Package ‘graphlayouts’

September 24, 2024

Title Additional Layout Algorithms for Network Visualizations

Version 1.2.0

Description

Several new layout algorithms to visualize networks are provided which are not part of 'igraph'. Most are based on the concept of stress majorization by Gansner et al. (2004) <doi:10.1007/978-3-540-31843-9_25>.
Some more specific algorithms allow the user to emphasize hidden group structures in networks or focus on specific nodes.

URL https://github.com/schochastics/graphlayouts,

https://schochastics.github.io/graphlayouts/

BugReports https://github.com/schochastics/graphlayouts/issues

Depends R (>= 3.6.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports igraph (>= 2.0.0), Rcpp

Suggests testthat, ggplot2, uwot

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.3.2

NeedsCompilation yes

Author David Schoch [aut, cre] (<https://orcid.org/0000-0003-2952-4812>),
Mark Ortmann [ctb] (author of R package oaqc),
Felix Schoenenberger [ctb] (author of R package oaqc)

Maintainer David Schoch <david@schochastics.net>

Repository CRAN

Date/Publication 2024-09-24 07:10:02 UTC

Contents

`annotate_circle` *annotate concentric circles*

Description

annotate concentric circles

Usage

```
annotate_circle(cent, col = "#00BFFF", format = "", pos = "top", text_size = 3)
```

Arguments

`cent`	centrality scores used for layout
`col`	color of text
`format`	either empty string or 'scientific'
`pos`	position of text ('top' or 'bottom')
`text_size`	font size for annotations

Details

this function is best used with layout_with_centrality together with draw_circle.

Value

annotated concentric circles around origin

Examples

```
library(igraph)

g <- sample_gnp(10, 0.4)
## Not run:
library(ggraph)
ggraph(g, layout = "centrality", centrality = closeness(g)) +
    draw_circle(use = "cent") +
    annotate_circle(closeness(g), pos = "bottom", format = "scientific") +
    geom_edge_link() +
    geom_node_point(shape = 21, fill = "grey25", size = 5) +
    theme_graph() +
    coord_fixed()

## End(Not run)
```

---

annotate_result — *Annotates the igraph object with orbit labels.*

Description

Annotates the igraph object with orbit labels.

Usage

```
annotate_result(graph, orbits, non_ind_freq)
```

Arguments

`graph`	Unmodified input graph.
`orbits`	List with n_orbits, e_orbits matrices.
`non_ind_freq`	A flag indicating whether non-induced frequencies have to be written or not.

Value

`orbits` if the input is not an igraph, the annotated igraph instead.

`as.edge_list` *This files is adapted from the archived R package oaqc Coerce graph input.*

Description

This files is adapted from the archived R package oaqc Coerce graph input.

Usage

```
as.edge_list(graph)
```

Arguments

`graph`	A matrix, data.frame or graph object.

Value

Edge list matrix.

`draw_circle` *Draw concentric circles*

Description

Draw concentric circles

Usage

```
draw_circle(col = "#00BFFF", use = "focus", max.circle)
```

Arguments

`col`	color of circles
`use`	one of 'focus' or 'cent'
`max.circle`	if use = 'focus' specifies the number of circles to draw

Details

this function is best used with a concentric layout such as layout_with_focus and layout_with_centrality.

Value

concentric circles around origin

Examples

```
library(igraph)
g <- sample_gnp(10, 0.4)

## Not run:
library(ggraph)
ggraph(g, layout = "centrality", centrality = degree(g)) +
    draw_circle(use = "cent") +
    geom_edge_link() +
    geom_node_point(shape = 21, fill = "grey25", size = 5) +
    theme_graph() +
    coord_fixed()

## End(Not run)
```

---

graph_manipulate *Manipulate graph*

Description

functions to manipulate a graph

Usage

```
reorder_edges(g, attr, desc = TRUE)
```

Arguments

g	igraph object
attr	edge attribute name used to sort edges
desc	logical. sort in descending (default) or ascending order

Details

`reorder_edges()` allows to reorder edges according to an attribute so that edges are drawn in the given order.

Value

manipulated graph

Author(s)

David Schoch

Examples

```
library(igraph)

g <- sample_gnp(10, 0.5)
E(g)$attr <- 1:ecount(g)
gn <- reorder_edges(g,"attr")
```

---

layout_as_metromap *Metro Map Layout*

Description

Metro map layout based on multicriteria optimization

Usage

```
layout_as_metromap(object, xy, l = 2, gr = 0.0025, w = rep(1, 5), bsize = 5)
```

Arguments

`object`	original graph
`xy`	initial layout of the original graph
`l`	desired multiple of grid point spacing. (l*gr determines desired edge length)
`gr`	grid spacing. (l*gr determines desired edge length)
`w`	weight vector for criteria (see details)
`bsize`	number of grid points a station can move away rom its original position

Details

The function optimizes the following five criteria using a hill climbing algorithm:

- *Angular Resolution Criterion*: The angles of incident edges at each station should be maximized, because if there is only a small angle between any two adjacent edges, then it can become difficult to distinguish between them
- *Edge Length Criterion*: The edge lengths across the whole map should be approximately equal to ensure regular spacing between stations. It is based on the preferred multiple, l, of the grid spacing, g. The purpose of the criterion is to penalize edges that are longer than or shorter than lg.
- *Balanced Edge Length Criterion*: The length of edges incident to a particular station should be similar
- *Line Straightness Criterion*: (not yet implemented) Edges that form part of a line should, where possible, be co-linear either side of each station that the line passes through
- *Octilinearity Criterion*: Each edge should be drawn horizontally, vertically, or diagonally at 45 degree, so we penalize edges that are not at a desired angle

Value

new coordinates for stations

Author(s)

David Schoch

References

Stott, Jonathan, et al. "Automatic metro map layout using multicriteria optimization." IEEE Transactions on Visualization and Computer Graphics 17.1 (2010): 101-114.

Examples

```
# the algorithm has problems with parallel edges
library(igraph)
g <- simplify(metro_berlin)
xy <- cbind(V(g)$lon, V(g)$lat) * 100

# the algorithm is not very stable. try playing with the parameters
## Not run:
xy_new <- layout_as_metromap(g, xy, l = 2, gr = 0.5, w = c(100, 100, 1, 1, 100), bsize = 35)

## End(Not run)
```

`layout_backbone` *backbone graph layout*

Description

emphasizes a hidden group structure if it exists in the graph. Calculates a layout for a sparsified network only including the most embedded edges. Deleted edges are added back after the layout is calculated.

Usage

```
layout_as_backbone(g, keep = 0.2, backbone = TRUE)

layout_igraph_backbone(g, keep = 0.2, backbone = TRUE, circular)
```

Arguments

g	igraph object
keep	fraction of edges to keep during backbone calculation
backbone	logical. Return edge ids of the backbone (Default: TRUE)
circular	not used

Details

The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'. 'ggraph' natively supports the layout.

Value

list of xy coordinates and vector of edge ids included in the backbone

References

Nocaj, A., Ortmann, M., & Brandes, U. (2015). Untangling the hairballs of multi-centered, small-world online social media networks. Journal of Graph Algorithms and Applications: JGAA, 19(2), 595-618.

Examples

```
library(igraph)

g <- sample_islands(9, 20, 0.4, 9)
g <- simplify(g)
V(g)$grp <- as.character(rep(1:9, each = 20))
bb <- layout_as_backbone(g, keep = 0.4)

# add backbone links as edge attribute
E(g)$col <- FALSE
E(g)$col[bb$backbone] <- TRUE
```

`layout_centrality` *radial centrality layout*

Description

arranges nodes in concentric circles according to a centrality index.

Usage

```
layout_with_centrality(
  g,
  cent,
  scale = TRUE,
  iter = 500,
  tol = 1e-04,
  tseq = seq(0, 1, 0.2)
)

layout_igraph_centrality(
  g,
```

```
    cent,
    scale = TRUE,
    iter = 500,
    tol = 1e-04,
    tseq = seq(0, 1, 0.2),
    circular
  )
```

Arguments

g	igraph object
cent	centrality scores
scale	logical. should centrality scores be scaled to $[0, 100]$? (Default: TRUE)
iter	number of iterations during stress optimization
tol	stopping criterion for stress optimization
tseq	numeric vector. increasing sequence of coefficients to combine regular stress and constraint stress. See details.
circular	not used

Details

The function optimizes a convex combination of regular stress and a constrained stress function which forces nodes to be arranged on concentric circles. The vector `tseq` is the sequence of parameters used for the convex combination. In iteration i of the algorithm $tseq[i]$ is used to combine regular and constraint stress as $(1 - tseq[i]) * stress_{regular} + tseq[i] * stress_{constraint}$. The sequence must be increasing, start at zero and end at one. The default setting should be a good choice for most graphs.

The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'. 'ggraph' natively supports the layout.

Value

matrix of xy coordinates

References

Brandes, U., & Pich, C. (2011). More flexible radial layout. Journal of Graph Algorithms and Applications, 15(1), 157-173.

See Also

layout_centrality_group

Examples

```
library(igraph)

g <- sample_gnp(10, 0.4)
## Not run:
library(ggraph)
ggraph(g, layout = "centrality", centrality = closeness(g)) +
    draw_circle(use = "cent") +
    geom_edge_link0() +
    geom_node_point(shape = 21, fill = "grey25", size = 5) +
    theme_graph() +
    coord_fixed()

## End(Not run)
```

---

layout_centrality_group

radial centrality group layout

Description

arranges nodes in concentric circles according to a centrality index and keeping groups within a angle range

Usage

```
layout_with_centrality_group(g, cent, group, shrink = 10, ...)

layout_igraph_centrality_group(g, cent, group, shrink = 10, circular, ...)
```

Arguments

`g`	igraph object
`cent`	centrality scores
`group`	vector indicating grouping of nodes
`shrink`	shrink the reserved angle range for a group to increase the gaps between groups
`...`	additional arguments to layout_with_centrality The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'. 'ggraph' natively supports the layout.
`circular`	not used

Value

matrix of xy coordinates

See Also

layout_centrality

Examples

```
library(igraph)
```

`layout_constrained_stress` *constrained stress layout*

Description

force-directed graph layout based on stress majorization with variable constrained

Usage

```
layout_with_constrained_stress(
  g,
  coord,
  fixdim = "x",
  weights = NA,
  iter = 500,
  tol = 1e-04,
  mds = TRUE,
  bbox = 30
)

layout_igraph_constrained_stress(
  g,
  coord,
  fixdim = "x",
  weights = NA,
  iter = 500,
  tol = 1e-04,
  mds = TRUE,
  bbox = 30,
  circular
)
```

Arguments

`g`	igraph object
`coord`	numeric vector. fixed coordinates for dimension specified in `fixdim`.
`fixdim`	string. which dimension should be fixed. Either "x" or "y".

weights	possibly a numeric vector with edge weights. If this is NULL and the graph has a weight edge attribute, then the attribute is used. If this is NA then no weights are used (even if the graph has a weight attribute). By default, weights are ignored. See details for more.
iter	number of iterations during stress optimization
tol	stopping criterion for stress optimization
mds	should an MDS layout be used as initial layout (default: TRUE)
bbox	constrain dimension of output. Only relevant to determine the placement of disconnected graphs
circular	not used

Details

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of an edges with higher weights are closer together (weights=1/E(g)$weight).

The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'. 'ggraph' natively supports the layout.

Value

matrix of xy coordinates

References

Gansner, E. R., Koren, Y., & North, S. (2004). Graph drawing by stress majorization. *In International Symposium on Graph Drawing* (pp. 239-250). Springer, Berlin, Heidelberg.

See Also

layout_constrained_stress3D

`layout_constrained_stress3D` *constrained stress layout in 3D*

Description

force-directed graph layout based on stress majorization with variable constrained in 3D

Usage

```
layout_with_constrained_stress3D(
  g,
  coord,
  fixdim = "x",
  weights = NA,
  iter = 500,
  tol = 1e-04,
  mds = TRUE,
  bbox = 30
)
```

Arguments

`g`	igraph object
`coord`	numeric vector. fixed coordinates for dimension specified in `fixdim`.
`fixdim`	string. which dimension should be fixed. Either "x", "y" or "z".
`weights`	possibly a numeric vector with edge weights. If this is NULL and the graph has a weight edge attribute, then the attribute is used. If this is NA then no weights are used (even if the graph has a weight attribute). By default, weights are ignored. See details for more.
`iter`	number of iterations during stress optimization
`tol`	stopping criterion for stress optimization
`mds`	should an MDS layout be used as initial layout (default: TRUE)
`bbox`	constrain dimension of output. Only relevant to determine the placement of disconnected graphs

Details

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of an edges with higher weights are closer together (weights=1/E(g)$weight).

This function does not come with direct support for igraph or ggraph.

Value

matrix of xyz coordinates

References

Gansner, E. R., Koren, Y., & North, S. (2004). Graph drawing by stress majorization. *In International Symposium on Graph Drawing* (pp. 239-250). Springer, Berlin, Heidelberg.

See Also

layout_constrained_stress

`layout_dynamic` *dynamic graph layout*

Description

Create layouts for longitudinal networks.

Usage

```
layout_as_dynamic(gList, weights = NA, alpha = 0.5, iter = 500, tol = 1e-04)
```

Arguments

`gList`	list of igraph objects. Each network must contain the same set of nodes.
`weights`	possibly a numeric vector with edge weights. If this is NULL and the graph has a weight edge attribute, then the attribute is used. If this is NA then no weights are used (even if the graph has a weight attribute). By default, weights are ignored. See details for more.
`alpha`	weighting of reference layout. See details.
`iter`	number of iterations during stress optimization
`tol`	stopping criterion for stress optimization

Details

The reference layout is calculated based on the union of all graphs. The parameter alpha controls the influence of the reference layout. For alpha=1, only the reference layout is used and all graphs have the same layout. For alpha=0, the stress layout of each individual graph is used. Values in-between interpolate between the two layouts.

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of an edges with higher weights are closer together (weights=1/E(g)$weight).

Value

list of coordinates for each graph

References

Brandes, U. and Indlekofer, N. and Mader, M. (2012). Visualization methods for longitudinal social networks and stochastic actor-oriented modeling. *Social Networks* 34 (3) 291-308

Examples

```
library(igraph)
g1 <- sample_gnp(20, 0.2)
g2 <- sample_gnp(20, 0.2)
g3 <- sample_gnp(20, 0.2)

xy <- layout_as_dynamic(list(g1, g2, g3))

# layout for first network
xy[[1]]
```

---

layout_fixed_coords *Layout with fixed coordinates*

Description

force-directed graph layout based on stress majorization with fixed coordinates for some nodes

Usage

```
layout_with_fixed_coords(
  g,
  coords,
  weights = NA,
  iter = 500,
  tol = 1e-04,
  mds = TRUE,
  bbox = 30
)

layout_igraph_fixed_coords(
  g,
  coords,
  weights = NA,
  iter = 500,
  tol = 1e-04,
  mds = TRUE,
  bbox = 30,
  circular
)
```

Arguments

g	igraph object
coords	numeric n x 2 matrix, where n is the number of nodes. values are either NA or fixed coordinates. coordinates are only calculated for the NA values.

weights	possibly a numeric vector with edge weights. If this is NULL and the graph has a weight edge attribute, then the attribute is used. If this is NA then no weights are used (even if the graph has a weight attribute). By default, weights are ignored. See details for more.
iter	number of iterations during stress optimization
tol	stopping criterion for stress optimization
mds	should an MDS layout be used as initial layout (default: TRUE)
bbox	constrain dimension of output. Only relevant to determine the placement of disconnected graphs
circular	not used

Details

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of an edges with higher weights are closer together (weights=1/E(g)$weight).

The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'. 'ggraph' natively supports the layout.

Value

matrix of xy coordinates

See Also

layout_constrained_stress

Examples

```
library(igraph)
set.seed(12)
g <- sample_bipartite(10, 5, "gnp", 0.5)
fxy <- cbind(c(rep(0, 10), rep(1, 5)), NA)
xy <- layout_with_fixed_coords(g, fxy)
```

---

layout_focus *radial focus layout*

Description

arrange nodes in concentric circles around a focal node according to their distance from the focus.

Usage

```
layout_with_focus(g, v, weights = NA, iter = 500, tol = 1e-04)

layout_igraph_focus(g, v, weights = NA, iter = 500, tol = 1e-04, circular)
```

Arguments

g	igraph object
v	id of focal node to be placed in the center
weights	possibly a numeric vector with edge weights. If this is NULL and the graph has a weight edge attribute, then the attribute is used. If this is NA then no weights are used (even if the graph has a weight attribute). By default, weights are ignored. See details for more.
iter	number of iterations during stress optimization
tol	stopping criterion for stress optimization
circular	not used

Details

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of an edges with higher weights are closer together (weights=1/E(g)$weight).

Value

a list containing xy coordinates and the distances to the focal node

References

Brandes, U., & Pich, C. (2011). More flexible radial layout. *Journal of Graph Algorithms and Applications*, 15(1), 157-173.

See Also

layout_focus_group The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'. 'ggraph' natively supports the layout.

Examples

```
library(igraph)
g <- sample_gnp(10, 0.4)
coords <- layout_with_focus(g, v = 1)
coords
```

---

layout_focus_group *radial focus group layout*

Description

arrange nodes in concentric circles around a focal node according to their distance from the focus and keep predefined groups in the same angle range.

Usage

```
layout_with_focus_group(
  g,
  v,
  group,
  shrink = 10,
  weights = NA,
  iter = 500,
  tol = 1e-04
)

layout_igraph_focus_group(
  g,
  v,
  group,
  shrink = 10,
  weights = NA,
  iter = 500,
  tol = 1e-04,
  circular
)
```

Arguments

g	igraph object
v	id of focal node to be placed in the center
group	vector indicating grouping of nodes
shrink	shrink the reserved angle range for a group to increase the gaps between groups
weights	possibly a numeric vector with edge weights. If this is NULL and the graph has a weight edge attribute, then the attribute is used. If this is NA then no weights are used (even if the graph has a weight attribute). By default, weights are ignored. See details for more.
iter	number of iterations during stress optimization
tol	stopping criterion for stress optimization
circular	not used

Details

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of an edges with higher weights are closer together (weights=1/E(g)$weight).

Value

matrix of xy coordinates

See Also

layout_focus The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'.

Examples

```
library(igraph)
g <- sample_islands(4, 5, 0.8, 2)
grp <- as.character(rep(1:4, each = 5))
layout_with_focus_group(g, v = 1, group = grp, shrink = 10)
```

---

layout_manipulate *manipulate layout*

Description

functions to manipulate an existing layout

Usage

```
layout_rotate(xy, angle)

layout_mirror(xy, axis = "vertical")
```

Arguments

xy	graph layout
angle	angle for rotation
axis	mirror horizontal or vertical

Details

These functions are mostly useful for deterministic layouts such as layout_with_stress

Value

manipulated matrix of xy coordinates

Author(s)

David Schoch

Examples

```
library(igraph)
g <- sample_gnp(50, 0.3)

xy <- layout_with_stress(g)

# rotate 90 degrees
xy <- layout_rotate(xy, 90)

# flip horizontally
xy <- layout_mirror(xy, "horizontal")
```

`layout_multilevel` *multilevel layout*

Description

Layout algorithm to visualize multilevel networks

Usage

```
layout_as_multilevel(
  g,
  type = "all",
  FUN1,
  FUN2,
  params1 = NULL,
  params2 = NULL,
  ignore_iso = TRUE,
  project2D = TRUE,
  alpha = 35,
  beta = 45
)

layout_igraph_multilevel(
  g,
  type = "all",
  FUN1,
  FUN2,
  params1 = NULL,
  params2 = NULL,
  ignore_iso = TRUE,
  alpha = 35,
  beta = 45,
  circular
)
```

Arguments

g	igraph object. Must have a vertex attribute "lvl" which is 1 or 2.
type	one of "all", "separate","fix1" or "fix2". see details
FUN1	if type="separate", the layout function to be used for level 1
FUN2	if type="separate", the layout function to be used for level 2
params1	named list of parameters for FUN1
params2	named list of parameters for FUN2
ignore_iso	treatment of isolates within levels. see details
project2D	logical. Defaults to TRUE (project to 2D).
alpha	angle for isometric projection between 0 and 90
beta	angle for isometric projection between 0 and 90
circular	not used

Details

The algorithm internally computes a 3D layout where each level is in a separate y-plane. The layout is then projected into 2D via an isometric mapping, controlled by the parameters `alpha` and `beta`. It may take some adjusting to `alpha` and `beta` to find a good perspective.

If type="all", the layout is computed at once for the complete network. For type="separate", two user specified layout algorithms (`FUN1` and `FUN2`) are used for the levels. The named lists `param1` and `param2` can be used to set parameters for `FUN1` and `FUN2`. This option helpful for situations where different structural features of the levels should be emphasized.

For type="fix1" and type="fix2" only one of the level layouts is fixed. The other one is calculated by optimizing the inter level ties, such that they are drawn (almost) vertical.

The `ignore_iso` parameter controls the handling of isolates. If TRUE, nodes without inter level edges are ignored during the layout process and added at the end. If FALSE they are left unchanged

The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'.

Value

matrix of xy coordinates

Examples

```
library(igraph)
data("multilvl_ex")

# compute a layout for the whole network
xy <- layout_as_multilevel(multilvl_ex, type = "all", alpha = 25, beta = 45)

# compute a layout for each level separately and combine them
xy <- layout_as_multilevel(multilvl_ex,
    type = "separate",
    FUN1 = layout_as_backbone,
```

```
    FUN2 = layout_with_stress,
    alpha = 25, beta = 45
)
```

`layout_pmds` *pivot MDS graph layout*

Description

similar to layout_with_mds but uses only a small set of pivots for MDS. Considerably faster than MDS and thus applicable for larger graphs.

Usage

```
layout_with_pmds(g, pivots, weights = NA, D = NULL, dim = 2)

layout_igraph_pmds(g, pivots, weights = NA, D = NULL, circular)
```

Arguments

`g`	igraph object
`pivots`	number of pivots
`weights`	possibly a numeric vector with edge weights. If this is NULL and the graph has a weight edge attribute, then the attribute is used. If this is NA then no weights are used (even if the graph has a weight attribute). By default, weights are ignored. See details for more.
`D`	precomputed distances from pivots to all nodes (if available, default: NULL)
`dim`	dimensionality of layout (defaults to 2)
`circular`	not used

Details

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of an edges with higher weights are closer together (weights=1/E(g)$weight)

The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'. 'ggraph' natively supports the layout.

Value

matrix of coordinates

Author(s)

David Schoch

References

Brandes, U. and Pich, C. (2006). Eigensolver Methods for Progressive Multidimensional Scaling of Large Data. In *International Symposium on Graph Drawing* (pp. 42-53). Springer

Examples

```
## Not run:
library(igraph)
library(ggraph)

g <- sample_gnp(1000, 0.01)

xy <- layout_with_pmds(g, pivots = 100)

## End(Not run)
```

---

layout_sparse_stress   *sparse stress graph layout*

Description

stress majorization for larger graphs based on a set of pivot nodes.

Usage

```
layout_with_sparse_stress(g, pivots, weights = NA, iter = 500)

layout_igraph_sparse_stress(g, pivots, weights = NA, iter = 500, circular)
```

Arguments

g	igraph object
pivots	number of pivots
weights	ignored
iter	number of iterations during stress optimization
circular	not used

Details

The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'. 'ggraph' natively supports the layout.

Value

matrix of xy coordinates

Author(s)

David Schoch

References

Ortmann, M. and Klimenta, M. and Brandes, U. (2016). A Sparse Stress Model. https://arxiv.org/pdf/1608.08909.pdf

Examples

```
## Not run:
library(igraph)
library(ggraph)

g <- sample_gnp(1000, 0.005)

ggraph(g, layout = "sparse_stress", pivots = 100) +
    geom_edge_link0(edge_colour = "grey66") +
    geom_node_point(shape = 21, fill = "grey25", size = 5) +
    theme_graph()

## End(Not run)
```

`layout_spectral` *spectral graph layouts*

Description

Using a set of eigenvectors of matrices associated with a graph as coordinates

Usage

```
layout_with_eigen(g, type = "laplacian", ev = "smallest")

layout_igraph_eigen(g, type = "laplacian", ev = "smallest", circular)
```

Arguments

`g`	igraph object
`type`	matrix to be used for spectral decomposition. either 'adjacency' or 'laplacian'
`ev`	eigenvectors to be used. Either 'smallest' or 'largest'.
`circular`	not used

Details

The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'. 'ggraph' natively supports the layout.

Value

matrix of xy coordinates

Author(s)

David Schoch

Examples

```
library(igraph)

g <- sample_gnp(50, 0.2)

xy <- layout_with_eigen(g, type = "adjacency", ev = "largest")

xy <- layout_with_eigen(g, type = "adjacency", ev = "smallest")

xy <- layout_with_eigen(g, type = "laplacian", ev = "largest")

xy <- layout_with_eigen(g, type = "laplacian", ev = "smallest")
```

---

layout_stress *stress majorization layout*

Description

force-directed graph layout based on stress majorization. Similar to Kamada-Kawai, but generally faster and with better results.

Usage

```
layout_with_stress(
  g,
  weights = NA,
  iter = 500,
  tol = 1e-04,
  mds = TRUE,
  bbox = 30
)

layout_igraph_stress(
  g,
  weights = NA,
  iter = 500,
  tol = 1e-04,
  mds = TRUE,
  bbox = 30,
  circular
)
```

Arguments

`g`	igraph object
`weights`	possibly a numeric vector with edge weights. If this is NULL and the graph has a weight edge attribute, then the attribute is used. If this is NA then no weights are used (even if the graph has a weight attribute). By default, weights are ignored. See details for more.
`iter`	number of iterations during stress optimization
`tol`	stopping criterion for stress optimization
`mds`	should an MDS layout be used as initial layout (default: TRUE)
`bbox`	width of layout. Only relevant to determine the placement of disconnected graphs
`circular`	not used

Details

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of an edges with higher weights are closer together (weights=1/E(g)$weight).

The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'. 'ggraph' natively supports the layout.

Value

matrix of xy coordinates

References

Gansner, E. R., Koren, Y., & North, S. (2004). Graph drawing by stress majorization. *In International Symposium on Graph Drawing* (pp. 239-250). Springer, Berlin, Heidelberg.

See Also

layout_stress3D

Examples

```
library(igraph)
set.seed(665)

g <- sample_pa(100, 1, 1, directed = FALSE)

# calculate layout manually
xy <- layout_with_stress(g)

# use it with ggraph
## Not run:
library(ggraph)
ggraph(g, layout = "stress") +
    geom_edge_link0(edge_width = 0.2, colour = "grey") +
```

```
    geom_node_point(col = "black", size = 0.3) +
    theme_graph()

## End(Not run)
```

`layout_stress3D` *stress majorization layout in 3D*

Description

force-directed graph layout based on stress majorization in 3D.

Usage

```
layout_with_stress3D(
  g,
  weights = NA,
  iter = 500,
  tol = 1e-04,
  mds = TRUE,
  bbox = 30
)
```

Arguments

`g`	igraph object
`weights`	possibly a numeric vector with edge weights. If this is NULL and the graph has a weight edge attribute, then the attribute is used. If this is NA then no weights are used (even if the graph has a weight attribute). By default, weights are ignored. See details for more.
`iter`	number of iterations during stress optimization
`tol`	stopping criterion for stress optimization
`mds`	should an MDS layout be used as initial layout (default: TRUE)
`bbox`	width of layout. Only relevant to determine the placement of disconnected graphs

Details

Be careful when using weights. In most cases, the inverse of the edge weights should be used to ensure that the endpoints of an edges with higher weights are closer together (weights=1/E(g)$weight).

Value

matrix of xyz coordinates

References

Gansner, E. R., Koren, Y., & North, S. (2004). Graph drawing by stress majorization. *In International Symposium on Graph Drawing* (pp. 239-250). Springer, Berlin, Heidelberg.

See Also

layout_stress

---

layout_umap *UMAP graph layouts*

Description

Using the UMAP dimensionality reduction algorithm as a graph layout

Usage

```
layout_with_umap(g, pivots = NULL, ...)

layout_igraph_umap(g, circular, ...)
```

Arguments

`g`	igraph object
`pivots`	if not NULL, number of pivot nodes to use for distance calculation (for large graphs).
`...`	additional parameters for umap. See the `?uwot::umap` for help.
`circular`	not used

Details

The layout_igraph_* function should not be used directly. It is only used as an argument for plotting with 'igraph'. UMAP can be tuned by many different parameters. Refer to the documentation at https://github.com/jlmelville/uwot for help

Value

matrix of xy coordinates

Author(s)

David Schoch

References

McInnes, Leland, John Healy, and James Melville. "Umap: Uniform manifold approximation and projection for dimension reduction." arXiv preprint arXiv:1802.03426 (2018).

Examples

```
library(igraph)

g <- sample_islands(10, 20, 0.6, 10)
# xy <- layout_with_umap(g, min_dist = 0.5)
```

`metro_berlin` *Subway network of Berlin*

Description

A dataset containing the subway network of Berlin

Usage

```
metro_berlin
```

Format

igraph object

References

Kujala, Rainer, et al. "A collection of public transport network data sets for 25 cities." Scientific data 5 (2018): 180089.

`multilvl_ex` *Multilevel example Network*

Description

Multilevel network, where both levels have different structural features

Usage

```
multilvl_ex
```

Format

igraph object

oaqc *Calculates the orbit-aware quad census on an edge and node level, see* `vignette('oaqc')`.

Description

Calculates the orbit-aware quad census on an edge and node level, see `vignette('oaqc')`.

Usage

```
oaqc(graph, non_ind_freq = F, file = "")
```

Arguments

`graph`	A matrix, data.frame or graph object.
`non_ind_freq`	A flag indicating whether non-induced frequencies have to be returned or not.
`file`	Name (and location) of the file to be written.

Value

orbit-aware quad census on a node and edge level. Consult `vignette('oaqc')` to see the correspondence between orbit and quad.

Examples

```
k4 <- data.frame(
    source = c(0, 0, 0, 1, 1, 2),
    target = c(1, 2, 3, 2, 3, 3)
)

k4orbits <- oaqc(k4, non_ind_freq = TRUE)
print(k4orbits)
```

Index