

# Package: graphlayouts (via r-universe)

May 31, 2026

**Title** Additional Layout Algorithms for Network Visualizations

**Version** 1.2.3

**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](https://doi.org/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, oaqc

**LinkingTo** Rcpp, RcppArmadillo

**RoxygenNote** 7.3.2

**Roxygen** list(markdown = TRUE)

**Config/pak/sysreqs** libglpk-dev libxml2-dev

**Repository** <https://cynkra.r-universe.dev>

**Date/Publication** 2026-05-08 13:15:53 UTC

**RemoteUrl** <https://github.com/schochastics/graphlayouts>

**RemoteRef** HEAD

**RemoteSha** 095cf074d3aab8270dedbf6247303c00e1d016e0

## Contents

annotate_circle . . . . .	2
draw_circle . . . . .	3
graph_manipulate . . . . .	4
layout_as_metromap . . . . .	5
layout_backbone . . . . .	6
layout_centrality . . . . .	7
layout_centrality_group . . . . .	9
layout_constrained_stress . . . . .	10
layout_constrained_stress3D . . . . .	11
layout_dynamic . . . . .	13
layout_fixed_coords . . . . .	14
layout_focus . . . . .	15
layout_focus_group . . . . .	16
layout_manipulate . . . . .	18
layout_multilevel . . . . .	19
layout_pmds . . . . .	21
layout_sparse_stress . . . . .	22
layout_spectral . . . . .	23
layout_stress . . . . .	24
layout_stress3D . . . . .	26
layout_umap . . . . .	27
metro_berlin . . . . .	28
multlvl_ex . . . . .	28
<b>Index</b>	<b>29</b>

---

annotate_circle	<i>annotate concentric circles</i>
-----------------	------------------------------------

---

### 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)
```

---

draw_circle	<i>Draw concentric circles</i>
-------------	--------------------------------

---

**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. ( $l \cdot gr$ determines desired edge length)
gr	grid spacing. ( $l \cdot gr$ determines desired edge length)
w	weight vector for criteria (see details)
bsize	number of grid points a station can move away from 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	<i>backbone graph layout</i>
-----------------	------------------------------

---

**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**

<code>g</code>	igraph object
<code>keep</code>	fraction of edges to keep during backbone calculation
<code>backbone</code>	logical. Return edge ids of the backbone (Default: TRUE)
<code>circular</code>	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)
## Not run:
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

## End(Not run)
```

---

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

<code>g</code>	igraph object
<code>cent</code>	centrality scores
<code>scale</code>	logical. should centrality scores be scaled to $[0, 100]$ ? (Default: TRUE)
<code>iter</code>	number of iterations during stress optimization
<code>tol</code>	stopping criterion for stress optimization
<code>tseq</code>	numeric vector. increasing sequence of coefficients to combine regular stress and constraint stress. See details.
<code>circular</code>	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**

<code>g</code>	igraph object
<code>coord</code>	numeric vector. fixed coordinates for dimension specified in <code>fixdim</code> .
<code>fixdim</code>	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**

<code>g</code>	igraph object
<code>coord</code>	numeric vector. fixed coordinates for dimension specified in <code>fixdim</code> .
<code>fixdim</code>	string. which dimension should be fixed. Either "x", "y" or "z".
<code>weights</code>	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.
<code>iter</code>	number of iterations during stress optimization
<code>tol</code>	stopping criterion for stress optimization
<code>mds</code>	should an MDS layout be used as initial layout (default: TRUE)
<code>bbox</code>	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	<i>dynamic graph layout</i>
----------------	-----------------------------

---

### Description

Create layouts for longitudinal networks.

### Usage

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

### Arguments

<code>gList</code>	list of igraph objects. Each network must contain the same set of nodes.
<code>weights</code>	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.
<code>alpha</code>	weighting of reference layout. See details.
<code>iter</code>	number of iterations during stress optimization
<code>tol</code>	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	<i>radial focus layout</i>
--------------	----------------------------

---

### 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**

<code>g</code>	igraph object
<code>v</code>	id of focal node to be placed in the center
<code>weights</code>	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.
<code>iter</code>	number of iterations during stress optimization
<code>tol</code>	stopping criterion for stress optimization
<code>circular</code>	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**

<code>g</code>	igraph object
<code>v</code>	id of focal node to be placed in the center
<code>group</code>	vector indicating grouping of nodes
<code>shrink</code>	shrink the reserved angle range for a group to increase the gaps between groups
<code>weights</code>	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.
<code>iter</code>	number of iterations during stress optimization
<code>tol</code>	stopping criterion for stress optimization
<code>circular</code>	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	<i>manipulate layout</i>
-------------------	--------------------------

---

**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	<i>multilevel layout</i>
-------------------	--------------------------

---

**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**

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

  ## End(Not run)

```

---

layout_pmds	<i>pivot MDS graph layout</i>
-------------	-------------------------------

---

### 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

<code>g</code>	igraph object
<code>pivots</code>	number of pivots
<code>weights</code>	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.
<code>D</code>	precomputed distances from pivots to all nodes (if available, default: NULL)
<code>dim</code>	dimensionality of layout (defaults to 2)
<code>circular</code>	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	<i>spectral graph layouts</i>
-----------------	-------------------------------

---

**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**

<code>g</code>	igraph object
<code>weights</code>	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.
<code>iter</code>	number of iterations during stress optimization
<code>tol</code>	stopping criterion for stress optimization
<code>mds</code>	should an MDS layout be used as initial layout (default: TRUE)
<code>bbox</code>	width of layout. Only relevant to determine the placement of disconnected graphs
<code>circular</code>	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

<code>g</code>	igraph object
<code>weights</code>	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.
<code>iter</code>	number of iterations during stress optimization
<code>tol</code>	stopping criterion for stress optimization
<code>mds</code>	should an MDS layout be used as initial layout (default: TRUE)
<code>bbox</code>	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	<i>UMAP graph layouts</i>
-------------	---------------------------

---

**Description**

Using the UMAP dimensionality reduction algorithm as a graph layout

**Usage**

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

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

**Arguments**

<code>g</code>	igraph object
<code>pivots</code>	if not NULL, number of pivot nodes to use for distance calculation (for large graphs).
<code>...</code>	additional parameters for umap. See the <code>?uwot::umap</code> for help.
<code>circular</code>	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	<i>Subway network of Berlin</i>
--------------	---------------------------------

---

**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.

---

multlvl_ex	<i>Multilevel example Network</i>
------------	-----------------------------------

---

**Description**

Multilevel network, where both levels have different structural features

**Usage**

```
multlvl_ex
```

**Format**

igraph object

# Index

- \* **datasets**
  - metro\_berlin, 28
  - multilvl\_ex, 28
- annotate\_circle, 2
- draw\_circle, 3, 3
- graph\_manipulate, 4
- layout\_as\_backbone (layout\_backbone), 6
- layout\_as\_dynamic (layout\_dynamic), 13
- layout\_as\_metromap, 5
- layout\_as\_multilevel
  - (layout\_multilevel), 19
- layout\_backbone, 6
- layout Centrality, 7, 10
- layout Centrality\_group, 8, 9
- layout\_constrained\_stress, 10, 12, 15
- layout\_constrained\_stress3D, 11, 11
- layout\_dynamic, 13
- layout\_fixed\_coords, 14
- layout\_focus, 15, 18
- layout\_focus\_group, 16, 16
- layout\_igraph\_backbone
  - (layout\_backbone), 6
- layout\_igraph\_Centrality
  - (layout\_Centrality), 7
- layout\_igraph\_Centrality\_group
  - (layout\_Centrality\_group), 9
- layout\_igraph\_constrained\_stress
  - (layout\_constrained\_stress), 10
- layout\_igraph\_eigen (layout\_spectral), 23
- layout\_igraph\_fixed\_coords
  - (layout\_fixed\_coords), 14
- layout\_igraph\_focus (layout\_focus), 15
- layout\_igraph\_focus\_group
  - (layout\_focus\_group), 16
- layout\_igraph\_multilevel
  - (layout\_multilevel), 19
- layout\_igraph\_pmds (layout\_pmds), 21
- layout\_igraph\_sparse\_stress
  - (layout\_sparse\_stress), 22
- layout\_igraph\_stress (layout\_stress), 24
- layout\_igraph\_umap (layout\_umap), 27
- layout\_manipulate, 18
- layout\_mirror (layout\_manipulate), 18
- layout\_multilevel, 19
- layout\_pmds, 21
- layout\_rotate (layout\_manipulate), 18
- layout\_sparse\_stress, 22
- layout\_spectral, 23
- layout\_stress, 24, 27
- layout\_stress3D, 25, 26
- layout\_umap, 27
- layout\_with\_Centrality, 3
- layout\_with\_Centrality
  - (layout\_Centrality), 7
- layout\_with\_Centrality\_group
  - (layout\_Centrality\_group), 9
- layout\_with\_constrained\_stress
  - (layout\_constrained\_stress), 10
- layout\_with\_constrained\_stress3D
  - (layout\_constrained\_stress3D), 11
- layout\_with\_eigen (layout\_spectral), 23
- layout\_with\_fixed\_coords
  - (layout\_fixed\_coords), 14
- layout\_with\_focus, 3
- layout\_with\_focus (layout\_focus), 15
- layout\_with\_focus\_group
  - (layout\_focus\_group), 16
- layout\_with\_mds, 21
- layout\_with\_pmds (layout\_pmds), 21
- layout\_with\_sparse\_stress
  - (layout\_sparse\_stress), 22
- layout\_with\_stress, 18
- layout\_with\_stress (layout\_stress), 24
- layout\_with\_stress3D (layout\_stress3D),

[26](#)  
layout\_with\_umap (layout\_umap), [27](#)  
metro\_berlin, [28](#)  
multilvl\_ex, [28](#)  
reorder\_edges (graph\_manipulate), [4](#)