Package 'RGraphSpace'

May 7, 2026

Type Package

Version 1.2.0

Title A Lightweight Interface Between 'igraph' and 'ggplot2' Graphics

Description An interface to integrate 'igraph' and 'ggplot2' graphics within a normalized coordinate system. 'RGraphSpace' implements geometric objects based on 'ggplot2' prototypes, optimized for the representation of large networks. The package provides three specialized 'geoms' to translate graph data into geometric layers, supporting customization of aesthetics and visual styles. These 'geoms' use a dual-anchor normalization approach to align layers, required for analyses where network elements must be referenced to a spatial map. 'RGraphSpace' aims to facilitate side-by-side visualization of multiple graphs spatially aligned with reference maps and images.

Depends R(>= 4.5), methods, ggplot2

Imports grDevices, grid, igraph, scales, rlang, ggrastr, lifecycle

Suggests knitr, rmarkdown, testthat, ggnewscale, patchwork

Enhances RedeR

License Artistic-2.0

VignetteBuilder knitr

URL https://github.com/sysbiolab/RGraphSpace

BugReports https://github.com/sysbiolab/RGraphSpace/issues

Collate 'geom-edgespace.R' 'geom-graphspace.R' 'geom-nodespace.R' 'gspace-checks.R' 'gspace-classes.R' 'gspace-constructor.R' 'gspace-generics.R' 'gspace-methods.R' 'gspace-misc.R' 'gspace-normalize.R' 'gspace-supplements.R' 'gspace-themes.R' 'gspace-validation.R' 'inject-nodespace.R'

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Sysbiolab Team [aut],
Flávio Kessler [ctb],
Jonathan Back [ctb],
Lana Querne [ctb],
Victor Apolonio [ctb],
Vinicius Chagas [ctb],
Mauro Castro [cre] (ORCID: <https://orcid.org/0000-0003-4942-8131>)

Maintainer Mauro Castro <mauro.a.castro@gmail.com>

Repository CRAN

Date/Publication 2026-04-25 09:50:02 UTC

Contents

`as_colorraster` *Map numeric values to a color raster*

Description

Helper function that converts numeric values to colors and returns a raster image. Useful for visualizing numeric matrices as color backgrounds.

Usage

```
as_colorraster(x, palette = hcl.colors(30), na.color = "white")
```

Arguments

x	A numeric vector or matrix containing values to be mapped to colors.
palette	A vector of colors used as the palette. By default, hcl.colors(30) is used.
na.color	Color used for NA values. Defaults to NA.

Details

Values in x are rescaled to the range of the palette using scales::rescale(), and each value is mapped to a corresponding color. If x is a matrix, the resulting raster preserves the same dimensions.

Value

A raster object as produced by as.raster().

Examples

```
# Convert the volcano matrix to a color raster
img <- as_colorraster(volcano)
plot(img)
```

GeomEdgeSpace *GeomEdgeSpace: a ggplot2 prototype for GraphSpace-class methods*

Description

GeomEdgeSpace is the underlying ggproto object used by geom_edgespace to draw edge elements in a graph layout.

This geom is designed for network diagrams, where graph attributes are often already in their final form (e.g., hex colors).

Usage

```
GeomEdgeSpace
```

Format

An object of class GeomEdgeSpace (inherits from GeomSegment, Geom, ggproto, gg) of length 7.

Aesthetics

GeomEdgeSpace understands ggplot2's conventions for segment-like geoms.

See Also

geom_edgespace, geom_segment

GeomGraphSpace *GeomGraphSpace: a ggplot2 prototype for GraphSpace-class methods*

Description

GeomGraphSpace is the underlying ggproto object used by geom_graphspace to draw node and edge elements in a graph layout.

This geom is designed for network diagrams, where graph attributes are often already in their final form (e.g., hex colors).

Usage

```
GeomGraphSpace
```

Format

An object of class GeomGraphSpace (inherits from Geom, ggproto, gg) of length 6.

Aesthetics

GeomGraphSpace understands ggplot2's conventions for point-like geoms.

See Also

geom_graphspace, geom_point

GeomNodeSpace *GeomNodeSpace: a ggplot2 prototype for GraphSpace-class methods*

Description

GeomNodeSpace is the underlying ggproto object used by geom_nodespace to draw node elements in a graph layout.

This geom is designed for network diagrams, where graph attributes are often already in their final form (e.g., hex colors).

Usage

```
GeomNodeSpace
```

Format

An object of class GeomNodeSpace (inherits from GeomPoint, Geom, ggproto, gg) of length 6.

Aesthetics

GeomNodeSpace understands ggplot2's conventions for point-like geoms.

See Also

geom_nodespace, geom_point

---

geom_edgespace *Draw edge elements in a 2D graph layout*

Description

Constructor for GeomEdgeSpace ggproto objects.

A wrapper around geom_segment that enables direct use of edge attributes stored in GraphSpace objects as aesthetics.

This geom is designed to create edge-level aesthetics such as `colour` and `linewidth`, or any custom aesthetics defined in GeomEdgeSpace.

Usage

```
geom_edgespace(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = FALSE,
  size_aes = NULL,
  range = c(1, 6),
  arrow_size = 1,
  arrow_offset = 0.01,
  lineend = "butt",
  linejoin = "mitre"
)
```

Arguments

mapping	Set of aesthetic mappings created by `ggplot2::aes()`. These mappings override global aesthetics and are not inherited from the top-level plot.
data	A GraphSpace object.
stat	The statistical transformation to use on the data. Defaults to `identity`.
position	Position adjustment, either as a string or the result of a call to a position adjustment function.

...	Additional parameters passed to the underlying drawing function in GeomEdgeSpace.
na.rm	Logical. Should missing values be removed? Defaults to FALSE.
show.legend	Logical or a named logical vector indicating whether this layer should be included in legends.
inherit.aes	Logical. If FALSE (default), the layer will use aesthetics defined in mapping.
size_aes	Optional node size mapping, created with ggplot2::aes(). Can be used to synchronize geom_nodespace and geom_edgespace to ensure directed edges and arrows are offset from node boundaries. This prevents overlaps when geom_nodespace includes a dynamic size or stroke mappings. For automated scale detection, use inject_nodespace instead.
range	Optional numeric vector of length 2 (default c(1, 6)) specifying the min/max node diameters in mm. Used with size_aes and should match the range in ggplot2::scale_size_continuous() for accurate edge clipping. For automated scale detection, use inject_nodespace instead.
arrow_size	Numeric scaling factor controlling arrowhead geometry (see 'drawing' section).
arrow_offset	Numeric value controlling the base offset of arrows at edge endpoints (see 'drawing' section).
lineend	Line end style (round, butt, square). Supplied for compatibility with geom_segment.
linejoin	Line join style (round, mitre, bevel). Supplied for compatibility with geom_segment.

Details

arrow_size is a numeric scaling factor controlling arrowhead geometry. The value is interpreted in the same numeric space as line width (lwd), ensuring consistent scaling between edge strokes and arrowheads.

arrow_offset is an additive term that offsets arrow endpoints uniformly in graph space and is bounded by the edge length, in NPC units.

Arrowhead types are specified in the GraphSpace constructor.

Value

A ggplot2 layer that renders edge segments defined by GeomEdgeSpace.

Aesthetics

geom_edgespace() understands geom_segment aesthetics.

If these aesthetics are not explicitly provided in aes(), they are automatically retrieved from the GraphSpace object.

x, y, xend, yend	Required (automatically supplied).
colour	Node border colour (see aes_colour_fill_alpha).
alpha	Transparency (see aes_colour_fill_alpha).
linetype	Edge line type (see aes_linetype_size_shape).
linewidth	Edge line width (see aes_linetype_size_shape).

Required aesthetics (x, y, xend, yend, ...) are supplied from the GraphSpace object and do not need to be manually mapped.

Additional parameters can be passed to control fixed values for the layer. For example: `colour = "grey"`, `linetype = 2`, `linewidth = 1`.

Arrows can be further adjusted by `arrow_size` and `arrow_offset` arguments (see *details*).

See Also

GraphSpace, geom_nodespace, geom_graphspace, geom_segment

Examples

```
# Load a demo igraph
data('gtoy1', package = 'RGraphSpace')

# Create a GraphSpace object
gs <- GraphSpace(gtoy1)

## Not run:

ggplot() +
  geom_edgespace(data = gs) +
  geom_nodespace(data = gs) +
  theme(aspect.ratio = 1)


## End(Not run)
```

`geom_graphspace` *Draw node and edge elements in a 2D graph layout*

Description

Constructor for GeomGraphSpace ggproto objects.

A wrapper around geom_point that enables direct use of node attributes stored in GraphSpace objects as aesthetics.

This geom is designed to map node-level attributes (e.g., `fill`, `size`) or any aesthetics supported by GeomPoint.

Usage

```
geom_graphspace(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
```

```
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = FALSE,
  arrow_size = 1,
  arrow_offset = 0.01
)
```

Arguments

mapping	Set of aesthetic mappings created by ggplot2::aes(). These mappings override global aesthetics and are not inherited from the top-level plot.
data	A GraphSpace object.
stat	The statistical transformation to use on the data. Defaults to identity.
position	Position adjustment, either as a string or the result of a call to a position adjustment function.
...	Additional parameters passed to the underlying drawing function in GeomGraphSpace.
na.rm	Logical. Should missing values be removed? Defaults to FALSE.
show.legend	Logical or a named logical vector indicating whether this layer should be included in legends.
inherit.aes	Logical. If FALSE (default), the layer will use aesthetics defined in mapping.
arrow_size	Numeric scaling factor controlling arrowhead geometry (see 'drawing' section).
arrow_offset	Numeric value controlling the base offset of arrows at edge endpoints (see 'drawing' section).

Value

A ggplot2 layer that renders node glyphs defined by GeomGraphSpace.

Aesthetics for node drawing

Nodes are drawn in the main layer of geom_graphspace(), which understands geom_point aesthetics.

If these aesthetics are not explicitly provided in aes(), they are automatically retrieved from the GraphSpace object.

x, y, vertex	Required (automatically supplied).
fill	Node interior colour (see aes_colour_fill_alpha).
colour	Node border colour (see aes_colour_fill_alpha).
alpha	Transparency (see aes_colour_fill_alpha).
shape	Node shape (see points and aes_linetype_size_shape).
size	Node size (see *drawing* section and aes_linetype_size_shape).
stroke	Node line width (see gg_par and aes_linetype_size_shape).

Required aesthetics x, y, and vertex are supplied from the GraphSpace object and do not need to be manually mapped.

Additional parameters can be passed to control fixed values for the layer. For example: `fill = "red"`, `stroke = 3`, `alpha = 0.5`, or `shape = 21`.

The interpretation of `size` depends on how it is provided:

- **As an aesthetic**: When mapped within `aes()`, `size` follows the behavior of geom_point, using absolute units to ensure consistency with the plot legends.
- **As a parameter**: When set outside `aes()`, `size` is treated as a percentage of the viewport (`[0, 100]`), scaling in `npc` units. This allows nodes to resize dynamically with viewport changes.

Edge context-aware parameters

These parameters control the edge appearance. If not explicitly provided, they are automatically retrieved from the GraphSpace object. They can be a single value or a vector matching the number of edges:

`edge_colour`	Node border colour.
`edge_linetype`	Edge line type.
`edge_linewidth`	Edge line width.
`edge_alpha`	Edge transparency.

Edge global parameters

These parameters apply globally to all edges in the layer:

`arrow_size`	Arrow scaling factor (default = 1).
`arrow_offset`	Arrow offset from nodes (default = 0.01).
`arrow_lineend`	Line end style (see gpar).
`arrow_linejoin`	Line join style (see gpar).

arrow_size is a numeric scaling factor controlling arrowhead geometry. The value is interpreted in the same numeric space as line width (`lwd`), ensuring consistent scaling between edge strokes and arrowheads.

arrow_offset is an additive term that offsets arrow endpoints uniformly in graph space and is bounded by the edge length, in NPC units.

Arrowhead types are specified in the GraphSpace constructor.

See Also

GraphSpace, geom_nodespace, geom_edgespace, geom_point

Examples

```
# Make a demo igraph
library(igraph)
gtoy1 <- make_star(15, mode="out")

# Set some node attributes
V(gtoy1)$nodeSize <- runif(vcount(gtoy1), 1, 20)
V(gtoy1)$nodeColor <- rainbow(vcount(gtoy1))
```

```
# Set some variables
V(gtoy1)$user_var1 <- runif(vcount(gtoy1), 1, 3)^3
V(gtoy1)$user_var2 <-  rep(c(1, 2, 3), each = 5)

# Create a GraphSpace object
gs <- GraphSpace(gtoy1, layout = layout_in_circle(gtoy1))

## Not run:

# Example 1: Nodes scaling with the legend
# When 'size' is mapped inside aes(), it follows
# ggplot2 default behavior: sizes are translated
# to absolute units (mm) via 'scale_size()'.

ggplot() +
  geom_graphspace(
  mapping = aes(size = nodeSize, fill = user_var2),
  data = gs, arrow_offset = 0.01) +
  scale_size(range = c(1, 12)) +
  theme(aspect.ratio = 1)

# Example 2: Nodes scaling with the viewport
# When 'size' is passed as a node attribute,
# as inherited from the igraph object, it is
# interpreted as a percentage of the plot area
# and translated to NPC units.

ggplot() +
  geom_graphspace(mapping = aes(fill = user_var2),
  data = gs, arrow_offset = 0.01) +
  theme(aspect.ratio = 1)


## End(Not run)
```

`geom_nodespace` *Draw node elements in a 2D graph layout*

Description

Constructor for GeomNodeSpace ggproto objects.

A wrapper around geom_point that enables direct use of node attributes stored in GraphSpace objects as aesthetics.

This geom is designed to map node-level attributes (e.g., `fill`, `size`) or any aesthetics supported by GeomPoint.

Usage

```
geom_nodespace(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = FALSE
)
```

Arguments

mapping	Set of aesthetic mappings created by ggplot2::aes(). These mappings override global aesthetics and are not inherited from the top-level plot.
data	A GraphSpace object.
stat	The statistical transformation to use on the data. Defaults to identity.
position	Position adjustment, either as a string or the result of a call to a position adjustment function.
...	Additional parameters passed to the underlying drawing function in GeomNodeSpace.
na.rm	Logical. Should missing values be removed? Defaults to FALSE.
show.legend	Logical or a named logical vector indicating whether this layer should be included in legends.
inherit.aes	Logical. If FALSE (default), the layer will use aesthetics defined in mapping.

Value

A ggplot2 layer that renders node glyphs defined by GeomNodeSpace.

Aesthetics

geom_nodespace() understands geom_point aesthetics.

If these aesthetics are not explicitly provided in aes(), they are automatically retrieved from the GraphSpace object.

x, y	Required (automatically supplied).
fill	Node interior colour (see aes_colour_fill_alpha).
colour	Node border colour (see aes_colour_fill_alpha).
alpha	Transparency (see aes_colour_fill_alpha).
shape	Node shape (see points and aes_linetype_size_shape).
size	Node size (see *drawing* section and aes_linetype_size_shape).
stroke	Node line width (see gg_par and aes_linetype_size_shape).

Required aesthetics x and y are supplied from the GraphSpace object and do not need to be manually mapped.

Additional parameters can be passed to control fixed values for the layer. For example: `fill = "red"`, `stroke = 3`, `alpha = 0.5`, or `shape = 21`.

Drawing

The interpretation of `size` depends on how it is provided:

- **As an aesthetic**: When mapped within `aes()`, `size` follows the behavior of geom_point, using absolute units to ensure consistency with the plot legends.
- **As a parameter**: When set outside `aes()`, `size` is treated as a percentage of the viewport (`[0, 100]`), scaling in npc units. This allows nodes to resize dynamically with viewport changes.

See Also

GraphSpace, geom_edgespace, geom_graphspace, geom_point

Examples

```
# Make a demo igraph
library(igraph)
gtoy1 <- make_star(15, mode="out")

# Set some node attributes
V(gtoy1)$nodeSize <- runif(vcount(gtoy1), 1, 20)
V(gtoy1)$nodeColor <- rainbow(vcount(gtoy1))

# Set some variables
V(gtoy1)$user_var1 <- runif(vcount(gtoy1), 1, 3)^3
V(gtoy1)$user_var2 <-  rep(c(1, 2, 3), each = 5)

# Create a GraphSpace object
gs <- GraphSpace(gtoy1, layout = layout_in_circle(gtoy1))

## Not run:

# Example 1: Nodes scaling with the legend
# When 'size' is mapped inside aes(), it follows
# ggplot2 default behavior: sizes are translated
# to absolute units (mm) via 'scale_size()'.

ggplot() +
geom_edgespace(data = gs, arrow_offset = 0.01) +
  geom_nodespace(mapping = aes(size = nodeSize, fill = user_var2),
  data = gs) +
  scale_size(range = c(1, 12)) +
  theme(aspect.ratio = 1)

# Example 2: Nodes scaling with the viewport
# When 'size' is passed as a node attribute,
# as inherited from the igraph object, it is
# interpreted as a percentage of the plot area
# and translated to NPC units.
```

```
ggplot() +
geom_edgespace(data = gs, arrow_offset = 0.01) +
geom_nodespace(mapping = aes(fill = user_var2), data = gs) +
theme(aspect.ratio = 1)


## End(Not run)
```

---

getGraphSpace,GraphSpace-method
Accessors for fetching slots from a GraphSpace object

Description

getGraphSpace retrives information from individual slots available in a GraphSpace object.

Usage

```
## S4 method for signature 'GraphSpace'
getGraphSpace(gs, what = "graph")
```

Arguments

gs	A preprocessed GraphSpace class object
what	A single character value specifying which information should be retrieved from the slots. Options: 'graph','gxy','gxyz','pars','misc','status','summits', 'summit_mask', and 'summit_contour'.

Value

Content from slots in the GraphSpace object.

Examples

```
# Load a demo igraph
data('gtoy1', package = 'RGraphSpace')

# Create a new GraphSpace object
gs <- GraphSpace(gtoy1)

# Get the 'summary' slot in gs
getGraphSpace(gs, what = 'graph')
```

GraphSpace,igraph-method

Create a GraphSpace object

Description

GraphSpace is the main constructor for GraphSpace objects, designed to store graph data and metadata for optimized rendering in RGraphSpace.

Usage

```
## S4 method for signature 'igraph'
GraphSpace(
  g,
  layout = NULL,
  verbose = TRUE,
  mar = deprecated(),
  image = deprecated(),
  ...
)

## S4 method for signature 'data.frame'
GraphSpace(g, verbose = TRUE, ...)
```

Arguments

g	An igraph object or a data.frame used to initialize a GraphSpace object. If an igraph is provided, it should include vertex coordinates in x and y attributes, and vertex labels in the name attribute. If a data.frame is provided, it must contain at least x and y columns representing the node coordinates; additional columns will be treated as vertex attributes. For graphs requiring edge definitions, use the igraph initialization.
layout	An optional numeric matrix with two columns for x and y vertex coordinates. If provided, it overrides coordinates in g.
verbose	A logical value. If TRUE, displays detailed messages.
mar	**[Deprecated]** Deprecated since RGraphSpace 1.1.1; use normalizeGraphSpace instead.
image	**[Deprecated]** Deprecated since RGraphSpace 1.1.1; use normalizeGraphSpace instead.
...	Additional arguments passed to the GraphSpace constructor.

Details

GraphSpace objects are designed to bridge the gap between network analysis (via igraph) and high-quality visualization (via ggplot2). The constructor ensures that all necessary aesthetics for geom_graphspace are pre-processed and validated.

Coordinate System and Normalization: By default, the constructor expects coordinates in the `x` and `y` vertex attributes, along with unique IDs in the `name` vertex attribute. If these are not provided, the constructor will generate sequential IDs and assign a layout using the `layout_nicely` function. These coordinates define the relative positioning of nodes. For optimal rendering, it is recommended to pass the object through `normalizeGraphSpace` after construction. This converts vertex positions to Normalized Parent Coordinates (NPC), ensuring the graph remains centered and scaled relative to the plotting area.

Data Structure: The resulting object stores nodes and edges in separate internal slots, preserving metadata such as `nodeSize` and `edgeLineColor`. If an `igraph` object is provided without specific styling attributes, `GraphSpace` will assign the default values defined in the `geom_graphspace` aesthetics. Users can also specify custom variables in the input graph to be used as aesthetics within the `ggplot2` grammar.

Arrowhead Mapping: The `arrowType` attribute (see *Arrowhead types* section) allows for a mapping between symbolic aliases (such as `"-->"`) and internal integer codes. This is useful for assigning interaction types in directed or undirected graphs (e.g., activation vs. inhibition).

Value

A GraphSpace class object.

Vertex attributes

The following attributes in g are evaluated by the constructor:

`nodeSize`	Numeric `[0, 100]`, representing % of the plotting space.
`nodeShape`	Integer code `[0-25]`; see points.
`nodeColor`	A valid color name or hexadecimal code.
`nodeLineWidth`	Border thickness; see gpar.
`nodeLineColor`	A valid color name or hexadecimal code.
`nodeLabel`	Character string (`NA` will omit labels).
`nodeLabelSize`	Font size in `pts`; see gpar.
`nodeLabelColor`	A valid color name or hexadecimal code.

Edge attributes

The following attributes in g are evaluated by the constructor:

`edgeLineWidth`	Edge thickness; see `gpar`.
`edgeLineColor`	A valid color name or hexadecimal code.
`edgeLineType`	Line style (e.g., "solid", "dashed"); see `gpar`.
`arrowType`	Arrowhead style (see *Arrowhead types* section).

Arrowhead types

Arrowheads are controlled via the `arrowType` attribute using integer or character codes (see examples in the *RGraphSpace* vignette).

In directed graphs, arrows follow the edge list orientation by default, representing forward directions (*e.g.*, `A -> B`). While undirected graphs do not show arrows by default, specific styles can be manually assigned for detailed visualization, including forward, backward, or bidirectional arrowheads.

Directed graphs (A -> B)::

Code	Alias	Description
0	"–"	No arrow
1	"->"	Forward arrow
-1	"-|"	Forward bar

Undirected graphs (A – B)::

Code	Alias	Description
0	"–"	No arrow
1	"->"	Forward arrow
2	"<-"	Backward arrow
3	"<->"	Bidirectional arrow
4	"|->"	Forward arrow / backward bar
-1	"-|"	Forward bar
-2	"|-"	Backward bar
-3	"|-|"	Bidirectional bar
-4	"<-|"	Backward arrow / forward bar

Author(s)

Sysbiolab.

See Also

geom_graphspace, plotGraphSpace

Examples

```
library(igraph)

# Create a star graph
gtoy1 <- make_full_graph(15)

# Custom attributes
V(gtoy1)$nodeSize <- 5
E(gtoy1)$edgeLineColor <- "red"
E(gtoy1)$arrowType <- "-->"

# Create a GraphSpace
gs <- GraphSpace(gtoy1)
```

GraphSpace-class *GraphSpace: An S4 class for igraph objects*

Description

GraphSpace: An S4 class for igraph objects

Value

An S4 class object.

Slots

`nodes` A data frame with xy-vertex coordinates.

`edges` A data frame with edges.

`graph` An igraph object.

`image` A raster background image matrix.

`pars` A list with parameters.

`misc` A list with intermediate objects for downstream methods.

Constructor

see `GraphSpace` constructor.

gs_nodes,GraphSpace-method *Accessors for applying essential igraph methods to modify attributes of GraphSpace objects.*

Description

Access and modify individual slots of a GraphSpace object. Selected 'igraph' methods are applied to the 'graph' slot and propagated to downstream components.

Usage

```
## S4 method for signature 'GraphSpace'
gs_nodes(x)

## S4 method for signature 'GraphSpace'
gs_edges(x)

## S4 method for signature 'GraphSpace'
gs_vcount(x)
```

```
## S4 method for signature 'GraphSpace'
gs_ecount(x)

## S4 method for signature 'GraphSpace'
names(x)

## S4 replacement method for signature 'GraphSpace'
names(x) <- value

## S4 method for signature 'GraphSpace'
gs_vertex_attr(x, name, ...)

## S4 replacement method for signature 'GraphSpace'
gs_vertex_attr(x, name, ...) <- value

## S4 method for signature 'GraphSpace'
gs_edge_attr(x, name, ...)

## S4 replacement method for signature 'GraphSpace'
gs_edge_attr(x, name, ...) <- value
```

Arguments

x	A GraphSpace class object
value	The new value of the attribute.
name	Name of the attribute.
...	Additional arguments passed to igraph methods.

Value

Updated GraphSpace object.

See Also

vertex_attr, edge_attr

Examples

```
# Load a demo igraph
data('gtoy1', package = 'RGraphSpace')

# Create a new GraphSpace object
gs <- GraphSpace(gtoy1)

#--- Usage of GraphSpace attribute accessors:

# Get a data frame with nodes for plotting methods
gs_nodes(gs)
```

```
# Get a data frame with edges for plotting methods
gs_edges(gs)

# Get vertex count
gs_vcount(gs)

# Get edge count
gs_ecount(gs)

# Get vertex names
names(gs)

# Access all vertex attributes
gs_vertex_attr(gs)

# Access a specific vertex attribute
gs_vertex_attr(gs, "nodeLabel")

# Modify a single value within a vertex attribute
gs_vertex_attr(gs, "nodeSize")["n1"] <- 10

# Replace an entire vertex attribute
gs_vertex_attr(gs, "nodeSize") <- 10

# Alternative syntax using `$` accessor
gs_vertex_attr(gs)$nodeSize <- 10

# Access a specific edge attribute
gs_edge_attr(gs, "edgeLineColor")

# Replace an entire edge attribute
gs_edge_attr(gs, "edgeLineWidth") <- 1

 # Alternative syntax using `$` for edge attributes
gs_edge_attr(gs)$edgeLineWidth <- 3
```

`gtoys` *Toy 'igraph' objects*

Description

Small 'igraph' objects used for workflow demonstrations. All graphs include 'x', 'y', and 'name' vertex attributes.

Usage

```
data(gtoy1)
```

Format

igraph

Value

A pre-processed igraph object.

Source

This package.

Examples

```
data(gtoy1)
data(gtoy2)
```

---

inject_nodespace *Dynamic Scale Injection for Edge Clipping*

Description

Utility function for **RGraphSpace** that enables edge layers to scan adjacent nodes and determine their dimensions. This information is used to compute arrow clipping offsets, preventing edge geometry from overlapping node symbols.

Usage

```
inject_nodespace(...)
```

Arguments

...	Additional parameters passed to other methods (currently ignored).

Details

This function operates in two stages within the `ggplot2` workflow:

1. **Capture:** It scans the `plot` layers for a GeomNodeSpace to extract both mapping variables (from `aes()`) and static parameters (specifically `size` and `stroke`).
2. **Injection:** It locates GeomEdgeSpace layers and injects scale rules, captured mappings, and fixed parameters into the geometry parameters.

This "lazy injection" calculates edge clipping based on the actual scales used by the nodes, even if scales are defined after the layers.

Note: `inject_nodespace()` must be called last in the `ggplot` chain to allow the function to correctly scan all previously added layers and scales.

Value

An object of class `inject_nodespace`, which interacts with the ggplot2 + operator.

See Also

geom_edgespace, geom_nodespace

Examples

```
library(igraph)
library(ggplot2)

# Generate a toy star graph
gtoy1 <- make_star(15, mode="out")

# Set node and edge attributes
V(gtoy1)$my_node_var <- runif(vcount(gtoy1), 1, 20)
E(gtoy1)$my_edge_var <-  runif(ecount(gtoy1), 1, 20)

# Create a GraphSpace object with a circular layout
gs <- GraphSpace(gtoy1, layout = layout_in_circle(gtoy1))

## Not run:
# Build the plot
# Note that inject_nodespace() is called at the end to
# synchronize node sizes with edge clipping.
ggplot() +
  geom_edgespace(aes(colour = my_edge_var), data = gs) +
  geom_nodespace(aes(size = my_node_var), data = gs) +
  scale_size(range = c(2, 15)) +
  inject_nodespace()

## End(Not run)
```

---

normalizeGraphSpace,GraphSpace,missing-method

Normalize node coordinates to graph and image spaces

Description

Accessory functions to normalize node coordinates in GraphSpace, either by centering them within the graph boundaries or by mapping them to pixel coordinates of a background image.

Usage

```
normalizeGraphSpace(gs, image = NULL, ...)

## S4 method for signature 'GraphSpace,ANY'
```

```
normalizeGraphSpace(
  gs,
  image,
  ...,
  mar = 0.1,
  flip.x = FALSE,
  flip.y = FALSE,
  rotate.xy = FALSE,
  flip.v = FALSE,
  flip.h = FALSE,
  verbose = TRUE
)

## S4 method for signature 'GraphSpace'
cropGraphSpace(gs, crop.coord = c(0, 1, 0, 1), verbose = TRUE)
```

Arguments

gs	A GraphSpace object to be normalized.
image	An optional background image. When provided, x and y coordinates must represent pixel positions in the image matrix.
...	Additional arguments passed to specific normalization workflows.
mar	A single numeric value in [0, 0.5] controlling the size of the outer margins around the graph. Without an image, mar specifies symmetric margins as a fraction of the graph space. With an image, mar is interpreted as a fraction of the available image margins surrounding the graph.
flip.x	Logical; whether to flip the node coordinates along the x-axis.
flip.y	Logical; whether to flip the node coordinates along the y-axis. Useful for aligning nodes with image backgrounds, which often use an inverted coordinate system.
rotate.xy	Logical; whether to rotate x-y coordinates.
verbose	A single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).
flip.v	Logical; whether to vertically flip the background image matrix (top-to-bottom) to align with the graph coordinate system.
flip.h	Logical; whether to horizontally flip the background image matrix (left-to-right) to align with the graph coordinate system.
crop.coord	An optional numeric vector of length four specifying a cropping region (xmin, xmax, ymin, ymax), with values in normalized coordinates [0,1].

Details

These functions provide different strategies for coordinate transformation:

- **normalizeGraphSpace**: Re-scales node coordinates to a [0, 1] unit square based on the graph's bounding box (when image is missing) or maps them to pixel coordinates (when image

is provided). It handles image-to-graph alignment via `flip.` and `rotate.` arguments, used to adjust the graph origin with the image matrix layout. Users should be aware of the potential discrepancy between image matrix orientation (top-down) and graph coordinates (bottom-up). The function attempts to automatically adjust the y-axis to align the graph's bottom-up coordinates with the image's top-down layout, but further manual adjustments might be required.

- **cropGraphSpace**: Subsets the normalized graph space into a specific region defined by `crop.coord`. It recalculates node positions and background image boundaries to maintain spatial consistency after cropping. This function requires a previously normalized `GraphSpace` object.

Value

A `GraphSpace` object with updated `nodes` and `image` slots.

Note

This is an accessory function typically called during the preprocessing of `GraphSpace` objects before rendering.

Examples

```
library(igraph)

# Create a star graph
gtoy1 <- make_full_graph(15)

# Create a GraphSpace
gs <- GraphSpace(gtoy1)

gs <- normalizeGraphSpace(gs)
```

---

plot.GraphSpace *Plot GraphSpace objects*

Description

Plot GraphSpace objects

Usage

```
## S3 method for class 'GraphSpace'
plot(x, ...)
```

Arguments

`x`	A GraphSpace class object.
`...`	Additional arguments passed to the `plotGraphSpace` function.

See Also

plotGraphSpace

---

plotGraphSpace,GraphSpace-method

Wrapper function to plot GraphSpace objects in ggplot2

Description

plotGraphSpace() is a High-level plotting interface that translates igraph and GraphSpace data objects into ggplot2 layers.

Usage

```
## S4 method for signature 'GraphSpace'
plotGraphSpace(
  gs,
  theme = "th0",
  xlab = "Graph coordinates 1",
  ylab = "Graph coordinates 2",
  font.size = 1,
  bg.color = "grey95",
  add.labels = FALSE,
  node.labels = NULL,
  label.size = 3,
  label.color = "grey20",
  add.image = FALSE,
  raster = FALSE,
  dpi = 300,
  dev = "cairo_png"
)

## S4 method for signature 'igraph'
plotGraphSpace(gs, ...)
```

Arguments

gs	Either an igraph or GraphSpace class object. If gs is an igraph, then it must include x, y, and name vertex attributes (see GraphSpace).
theme	Name of a custom RGraphSpace theme. These themes (from 'th0' to 'th3') consist of preconfigured ggplot settings, which can subsequently refine using ggplot2.
xlab	The title for the 'x' axis of a 2D-image space.
ylab	The title for the 'y' axis of a 2D-image space.
font.size	A single numeric value passed to ggplot themes.

bg.color	A single color for background.
add.labels	A logical value indicating whether to plot vertex labels.
node.labels	A vector of vertex names to be highlighted in the graph space. This argument overrides 'add.labels'.
label.size	A size argument passed to geom_text.
label.color	A color passed to geom_text.
add.image	A logical value indicating whether to add a background image, when one is available (see GraphSpace).
raster	A logical value indicating whether to rasterize the main plot. See rasterise for further specifications.
dpi	Raster resolution, in dots per inch.
dev	Device used in the rasterise call.
...	Additional arguments passed to the plotGraphSpace function.

Value

A ggplot-class object.

Author(s)

Sysbiolab.

See Also

GraphSpace

Examples

```
library(igraph)

# Load a demo igraph
data('gtoy1', package = 'RGraphSpace')

# Generate a ggplot for gtoy1
plotGraphSpace(gtoy1)

# Create a star graph
gtoy_star <- make_full_graph(15)

# Example of setting node and edge attributes
V(gtoy_star)$nodeSize <- 5
E(gtoy_star)$edgeLineColor <- "red"
E(gtoy_star)$arrowType <- "<->"

# Create a GraphSpace object
gs_star <- GraphSpace(gtoy_star)

# Normalize graph coordinates
```

```
gs_star <- normalizeGraphSpace(gs_star)

# Generate a ggplot for gs_star
plotGraphSpace(gs_star)
```

---

theme_gspace *RGraphSpace ggplot2 themes*

Description

A set of **ggplot2** themes used by **RGraphSpace** plots.

Usage

```
theme_gspace_th0(
  txt_size = 1,
  leg_size = 1,
  bg_color = "grey95",
  key_fill = FALSE,
  key_colour = FALSE,
  ...
)

theme_gspace_th1(
  txt_size = 1,
  leg_size = 1,
  bg_color = "grey95",
  key_fill = FALSE,
  key_colour = FALSE,
  ...
)

theme_gspace_th2(
  txt_size = 1,
  leg_size = 1,
  bg_color = "grey95",
  key_fill = FALSE,
  key_colour = FALSE,
  ...
)

theme_gspace_th3(
  txt_size = 1,
  leg_size = 1,
  bg_color = "grey95",
  key_fill = FALSE,
```

```
  key_colour = FALSE,
  ...
)

theme_gspace_coords(
  theme = "th0",
  is_norm = FALSE,
  xlab = "Graph coordinates 1",
  ylab = "Graph coordinates 2",
  expand = NULL,
  ...
)

theme_gspace_legend(leg_size = 1, key_fill = FALSE, key_colour = FALSE, ...)
```

Arguments

txt_size	Numeric value to scale plot- and axis-related text elements.
leg_size	Numeric value to scale legend-related elements.
bg_color	A color name or hex code specifying the panel background.
key_fill	Logical; if TRUE, treats the fill legend as discrete to adjust key size.
key_colour	Logical; if TRUE, treats the colour legend as discrete to adjust key size.
...	Additional arguments passed to theme_gspace_th* and ggtheme.
theme	Character string specifying the GraphSpace theme variant. Options: th0, th1, th2, and th3.
is_norm	Logical; if TRUE, assumes plot coordinates are already normalized in [0, 1].
xlab	The title for the 'x' axis.
ylab	The title for the 'y' axis.
expand	A range expansion factor applied to both the lower and upper limits of the 'x' and 'y' scales.

Details

theme_gspace_th0() is a minimal wrapper around theme_gray that simplifies axis and legend scaling. The txt_size and leg_size arguments aggregate related theme parameters for quick thematic overrides.

theme_gspace_th1() builds on theme_gspace_th0() and modifies grid lines, axis appearance, and panel borders.

theme_gspace_th2() is similar to theme_gspace_th1() with simplified grid elements and a customizable panel background.

theme_gspace_th3() is similar to theme_gspace_th2() but with slightly adjusted margins, tick appearance, and legend formatting.

The theme_gspace_coords() is a helper function that also adds axes scales for normalized coordinates. It configures axis breaks, limits, and expansion for graph layouts. Plot coordinates are ideally normalized to the interval [0, 1].

`theme_gspace_legend()` is helper function that adjusts legend text, title, and key sizes by a single scaling factor.

Value

`theme_gspace_th*()` return a `ggplot2` theme object.

`theme_gspace_coords()` returns a list containing scale and theme components that can be added to a **ggplot2** plot.

`theme_gspace_legend()` returns a list of theme and guide components.

See Also

ggtheme, theme

Examples

```
library(ggplot2)

ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  theme_gspace_th0()

ggplot(mtcars,
  aes(scales::rescale(wt),
      scales::rescale(mpg))) +
  geom_point() +
  theme_gspace_coords("th2", is_norm = TRUE)

# Small scale legends
ggplot(mtcars, aes(wt, mpg, fill = factor(cyl))) +
  geom_point(shape = 21) +
  theme_gspace_legend(0.8)
```

Index