Package 'RGraphSpace'

August 24, 2026

Type Package

Version 1.5.2

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

Description An interface for rendering 'igraph' objects as 'ggplot2' graphics within a normalized coordinate space. 'RGraphSpace' implements new geometries that treat a graph as a single coherent object, synchronizing node and edge layers under standard aesthetic mappings. Node features are resolved on demand, supporting high-dimensional data without expanding node tables. Spatial alignment is available at the pixel level, with node coordinates anchored to pixel centers through a half-pixel offset, enabling precise node positioning over external reference frames such as images and maps.

Depends R(>= 4.5), methods, ggplot2 (>= 4.0)

Imports grDevices, grid, igraph, tidygraph, scales, rlang, ggrastr, Matrix, sf, stats, lifecycle

Suggests knitr, rmarkdown, testthat, ggraph, ggnewscale, patchwork, Seurat, SeuratObject, SummarizedExperiment, SpatialExperiment

Enhances RedeR

License Artistic-2.0

VignetteBuilder knitr

URL https://github.com/sysbiolab/RGraphSpace, https://sysbiolab.github.io/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-accessors.R' 'gspace-misc.R' 'gspace-subset.R' 'gspace-addition.R' 'gspace-subscript.R' 'gspace-normalize.R' 'gspace-transform.R' 'gspace-geometry.R' 'gspace-themes.R' 'gspace-validation.R' 'gspace-ggplot-constructor.R' 'annotation-gspace.R' 'gspace-coercion.R' 'gspace-features.R'

Encoding UTF-8

RdMacros lifecycle

Config/roxygen2/version 8.0.0

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 2026-08-24 05:20:40 UTC

Contents

`annotation_gspace_image`

Annotate a GraphSpace Plot with an Image

Description

`annotation_gspace_image()` adds an image annotation layer to a `ggplot`-based `GraphSpace` plot.

Usage

```
annotation_gspace_image(
  x,
  interpolate = FALSE,
  opacity = 1,
  flip.v = FALSE,
  flip.h = FALSE,
  na.color = NA
)

annotation_gspace(...)
```

Arguments

`x`	An image to be displayed. Accepted types: • A `GraphSpace` object — the image is extracted via `gs_image`. • A `raster` object (see `as.raster`). • A `matrix` or 3D array (RGB/RGBA), coerced to `raster` automatically.
`interpolate`	A logical value indicating whether to apply linear interpolation when the image is rendered at a different resolution than its native size. Defaults to `FALSE`.
`opacity`	A numeric value in `[0, 1]` controlling the transparency of the image. `1` is fully opaque (default); `0` is fully transparent.
`flip.v`	A logical value; if `TRUE`, the image is flipped vertically (top-to-bottom). Defaults to `FALSE`.
`flip.h`	A logical value; if `TRUE`, the image is flipped horizontally (left-to-right). Defaults to `FALSE`.
`na.color`	The colour to map to NA values. Defaults to `NA`.
`...`	Additional arguments (currently unused).

Value

A ggplot2 layer object that can be added to a `ggplot()` call with `+`, or `invisible(NULL)` with a warning if the image could not be resolved.

Note

`annotation_gspace()` is deprecated as of v1.4.0; use `annotation_gspace_image()` instead.

See Also

`annotation_raster`, `gs_image`, `geom_nodespace`, `geom_edgespace`

Examples

```
library(RGraphSpace)
library(igraph)

# Load a demo igraph
data('gtoy1', package = 'RGraphSpace')
gs <- GraphSpace(gtoy1)

# Normalize node coordinates
gs <- normalizeGraphSpace(gs)

# Add a raster image
gs_image(gs) <- as_colorraster(volcano)

## Not run:
# Pass a GraphSpace object directly
ggplot(gs) +
  annotation_gspace_image(gs) +
  geom_edgespace() +
  geom_nodespace()

# Extract the image explicitly
ggplot(gs) +
  annotation_gspace_image(gs_image(gs)) +
  geom_edgespace() +
  geom_nodespace()

# Dim the background and flip vertically
ggplot(gs) +
  annotation_gspace_image(gs, opacity = 0.5, flip.v = TRUE) +
  geom_edgespace() +
  geom_nodespace()


## End(Not run)
```

`as.GraphSpace` *Convert objects to GraphSpace*

Description

S3 generic function for coercing objects into a `GraphSpace` object.

Usage

```
as.GraphSpace(x, ...)

## Default S3 method:
as.GraphSpace(x, ...)

## S3 method for class 'igraph'
as.GraphSpace(x, ...)

## S3 method for class 'tbl_graph'
as.GraphSpace(x, ...)

## S3 method for class 'data.frame'
as.GraphSpace(x, ...)

## S3 method for class 'DFrame'
as.GraphSpace(x, ...)

## S3 method for class 'matrix'
as.GraphSpace(x, ...)

## S3 method for class 'SpatialExperiment'
as.GraphSpace(x, assay = "counts", ...)

## S3 method for class 'Seurat'
as.GraphSpace(x, layer = NULL, space = c("embedding", "spatial"), ...)
```

Arguments

`x`	An object to be converted.
`...`	Additional arguments passed to coercion methods.
`assay`	Name of the assay in the `SpatialExperiment` object from which data should be retrieved (see `assay`).
`layer`	Name of the layer in the `Seurat` object from which node data should be retrieved (see `LayerData`).
`space`	Character specifying the coordinate space used for node geometry. Either `"embedding"` or `"spatial"`. See details.

Details

Unified entry point for converting graph, spatial, and high-dimensional data into a GraphSpace object.

Graph objects are imported either through native methods or via as_tbl_graph when available.

For **Seurat** objects, coordinate extraction depends on the selected space:

- space = "embedding" uses the first two dimensions returned by Embeddings.
- space = "spatial" uses tissue coordinates returned by GetTissueCoordinates.

Assay data are stored in the data slot of the resulting GraphSpace object. Node metadata from x@meta.data are appended to the node table.

Value

A GraphSpace object.

See Also

GraphSpace

---

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 white.

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

```
library(RGraphSpace)

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

`cropGraphSpace,GraphSpace-method`

Crop, rotate, flip, and transpose a GraphSpace

Description

Accessory functions to spatially transform a normalized GraphSpace object. `cropGraphSpace()` subsets the plotting area to a rectangular region; `rotateGraphSpace()` rotates by a quarter turn; `flipGraphSpace()` mirrors horizontally or vertically; `transposeGraphSpace()` swaps the x and y axes.

Usage

```
## S4 method for signature 'GraphSpace'
cropGraphSpace(gs, xmin = 0, xmax = 1, ymin = 0, ymax = 1, verbose = TRUE)

## S4 method for signature 'GraphSpace'
rotateGraphSpace(gs, clockwise = FALSE, verbose = TRUE)

## S4 method for signature 'GraphSpace'
flipGraphSpace(gs, vertical = FALSE, verbose = TRUE)

## S4 method for signature 'GraphSpace'
transposeGraphSpace(gs, verbose = TRUE)
```

Arguments

`gs`	A normalized `GraphSpace` object.
`xmin`	A single number in `[0,1]` specifying the lower x-boundary of the plotting area.
`xmax`	A single number in `[0,1]` specifying the upper x-boundary of the plotting area.
`ymin`	A single number in `[0,1]` specifying the lower y-boundary of the plotting area.
`ymax`	A single number in `[0,1]` specifying the upper y-boundary of the plotting area.
`verbose`	A single logical value specifying to display detailed messages (when `verbose=TRUE`) or not (when `verbose=FALSE`).
`clockwise`	A single logical value. If `FALSE` (default), the 90-degree turn is counter-clockwise; if `TRUE`, clockwise (`rotateGraphSpace` only).
`vertical`	A single logical value. If `FALSE` (default), the flip is horizontal (mirror left-right); if `TRUE`, vertical (mirror top-bottom). (`flipGraphSpace` only).

Details

`cropGraphSpace()` subsets a normalized graph space to a specific region defined by the cropping boundaries. It recalculates node positions and background image boundaries to maintain spatial consistency after cropping, and drops nodes (and edges) that fall outside the window.

`rotateGraphSpace()`, `flipGraphSpace()`, and `transposeGraphSpace()` are all exact, a coordinate/pixel permutation, with no resampling, no interpolation, and no risk of misaligning nodes against the background image. `rotateGraphSpace()` is restricted to a single 90-degree turn: apply it again to its own output for 180 or 270 degrees, e.g. `rotateGraphSpace(rotateGraphSpace(gs))` for 180 degrees. Combine all three with each other to reach any of the 8 symmetries of a square.

Value

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

Note

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

See Also

`normalizeGraphSpace`

Examples

```
library(RGraphSpace)
library(igraph)

# Create a star graph
gtoy1 <- make_full_graph(30)

# Create a GraphSpace
gs <- GraphSpace(gtoy1)

gs <- normalizeGraphSpace(gs)

gs_crop <- cropGraphSpace(gs, ymax = 0.5)
gs_rot90 <- rotateGraphSpace(gs)
gs_flip <- flipGraphSpace(gs)
gs_t <- transposeGraphSpace(gs)

plotGraphSpace(gs, add.labels = TRUE)

plotGraphSpace(gs_crop, add.labels = TRUE)
```

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
```

Aesthetics

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

See Also

geom_edgespace, geom_segment

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
```

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 bridges GraphSpace edge attributes with ggplot2 rendering via two distinct aesthetic interfaces that coexist without collision (see *Two aesthetic interfaces* section).

Usage

```
geom_edgespace(
  mapping = NULL,
  data = NULL,
  stat = StatEdgeSpace,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = FALSE,
  arrow_size = 0.5,
  arrow_offset = 0.01,
  curve = 0,
  coord_warp = 1,
  parallel_spread = 1,
  loop_direction = "adaptive",
  lineend = "butt",
  linejoin = "mitre",
  raster = FALSE,
  dpi = NULL,
  dev = "cairo",
  scale = 1
)

edgespace_handler()
```

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	The data to be displayed in this layer. It can be a GraphSpace object, an igraph object, or the edgespace_handler() closure. When NULL (default), a handler is created internally.
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.
arrow_size	Numeric scaling factor controlling arrowhead geometry (see 'details').
arrow_offset	Numeric value controlling the base offset of arrows at edge endpoints (see 'details').
curve	Numeric. Controls edge curvature, as a fraction of edge length. Non-zero values bow the edge into a smooth curve, and the sign controls which side it bows toward. Ignored for loops and parallel edges (see 'details').
coord_warp	Numeric (>=0). Bend applied to edges under non-linear coordinate systems, so that curvature follows the coordinate system's warping. Defaults to 1; has no effect under linear coordinates (see 'details').
parallel_spread	Numeric (>=0). Controls the lateral spread of parallel edges and self-loops. Ignored for simple non-loop edges (see 'details').
loop_direction	Controls how self-loops are oriented around their node. Options: 'adaptive' (default), 'opposite', and an angle in degrees (see 'details').
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.
raster	Logical. Should node glyphs be rasterized? Rasterization support is based on rasterise.
dpi	Numeric. Rasterization resolution.
dev	Character. Rasterization backend. One of 'cairo', 'ragg', 'ragg_png', or 'cairo_png'.
scale	Numeric. Rasterization scaling factor (see rasterise).

Details

arrow_size is a numeric scaling factor controlling arrowhead geometry. The value is interpreted in the same numeric space as line width (lwd).

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.

curve bows an edge through a control point displaced perpendicular to the edge, by curve times the edge length. curve = 0 (default) renders a straight edge. Typical visible values range from about 0.1 to 0.4; sign sets which side the edge bows toward.

coord_warp bends edges under non-linear coordinate systems, for example coord_sf with a `default_crs`, coord_polar, or coord_trans, so that edge curvature follows the coordinate system's warping rather than cutting across it. `coord_warp = 1` (default) applies the exact deviation between the warped edge midpoint and the midpoint of the warped endpoints; `coord_warp = 0` disables it. Values above 1 exaggerate the bend, but may give erratic results under strongly warped coordinate systems. The bend indicates the coordinate system's influence on the graph's extent; it does not depict a path through space.

parallel_spread controls the fan opening for parallel edges, reciprocal `A->B/B->A` pairs, and self-loops – anything where multiple edges share the same vertex pair. `curve` has no effect on these edges; `parallel_spread` governs both their curvature magnitude and how far apart they fan. A value of `0` collapses all edges in a group onto the same position; increasing values progressively open the fan. Self-loops behave the same way: a single loop uses `parallel_spread` to set its own size, and multiple loops at the same node fan out accordingly. A built-in minimum, tied to `arrow_size` and node size, keeps small `parallel_spread` values from producing a loop whose arrowhead looks skewed against its own curvature.

loop_direction determines where self-loops sit relative to their node. `"adaptive"` (default) points each loop in the direction that faces away from the graph's centroid. `"opposite"` is a two-sided arrangement: loops are split into two groups placed above and below the node. A numeric angle (in degrees) places all loops at a fixed direction regardless of their node's position in the layout. When node position data is unavailable, `"adaptive"` silently falls back to `"opposite"`.

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`	Edge 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).

All required aesthetics are supplied from the GraphSpace object and do not need to be manually mapped.

Fixed identity values can also be passed directly as parameters, bypassing both graph attributes and scale training. For example: `colour = "grey"`, `linetype = 2`, `linewidth = 1`.

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

Two aesthetic interfaces

`geom_edgespace()` supports two interfaces that coexist without collision: graph attributes (camelCase names such as `edgeColor`, `edgeLineWidth`) and ggplot2 mappings (via `aes()`). See comments in the vignette.

When multiple sources provide the same aesthetic, priority follows: `aes()` mapping > fixed parameter > graph attribute.

Label aesthetics

When `label` is mapped via `aes()`, a text label is drawn at the visual midpoint of each edge. Labels follow the rendered edge geometry: the chord midpoint for straight edges, the Bezier midpoint for curved edges, and the loop apex for self-loops. Edges with `NA` labels are silently skipped.

The `label_colour` aesthetic defaults to the edge `colour`, and `label_alpha` defaults to the edge `alpha`. All other `label_*` aesthetics default to `geom_label` when not set.

`label`	Required to activate label rendering.
`label_colour`	Label text colour (see `geom_label`).
`label_alpha`	Transparency (see `geom_label`).
`label_fill`	Background colour (see `geom_label`).
`label_size`	Font size (see `geom_label`).
`label_angle`	Rotation angle (see `geom_label`).
`label_hjust`	Horizontal justification (see `geom_label`).
`label_vjust`	Vertical justification (see `geom_label`).
`label_lwd`	Border linewidth (see `geom_label`).
`label_lty`	Border linetype (see `geom_label`).
`label_family`	Font family (see `geom_label`).
`label_fontface`	Font face (see `geom_label`).
`label_lineheight`	Line height (see `geom_label`).

See Also

GraphSpace, geom_nodespace, geom_graphspace, geom_segment, geom_label

Examples

```
library(RGraphSpace)
library(igraph)
library(ggplot2)

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

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

## Not run:

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


## End(Not run)
```

geom_graphspace *Convenience wrapper for node and edge geoms*

Description

geom_graphspace() adds both node and edge layers to a **ggplot2** plot by calling geom_nodespace and geom_edgespace in sequence. It is a convenience wrapper with no logic of its own; any argument accepted by either underlying geom can be passed via node.params or edge.params.

For independent control of node and edge layers, use geom_nodespace and geom_edgespace directly.

Usage

```
geom_graphspace(mapping = NULL, node.params = list(), edge.params = list())
```

Arguments

mapping	An optional aes call passed to geom_nodespace. The most common use is supplying node label aesthetics, e.g. aes(label = nodeLabel).
node.params	A named list of additional arguments forwarded to geom_nodespace.
edge.params	A named list of additional arguments forwarded to geom_edgespace.

Value

A list of two **ggplot2** layers, which **ggplot2** flattens automatically when added to a plot with +.

See Also

geom_nodespace, geom_edgespace, plotGraphSpace

Examples

```
library(ggplot2)
data("gtoy1", package = "RGraphSpace")
gs <- GraphSpace(gtoy1)

# Simplest use
ggplot(gs) + geom_graphspace()

# With node labels
ggplot(gs) + geom_graphspace(aes(label = nodeLabel))

# With independent node and edge customization
ggplot(gs) + geom_graphspace(
  node.params = list(aes(label = nodeLabel)),
  edge.params = list(curve = 0.3)
)
```

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

Description

Constructor for GeomNodeSpace ggproto objects.

A wrapper around geom_point that bridges GraphSpace node attributes with ggplot2 rendering via two distinct aesthetic interfaces that coexist without collision (see *Two aesthetic interfaces* section).

Usage

```
geom_nodespace(
  mapping = NULL,
  data = NULL,
  stat = StatNodeSpace,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = FALSE,
  raster = FALSE,
  dpi = NULL,
  dev = "cairo",
  scale = 1
)

nodespace_handler(mapping = NULL)
```

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	The data to be displayed in this layer. It can be a GraphSpace object, an igraph object, or the nodespace_handler() closure. When NULL (default), a handler is created internally from the mapping argument.
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.
raster	Logical. Should node glyphs be rasterized? Rasterization support is based on rasterise.

dpi	Numeric. Rasterization resolution.
dev	Character. Rasterization backend. One of 'cairo', 'ragg', 'ragg_png', or 'cairo_png'.
scale	Numeric. Rasterization scaling factor (see rasterise).

Details

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.

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	Node coordinates (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 are supplied from the GraphSpace object and do not need to be manually mapped.

Fixed identity values can also be passed directly as parameters, bypassing both graph attributes and scale training. For example: fill = "red", stroke = 3, alpha = 0.5, or shape = 21.

Two aesthetic interfaces

geom_nodespace() supports two interfaces that coexist without collision: graph attributes (camelCase names such as nodeColor, nodeSize) and ggplot2 mappings (via aes()). See comments in the vignette.

When multiple sources provide the same aesthetic, priority follows: aes() mapping > fixed parameter > graph attribute.

Label aesthetics

When `label` is mapped via `aes()`, a text label is drawn at each node's position using `geom_text`, rendered on top of the node glyph. Nodes with `NA` labels are silently skipped.

The `label_size` and `label_colour` aesthetics are automatically retrieved from the GraphSpace object when not explicitly provided in `aes()`. All other `label_*` aesthetics default to `geom_text` when not set.

`label`	Required to activate label rendering.
`label_size`	Font size (see `geom_text`).
`label_colour`	Label colour (see `geom_text`).
`label_alpha`	Label transparency (see `geom_text`).
`label_angle`	Rotation angle (see `geom_text`).
`label_hjust`	Horizontal justification (see `geom_text`).
`label_vjust`	Vertical justification (see `geom_text`).
`label_family`	Font family (see `geom_text`).
`label_fontface`	Font face (see `geom_text`).
`label_lineheight`	Line height (see `geom_text`).

See Also

GraphSpace, geom_edgespace, geom_graphspace, geom_point, geom_text

Examples

```
library(RGraphSpace)
library(igraph)
library(ggplot2)

# Make a demo 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: size is translated
# to absolute units (mm) via 'scale_size()'.

ggplot(gs) +
```

```
geom_edgespace(arrow_offset = 0.01) +
geom_nodespace(mapping = aes(size = nodeSize, fill = user_var2)) +
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,
# inherited from the igraph object, it is
# interpreted as a percentage of the plotting
# area and translated to NPC units.

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

# Example 3: Node labels
ggplot(gs) +
  geom_edgespace() +
  geom_nodespace(aes(label = nodeLabel)) +
  theme(aspect.ratio = 1)


## End(Not run)
```

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

Description

`getGraphSpace` retrieves 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 slot to retrieve from a 'GraphSpace' object. Options: "graph", "nodes", "edges", "pars", "misc", "image", "canvas", "fdata", "coords", and "uuid".

Value

Content from slots in the GraphSpace object.

Examples

```
library(RGraphSpace)
library(igraph)

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

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

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

---

ggplot-GraphSpace *Using ggplot2 with GraphSpace objects*

Description

GraphSpace objects can be used directly with **ggplot2**, allowing node attributes and high-dimensional feature data to be mapped through standard aesthetic mappings without manual data extraction. This integration enables:

- Lazy evaluation of node attributes and feature data.
- Automatic synchronization of node metadata for standard ggplot2 geoms such as geom_point.
- Automatic propagation of node metadata required for edge clipping and arrow placement in GraphSpace-native geoms.

Usage

```
## S3 method for class 'GraphSpace'
ggplot(data, mapping = NULL, ...)
```

Arguments

`data`	A GraphSpace object.
`mapping`	Set of aesthetic mappings created by aes. Passed to ggplot.
`...`	Additional arguments passed to ggplot.

Details

When a GraphSpace object is supplied to `ggplot()`, RGraphSpace extends the standard ggplot2 build process to automatically resolve GraphSpace variables and synchronize node metadata required for edge rendering.

When using `ggplot()`, neither nodespace_handler nor inject_nodespace need to be called explicitly.

Value

A gspace_plot object extending ggplot.

See Also

GraphSpace, geom_nodespace, geom_edgespace, inject_nodespace, nodespace_handler, edgespace_handler

Examples

```
library(RGraphSpace)
library(igraph)
library(ggplot2)

# Generate a toy star graph
gtoy1 <- make_star(15, mode = "out")
V(gtoy1)$my_node_var <- runif(vcount(gtoy1), 1, 20)

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

## Not run:

# Example 1: Using RGraphSpace-native geoms
# Edge clipping metadata are injected automatically
ggplot(gs) +
  geom_edgespace(colour = "red") +
  geom_nodespace(aes(size = my_node_var),
  fill = "steelblue", stroke = 2) +
  scale_size(range = c(2, 15))

# Example 2: Mixing native and general geoms
# Note possible clipping mismatch when combining
# geom_edgespace() with generic ggplot2 node geoms.
# Since geom_point() does not expose the final rendered
# node radius to RGraphSpace, edge clipping is estimated
# from layer parameters and may not exactly match the
# displayed node geometry.
ggplot(gs) +
  geom_edgespace(colour = "red") +
  geom_point(aes(x, y, size = my_node_var),
  fill = "steelblue", stroke = 2, shape = 21) +
  scale_size(range = c(2, 15))


## End(Not run)
```

GraphSpace,ANY-method *Create a GraphSpace object*

Description

GraphSpace is the main constructor for GraphSpace objects.

Usage

```
## S4 method for signature 'ANY'
GraphSpace(g, layout = NULL, simplify = TRUE, verbose = TRUE, ...)

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

Arguments

g	A graph object inheriting from the igraph class (such as igraph and tbl_graph) or a data.frame used to initialize a GraphSpace object. If a graph 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.
simplify	A logical value. If TRUE (default), removes loops and multiple edges (see simplify).
verbose	A logical value. If TRUE (default), displays detailed messages.
...	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 edgeColor. 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.
edgeColor	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(RGraphSpace)
library(igraph)

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

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

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

GraphSpace-accessors *Accessors and attribute utilities for GraphSpace objects*

Description

Access and modify individual components of a GraphSpace object. Selected **igraph** methods are applied to the internal graph representation and propagated to downstream node and edge components.

Usage

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

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

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

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

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

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

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

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

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

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

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

## S3 method for class 'GraphSpace'
as.igraph(x, ...)

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

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

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

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

```
## S4 method for signature 'GraphSpace'
gs_delete_v_attr(x, name)

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

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

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

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

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

## S4 method for signature 'GraphSpace'
gs_geometry(x, name = "geometry")

## S4 replacement method for signature 'GraphSpace'
gs_geometry(x, name = "geometry") <- value

## S4 method for signature 'GraphSpace'
x$name

## S4 replacement method for signature 'GraphSpace'
x$name <- value
```

Arguments

x	A GraphSpace class object
...	Additional arguments passed to extraction methods.
value	Replacement value for the selected slot or attribute.
name	Name of the attribute.

Details

For `gs_nodes()`, the optional `vars` argument specifies node-associated features retrieved from the `fdata` container. See also `gs_fetch_features`.

Value

Updated GraphSpace object.

See Also

vertex_attr, edge_attr, gs_fetch_features

Examples

```
library(RGraphSpace)
library(igraph)

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

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

#--- Usage of GraphSpace attribute accessors:

# Vertex names
names(gs)

# Vertex attribute names
gs_names(gs)

# Get a data frame with nodes
gs_nodes(gs)

# Get a data frame with edges
gs_edges(gs)

# Get vertex count
gs_vcount(gs)

# Get edge count
gs_ecount(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

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

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

```
# Add an image and rescale graph coordinates to image space
# Images may be provided as a raster or numeric matrix
gs_image(gs) <- as_colorraster(volcano)
gs <- normalizeGraphSpace(gs, image.space = FALSE)

# apply a scaling factor to node coordinates
gs_scale_factor(gs) <- 0.1
# undo scaling
gs_scale_factor(gs) <- 1

# add an 'sfc' geometry column
pts <- replicate(gs_vcount(gs), sf::st_point(runif(2)), simplify = FALSE)
gs_geometry(gs) <- sf::st_sfc(pts)
```

`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 containing node coordinates, attributes, and metadata.

`edges` A data frame containing edge relationships and attributes.

`graph` An `igraph` object representing the graph structure.

`image` A `raster` object (see `as.raster`) holding the original background image as supplied by the user. Never modified after construction; always serves as the stable source for `normalizeGraphSpace()`.

`canvas` A `raster` object holding the processed, render-ready image produced by `normalizeGraphSpace()`. Receives all centering, flipping, and margin adjustments. When this slot contains only the empty sentinel, downstream accessors fall back to `@image` automatically; see gs_image.

`fdata` A `Matrix` object storing high-dimensional feature data associated with graph nodes.

`coords` A data frame with raw coordinates. It also stores a raw `sfc` list column when a `geometry` is included by the gs_geometry function.

`pars` A list with parameters.

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

`uuid` A Universally Unique Identifier (UUID) for the object instance.

Constructor

see `GraphSpace` constructor.

GraphSpace-subscript *Subscript operators for GraphSpace objects*

Description

`[` subsets a `GraphSpace` object along two independent dimensions: the first index (`i`) selects nodes; the second (`j`) selects edges.

`[[` retrieves a single named slot from a `GraphSpace` object.

Usage

```
## S4 method for signature 'GraphSpace,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'GraphSpace'
x[[i, j, ...]]
```

Arguments

`x`	A `GraphSpace` object.
`i`	A node selection. Accepted forms: • A **character** vector of node names. • An **integer** vector of positional indices into `@nodes`. • A **logical** vector whose length matches the number of nodes. If omitted, all nodes are retained.
`j`	An edge selection. Accepted forms: • An **integer** vector of positional indices into `@edges`. • A **logical** vector whose length matches the number of edges. Because `[` evaluates its arguments in the calling environment before dispatch, unquoted column names such as `name1 == "n1"` cannot be used directly. Pre-evaluate the expression against the edge table first (e.g. `gs_edges(gs)$name1 == "n1"`), or use `gs_subset_edges` which supports unquoted predicates via data masking. If omitted, all edges are retained (subject to node-filter cascade).
`...`	Currently unused.
`drop`	Ignored; accepted for S4 method compatibility only.

Details

Mental model: unlike a data frame, where `[i, j]` means rows and columns of the same table, for `GraphSpace` the two indices address the two primary components of the graph: nodes (`i`) and edges (`j`). Neither index subsets columns — they select graph entities.

Synchronization rules:

- `x[i, ]` — node-induced subgraph. After selecting nodes, edges are automatically pruned to those whose both endpoints survived. Normalized coordinates are preserved.
- `x[, j]` — edge selection. The node set is not modified; no node pruning occurs.
- `x[i, j]` — combined selection. Node filtering is applied first. Edge index `j` is resolved against the **original** edge table; an edge survives only if it appears in `j` *and* both its endpoints survived node filtering (silent intersection).

Note for `[[`: the slot accessor is read-only. Use the dedicated replacement methods (`gs_image<-`, `gs_fdata<-`, `gs_vertex_attr<-`, etc.) to modify slot contents.

Value

`[` returns a `GraphSpace` object.

`[[` returns the content of the named slot.

See Also

`gs_subset_nodes`, `gs_subset_edges`, `getGraphSpace`, `cropGraphSpace`

Examples

```
library(RGraphSpace)
library(igraph)

g <- make_star(10, mode = "out")
V(g)$nodeSize <- runif(vcount(g), 1, 10)
E(g)$weight   <- runif(ecount(g), 0, 1)
gs <- GraphSpace(g)
gs <- normalizeGraphSpace(gs)

#--- [ examples ---

# Node-induced subgraph: keep named nodes, prune dangling edges
gs[c("n1", "n2", "n3"), ]

# Node-induced subgraph by integer position
gs[1:4, ]

# Node-induced subgraph by pre-evaluated logical mask
gs[gs$nodeSize > 5, ]

# Edge selection only: keep all nodes
gs[, 1:3]
gs[, gs_edges(gs)$weight > 0.5]

# Edge selection by endpoint: 'name1' and 'name2' must be pre-evaluated
# when using [, because [ evaluates j in the calling environment.
# Use gs_subset_edges() for unquoted predicate expressions instead.
gs[, gs_edges(gs)$name1 == "n1"]
gs[, gs_edges(gs)$name1 == "n1" & gs_edges(gs)$name2 == "n2"]
gs[, quote(name1 == "n1" & name2 == "n2")]
```

```
# Combined: node filter first, then edge intersection
gs[c("n1", "n2", "n3"), gs_edges(gs)$weight > 0.5]
gs[c("n1", "n2", "n3"), gs_edges(gs)$name1 == "n1"]

#--- [[ examples ---

gs[["nodes"]]   # same as getGraphSpace(gs, "nodes")
gs[["edges"]]   # same as getGraphSpace(gs, "edges")
gs[["graph"]]   # same as getGraphSpace(gs, "graph")
gs[["fdata"]]   # same as getGraphSpace(gs, "fdata")
```

`gs_add_edges` *Add edges to a GraphSpace object*

Description

`gs_add_edges()` and `gs_add_edges<-` add one or more edges to a `GraphSpace` object. Both endpoints of every new edge must already exist in the node set. The `@graph`, `@edges`, and all derived edge quantities are updated consistently; the node set and the normalized coordinate state are not affected.

`gs_add_edges(x, value)` is the pipe-friendly functional form and returns the modified object. `gs_add_edges(x) <- value` is the in-place replacement form and modifies `x` by reference in the calling environment. Both forms are equivalent.

Usage

```
## S4 method for signature 'GraphSpace'
gs_add_edges(x, value, ...)

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

Arguments

`x`	A `GraphSpace` object.
`value`	A data frame with at least two columns identifying the edge endpoints. Two column naming conventions are accepted: • `from` / `to` — the tidygraph / igraph convention. • `name1` / `name2` — the `@edges` slot convention, useful when constructing `value` directly from `gs_edges()`. If both conventions are present, `from`/`to` takes priority. Any additional columns are treated as edge attributes and passed through to `@edges`. Standard visual attributes (`edgeColor`, `arrowType`, etc.) are filled from package defaults when omitted; analytical attributes such as `weight` are stored as-is.
`...`	Additional arguments (currently unused; reserved for future use).

Details

Adding edges does not invalidate the normalized layout. Node coordinates in `@nodes` are left untouched and `normalizeGraphSpace` does not need to be re-run.

For objects built with `simplify = TRUE` (the default), loop edges (`from == to`), parallel edges, and duplicate rows within `value` are silently dropped with a warning. Admissible edges in the same call are still added. To allow loops or parallel edges, rebuild the object with `GraphSpace(g, simplify = FALSE)`.

Because adding an edge to a group of parallel edges changes the derived attributes `curve_weight`, `is_multiple`, and `is_loop` for all members of that group, the full edge table is recomputed from `@graph` after each assignment.

Value

A `GraphSpace` object with the new edges appended.

See Also

`gs_add_nodes`, `gs_edge_attr`, `gs_subset_edges`, `gs_edges`

Examples

```
library(RGraphSpace)
library(igraph)

g <- make_star(6, mode = "out")
gs <- GraphSpace(g)
gs <- normalizeGraphSpace(gs)

# Functional form (pipe-friendly): returns a modified copy
gs <- gs_add_edges(gs, data.frame(from = "n2", to = "n3"))

# Assignment form: modifies gs in place
gs_add_edges(gs) <- data.frame(from = "n3", to = "n4")

# Add multiple edges with an analytical attribute
gs <- gs_add_edges(gs, data.frame(
  from   = c("n4", "n5"),
  to     = c("n5", "n6"),
  weight = c(0.8, 0.4)
))
```

`gs_add_nodes` *Add nodes to a GraphSpace object*

Description

gs_add_nodes() and gs_add_nodes<- add one or more nodes to a GraphSpace object. The @graph, @nodes, and @fdata slots are updated consistently. Because new nodes introduce coordinates into the existing layout, the normalized state is invalidated and normalizeGraphSpace must be re-run afterwards.

gs_add_nodes(x, value) is the pipe-friendly functional form and returns the modified object. gs_add_nodes(x) <- value is the in-place replacement form and modifies x by reference in the calling environment. Both forms are equivalent.

Usage

```
## S4 method for signature 'GraphSpace'
gs_add_nodes(x, value, ...)

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

Arguments

x	A GraphSpace object.
value	A data frame with at minimum three columns: • name — unique node identifier (character). • x, y — node coordinates in raw graph space. Any additional columns are treated as node attributes. Standard visual attributes (nodeSize, nodeColor, nodeShape, etc.) are filled from package defaults when omitted. The column vertex is reserved and stripped automatically if present.
...	Additional arguments (currently unused; reserved for future use).

Details

Adding nodes always invalidates the normalized layout. The @pars normalization flags are cleared, @canvas is reset, and normalizeGraphSpace must be re-run to restore a renderable state. The @edges slot is not affected. Existing node coordinates in @nodes revert to raw graph-space values if the object was previously normalized, since normalization is cleared before @nodes is rebuilt.

Standard node attributes (nodeSize, nodeColor, etc.) are kept consistent across old and new nodes: attributes present on existing nodes but absent from value are filled from package defaults for the new rows, and vice versa.

nodeLabel defaults to the node name when not supplied, consistent with the behaviour of the GraphSpace constructor.

If @fdata is non-empty, new nodes are appended as NA rows so the feature matrix remains aligned with @nodes.

Value

A GraphSpace object with the new nodes appended and the normalized state cleared.

See Also

gs_add_edges, gs_vertex_attr, gs_subset_nodes, gs_nodes

Examples

```
library(RGraphSpace)
library(igraph)

g <- make_star(5, mode = "out")
gs <- GraphSpace(g)

# Functional form (pipe-friendly): returns a modified copy
gs <- gs_add_nodes(gs, data.frame(name = "n6", x = 0.5, y = 0.5))

# Assignment form: modifies gs in place
gs_add_nodes(gs) <- data.frame(name = "n7", x = 0.5, y = 0.5)

# Add multiple nodes with visual attributes
gs <- gs_add_nodes(gs, data.frame(
  name      = c("n8", "n9"),
  x         = c(0.5, 0.8),
  y         = c(0.5, 0.2),
  nodeSize  = c(8, 5),
  nodeColor = c("steelblue", "tomato")
))
```

---

gs_features-utils *Manipulate node features in a GraphSpace object*

Description

Utilities for extracting and adding node-associated features stored in the fdata container of a GraphSpace object.

Usage

```
gs_fetch_features(x, vars = NULL, as_df = FALSE)

gs_add_features(x, data)
```

Arguments

x	A GraphSpace object.
vars	Character vector specifying feature names to extract. If NULL, all features are returned.
as_df	Logical. If TRUE, returns a data.frame. Otherwise returns the original backend representation.

data A matrix-like or data.frame object containing node features. Rows must correspond to node identifiers.

Value

- gs_fetch_features() returns a matrix-like object or data.frame containing the selected node features.
- gs_add_features() returns a modified GraphSpace object.

---

gs_image_toy *Toy 'GraphSpace' object*

Description

A small GraphSpace object used for workflow demonstrations. It includes an embedded image, with node coordinates representing image indices.

Usage

```
data(gs_image_toy)
```

Format

An GraphSpace object ready for rendering.

Value

A pre-processed GraphSpace object.

Source

This package.

Examples

```
library(RGraphSpace)
data(gs_image_toy)
```

gs_subset *Filter nodes and edges in a GraphSpace object*

Description

`gs_subset_nodes()` retains a subset of nodes and automatically removes any edge whose endpoint is no longer present.

`gs_subset_edges()` retains a subset of edges without modifying the node set.

Usage

```
gs_subset_nodes(x, i)

gs_subset_edges(x, i)
```

Arguments

`x`	A `GraphSpace` object.
`i`	A filter specification. Accepted forms: • A **character** vector of node names (`gs_subset_nodes()` only; edges are identified by integer position or predicate, not by name). • An **integer** vector of positional indices into the node or edge table. • A **logical** vector whose length must match the number of nodes or edges, respectively. • An **unquoted predicate** evaluated against the node or edge data frame using data masking, such as `nodeSize > 5` or `weight > 0.5`. Column names from the relevant table are available directly as variables inside the expression.

Details

Node filtering preserves the normalized coordinate state. Coordinates for surviving nodes remain in their current space (`[0, 1]` if normalized, raw coordinates otherwise), so `normalizeGraphSpace` does not need to be re-run. The `@graph`, `@fdata`, `@nodes`, and `@edges` slots are all updated consistently. The `@canvas` and background image are not modified.

Edge filtering leaves the node set and the layout entirely intact. Because removing an edge from a group of parallel edges invalidates the derived attributes `curve_weight`, `is_multiple`, and `is_loop` for the remaining members of that group, the full edge table is recomputed from `@graph` after deletion.

Note on parallel edges: in non-simplified graphs containing parallel edges between the same vertex pair, integer or logical indexing is the most reliable approach. A predicate expression that matches a shared attribute (such as `edgeColor`) will match all parallel instances simultaneously, which is usually the intended behavior.

Value

A GraphSpace object with the selected subset of nodes or edges.

See Also

cropGraphSpace, gs_nodes, gs_edges, normalizeGraphSpace

Examples

```
library(RGraphSpace)
library(igraph)

# Create a directed star graph with numeric attributes
g <- make_star(10, mode = "out")
V(g)$nodeSize <- runif(vcount(g), 1, 10)
E(g)$weight   <- runif(ecount(g), 0, 1)
gs <- GraphSpace(g)
gs <- normalizeGraphSpace(gs)

#--- gs_subset_nodes examples ---

# By node name (character vector)
gs2 <- gs_subset_nodes(gs, c("n1", "n2", "n3"))

# By integer position
gs2 <- gs_subset_nodes(gs, 1:5)

# By predicate (data masking against @nodes columns)
gs2 <- gs_subset_nodes(gs, nodeSize > 5)

# By pre-evaluated logical vector
keep <- gs$nodeSize > 5
gs2  <- gs_subset_nodes(gs, keep)

# Combining with pipes
gs2 <- gs |>
  gs_subset_nodes(nodeSize > 5) |>
  gs_subset_edges(weight > 0.3)

#--- gs_subset_edges examples ---

# By predicate on an edge attribute
gs3 <- gs_subset_edges(gs, weight > 0.5)

# By endpoint names: name1 and name2 are columns in @edges and
# can be used directly inside any predicate expression
gs3 <- gs_subset_edges(gs, name1 == "n1")
gs3 <- gs_subset_edges(gs, name2 == "n1")

# Combining endpoint and attribute conditions
gs3 <- gs_subset_edges(gs, name1 == "n1" & weight > 0.5)
```

```
# By integer position
gs3 <- gs_subset_edges(gs, 1:3)

# By logical vector
gs3 <- gs_subset_edges(gs, gs_edges(gs)$weight > 0.5)
```

---

gtoys *Toy 'igraph' objects*

Description

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

Usage

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

Format

igraph

Value

A pre-processed igraph object.

Source

This package.

Examples

```
library(RGraphSpace)
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(RGraphSpace)
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)
```

`normalizeGeometry,GraphSpace-method`
Normalize or fit node geometry

Description

Two related operations for keeping an `sfc` geometry column attached to a `GraphSpace`'s nodes in registration with the node coordinates, for two different situations.

Usage

```
## S4 method for signature 'GraphSpace'
normalizeGeometry(gs, name = "geometry", verbose = TRUE)

## S4 method for signature 'GraphSpace'
fitGeometry(gs, name = "geometry", use_node_size = TRUE, verbose = TRUE)
```

Arguments

`gs`	A `GraphSpace` object.
`name`	Character. Name of the geometry column to operate on.
`verbose`	Logical. Whether to report progress messages.
`use_node_size`	Logical. If `TRUE` (the default), `fitGeometry()` also rescales each geometry to match its node's `nodeSize`. If `FALSE`, only repositioning happens, each feature keeps its current size.

Details

normalizeGeometry is for geometry that is already spatially meaningful, with its own coordinates genuinely correspond to the nodes (e.g. real cell-segmentation boundaries) and only needs re-aligning to the current, normalized node frame. It fits a linear regression between the geometry's centroids and the node coordinates and rescales the geometry accordingly, warning if the fit is poor (the geometry did not, in fact, scale linearly with the nodes).

fitGeometry is for geometry that is not yet spatially related to the nodes, as arbitrary shapes used for node markers. It repositions every shape so its centroid sits exactly at its node's coordinates and, when use_node_size = TRUE, also rescales each shape so its diameter matches nodeSize.

Both require gs to already be normalized (see normalizeGraphSpace), and both operate on a single named geometry column, leaving any other geometry columns untouched.

Value

The updated GraphSpace object.

Examples

```
## Not run:
# Mode 1: geometry already spatially meaningful, just needs realigning
gs_geometry(gs, "geometry") <- real_cell_boundaries
gs <- normalizeGeometry(gs)

# Mode 2: arbitrary shapes, sized and positioned like nodes
gs_geometry(gs, "geometry") <- arbitrary_shapes
gs <- fitGeometry(gs, use_node_size = TRUE)

## End(Not run)
```

---

normalizeGraphSpace,GraphSpace-method

Normalize node coordinates to graph and image spaces

Description

Accessory function to normalize node coordinates of a GraphSpace object, either by centering nodes within the plot boundaries or by mapping nodes to pixel coordinates of a background image.

Usage

```
## S4 method for signature 'GraphSpace'
normalizeGraphSpace(
  gs,
  mar = 0.1,
  image.space = .has_image(gs),
  flip.x = FALSE,
```

```
  flip.y = image.space,
  flip.v = FALSE,
  flip.h = FALSE,
  swap.xy = FALSE,
  equal.mar = FALSE,
  verbose = TRUE
)
```

Arguments

gs	A GraphSpace object to be normalized.
mar	A single numeric value in [0, 0.5] setting the margins around the graph, as a fraction of the final normalized space. For example, mar = 0.1 leaves a margin of 0.1 on each side, so the graph occupies the central 0.8 of the space. With an image, the image is cropped to the same proportions; if the graph lies close to an image border, the crop is shifted or truncated to stay within the image, and the requested margin may not be reached.
image.space	Logical; if an image is available, whether to use it as a background reference map. When enabled, x and y graph coordinates are interpreted as pixel coordinates in the image matrix. Images can be inspected and assigned with gs_image.
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. Defaults to image.space.
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.
swap.xy	Logical; whether to swap x and y node coordinates. Useful when the graph coordinate system is transposed relative to the image or reference map.
equal.mar	Logical; when an image is available, whether to fit the image with equal margins around the graph, resulting in a tighter crop of the image. If FALSE (default), the image is fitted to the full square figure area, resulting in unequal margins when the graph aspect ratio differs from 1. Both methods preserve the aspect ratios of the image and graph.
verbose	A single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Details

This function re-scales node coordinates to a [0, 1] unit square based on the graph's bounding box when image.space = FALSE or, when an image is provided and image.space = TRUE, it maps nodes to pixel coordinates. It handles image-to-graph alignment via flip.* and swap.* 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.

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.

See Also

cropGraphSpace, gs_image

Examples

```
library(RGraphSpace)
library(igraph)

# Create a star graph
gtoy1 <- make_full_graph(30)

# Create a GraphSpace
gs <- GraphSpace(gtoy1)

gs <- normalizeGraphSpace(gs)

plotGraphSpace(gs, add.labels = TRUE)
```

---

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",
  node.labels = FALSE,
  label.size = 3,
  label.color = "grey20",
  add.image = TRUE,
  raster = FALSE,
  dpi = 300,
  dev = "cairo_png",
  add.labels = deprecated(),
  font.size = deprecated(),
  bg.color = deprecated()
)

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

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

## S4 method for signature 'gs_graph'
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 be subsequently refine using theme.
xlab	The title for the 'x' axis of a 2D-image space.

ylab	The title for the 'y' axis of a 2D-image space.
node.labels	Optional specification of node labels to display. If FALSE (default), no labels are shown. If TRUE, labels are displayed for all nodes. Otherwise, a character vector may be supplied to display labels only for the specified nodes. Character values are matched against the nodeLabel attribute.
label.size	Font size passed to geom_text.
label.color	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.
add.labels	Deprecated. Use node.labels instead.
font.size	Deprecated. Use theme customization instead.
bg.color	Deprecated. Use theme customization instead.
...	Additional arguments passed to the plotGraphSpace function.

Value

A ggplot-class object.

Author(s)

Sysbiolab.

See Also

GraphSpace

Examples

```
library(RGraphSpace)
library(igraph)

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

# Generate a ggplot for gtoy1
plotGraphSpace(gtoy1, node.labels = TRUE)


# 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)$edgeColor <- "red"
E(gtoy_star)$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)
```

`sfshape_ngon` *Build regular polygons*

Description

Construct one or more regular polygons (equal sides and angles) as `sfg`/`sfc` `POLYGON` geometry. `sfshape_ngon()` builds a single polygon at a given center; `sfshape_ngons()` builds `n` polygons, automatically arranged in a compact, non-overlapping grid. Useful for building varied, non-hand-typed node geometries; see the geometry vignette for examples, and `sfshape_stars` for the star-shaped equivalent.

Usage

```
sfshape_ngon(cx = 0, cy = 0, sides = 5, radius = 1)

sfshape_ngons(n, sides = c(3, 5, 7), radius = 0.3, spacing = NULL)
```

Arguments

`cx`, `cy`	Numeric. Coordinates of the polygon's center. (`sfshape_ngon()` only.)
`sides`	Integer. Number of sides; must be 3 or more. For `sfshape_ngons()`, may be a vector, recycled across the `n` polygons to vary shape per polygon.
`radius`	Numeric. Circumradius – distance from the center to each vertex. For `sfshape_ngons()`, may be a vector, recycled across the `n` polygons.
`n`	Integer. Number of polygons to build. (`sfshape_ngons()` only.)
`spacing`	Numeric. Distance between polygon centers in the auto-generated grid. Defaults to `max(radius) * 2.5`, which guarantees no overlap. (`sfshape_ngons()` only.)

Value

`sfshape_ngon()` returns a single `sfg` object of type `POLYGON`. `sfshape_ngons()` returns an `sfc` of `n` such polygons.

See Also

sfshape_stars

Examples

```
pentagon <- sfshape_ngon(0, 0, sides = 5, radius = 1)
hexagon  <- sfshape_ngon(2, 0, sides = 6, radius = 1)
plot(sf::st_sfc(pentagon, hexagon))

many <- sfshape_ngons(7, sides = c(3, 5, 8), radius = 0.4)
plot(many)
```

---

sfshape_star *Build star polygons*

Description

Construct one or more star-shaped polygons, alternating between an outer and an inner radius at each vertex, as sfg/sfc POLYGON geometry. sfshape_star() builds a single star at a given center; sfshape_stars() builds n stars, automatically arranged in a compact, non-overlapping grid. Useful for building varied, non-hand-typed node geometries; see the geometry vignette for examples, and sfshape_ngons for the regular-polygon equivalent.

Usage

```
sfshape_star(cx = 0, cy = 0, points = 5, r_outer = 0.3, r_inner = 0.1)

sfshape_stars(
  n,
  points = c(3, 4, 5),
  r_outer = 0.3,
  r_inner = 0.1,
  spacing = NULL
)
```

Arguments

cx, cy	Numeric. Coordinates of the star's center. (sfshape_star() only.)
points	Integer. Number of star points; must be 2 or more. For sfshape_stars(), may be a vector, recycled across the n stars to vary shape per star.
r_outer	Numeric. Radius to each outer (point) vertex. For sfshape_stars(), may be a vector, recycled across the n stars.
r_inner	Numeric. Radius to each inner (valley) vertex. Smaller values relative to r_outer produce sharper points; values closer to r_outer produce a rounder, less pronounced star. For sfshape_stars(), may be a vector, recycled across the n stars.

n	Integer. Number of stars to build. (sfshape_stars() only.)
spacing	Numeric. Distance between star centers in the auto-generated grid. Defaults to max(r_outer) * 2.5, which guarantees no overlap. (sfshape_stars() only.)

Value

sfshape_star() returns a single sfg object of type POLYGON. sfshape_stars() returns an sfc of n such stars.

See Also

sfshape_ngons

Examples

```
star5 <- sfshape_star(0, 0, points = 5, r_outer = 1, r_inner = 0.4)
star8 <- sfshape_star(3, 0, points = 8, r_outer = 1, r_inner = 0.7)
plot(sf::st_sfc(star5, star8))

many <- sfshape_stars(6, points = 5, r_outer = 0.4, r_inner = 0.16)
plot(many)
```

StatEdgeSpace *Attribute Processing for GeomEdgeSpace*

Description

Manage visual attribute precedence (colour, size, shape) for GeomEdgeSpace objects.

Usage

```
StatEdgeSpace
```

Format

A ggproto object.

Attribute Priority

1. Explicit aes() mappings.
2. Fixed geom_edgespace() arguments.
3. Original graph attributes (via optional_aes).

During the setup_data stage, the Stat invokes internal functions to resolve value priority:

1. **Explicit Mapping**: Values defined by the user inside aes().
2. **Fixed Parameters**: Constant values passed as arguments in the geom_edgespace() call.
3. **Graph Attributes**: Original attributes stored within the GraphSpace object, retrieved from the data columns.

See Also

geom_edgespace

StatNodeSpace *Attribute Processing for GeomNodeSpace*

Description

Manage visual attribute precedence (color, size, shape) for GeomNodeSpace objects.

Usage

```
StatNodeSpace
```

Format

A ggproto object.

Attribute Priority

1. Explicit aes() mappings.
2. Fixed geom_nodespace() arguments.
3. Original graph attributes (via optional_aes).

During the setup_data stage, the Stat invokes internal functions to resolve value priority:

1. **Explicit Mapping**: Values defined by the user inside aes().
2. **Fixed Parameters**: Constant values passed as arguments in the geom_nodespace() call.
3. **Graph Attributes**: Original attributes stored within the GraphSpace object, retrieved from the data columns.

See Also

geom_nodespace

summary,GraphSpace-method *Summarise a GraphSpace object*

Description

Prints a structured summary of a GraphSpace object, including graph topology, optional feature data, and spatial boundaries for nodes and, when present, the background image.

Node boundaries are always drawn from @graph (original pixel coordinates, never modified). Image boundaries reflect @canvas after normalization with image.space = TRUE, and @image otherwise. When normalized, both boundary lines show the source range and [0,1] target to make the transformation explicit.

Usage

```
## S4 method for signature 'GraphSpace'
summary(object, ...)
```

Arguments

object	A GraphSpace object.
...	Currently unused; present for S4 generic compatibility.

Value

Invisibly returns object, allowing the call to be used inside a pipeline without side effects beyond the printed output.

See Also

GraphSpace, normalizeGraphSpace

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_colour = "grey95",
  discrete_fill = FALSE,
  discrete_colour = FALSE,
  ...
)

theme_gspace_th1(
  txt_size = 1,
  leg_size = 1,
  bg_colour = "grey95",
  discrete_fill = FALSE,
  discrete_colour = FALSE,
  ...
)

theme_gspace_th2(
  txt_size = 1,
  leg_size = 1,
  bg_colour = "grey95",
  discrete_fill = FALSE,
  discrete_colour = FALSE,
  ...
)

theme_gspace_th3(
  txt_size = 1,
  leg_size = 1,
  bg_colour = "grey95",
  discrete_fill = FALSE,
  discrete_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,
```

```
  discrete_fill = FALSE,
  discrete_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_colour	A colour name or hex code specifying the panel background.
discrete_fill	Logical; if TRUE, treats the fill legend as discrete to adjust key size.
discrete_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(RGraphSpace)
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)

# Reduce legend element sizes
ggplot(mtcars, aes(wt, mpg, fill = factor(cyl))) +
  geom_point(shape = 21) +
  theme_gspace_legend(0.8)
```

---

updateGraphSpace,GraphSpace-method

Update a GraphSpace object

Description

Updates GraphSpace objects serialized from previous package versions, adding any missing slots with default values.

Usage

```
## S4 method for signature 'GraphSpace'
updateGraphSpace(x, verbose = TRUE)
```

Arguments

x	A GraphSpace object.
verbose	Logical; if TRUE, reports which slots were added.

Value

An updated GraphSpace object.

Index