Package 'ggnetwork'

August 29, 2016

Type Package

Title Geometries to Plot Networks with 'ggplot2'

Version 0.5.1

Date 2016-03-24

Author Francois Briatte

Maintainer Francois Briatte <f.briatte@gmail.com>

Description Geometries to plot network objects with 'ggplot2'.

License GPL-3

LazyData TRUE

VignetteBuilder knitr

NeedsCompilation no

Depends ggplot2 (>= 2.0.0), R (>= 3.1),

Imports ggrepel (>= 0.5), network, sna

Suggests intergraph, knitr, testthat

RoxygenNote 5.0.1

Repository CRAN

Date/Publication 2016-03-25 01:15:36

R topics documented:

fortify.igraph 2
fortify.network 2
geom_edges 4
geom_edgetext 6
geom_edgetext_repel 8
geom_nodes 10
geom_nodetext 12
geom_nodetext_repel 14
ggnetwork 16
theme_blank 17
theme_facet 17

Index **18**

`fortify.igraph` *Fortify method for networks of class* `igraph`

Description

This function requires the `intergraph` package to be installed.

Usage

```
## S3 method for class 'igraph'
fortify(model, ...)
```

Arguments

`model`	an object of class `igraph`, which will be converted to an object of class `network` with the `asNetwork` function before being passed to the `fortify.network` function.
`...`	additional parameters for the `fortify.network` function; see `fortify.network`.

`fortify.network` *Fortify method for networks of class* `network`

Description

See the vignette at `https://briatte.github.io/ggnetwork/` for a description of both this function and the rest of the `ggnetwork` package.

Usage

```
## S3 method for class 'network'
fortify(model, data = NULL,
  layout = "fruchtermanreingold", weights = NULL,
  arrow.gap = ifelse(network::is.directed(x), 0.025, 0), by = NULL, ...)
```

Arguments

`model`	an object of class `network`.
`data`	not used by this method.
`layout`	a network layout supplied by `gplot.layout`, such as `"fruchtermanreingold"` (the default), or a two-column matrix with as many rows as there are nodes in the network, in which case the matrix is used as nodes coordinates.
`weights`	the name of an edge attribute to use as edge weights when computing the network layout, if the layout supports such weights (see 'Details'). Defaults to `NULL` (no edge weights).

arrow.gap	a parameter that will shorten the network edges in order to avoid overplotting edge arrows and nodes; defaults to 0 when the network is undirected (no edge shortening), or to 0.025 when the network is directed. Small values near 0.025 will generally achieve good results when the size of the nodes is reasonably small.
by	a character vector that matches an edge attribute, which will be used to generate a data frame that can be plotted with facet_wrap or facet_grid. The nodes of the network will appear in all facets, at the same coordinates. Defaults to NULL (no faceting).
...	additional parameters for the layout argument; see gplot.layout for available options.

Details

fortify.network will return a warning if it finds duplicated edges after converting the network to an edge list. Duplicated edges should be eliminated in favour of single weighted edges before using a network layout that supports edge weights, such as the Kamada-Kawai force-directed placement algorithm.

Value

a data.frame object.

Examples

```
if (require(ggplot2) && require(network)) {

  # source: ?network::flo
  data(flo)

  # data example
  ggnetwork(flo)

  # plot example
  ggplot(ggnetwork(flo), aes(x, y, xend = xend, yend = yend)) +
    geom_edges(alpha = 0.5) +
    geom_nodes(size = 12, color = "white") +
    geom_nodetext(aes(label = vertex.names), fontface = "bold") +
    theme_blank()

  # source: ?network::emon
  data(emon)

  # data example
  ggnetwork(emon[[1]], layout = "target", niter = 100)

  # data example with edge weights
  ggnetwork(emon[[1]], layout = "kamadakawai", weights = "Frequency")

  # plot example with straight edges
  ggplot(ggnetwork(emon[[1]], layout = "kamadakawai", arrow.gap = 0.025),
```

```
      aes(x, y, xend = xend, yend = yend)) +
  geom_edges(aes(color = Frequency),
             arrow = arrow(length = unit(10, "pt"), type = "closed")) +
  geom_nodes(aes(size = Formalization)) +
  scale_color_gradient(low = "grey50", high = "tomato") +
  scale_size_area(breaks = 1:3) +
  theme_blank()

# plot example with curved edges
ggplot(ggnetwork(emon[[1]], layout = "kamadakawai", arrow.gap = 0.025),
       aes(x, y, xend = xend, yend = yend)) +
  geom_edges(aes(color = Frequency), curvature = 0.1,
             arrow = arrow(length = unit(10, "pt"), type = "open")) +
  geom_nodes(aes(size = Formalization)) +
  scale_color_gradient(low = "grey50", high = "tomato") +
  scale_size_area(breaks = 1:3) +
  theme_blank()

# facet by edge attribute
ggplot(ggnetwork(emon[[1]], arrow.gap = 0.02, by = "Frequency"),
       aes(x, y, xend = xend, yend = yend)) +
  geom_edges(arrow = arrow(length = unit(5, "pt"), type = "closed")) +
  geom_nodes() +
  theme_blank() +
  facet_grid(. ~ Frequency, labeller = label_both)

# user-provided layout
ggplot(ggnetwork(emon[[1]], layout = matrix(runif(28), ncol = 2)),
       aes(x, y, xend = xend, yend = yend)) +
  geom_edges(arrow = arrow(length = unit(5, "pt"), type = "closed")) +
  geom_nodes() +
  theme_blank()

}
```

`geom_edges` *Draw the edges of a network.*

Description

All arguments to this geom are identical to those of `geom_segment`, including `arrow`, which is useful to plot directed networks in conjunction with the `arrow.gap` argument of `fortify.network`. The `curvature`, `angle` and `ncp` arguments of `geom_curve` are also available: if `curvature` is set to any value above `0` (the default), the edges produced by `geom_edges` will be curved.

Usage

```
geom_edges(mapping = NULL, data = NULL, position = "identity",
  arrow = NULL, curvature = 0, angle = 90, ncp = 5, na.rm = FALSE,
  show.legend = NA, inherit.aes = TRUE, ...)
```

Arguments

mapping	Set of aesthetic mappings created by aes or aes_. If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot. A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame., and will be used as the layer data.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
arrow	specification for arrow heads, as created by arrow()
curvature	A numeric value giving the amount of curvature. Negative values produce left-hand curves, positive values produce right-hand curves, and zero produces a straight line.
angle	A numeric value between 0 and 180, giving an amount to skew the control points of the curve. Values less than 90 skew the curve towards the start point and values greater than 90 skew the curve towards the end point.
ncp	The number of control points used to draw the curve. More control points creates a smoother curve.
na.rm	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
...	other arguments passed on to layer. These are often aesthetics, used to set an aesthetic to a fixed value, like color = "red" or size = 3. They may also be parameters to the paired geom/stat.

Examples

```
if (require(network) && require(sna)) {

# rerun if the example does not produce reciprocated ties
n <- network(rgraph(10, tprob = 0.2), directed = TRUE)

# just edges
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(size = 1, color = "steelblue") +
  theme_blank()
```

```
# with nodes
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(size = 1, color = "steelblue") +
  geom_nodes(size = 3, color = "steelblue") +
  theme_blank()

# with arrows
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(size = 1, color = "steelblue",
             arrow = arrow(length = unit(0.5, "lines"), type = "closed")) +
  geom_nodes(size = 3, color = "steelblue") +
  theme_blank()

# with curvature
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(size = 1, color = "steelblue", curvature = 0.15,
             arrow = arrow(length = unit(0.5, "lines"), type = "closed")) +
  geom_nodes(size = 3, color = "steelblue") +
  theme_blank()

# arbitrary categorical edge attribute
e <- sample(letters[ 1:2 ], network.edgecount(n), replace = TRUE)
set.edge.attribute(n, "type", e)
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(aes(linetype = type), size = 1, curvature = 0.15,
             arrow = arrow(length = unit(0.5, "lines"), type = "closed")) +
  geom_nodes(size = 3, color = "steelblue") +
  theme_blank()

# arbitrary numeric edge attribute (signed network)
e <- sample(-2:2, network.edgecount(n), replace = TRUE)
set.edge.attribute(n, "weight", e)
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(aes(color = weight), curvature = 0.15,
             arrow = arrow(length = unit(0.5, "lines"), type = "closed")) +
  geom_nodes(size = 3, color = "grey50") +
  scale_color_gradient(low = "steelblue", high = "tomato") +
  theme_blank()

# draw only a subset of all edges
positive_weight <- function(x) { x[ x$weight >= 0, ] }
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(aes(color = weight), data = positive_weight) +
  geom_nodes(size = 4, color = "grey50") +
  scale_color_gradient(low = "gold", high = "tomato") +
  theme_blank()

}
```

`geom_edgetext` *Label the edges of a network.*

Description

All arguments to both geom_edgetext and geom_edgelabel are identical to those of geom_label, with the only difference that the label.size argument defaults to 0 in order to avoid drawing a border around the edge labels. The labels will be drawn at mid-edges. geom_text and geom_label produce strictly identical results.

Usage

```
geom_edgetext(mapping = NULL, data = NULL, position = "identity",
  parse = FALSE, ..., nudge_x = 0, nudge_y = 0,
  label.padding = unit(0.25, "lines"), label.r = ggplot2::unit(0.15,
  "lines"), label.size = 0, na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE)

geom_edgelabel(mapping = NULL, data = NULL, position = "identity",
  parse = FALSE, ..., nudge_x = 0, nudge_y = 0,
  label.padding = unit(0.25, "lines"), label.r = ggplot2::unit(0.15,
  "lines"), label.size = 0, na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE)
```

Arguments

mapping	Set of aesthetic mappings created by aes or aes_. If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot. A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame., and will be used as the layer data.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
parse	If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath
...	other arguments passed on to layer. These are often aesthetics, used to set an aesthetic to a fixed value, like color = "red" or size = 3. They may also be parameters to the paired geom/stat.
nudge_x, nudge_y	Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales.
label.padding	Amount of padding around label. Defaults to 0.25 lines.
label.r	Radius of rounded corners. Defaults to 0.15 lines.
label.size	Size of label border, in mm.

na.rm	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.

Examples

```
if (require(network) && require(sna)) {

data(flo, package = "network")
n <- network(flo, directed = FALSE)

# arbitrary categorical edge attribute
e <- sample(letters[ 1:4 ], network.edgecount(n), replace = TRUE)
set.edge.attribute(n, "type", e)

# with labelled edges
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(aes(color = type)) +
  geom_edgetext(aes(label = type, color = type)) +
  geom_nodes(size = 4, color = "grey50") +
  theme_blank()

# label only a subset of all edges with arbitrary symbol
edge_type <- function(x) { x[ x$type == "a", ] }
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges() +
  geom_edgetext(label = "=", data = edge_type) +
  geom_nodes(size = 4, color = "grey50") +
  theme_blank()

}
```

---

geom_edgetext_repel *Draw repulsive edge labels.*

Description

All arguments to both geom_edgetext_repel and geom_edgelabel_repel are identical to those of geom_label_repel. geom_text_repel and geom_label_repel produce strictly identical results.

Usage

```
geom_edgetext_repel(mapping = NULL, data = NULL, parse = FALSE, ...,
  box.padding = unit(0.25, "lines"), label.padding = unit(0.25, "lines"),
  point.padding = unit(1e-06, "lines"), label.r = unit(0.15, "lines"),
```

```
  label.size = 0.25, segment.color = "#666666", segment.size = 0.5,
  arrow = NULL, force = 1, max.iter = 2000, nudge_x = 0, nudge_y = 0,
  na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)

geom_edgelabel_repel(mapping = NULL, data = NULL, parse = FALSE, ...,
  box.padding = unit(0.25, "lines"), label.padding = unit(0.25, "lines"),
  point.padding = unit(1e-06, "lines"), label.r = unit(0.15, "lines"),
  label.size = 0.25, segment.color = "#666666", segment.size = 0.5,
  arrow = NULL, force = 1, max.iter = 2000, nudge_x = 0, nudge_y = 0,
  na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes` or `aes_`. If specified and `inherit.aes = TRUE` (the default), is combined with the default mapping at the top level of the plot. You only need to supply `mapping` if there isn't a mapping defined for the plot.
`data`	A data frame. If specified, overrides the default data frame defined at the top level of the plot.
`parse`	If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath
`...`	other arguments passed on to `layer`. There are three types of arguments you can use here: • Aesthetics: to set an aesthetic to a fixed value, like `color = "red"` or `size = 3`. • Other arguments to the layer, for example you override the default `stat` associated with the layer. • Other arguments passed on to the stat.
`box.padding`	Amount of padding around bounding box. Defaults to `unit(0.25, "lines")`.
`label.padding`	Amount of padding around label. Defaults to 0.25 lines.
`point.padding`	Amount of padding around labeled point. Defaults to `unit(0, "lines")`.
`label.r`	Radius of rounded corners. Defaults to 0.15 lines.
`label.size`	Size of label border, in mm.
`segment.color`	Color of the line segment connecting the data point to the text labe. Defaults to `#666666`.
`segment.size`	Width of segment, in mm.
`arrow`	specification for arrow heads, as created by `arrow`
`force`	Force of repulsion between overlapping text labels. Defaults to 1.
`max.iter`	Maximum number of iterations to try to resolve overlaps. Defaults to 2000.
`nudge_x, nudge_y`	Horizontal and vertical adjustments to nudge the starting position of each text label.
`na.rm`	If `FALSE` (the default), removes missing values with a warning. If `TRUE` silently removes missing values.

show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.

Examples

```
if (require(network) && require(sna)) {

data(flo, package = "network")
n <- network(flo, directed = FALSE)

# arbitrary categorical edge attribute
e <- sample(1:4, network.edgecount(n), replace = TRUE)
set.edge.attribute(n, "day", e)

# with repulsive edge labels
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges() +
  geom_edgetext_repel(aes(label = day), box.padding = unit(0.5, "lines")) +
  geom_nodes(size = 4, color = "grey50") +
  theme_blank()

# repulsive edge labels for only a subset of all edges
edge_day <- function(x) { x[ x$day > 2, ] }
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(aes(color = cut(day, (4:0)[ -3 ]))) +
  geom_edgetext_repel(aes(label = paste("day", day),
                      color = cut(day, (4:0)[ -3 ])), data = edge_day) +
  geom_nodes(size = 4, color = "grey50") +
  scale_color_manual("day", labels = c("old ties", "day 3", "day 4"),
                     values = c("grey50", "gold", "tomato")) +
  theme_blank()

}
```

---

geom_nodes *Draw the nodes of a network.*

Description

All arguments to this geom are identical to those of geom_point.

Usage

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

Arguments

mapping	Set of aesthetic mappings created by aes or aes_. If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot. A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame., and will be used as the layer data.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
na.rm	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
...	other arguments passed on to layer. These are often aesthetics, used to set an aesthetic to a fixed value, like color = "red" or size = 3. They may also be parameters to the paired geom/stat.

Examples

```
if (require(network) && require(sna)) {

data(flo, package = "network")
n <- network(flo, directed = FALSE)

# just nodes
ggplot(n, aes(x, y)) +
  geom_nodes(size = 3, shape = 21, color = "steelblue") +
  theme_blank()

# with edges
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(color = "steelblue") +
  geom_nodes(size = 3, shape = 21, color = "steelblue", fill = "white") +
  theme_blank()

# with nodes sized according to degree centrality
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(color = "steelblue") +
  geom_nodes(size = degree(n), shape = 21, color = "steelblue", fill = "white") +
  theme_blank()
```

```
# with nodes colored according to betweenness centrality

n %v% "betweenness" <- betweenness(flo)
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(color = "grey50") +
  geom_nodes(aes(color = betweenness), size = 3) +
  scale_color_gradient(low = "gold", high = "tomato") +
  theme_blank() +
  theme(legend.position = "bottom")

}
```

`geom_nodetext` *Label the nodes of a network.*

Description

All arguments to these geoms are identical to those of `geom_text` and `geom_label`.

Usage

```
geom_nodetext(mapping = NULL, data = NULL, position = "identity", ...,
  parse = FALSE, nudge_x = 0, nudge_y = 0, check_overlap = FALSE,
  na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)

geom_nodelabel(mapping = NULL, data = NULL, position = "identity", ...,
  parse = FALSE, nudge_x = 0, nudge_y = 0, label.padding = unit(0.25,
  "lines"), label.r = unit(0.15, "lines"), label.size = 0.25,
  na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes` or `aes_`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame.`, and will be used as the layer data.
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.

...	other arguments passed on to layer. These are often aesthetics, used to set an aesthetic to a fixed value, like color = "red" or size = 3. They may also be parameters to the paired geom/stat.
parse	If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath
nudge_x, nudge_y	Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales.
check_overlap	If TRUE, text that overlaps previous text in the same layer will not be plotted. A quick and dirty way
na.rm	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
label.padding	Amount of padding around label. Defaults to 0.25 lines.
label.r	Radius of rounded corners. Defaults to 0.15 lines.
label.size	Size of label border, in mm.

Examples

```
## geom_nodetext examples

if (require(network) && require(sna)) {

n <- network(rgraph(10, tprob = 0.2), directed = FALSE)

# just node labels
ggplot(n, aes(x, y)) +
  geom_nodetext(aes(label = vertex.names)) +
  theme_blank()

# with nodes underneath
ggplot(n, aes(x, y)) +
  geom_nodes(color = "gold", size = 9) +
  geom_nodetext(aes(label = vertex.names)) +
  theme_blank()

# with nodes and edges
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(color = "gold") +
  geom_nodes(color = "gold", size = 9) +
  geom_nodetext(aes(label = vertex.names)) +
  theme_blank()

}
```

```
## geom_nodelabel examples

if (require(network) && require(sna)) {

data(flo, package = "network")
n <- network(flo, directed = FALSE)

# with text labels
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(color = "grey50") +
  geom_nodelabel(aes(label = vertex.names)) +
  theme_blank()

# with text labels coloured according to degree centrality
n %v% "degree" <- degree(n)
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(color = "grey50") +
  geom_nodelabel(aes(label = vertex.names, fill = degree)) +
  scale_fill_gradient(low = "gold", high = "tomato") +
  theme_blank()

# label only a subset of all nodes
high_degree <- function(x) { x[ x$degree > median(x$degree), ] }
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(color = "steelblue") +
  geom_nodes(aes(size = degree), color = "steelblue") +
  geom_nodelabel(aes(label = vertex.names), data = high_degree,
                 color = "white", fill = "tomato") +
  theme_blank()
}
```

`geom_nodetext_repel` *Draw repulsive node labels*

Description

All arguments to these geoms are identical to those of `geom_text_repel` and `geom_label_repel`.

Usage

```
geom_nodetext_repel(mapping = NULL, data = NULL, parse = FALSE, ...,
  box.padding = unit(0.25, "lines"), point.padding = unit(1e-06, "lines"),
  segment.color = "#666666", segment.size = 0.5, arrow = NULL,
  force = 1, max.iter = 2000, nudge_x = 0, nudge_y = 0, na.rm = FALSE,
  show.legend = NA, inherit.aes = TRUE)

geom_nodelabel_repel(mapping = NULL, data = NULL, parse = FALSE, ...,
  box.padding = unit(0.25, "lines"), label.padding = unit(0.25, "lines"),
```

```
  point.padding = unit(1e-06, "lines"), label.r = unit(0.15, "lines"),
  label.size = 0.25, segment.color = "#666666", segment.size = 0.5,
  arrow = NULL, force = 1, max.iter = 2000, nudge_x = 0, nudge_y = 0,
  na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)
```

Arguments

mapping	Set of aesthetic mappings created by aes or aes_. If specified and inherit.aes = TRUE (the default), is combined with the default mapping at the top level of the plot. You only need to supply mapping if there isn't a mapping defined for the plot.
data	A data frame. If specified, overrides the default data frame defined at the top level of the plot.
parse	If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath
...	other arguments passed on to layer. There are three types of arguments you can use here: • Aesthetics: to set an aesthetic to a fixed value, like color = "red" or size = 3. • Other arguments to the layer, for example you override the default stat associated with the layer. • Other arguments passed on to the stat.
box.padding	Amount of padding around bounding box. Defaults to unit(0.25, "lines").
point.padding	Amount of padding around labeled point. Defaults to unit(0, "lines").
segment.color	Color of the line segment connecting the data point to the text labe. Defaults to #666666.
segment.size	Width of segment, in mm.
arrow	specification for arrow heads, as created by arrow
force	Force of repulsion between overlapping text labels. Defaults to 1.
max.iter	Maximum number of iterations to try to resolve overlaps. Defaults to 2000.
nudge_x, nudge_y	Horizontal and vertical adjustments to nudge the starting position of each text label.
na.rm	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
label.padding	Amount of padding around label. Defaults to 0.25 lines.
label.r	Radius of rounded corners. Defaults to 0.15 lines.
label.size	Size of label border, in mm.

Examples

```
## geom_nodetext_repel example

if (require(network) && require(sna)) {

n <- network(rgraph(10, tprob = 0.2), directed = FALSE)
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(color = "steelblue") +
  geom_nodetext_repel(aes(label = paste("node", vertex.names)),
                      box.padding = unit(1, "lines")) +
  geom_nodes(color = "steelblue", size = 3) +
  theme_blank()

}
## geom_nodelabel_repel examples

if (require(network) && require(sna)) {

data(flo, package = "network")
n <- network(flo, directed = FALSE)

ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(color = "steelblue") +
  geom_nodelabel_repel(aes(label = vertex.names),
                       box.padding = unit(1, "lines")) +
  geom_nodes(color = "steelblue", size = 3) +
  theme_blank()

# label only a subset of all nodes
n %v% "degree" <- degree(n)
low_degree <- function(x) { x[ x$degree < median(x$degree), ] }
ggplot(n, aes(x, y, xend = xend, yend = yend)) +
  geom_edges(color = "steelblue") +
  geom_nodelabel_repel(aes(label = vertex.names),
                       box.padding = unit(1.5, "lines"),
                       data = low_degree,
                       segment.color = "tomato",
                       color = "white", fill = "tomato") +
  geom_nodes(aes(size = degree), color = "steelblue") +
  theme_blank()

}
```

`ggnetwork` *Fortify network objects.*

Description

A wrapper for the `fortify.network` and `fortify.igraph` functions that will also try to coerce matrices and data frames to network objects.

Usage

```
ggnetwork(x, ...)
```

Arguments

x	an object of class network, or any object that can be coerced to this class, such as an adjacency or incidence matrix, or an edge list: see edgeset.constructors and network for details. If the object is of class igraph and the intergraph package is installed, it will be used to convert the object: see fortify.igraph for details.
...	arguments passed to the fortify.network function.

---

theme_blank *Blank ggplot2 theme, suited for plotting networks.*

Description

A ggplot2 theme without lines, borders, axis text or titles, suited for plotting networks.

Usage

```
theme_blank(base_size = 12, base_family = "", ...)
```

Arguments

base_size	base font size
base_family	base font family
...	other theme arguments

---

theme_facet *Blank ggplot2 theme with a panel border.*

Description

A variation of theme_blank that adds a panel border to the plot, which is often suitable for plotting faceted networks.

Usage

```
theme_facet(base_size = 12, base_family = "", ...)
```

Arguments

base_size	base font size
base_family	base font family
...	other theme arguments

Index

aes, *5*, *7*, *9*, *11, 12*, *15*
aes_, *5*, *7*, *9*, *11, 12*, *15*
arrow, *9*, *15*
asNetwork, *2*

borders, *5*, *8*, *10, 11*, *13*, *15*

edgeset.constructors, *17*

facet_grid, *3*
facet_wrap, *3*
fortify, *5*, *7*, *11, 12*
fortify.igraph, 2, *16, 17*
fortify.network, *2*, 2, *4*, *16, 17*

geom_curve, *4*
geom_edgelabel, *7*
geom_edgelabel *(*geom_edgetext*)*, 7
geom_edgelabel_repel, *8*
geom_edgelabel_repel
*(*geom_edgetext_repel*)*, 8
geom_edges, 4
geom_edgetext, 6, *7*
geom_edgetext_repel, *8*, 8
geom_label, *7*, *12*
geom_label_repel, *8*, *14*
geom_nodelabel *(*geom_nodetext*)*, 12
geom_nodelabel_repel
*(*geom_nodetext_repel*)*, 14
geom_nodes, 10
geom_nodetext, 12
geom_nodetext_repel, 14
geom_point, *10*
geom_segment, *4*
geom_text, *7*, *12*
geom_text_repel, *8*, *14*
ggnetwork, 16
ggplot, *5*, *7*, *11, 12*
gplot.layout, *2, 3*

igraph, *2*, *17*

intergraph, *2*, *17*

layer, *5*, *7*, *9*, *11*, *13*, *15*

network, *2*, *17*

theme, *17*
theme_blank, *17*, 17
theme_facet, 17