graphlayouts: Layout algorithms for network visualizations in R

David Schoch

1 GESIS - Leibniz Institute for the Social Sciences

Summary

Network analysis offers a powerful set of methods to understand the relationships among entities, such as people, or organizations, and the patterns that emerge from these connections. It is increasingly popular in various fields, including sociology, biology, economics, and computer science, and has been used to study diverse phenomena including the spread of diseases, flow of information, and the structure of political organizations. Network visualization is a powerful tool for exploring, analyzing, and communicating network structures and patterns therein. However, in contrast to tabular data, nodes can technically be placed arbitrarily on the plane and it is easy to draw wrong conclusion based on an inadequate layout. To circumvent arbitrary placement of nodes, many different layout algorithms have been developed which optimize different stylistic features and can serve purposes in communicating structural properties. The package graphlayouts implements several state-of-the-art algorithms which are so far not available for R. This includes algorithms for large graphs, to emphasize hidden group structures, and important nodes within a network.

Statement of need

The all-purpose network analysis package igraph (Csardi & Nepusz, 2006) already implements a great variety of layout algorithms for networks. graphlayouts complements these by adding a faster general purpose algorithm and a series of specialized algorithms that serve very specific purposes, either to emphasize group structures or the position of individual nodes within a network. It further adds support for dynamic and multilevel networks. The package is already well integrated into the R ecosystem. ggraph (Pedersen, 2022), the ggplot2 for networks, imports the package and uses the stress based layout as its default layout algorithm. All layout algorithms included in the package by default return a matrix of coordinates. This allows to use the layouts with most other network visualization packages too, including static ones such as sna (Butts, 2008) and statnet (Handcock et al., 2008), and interactive ones such as vlsNetwork (Almende et al., 2022).

Overview of algorithms

In this section, the most prominent implemented layout algorithms are presented. The description of algorithms are kept at a minimum and just for illustrative purposes. More details on all algorithms can be found in a short vignette, the online documentation http://graphlayouts.schochastics.net and a comprehensive tutorial on network visualization https://www.mr.schochastics.net/material/netVizR/.

Stress majorization

Stress majorization (Gansner et al., 2005) is, in contrast to many other layout algorithms, deterministic and able to produce high quality layouts for a great variety of graph classes. graphlayouts implements classic stress majorization in `layout_with_stress()`. An example is shown in Figure 1.

```r
library(graphlayouts)

# Example graph
g <- sample_islands(9, 40, 0.4, 15)
g <- simplify(g)
V(g)$grp <- as.character(rep(1:9, each=40))

# Stress layout
gg <- ggraph(g, layout = "stress") +
    geom_edge_link(edge_width = 0.2, edge_colour = "grey") +
    geom_node_point(size = 0.3) +
    theme_graph()

# Display
gg
```

Figure 1: Example of a stress based layout.

Sparse stress majorization

Stress majorization requires the computation of the full distance matrix, which becomes computationally expensive for large graphs. The function `layout_with_sparse_stress()` calculates the distances only for a small set of pivots and creates the layout based on these distances (Ortmann et al., 2016). The resulting layout is not as good as the full stress, but it can reasonably be used for graphs with around 100k nodes and 5 million edges. The wiki https://github.com/schochastics/graphlayouts/wiki contains several benchmark results in comparison with layouts from igraph.

Backbone layout

`layout_as_backbone()` is a layout algorithm that can help emphasize hidden group structures in hairball graphs (Nocaj et al., 2015). To illustrate the performance of the algorithm, we use an artificial network with a subtle group structure (cf. Figure 2).

```r
# Set seed for reproducibility
set.seed(665)

# Create network with group structure
g <- sample_islands(9, 40, 0.4, 15)
g <- simplify(g)
V(g)$grp <- as.character(rep(1:9, each=40))

# Backbone layout
gg <- ggraph(g, layout = "stress") +
    geom_edge_link(edge_width = 0.1, colour = rgb(0,0,0.5)) +
    geom_node_point(size = 0.3)

# Display
gg
```

The backbone layout helps to uncover potential group structures based on edge embeddedness and puts more emphasis on this structure in the layout (cf. Figure 3).

```r
bb <- layout_as_backbone(g, keep = 0.4)
E(g)$col <- FALSE
E(g)$col[bb$backbone] <- TRUE

ggraph(g, layout = "manual", x = bb$xy[, 1], y = bb$xy[, 2]) +
  geom_edge_link(aes(col = col), edge_width = 0.1) +
  geom_node_point(aes(col = grp)) +
  scale_color_brewer(palette = "Set1") +
  scale_edge_color_manual(values = c(rgb(0, 0, 0, 0.3), rgb(0, 0, 0, 1))) +
  theme_graph() +
  theme(legend.position = "none")
```

Figure 2: Hairball graph with stress layout.

Radial layouts

The function `layout_with_focus()` creates a radial layout around a focal node (Brandes & Pich, 2010). All nodes with the same distance from the focal node are on the same circle (cf. Figure 4).

```r
library(igraphdata)
library(patchwork)
data("karate")

p1 <- ggraph(karate, layout = "focus", focus = 1) +
draw_circle(use = "focus", max.circle = 3) +
geom_edge_link0(edge_color = "black", edge_width = 0.3) +
geom_node_point(aes(fill = as.factor(Faction)), size = 2, shape = 21) +
scale_fill_manual(values = c("#8B2323", "#EEAD0E")) +
theme_graph() +
theme(legend.position = "none") +
coord_fixed() +
labs(title = "Focus on Mr. Hi")

p2 <- ggraph(karate, layout = "focus", focus = 34) +
draw_circle(use = "focus", max.circle = 4) +
geom_edge_link0(edge_color = "black", edge_width = 0.3) +
geom_node_point(aes(fill = as.factor(Faction)), size = 2, shape = 21) +
scale_fill_manual(values = c("#8B2323", "#EEAD0E")) +
theme_graph() +
theme(legend.position = "none") +
coord_fixed() +
labs(title = "Focus on John A.")

p1 + p2
```

Figure 3: Backbone layout of the graph shown in Figure 2.
The function `layout_with_centrality` creates a radial layout around the node with the highest centrality value. The further outside a node is, the more peripheral it is (cf. Figure 5).

```r
bc <- betweenness(karate)
p1 <- ggraph(karate, layout = "centrality", centrality = bc, tseq = seq(0, 1, 0.15)) +
  draw_circle(use = "cent") +
  annotate_circle(bc, format = "", pos = "bottom") +
  geom_edge_link0(edge_color = "black", edge_width = 0.3) +
  geom_node_point(aes(fill = as.factor(Faction)), size = 2, shape = 21) +
  scale_fill_manual(values = c("#8B2323", "#EEAD0E") ) +
  theme_graph() +
  theme(legend.position = "none") +
  coord_fixed() +
  labs(title = "betweenness centrality")

cc <- closeness(karate)
p2 <- ggraph(karate, layout = "centrality", centrality = cc, tseq = seq(0, 1, 0.2)) +
  draw_circle(use = "cent") +
  annotate_circle(cc, format = "scientific", pos = "bottom") +
  geom_edge_link0(edge_color = "black", edge_width = 0.3) +
  geom_node_point(aes(fill = as.factor(Faction)), size = 2, shape = 21) +
  scale_fill_manual(values = c("#8B2323", "#EEAD0E") ) +
  theme_graph() +
  theme(legend.position = "none") +
  coord_fixed() +
  labs(title = "closeness centrality")

p1 + p2
```

Figure 4: Example of focus layouts.
Figure 5: Example of centrality layout using betweenness and closeness.

References


