graphlayouts: Layout algorithms for network visualizations in R

David Schoch [1]

1 GESIS - Leibniz Institute for the Social Sciences

DOI: 10.21105/joss.05238

Software

- Review
- Repository
- Archive

Editor: Chris Vernon

Reviewers:

- @bpbond
- @ndjackso

Submitted: 03 March 2023
Published: 27 April 2023

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 `visNetwork` (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.

```
ggraph(pa,layout = "stress")+
  geom_edge_link0(edge_width = 0.2, edge_colour = "grey")+
  geom_node_point(size = 0.3)+
  theme_graph()
```

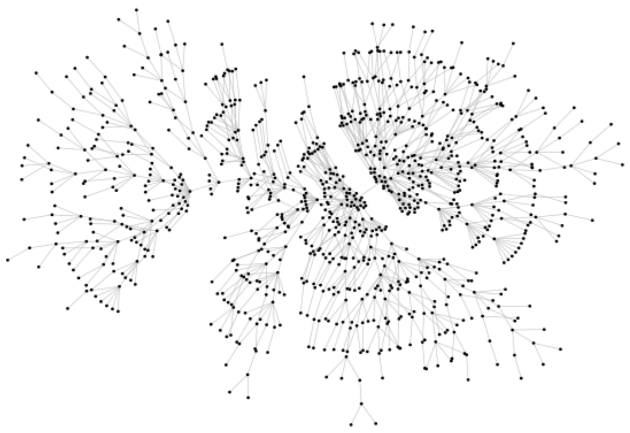

Figure 1: Example of a stress based layout.

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).

```
set.seed(665)
#create network with a group structure
g <- sample_islands(9,40,0.4,15)
g <- simplify(g)
V(g)$grp <- as.character(rep(1:9,each=40))

ggraph(g,layout = "stress")+
  geom_edge_link0(colour = rgb(0,0,0,0.5), edge_width = 0.1)+
  geom_node_point(aes(col = grp))+
  scale_color_brewer(palette = "Set1")+
  theme_graph()+
  theme(legend.position = "none")
```

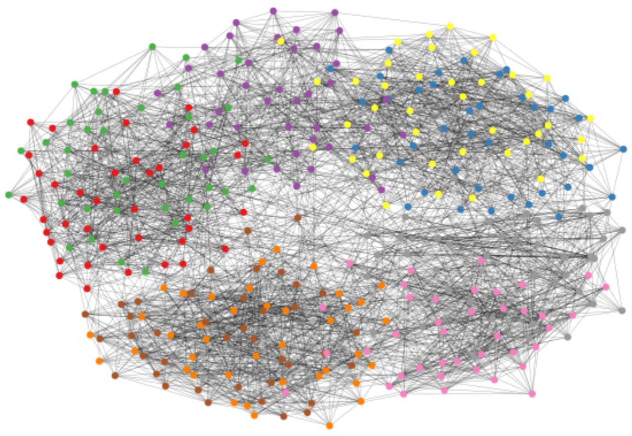

Figure 2: Hairball graph with stress layout.

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).

```
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_link0(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")
```

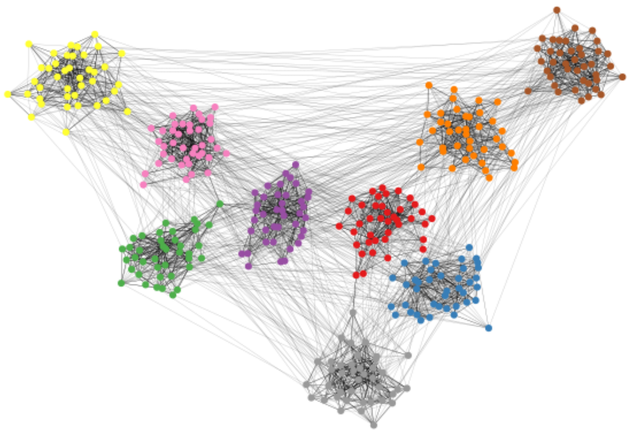

Figure 3: Backbone layout of the graph shown in Figure 2.

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
```

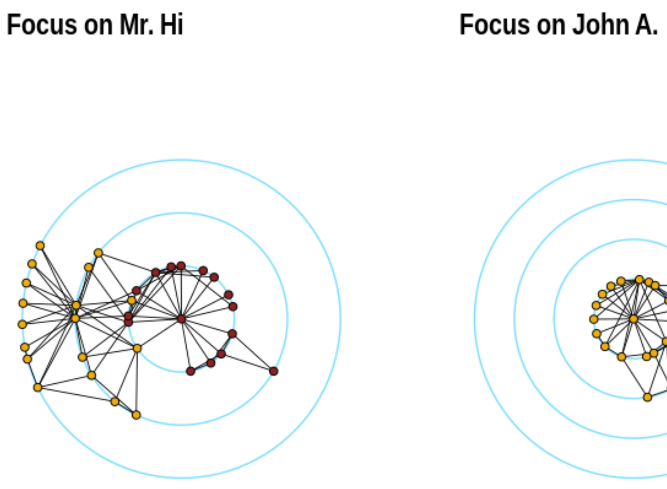

Figure 4: Example of focus layouts.

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).

```
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
```

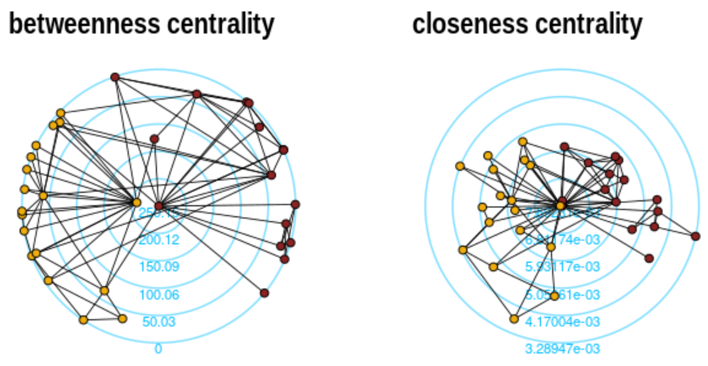

Figure 5: Example of centrality layout using betweenness and closeness.

References

Almende, B., Thieurmel, B., & Robert, T. (2022). *visNetwork: Network visualization using 'vis.js' library*. https://CRAN.R-project.org/package=visNetwork

Brandes, U., & Pich, C. (2010). More flexible radial layout. *Graph Drawing: 17th International Symposium, GD 2009, Chicago, IL, USA, September 22-25, 2009. Revised Papers 17*, 107–118. https://doi.org/10.1007/978-3-642-11805-0_12

Butts, C. T. (2008). Social Network Analysis with SNA. *Journal of Statistical Software*, *24*(6). https://doi.org/10.18637/jss.v024.i06

Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal, Complex Systems*, *1695*(5), 1–9.

Gansner, E. R., Koren, Y., & North, S. (2005). Graph drawing by stress majorization. *Graph Drawing: 12th International Symposium, GD 2004, New York, NY, USA, September 29-October 2, 2004, Revised Selected Papers 12*, 239–250. https://doi.org/10.1007/978-3-540-31843-9_25

Handcock, M. S., Hunter, D. R., Butts, C. T., Goodreau, S. M., & Morris, M. (2008). Statnet: Software tools for the representation, visualization, analysis and simulation of network data. *Journal of Statistical Software*, *24*(1), 1548. https://doi.org/10.18637/jss.v024.i01

Nocaj, A., Ortmann, M., & Brandes, U. (2015). Untangling the hairballs of multi-centered, small-world online social media networks. *Journal of Graph Algorithms and Applications: JGAA*, *19*(2), 595–618. https://doi.org/10.7155/jgaa.00370

Ortmann, M., Klimenta, M., & Brandes, U. (2016). A sparse stress model. *Graph Drawing and Network Visualization: 24th International Symposium, GD 2016, Athens, Greece, September 19-21, 2016, Revised Selected Papers 24*, 18–32. https://doi.org/10.1007/978-3-319-50106-2_2

Pedersen, T. L. (2022). *Ggraph: An implementation of grammar of graphics for graphs and networks*. https://CRAN.R-project.org/package=ggraph