1 help

Draw Network in R (control edge thickness plus non-overlapping edges)

I need to draw a network with 5 nodes and 20 directed edges (an edge connecting each 2 nodes) using R, but I need two features to exist:

1. To be able to control the thickness of each edge.
2. The edges not to be overlapping (i.e.,the edge form A to B is not drawn over the edge from B to A)

I've spent hours looking for a solution, and tried many packages, but there's always a problem.

Can anybody suggest a solution please and provide a complete example as possible?

Many Thanks in advance.

r

asked Sep 22 '11 at 20:54
Pansy
104 2 7
add a comment

4 Answers

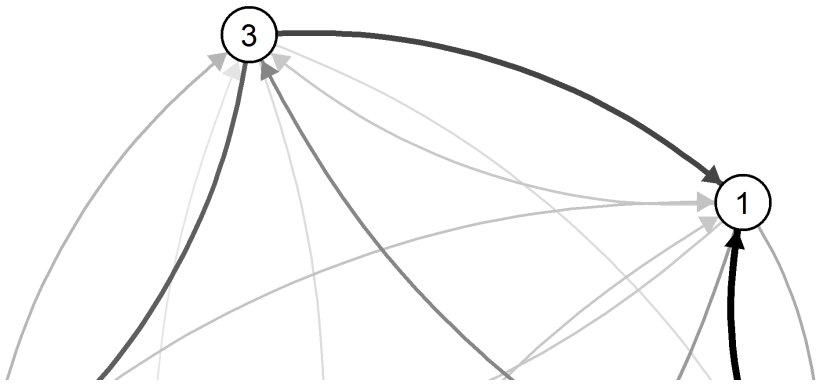
If it is ok for the lines to be curved then I know two ways. First I create an edgelist:

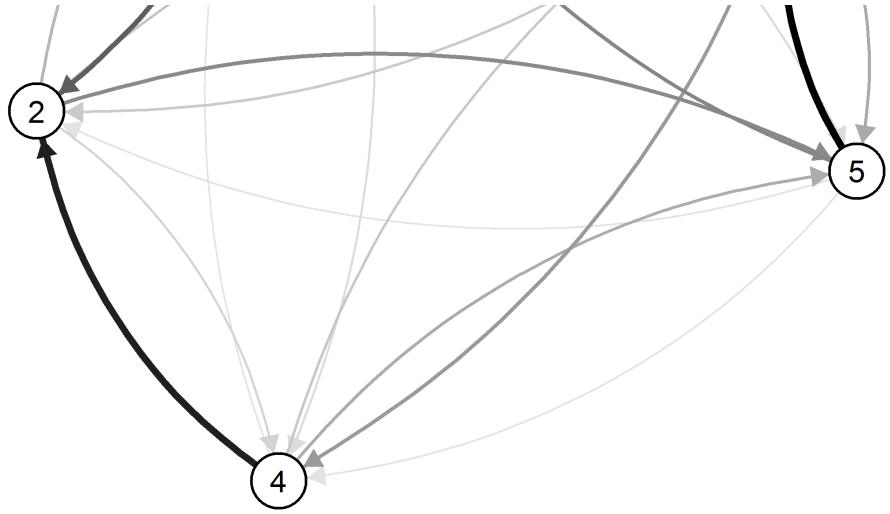
```
Edges <- data.frame(
    from = rep(1:5,each=5),
    to = rep(1:5,times=5),
    thickness = abs(rnorm(25)))

Edges <- subset(Edges,from!=to)
```

This contains the node of origin at the first column, node of destination at the second and weight at the third. You can use my pacake qgraph to plot a weighted graph using this. By default the edges are curved if there are multiple edges between two nodes:

```
library("qgraph")
qgraph(Edges,esize=5,gray=TRUE)
```





However this package is not really intended for this purpose and you can't change the edge colors (yet, working on it:)). You can only make all edges black with a small trick:

```
qgraph(Edges,esize=5,gray=TRUE,minimum=0,cut=.Machine$double.xmin)
```

For more control you can use the igraph package. First we make the graph:

```
library("igraph")
g <- graph.edgelist(as.matrix(Edges[,-3]))
```

Note the conversion to matrix and subtracting one because the first node is 0. Next we define the layout:

```
l <- layout.fruchterman.reingold(g)
```

Now we can change some of the edge parameters with the `E()` function:

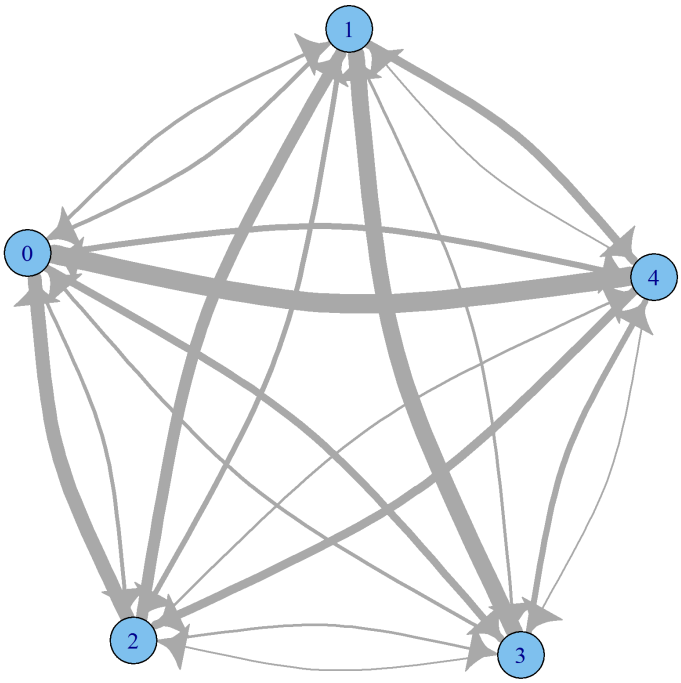
```
# Define edge widths:
E(g)$width <- Edges$thickness * 5

# Define arrow widths:
E(g)$arrow.width <- Edges$thickness * 5

# Make edges curved:
E(g)$curved <- 0.2
```

And finally plot the graph:

```
plot(g,layout=l)
```



edited Oct 3 '12 at 6:05

answered Sep 23 '11 at 11:04

Sacha Epskamp

14.6k 3 32 78

Thanks very much, The 2 solutions are working. But now I found another problem; I'm drawing 2 networks with the same node names but with different edge thickness, the same node has different location in each network, e.g., node1 in the 1st network is drawn on the top left and in the 2nd network is drawn on the top right. I need the nodes to be in the same location so it's easy to visually compare between the 2 networks. Do you know a way to fix this? Thanks – Pansy Sep 24 '11 at 18:30

You can manually define the placement of the nodes by using a fixed layout. In qgraph this can be done as followed: Store the layout of one graph with: `L <- qgraph( [arguments] )$layout` then use it in another graph with `qgraph( [arguments] , layout=L)` . In igraph you can save the layour as I already showed above and just use it again in the second graph. – Sacha Epskamp Sep 25 '11 at 0:03

I'm not sure why but the bit which has the code `g <- graph.edgelist(as.matrix(Edges[,-3]-1))` produces an error like `Error in graph(t(el), directed = directed) : At structure_generators.c:84 : Invalid (negative) vertex id, Invalid vertex id` any idea what the problem is? – h.l.m Oct 3 '12 at 1:23

This is because since this answer igraph changed node numbering starting from 1 now where it was 0. It works if you remove the `-1` . Edited the answer as well. – Sacha Epskamp Oct 3 '12 at 6:06

add a comment

While not an R answer specifically, I would recommend using Cytoscape to generate the network.

You can automate it using a RCytoscape.
http://bioconductor.org/packages/release/bioc/html/RCytoscape.html

answered Sep 22 '11 at 20:57

lain
245 1 2 6

add a comment

The package informatively named 'network' can draw directed networks fairly well, and handle your issues.

```
ex.net <- rbind(c(0, 1, 1, 1), c(1, 0, 0, 1), c(0, 0, 0, 1), c(1, 0, 1, 0))

plot(network(ex.net), usecurve = T, edge.curve = 0.00001,
     edge.lwd = c(4, rep(1, 7)))
```

The edge.curve argument, if set very low and combined with usecurve=T, separates the edges, although there might be a more direct way of doing this, and edge.lwd can take a vector as its argument for different sizes.

It's not always the prettiest result, I admit. But it's fairly easy to get decent looking network plots that can be customized in a number of different ways (see ?network.plot).

answered Sep 22 '11 at 22:38

Wine
196 3

add a comment

The 'non overlapping' constraint on edges is the big problem here. First, your network has to be 'planar' otherwise it's impossible in 2-dimensions (you cant connect three houses to gas, electric, phone company

buildings without crossovers).

I think an algorithm for planar graph layout essentially solves the 4-colour problem. Have fun with that. Heuristics exist, search for planar graph layout, and force-directed, and read Planar Graph Layouts

answered Sep 23 '11 at 11:22

Spacedman
35.2k 3 28 77

OP meant that edges between two nodes should not overlap, not all edges. – Sacha Epskamp Sep 23 '11 at 11:24

Ah yes. I'd just draw one edge and stick arrowheads on the appropriate end or ends. Job done :) – Spacedman Sep 23 '11 at 11:43

add a comment

Not the answer you're looking for? Browse other questions tagged r or ask your own question.