ReproProjectProgress

Lantz & Karubian (2017) Reproducibility Project Progress

My goal for this project to to recreate the social network figure (Figure 2) from Lantz & Karubian (2017) paper titled "Environmental Disturbance increases social connectivity in a passerine bird". - Found paper using Howard Tilton Library database search - Data is provided with the paper (imported to r), but the code is not provided
- Packages igraph, sna and asnipe are used to create social networks - Looking for other papers with similar methods that have posted the code - Having difficulties finding any similar papers that have posted the code -Decided to email Samantha Lantz (samlantz@gmail.com) on 10/5 asking for code -Her email was available on the PLOS ONE website (https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0183144)

Next steps: -Wait for email back -Study igraph and begin practicing

Studying up on igraph (info from https://www.rdocumentation.org/packages/igraph/versions/1.2.5/topics/igraph-package)

- In this project, nodes = color-banded individuals, edges = relationships/birds that were seen or captured together

Creating Graphs

To create small graphs with a given structure probably the graph_from_literal function is easiest. It uses R's formula interface, its manual page contains many examples. Another option is graph, which takes numeric vertex ids directly. graph.atlas creates graph from the Graph Atlas, make_graph can create some special graphs.

To create graphs from field data, graph_from_edgelist, graph_from_data_frame and graph_from_adjacency_matrix are probably the best choices.

The igraph package includes some classic random graphs like the Erdos-Renyi GNP and GNM graphs (sample_gnp, sample_gnm) and some recent popular models, like preferential attachment (sample_pa) and the small-world model (sample_smallworld).

Vertex & Edge IDs

Vertices and edges have numerical vertex ids in igraph. Vertex ids are always consecutive and they start with one. I.e. for a graph with n vertices the vertex ids are between 1 and n. If some operation changes the number of vertices in the graphs, e.g. a subgraph is created via induced_subgraph, then the vertices are renumbered to satisfty this criteria.

The same is true for the edges as well, edge ids are always between one and m, the total number of edges in the graph.

It is often desirable to follow vertices along a number of graph operations, and vertex ids don't allow this because of the renumbering. The solution is to assign attributes to the vertices. These are kept by all operations, if possible.

Attributes

In igraph it is possible to assign attributes to the vertices or edges of a graph, or to the graph itself. igraph provides flexible constructs for selecting a set of vertices or edges based on their attribute values, see vertex_attr, V and E for details.

Some vertex/edge/graph attributes are treated specially. One of them is the 'name' attribute. This is used for printing the graph instead of the numerical ids, if it exists. Vertex names can also be used to specify a vector or set of vertices, in all igraph functions. E.g. degree has a v argument that gives the vertices for which the degree is calculated. This argument can be given as a character vector of vertex names.

Edges can also have a 'name' attribute, and this is treated specially as well. Just like for vertices, edges can also be selected based on their names, e.g. in the delete_edges and other functions.

We note here, that vertex names can also be used to select edges. The form 'from|to', where 'from' and 'to' are vertex names, select a single, possibly directed, edge going from 'from' to 'to'. The two forms can also be mixed in the same edge selector.

Attribute values can be set to any R object, but note that storing the graph in some file formats might result the loss of complex attribute values. All attribute values are preserved if you use save and load to store/retrieve your graphs.

Visualization

igraph provides three different ways for visualization. The first is the plot.igraph function. (Actually you don't need to write plot.igraph, plot is enough. This function uses regular R graphics and can be used with any R device.

The second function is tkplot, which uses a Tk GUI for basic interactive graph manipulation. (Tk is quite resource hungry, so don't try this for very large graphs.)

The third way requires the rgl package and uses OpenGL. See the rglplot function for the details.