Software Tools for Networks

Sébastien Martin

MIT Operations Research Center

January 14, 2016

Real-World Networks in Operations Research

- Energy
  - Power grids
  - Oil and gas pipelines
- Marketing
  - The Internet
  - Social Networks
- Public Sector OR
  - Epidemiologic encounter and movement networks
  - Social networks for policing
  - Road networks for disaster response
- Supply chains
- Transportation
  - Road, rail and airline networks
  - Delivery networks

Road network around MIT (OpenStreetMap)

Web network (Opte Project)

Module summary

- **Focus: Software tools for network analysis**
- Data Wrangling to construct networks in R
- Visualizing networks
- Network Metrics
- Community detection

File Location

- The source file is `4-graphs/script_full.R`
- The skeleton file we will use to code today is in `4-graphs/script_skeleton.R`
- We will use the following CSVs: `data/area_info_full.csv` and `data/2013-05-14_neighborhoods.csv`
- Slides: `4-graphs/presentation.pdf`

Networks in R

- Our network: 05/14/2013 taxi rides
  - Vertices: NYC neighborhoods
  - Directed edge (a, b): at leas one taxi trip from a to b
- Fairly small network
  - 310 nodes
  - 5688 direct edges representing 490,347 taxi trips
- We will use `igraph` R package
  - Popular general-purpose network package
  - Sparse (edge list) representation
  - Many built-in metrics and algorithms
  - efficient: mostly implemented in c
- `network` and `sna` popular R alternatives
- Others: `BGL`(C++), `networkx`(Python), `JGraphT` (java)

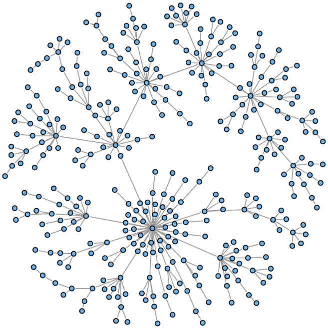
Data wrangling to construct a network

- `graph.data.frame(edges, directed, vertices)`
  - `edges`: Edge data frame, first two columns are endpoints and additional columns are metadata.
  - `vertices`: (Optional) vertext data frame, first column is name and additional columns are metadata
- IGraph also has a lot of other graph generation function (random graphs...)

Visualizing Networks

- Visual representation valuable for understanding networks
- Nodes typically represented by circles
  - Visual properties: size, color, text label
- Edges typically represented by lines
  - Visual properties: width, color, text label, arrowhead
- But where to plot the nodes?
  - From data
  - **Force-based layout**
  - Circular layout
  - ...

Tree graph plotted with `igraph`



Force-directed Graph Drawing

- Treat graph as physical system with opposing forces
  - Edges act as springs, pulling vertices together (Hooke's Law)
  - Vertices repel each other, spreading out graph (Coulomb's Law)
- Optimal vertex positioning is a nonlinear optimization problem
- Simulated annealing often used to optimize system
- Many similar force-directed layout algorithms in `igraph` (`?igraph::layout`)

- `layout.kamada.kawai` if ≤ 100 nodes
- `layout.fruchterman.reingold` for 100-1000 nodes
- `layout.lgl` and `layout.drl` for ≥ 1000 nodes

Network metrics

- Describe structural properties of network
- Vertex metrics
  - `degree(g)`: Number of incident edges (default: in-degree plus out-degree)
  - `closeness(g)`: Inverse of sum of shortest paths to other nodes
  - `betweenness(g)`: Number of shortest paths containing vertex
  - `page.rank(g)$vector`: Score based on score of linked nodes
  - `transitivity(g, ïocal)̈`: Probability pair of neighbors connected
- Edge metrics
  - `edge.betweenness(g)`: Number of shortest path containing edge
  - `degree(g)[get.edges(g,E(g))[,1]]`: Source degree
  - `degree(g)[get.edges(g,E(g))[,2]]`: Sink degree
- Full-network metrics
  - `graph.density(g)`: Proportion of possible edges present
  - `reciprocity(g)`: Proportion of all links that are bidirectional
  - `assortativity.degree(g)`: Correlation of degrees of linked nodes

Graph partitioning and community detection

- *Graph Partitioning*: Prespecified structure
  - Input: Number of groups, size of each
  - Output: Graph partition minimizing edge count between groups
  - Example algorithms: Kerninghan-Lin, Spectral Partitioning
- *Community Detection*: Find "natural" partitioning
  - No fixed group counts or sizes
  - Multiple definitions of "good partitioning"
  - Many algorithms (`igraph` has nine)
- Market segmentation
  - Groups typically cohesive (many links among members)
  - Groups typically don't mix (few links between groups)
- Communities easily detached
  - Useful for epidemiologist performing vaccination
  - Area of concern in telecommunication or transportation networks
- Community can be used in prediction algorithms

Modularity Maximization

- Popular community detection objective: modularity
- $Q = \frac{1}{2m} \sum_{i,j} (A_{i,j} - \frac{k_i k_j}{2m} \delta(c_i, c_j))$
  - m: graph edge count
  - A_{ij}: are i and j joined by an edge (binary)
  - k_i: degree of i
  - $\delta(c_i, c_j)$: are i and j in the same partition (binary)
  - $\frac{k_i k_j}{2m^2}$ probability i and j would be joined by chance
- Nodes connected at above-chance levels should be in same cluster
- Nodes connected at below-chance levels should be in different clusters
- Optimal number of clusters varies by graphics
- Heuristics typically employed to optimize modularity