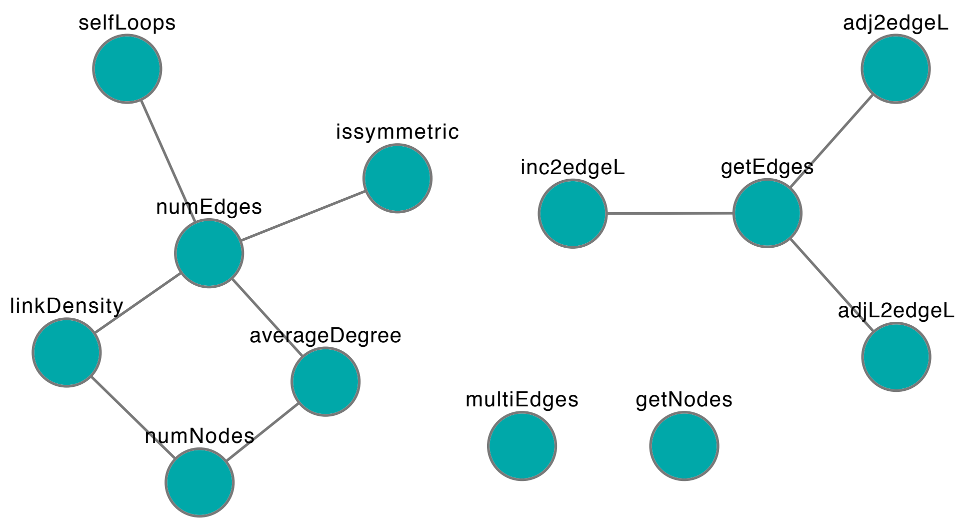
Octave routines for network analysis

GB

September 20, 2012

Contents



0 Basic network routines

0.1 Basic network theory

0.2 Routines

0.2.1 getNodes.m

Returns the list of nodes for varying graph representations.

```
% Returns the list of nodes for varying graph representation types
% Inputs: graph structure (matrix or cell or struct) and type of structure (string)
%       'type' can be: 'adj','edgelist','adjlist' (neighbor list),'inc' (incidence matrix)
% Note 1: only the edge list allows/returns non-consecutive node indexing
% Note 2: no build-in error check for graph structure
%
% Example representations of a directed triangle: 1->2->3->1
%           'adj' - [0 1 0; 0 0 1; 1 0 0]
%           'adjlist' - {1: [2], 2: [3], 3: [1]}
%           'edgelist' - [1 2; 2 3; 3 1] or [1 2 1; 2 3 1; 3 1 1] (1 is the edge weight)
%           'inc' - [-1  0  1
%                     1 -1  0
%                     0  1 -1]
%
% GB: last updated, Sep 18 2012
```

0.2.2 getEdges.m

Returns the list of edges for varying graph representations.

```
% Returns the list of edges for graph varying representation types
% Inputs: graph structure (matrix or cell or struct) and type of structure (string)
% Outputs: edge list, mx3 matrix, where the third column is edge weight
%
% Note 1: 'type' can be: 'adj','edgelist','adjlist' (neighbor list), 'inc' (incidence matrix)
% Note 2: symmetric edges will appear twice, also in undirected graphs, (i.e. [n1,n2] and [n2,n1])
% Other routines used: adj2edgeL.m, adjL2edgeL.m, inc2edgeL.m
%
% Example representations of a directed triangle: 1->2->3->1
%           'adj' - [0 1 0; 0 0 1; 1 0 0]
%           'adjlist' - {1: [2], 2: [3], 3: [1]}
%           'edgelist' - [1 2; 2 3; 3 1] or [1 2 1; 2 3 1; 3 1 1] (1 is the edge weight)
%           'inc' - [-1  0  1
%                     1 -1  0
%                     0  1 -1]
%
% GB: last updated, Sep 18 2012
```

0.2.3 numNodes.m

Number of vertices/nodes in the network.

```
% Returns the number of nodes, given an adjacency list, or adjacency matrix
% INPUTs: adjacency list: {i:j_1,j_2 ..} or adjacency matrix, ex: [0 1; 1 0]
% OUTPUTs: number of nodes, integer
%
% GB: last update Sep 19, 2012

function n = numNodes(adjL)

n = length(adjL);
```

0.2.4 numEdges.m

Number of edges/links in the network.

```
% Returns the total number of edges given the adjacency matrix
% INPUTs: adjacency matrix, nxn
% OUTPUTs: m - total number of edges/links
%
% Note: Valid for both directed and undirected, simple or general graph
% Other routines used: selfloops.m, issymmetric.m
% GB, last updated Sep 19, 2012
```

0.2.5 linkDensity.m

The density of links of the graph. $Density = \frac{m}{n(n-1)/2}$ (n is the number of nodes and m is the number of edges).

```
% Computes the link density of a graph, defined as the number of edges divided by
% number_of_nodes(number_of_nodes-1)/2 where the latter is the maximum possible number of edges.
%
% Inputs: adjacency matrix, nxn
% Outputs: link density, a float between 0 and 1
%
% Note: The graph has to be non-trivial (more than 1 node).
% Other routines used: numNodes.m, numEdges.m
% GB: last update Sep 19, 2012
```

0.2.6 selfLoops.m

Number of selfloops, i.e. nodes connected to themselves.

```
% Counts the number of self-loops in the graph
%
% INPUT: adjacency matrix, nxn
% OUTPUT: integer, number of self-loops
%
% Note: in the adjacency matrix representation loops appear as non-zeros on the diagonal
% GB: last updated, Sep 20 2012
```

0.2.7 multiEdges.m

An edge counts towards the multi-edge total if it shares origin and destination nodes with another edge.

```
% Counts the number of multiple edges in the graph
% Multiple edges here are defined as two or more edges that have the same origin and destination nodes.
% Note 1: This creates a natural difference in counting for undirected and directed graphs.
```

```
%
% INPUT: adjacency matrix, nxn
% OUTPUT: integer, number of multiple edges
%
% Examples: multiEdges([0 2; 2 0])=2, and multiEdges([0 0 1; 2 0 0; 0 1 0])=2
%
% Note 2: The definition of number of multi-arcs (node pairs that have multiple edges across them)
% would be: mA = length(find(adj>1)) (normalized by 2 depending on whether the graph is directed)
%
% GB: last updated, Sep 20 2012
```

0.2.8 averageDegree.m

The average degree (# links) across all nodes. Defined as: $\frac{2m}{n}$, where n is the number of nodes and m is the number of edges. Also, $linkDensity = \frac{averageDegree}{n-1}$.

```
% Computes the average degree of a node in a graph, defined as
% 2 times the number of edges divided by the number of nodes (every edge is counted in degrees twice).
%
% Inputs: adjacency matrix, nxn
% Outputs: float, the average degree, a number between 0 and max(sum(adj))
%
% Note: The average degree is related to the link density, namely:
%       link_density = ave_degree/(n-1), where n is the number of nodes
%
% Other routines used: numNodes.m, numEdges.m
% GB: last update, September 20, 2012
```