Package 'influenceR'

July 22, 2025

Title Software Tools to Quantify Structural Importance of Nodes in a Network

Version 0.1.5

Author Simon Jacobs [aut], Aditya Khanna [aut, cre], Kamesh Madduri [ctb], David Bader [ctb]

Maintainer Aditya Khanna <khanna7.work@gmail.com>

Description Provides functionality to compute various node centrality measures on networks. Included are functions to compute betweenness centrality (by utilizing Madduri and Bader's SNAP library), implementations of constraint and effective network size by Burt (2000) <doi:10.1016/S0191-3085(00)22009-1>; algorithm to identify key players by Borgatti (2006) <doi:10.1007/s10588-006-7084-x>; and the bridging algorithm by Valente and Fujimoto (2010) <doi:10.1016/j.socnet.2010.03.003>. On Unix systems, the betweenness, Key Players, and bridging implementations are parallelized with OpenMP, which may run faster on systems which have OpenMP configured.

Depends R (>= 3.2.0)

License GPL-2

Imports igraph (>= 1.0.1), Matrix (>= 1.1-4), methods, utils

NeedsCompilation yes

Suggests testthat

URL https://github.com/khanna-lab/influenceR

RoxygenNote 7.2.3

Repository CRAN

Date/Publication 2023-05-18 10:00:02 UTC

Contents

`betweenness` *Vertex Betweenness centrality measure.*

Description

The Betweenness centrality score of a node u is the sum over all pairs s,t of the proportion of shortest paths between s and t that pass through u. This function allows the use of either the SNAP betweenness implementation (default), or the igraph betweenness function. The SNAP version makes use of OpenMP for parallelization, and may be faster in some circumstances.

Usage

```
betweenness(g, snap = T)
```

Arguments

`g`	The igraph object to analyze
`snap`	True to use the SNAP betweenness code, False to use igraph::betweenness

Value

A numeric vector with the betweenness centrality score for each vertex

References

https://snap-graph.sourceforge.net/

Examples

```
ig.ex <- igraph::erdos.renyi.game(100, p.or.m=0.3) # generate an undirected 'igraph' object
betweenness(ig.ex) # betweenness scores for each node in the graph
```

`bridging` *Valente's Bridging vertex measure.*

Description

Edges that reduces distances in a network are important structural bridges. Here we implement Valente and Fujimoto's metric, where a node's bridging score is the average decrease in cohesiveness if each of its edges were removed from the graph.

Usage

```
bridging(g)
```

Arguments

g	The igraph object to analyze.

Value

A numeric vector with the bridging score for each vertex

References

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2889704/

Examples

```
ig.ex <- igraph::erdos.renyi.game(100, p.or.m=0.3) # generate an undirected 'igraph' object
bridging(ig.ex) # bridging scores for each node in the graph
```

---

constraint *Burt's Constraint Index.*

Description

The igraph package provides an implementation of Constraint; this is an alternate implementation.

Usage

```
constraint(g, v = igraph::V(g))
```

Arguments

g	The igraph object to analyze.
v	vertices over which to compute constraint (default to all)

Value

A numeric vector with the constraint score for each vertex in v

Examples

```
ig.ex <- igraph::erdos.renyi.game(100, p.or.m=0.3) # generate an undirected 'igraph' object
constraint(ig.ex) # constraint scores for each node in the graph
```

csv.to.igraph *Convert a CSV file to an igraph graph object.*

Description

The first column should be sources, the second should be targets.

Usage

```
csv.to.igraph(fname)
```

Arguments

fname	A filename

Value

An igraph graph object built from the filename.

Examples

```
## Not run: ig.csv <- csv.to.igraph("edgelist.csv")
```

ens *Burt's Effective Network Size and Constraint index. The next two functions below provide ways to measure the actors' access to structural holes in a network. Structural holes "provide opportunities to broker connections between people" (Burt 2008).*

Description

Burt's Effective Network Size and Constraint index. The next two functions below provide ways to measure the actors' access to structural holes in a network. Structural holes "provide opportunities to broker connections between people" (Burt 2008).

Usage

```
ens(g)
```

Arguments

g	The igraph object to analyze.

Value

A numeric vector with the effective network size for each vertex

References

https://www.sciencedirect.com/science/article/abs/pii/S0378873397000038

Examples

```
ig.ex <- igraph::erdos.renyi.game(100, p.or.m=0.3) # generate an undirected 'igraph' object
ens(ig.ex) # Effective Network Size scores for each node in the graph
```

`influenceR` *influenceR: Software tools to quantify structural importance of nodes in a network.*

Description

The influenceR package includes functions to quantify the structural importance of nodes in a network. Algorithms include Betweenness Centrality, Bridging, Constraint Index, Effective Network Size, and Key Players. Currently, algorithms are only guaranteed to work on undirected graphs; work on directed graphs is in progress. These functions run on graph objects from the igraph package.

Details

In addition to igraph, this package makes use of the SNAP framework for a high-performance graph data structure and an OpenMP-parallelized implementation of Betweenness Centrality. See https://snap-graph.sourceforge.net

Funding

Development of this software package was supported by NIH grant R01 DA033875.

References

The website and source code is located at https://github.com/khanna-lab/influenceR.

`keyplayer` *Compute a KPP-Pos set for a given graph.*

Description

The "Key Player" family of node importance algorithms (Borgatti 2006) involves the selection of a metric of node importance and a combinatorial optimization strategy to choose the set S of vertices of size k that maximize that metric.

Usage

```
keyplayer(g, k, prob = 0, tol = 1e-04, maxsec = 120, roundsec = 30)
```

Arguments

g	The igraph object to analyze.
k	The size of the KP-set
prob	probability of accepting a state with a lower value
tol	tolerance within which to stop the optimization and accept the current value
maxsec	The total computation budget for the optimization, in seconds
roundsec	Number of seconds in between synchronizing workers' answer

Details

This function implements KPP-Pos, a metric intended to identify k nodes which optimize resource diffusion through the network. We sum over all vertices not in S the reciprocal of the shortest distance to a vertex in S.

According to Borgatti, a number of off-the-shelf optimization algorithms may be suitable to find S, such as tabu-search, K-L, simulated annealing, or genetic algorithms. He presents a simple greedy algorithm, which we excerpt here:

1. Select k nodes at random to populate set S
2. Set F = fit using appropriate key player metric.
3. For each node u in S and each node v not in S:
   - DELTAF = improvement in fit if u and v were swapped
4. Select pair with largest DELTAF
   - If DELTAF <= [tolerance] then terminate
   - Else, swap pair with greatest improvement in fit and set F = F + DELTAF
5. Go to step 3.

Our implementation uses a different optimization method which we call stochastic gradient descent. In tests on real world data, we found that our method discovered sets S with larger fits in less computation time. The algorithm is as follows:

1. Select k nodes at random to populate set S
2. Set F = fit using appropriate key player metric (KPP-Pos in our case)
3. Get a new state:
   - Pick a random u in S and v not in S.
   - F' = fit if u and v were swapped
   - If F' > F, swap u and v in S. Else, repeat step 3. (Alternatively, if a positive value is given for the 'prob' parameter, a swap will be accepted with a small probability regardless of whether it improves the fit).
4. If F' - F < tolerance or our maximum computation time is exceeded, return S. Else, go to step 3.

This implementation uses OpenMP (if available on the host system) so that multiple workers can explore the solution space in parallel. After a given of time, the workers synchronize their sets S to the one which maximizes the metric.

Value

a vector with the vertex number of each vertex in the selected set S.

References

https://link.springer.com/article/10.1007/s10588-006-7084-x

Examples

```
ig.ex <- igraph::erdos.renyi.game(100, p.or.m=0.3) # generate an undirected 'igraph' object
keyplayer(ig.ex, k=10) # key-player set consisting of 10 actors
```

Index