Package 'targbetween'

May 11, 2020

Title Targeted Betweenness Centrality

Version 0.9.0

Description This package provides several functions to compute targeted betweenness centrality for igraph-formatted network and specified pairs of vertices. If you use this package, please cite the original article: Britt, B. C., Hayes, J. L., Musaev, A., Sheinidashtegol, P., Parrott, M. S., & Albright, D. (under review). Targeted betweenness centrality: An approach to identify bridges between specified network alters. Manuscript submitted to Social Networks.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports igraph

RoxygenNote 7.1.0

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Brian Britt [aut, cre] (<https://orcid.org/0000-0002-1521-3258>)

Maintainer Brian Britt <bcbritt@ua.edu>

R topics documented:

assorted_dyads *Retrieve Shortest Paths for Assorted Dyads*

Description

This helper function retrieves the set of shortest paths connecting each dyad of vertices specified in a two-column data.frame (column 1 = source, column 2 = destination), for subsequent use by `compute_targbetween`.

Usage

```
assorted_dyads(network, dyads, directed = FALSE)
```

Arguments

`network`	An `igraph` network graph
`dyads`	A two-column data.frame, with two vertices in each row, indicating the dyads to be considered in the analysis
`directed`	A boolean value indicating whether the `network` should be treated as directed (default = `FALSE`)

Details

This function should not be called directly. Instead, provide the `network` and the set of `dyads` that you want to analyze to `targbetween`. That function calls `assorted_dyads` as needed.

WARNING: This function may be extremely slow when the number of rows in `dyads` is large, as `all_shortest_paths` must be separately run for each individual dyad. If all source vertices correspond to the same set of `destination` vertices, `targbetween_alldyads` should be used, as this calls `one_to_many`, instead of using `targbetween` to call `assorted_dyads`. The `targbetween_alldyads` function takes a vector of `source` vertices and a vector of `destination` vertices as arguments, and it is either `length(source)` or `length(destination)` times faster than `targbetween`, whichever is larger.

Value

A list of shortest paths between the pairs of vertices indicated in each row of `dyads`.

compute_targbetween *Compute Targeted Betweenness Centrality from Shortest Paths*

Description

This helper function takes the list of shortest paths, as retrieved by either `one_to_many` or `assorted_dyads`, and uses it to finish the computation of targeted betwenness centrality.

Usage

```
compute_targbetween(network, paths, num_dyads, update = 0)
```

Arguments

network	An igraph network graph
paths	Shortest paths retrieved by all_shortest_paths
num_dyads	Number of dyads used in the analysis that generated paths
update	A numeric value indicating how many loop iterations should elapse between progress updates (default = 0, which suppresses output)

Details

This function should not be called directly. Instead, provide the network and the set of dyads that you want to analyze to targbetween_alldyads or targbetween. Each of those functions calls compute_targbetween as needed.

Value

A data.frame with the results for targeted betweenness centrality

---

one_to_many *Retrieve Shortest Paths from One Source to Multiple Destinations*

Description

This helper function retrieves the set of shortest paths connecting all possible dyads that can be formed from a single source vertex to a vector of destination vertices, for subsequent use by compute_targbetween.

Usage

```
one_to_many(network, source, destination, mode)
```

Arguments

network	An igraph network graph
source	A single vertex or a vector of vertices in network
destination	A vector of one or more vertices in network
mode	A character value indicating the mode for all_shortest_paths ("in", "out", or "all")

Details

This function should not be called directly. Instead, provide the network and the set of dyads that you want to analyze to targbetween_alldyads. That function calls one_to_many as needed.

Note that other functions that call one_to_many may reverse source and destination in order to identify the shortest paths from many source vertices to a single destination vertex. When this is done for directed networks, the mode argument is set to "in" rather than "out" as its value. (If the network is treated as undirected in the analysis, mode = "all" regardless of whether source and destination are reversed.)

Value

A list of shortest paths from the source vertex to each of the destination vertices.

---

targbetween *Targeted Betweenness Centrality for Specified Dyads*

Description

This function computes targeted betweenness centrality for a specified set of dyads. targbetween is more flexible than targbetween_alldyads, allowing different source vertices to correspond to different destination vertices.

Usage

```
targbetween(network, dyads, filename = NULL, directed = FALSE, update = 0)
```

Arguments

network	An igraph network graph
dyads	A two-column data.frame, with two vertices in each row, indicating the dyads to be considered in the analysis
filename	A character value indicating the file location to output results (default = NULL)
directed	A boolean value indicating whether the network should be treated as directed (default = FALSE)
update	A numeric value indicating how many loop iterations should elapse between progress updates (default = 0, which suppresses output)

Details

WARNING: This function calls assorted_dyads, which may be extremely slow when the number of rows in dyads is large, as all_shortest_paths must be separately run for each individual dyad. If all source vertices correspond to the same set of destination vertices, targbetween_alldyads should be used instead of targbetween. The targbetween_alldyads function takes a vector of source vertices and a vector of destination vertices as arguments, and it is either length(source) or length(destination) times faster than targbetween, whichever is larger.

Value

A data.frame with the results for targeted betweenness centrality

Examples

```
my_network <- igraph::erdos.renyi.game(20, 0.5, directed = FALSE)
various_dyads <- cbind(c(1,1,2,2,3,3,4,4,5,5),c(3,4,1,8,18,20,1,15,3,10))
results <- targbetween(my_network, various_dyads, directed=FALSE)
results
#       vertices targeted_betweenness
#  [1,] "1"      "0.2"
#  [2,] "2"      "0"
#  [3,] "3"      "0.166666666666667"
```

```
#  [4,] "4"      "0"
#  [5,] "5"      "0.2"
#  [6,] "6"      "0.166666666666667"
#  [7,] "7"      "0.166666666666667"
#  [8,] "8"      "0.166666666666667"
#  [9,] "9"      "0"
# [10,] "10"     "0.166666666666667"
# [11,] "11"     "0.166666666666667"
# [12,] "12"     "0.166666666666667"
# [13,] "13"     "0.166666666666667"
# [14,] "14"     "0.166666666666667"
# [15,] "15"     "0.366666666666667"
# [16,] "16"     "0.166666666666667"
# [17,] "17"     "0"
# [18,] "18"     "0"
# [19,] "19"     "0.2"
# [20,] "20"     "0.366666666666667"
```

`targbetween_alldyads` *Targeted Betweenness Centrality for Specified Source and Destination Vertices*

Description

This function computes targeted betweenness centrality for one or more sources and one or more destinations. The set of dyads to be evaluated includes all possible pairs of source vertices and destination vertices from the specified `source` and `destination` vectors.

Usage

```
targbetween_alldyads(
  network,
  source,
  destination,
  filename = NULL,
  directed = FALSE,
  update = 0
)
```

Arguments

`network`	An `igraph` network graph
`source`	A vector of one or more vertices in `network`
`destination`	A vector of one or more vertices in `network`
`filename`	A character value indicating the file location to output results (default = `NULL`)
`directed`	A boolean value indicating whether the `network` should be treated as directed (default = `FALSE`)
`update`	A numeric value indicating how many loop iterations should elapse between progress updates (default = `0`, which suppresses output)

Value

A data.frame with the results for targeted betweenness centrality

Examples

```
my_network <- igraph::erdos.renyi.game(20, 0.5, directed = FALSE)
single_source <- igraph::V(my_network)[1]
single_destination <- igraph::V(my_network)[20]
multiple_sources <- igraph::V(my_network)[1:5]
multiple_destinations <- igraph::V(my_network)[6:10]
results1 <- targbetween_alldyads(my_network, single_source,
  single_destination)
results2 <- targbetween_alldyads(my_network, single_source,
  multiple_destinations, directed=FALSE)
results3 <- targbetween_alldyads(my_network, multiple_sources,
  single_destination, directed=TRUE)
results4 <- targbetween_alldyads(my_network, multiple_sources,
  multiple_destinations, directed=FALSE, update=1000)
results4
#       vertices targeted_betweenness
#  [1,] "1"      "0.533333333333333"
#  [2,] "2"      "0.866666666666667"
#  [3,] "3"      "0.5"
#  [4,] "4"      "0.291666666666667"
#  [5,] "5"      "1.11666666666667"
#  [6,] "6"      "1.86666666666667"
#  [7,] "7"      "0.541666666666667"
#  [8,] "8"      "0.291666666666667"
#  [9,] "9"      "0"
# [10,] "10"     "1.36666666666667"
# [11,] "11"     "0.991666666666667"
# [12,] "12"     "2.03333333333333"
# [13,] "13"     "0.366666666666667"
# [14,] "14"     "0.458333333333333"
# [15,] "15"     "1.15833333333333"
# [16,] "16"     "0.75"
# [17,] "17"     "0"
# [18,] "18"     "1.20833333333333"
# [19,] "19"     "0.866666666666667"
# [20,] "20"     "0.791666666666667"
```