CliquePercolation: An R Package for conducting and visualizing results of the clique percolation network community detection algorithm

Jens Lange[1]

1 University of Hamburg

DOI: 10.21105/joss.03210

Software

- Review
- Repository
- Archive

Editor: Karthik Ram

Reviewers:

- @donaldRwilliams
- @AlexChristensen

Submitted: 06 January 2021
Published: 31 May 2021

Summary and Statement of Need

Modeling complex phenomena as networks constitutes one – if not the most – versatile field of research (Barabási, 2011). Indeed, many interconnected entities can be represented as networks, in which entities are called *nodes* and their connections are called *edges*. For instance, networks can represent friendships between people, hyperlinks between web pages, or correlations between questionnaire items. One structural characteristic of networks that is investigated frequently across various sciences is the detection of *communities* (Fortunato, 2010). Communities are strongly connected subgraphs in the network such as groups of friends, thematic fields, or latent factors. Most community detection algorithms thereby put each node in only one community. However, nodes are often shared by multiple communities, e.g., when a person is part of multiple groups of friends, web pages belong to different thematic fields, or items load on multiple factors. The most popular community detection algorithm that is aimed at identifying such overlapping communities is the *clique percolation algorithm* (Farkas et al., 2007; Palla et al., 2005).

The clique percolation algorithm is not yet implemented in a package in **R** (R Core Team, 2020). So far, the primary software for running the algorithm is the standalone program **CFinder**, written in **C++** and **Java** (Adamcsek et al., 2006). However, **CFinder** cannot be used to construct networks from data or to visualize the solutions of the algorithm, requiring the simultaneous use of other software such as **R**. Handling multiple programs impedes a smooth workflow. Next to **CFinder**, an **R** function for running one variant of the clique percolation algorithm is available in a GitHub repository. However, it is not implemented in a package and lacks functions for optimizing parameters of the algorithm as well as plotting its results. **CliquePercolation** overcomes these limitations as it entails functions for helping to optimize parameters of the algorithm, running the algorithm, and plotting the results.

A minimal example

The structure of a network can be captured in a matrix. An undirected network of n nodes translates into a symmetric square n-by-n matrix. Each element a_{ij} takes the value 0, if there is no edge between nodes i and j. If there is an edge, in a *unweighted* network, a_{ij} takes the value 1, and in a *weighted* network, it takes any non-zero value. The **R** package **qgraph** (Epskamp et al., 2012) can visualize such networks. For instance, a weighted network with eight nodes a to h as depicted in Figure 1 results from running

```
library(qgraph) #version 1.6.5
W <- matrix(c(0 , .1, .3, .3,  0,  0,   0,   0,
                .1,  0, .2, .2,  0,  0,   0,   0,
                .3, .2,  0,  0,  0,  0,   0,   0,
                .3, .2,  0,  0, .1, .1,  .1,   0,
                 0,  0,  0, .1,  0, .1,  .1,   0,
                 0,  0,  0, .1, .1,  0,  .1,   0,
                 0,  0,  0, .1, .1, .1,   0, -.2,
                 0,  0,  0,  0,  0,  0, -.2,   0),
              nrow = 8, ncol = 8, byrow = TRUE)
rownames(W) <- letters[seq(from = 1, to = nrow(W))]
colnames(W) <- letters[seq(from = 1, to = nrow(W))]
W <- qgraph::qgraph(W,
                    theme = "colorblind",
                    cut = 0.02,
                    edge.labels = TRUE)
```

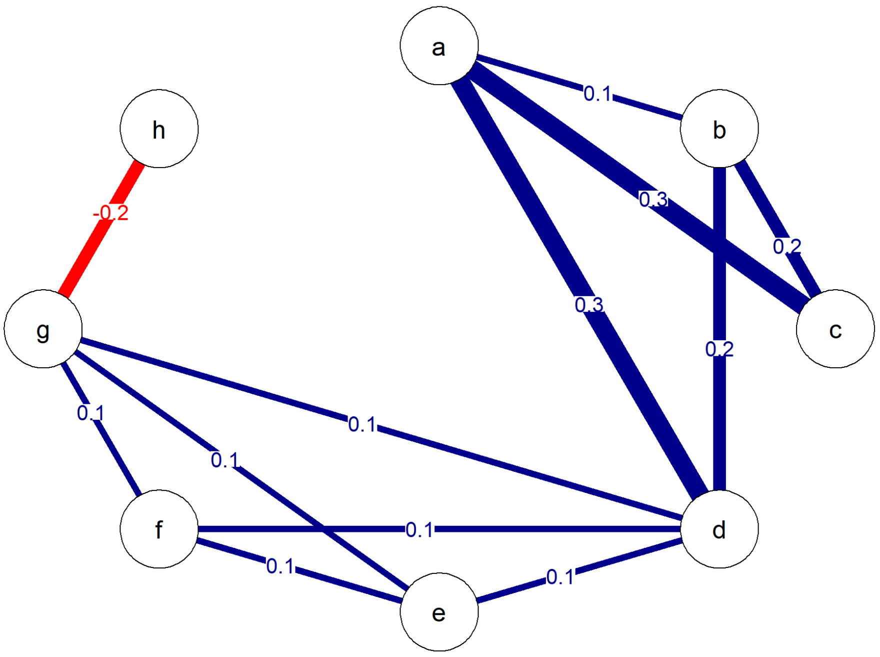

Figure 1: Weighted network with eight nodes.

The clique percolation algorithm proceeds in two steps. First, it identifies k-cliques in the network, i.e., fully conntected subgraphs with k nodes, when the geometric mean of their edge weights exceeds the Intensity threshold I. Second, communities are defined as sets of adjacent k-cliques, i.e., k-cliques that share $k-1$ nodes, allowing some nodes to be shared by communities or to be isolated.

The package **CliquePercolation** facilitates executing these steps. First, it helps identifying optimal values for k and I. For very small networks (as in Figure 1), the entropy of the community partition should be maximized (treating isolated nodes as a separate community).

$$Entropy = -\sum_{i=1}^{N} p_i * \log_2 p_i \tag{1}$$

where N is the number of communities and p_i is the probability of being in community i. Entropy is maximal when the resulting communities are equally sized with a small number of isolated nodes. A permutation test, which repeatedly randomly shuffles the edges in the network and recalculates entropy can point out which entropy values are higher than already expected by chance.

In **CliquePercolation** the cpThreshold function calculates entropy for a range of k and I values.

```
library(CliquePercolation) #version 0.3.0
threshold <- cpThreshold(W, method = "weighted",
                         k.range = c(3,4),
                         I.range = seq(0.3, 0.09, -0.01),
                         threshold = "entropy")
```

and the cpPermuteEntropy function runs the permutation test

```
thresholds.permute <- cpPermuteEntropy(W, cpThreshold.object = threshold, seed = 4186)
```

returning the combinations of k and I that are more surprising than chance

```
thresholds.permute
#>
#> Confidence intervals for entropy values of random permutations of original network
#>
#> --------------------
#>
#> User-specified Settings
#>
#> n = 100
#> interval = 0.95
#> CFinder = FALSE
#> ncores = 2
#> seed = 4186
#>
#>
#> --------------------
#>
#> Confidence intervals
#>
#>  k 95% CI lower 95% CI upper
#>  3        1.166        1.299
#>  4        0.113        0.267
#>
#>
#> --------------------
#>
#> Extracted rows from cpThreshold object
#>
#>  k Intensity Number.of.Communities Number.of.Isolated.Nodes Entropy.Threshold
#>  3      0.09                     2                        1             1.419
#>  4      0.09                     1                        4             1.000
```

The highest entropy results for $k = 3$ and $I = 0.09$, which can be used to run the clique percolation algorithm for weighted networks with the `cpAlgorithm` function.

```
cp.k3I.09 <- cpAlgorithm(W, k = 3, method = "weighted", I = 0.09)

cp.k3I.09
#>
#> Results of clique percolation community detection algorithm
#>
#> --------------------
#>
#> User-specified Settings
#>
#> method = weighted
#> k = 3
#> I = 0.09
#>
#> --------------------
#>
#> Results
#>
#> Number of communities: 2
#> Number of shared nodes: 1
#> Number of isolated nodes: 1
#>
#> --------------------
#>
#> For details, use summary() (see ?summary.cpAlgorithm).

summary(cp.k3I.09)
#>
#> --------------------
#> Communities (labels as identifiers of nodes)
#> --------------------
#>
#> Community 1 : d e f g
#> Community 2 : a b c d
#>
#>
#> --------------------
#> Shared nodes (labels as identifiers of nodes)
#> --------------------
#>
#> d
#>
#>
#> --------------------
#> Isolated nodes (labels as identifiers of nodes)
#> --------------------
#>
#> h
```

The algorithm identified two communities with one shared node and one isolated node. Hence, the two adjacent 3-cliques a–b–c and a–b–d form a community and the four adjacent 3-cliques

d–e–f, d–e–g, d–f–g, and e–f–g form another community, leading node d to be shared between both communities and node h to be isolated.

The function `cpColoredGraph` can visualize the results. For instance, using the default color scheme, all nodes that belong to the same community get the same color, shared nodes are split in multiple parts with colors for each community they belong to, and isolated nodes are white (see Figure 2).

```r
col_graph <-
  cpColoredGraph(W,
                 list.of.communities = cp.k3I.09$list.of.communities.labels,
                 theme = "colorblind",
                 cut = 0.02,
                 edge.labels = TRUE)
```

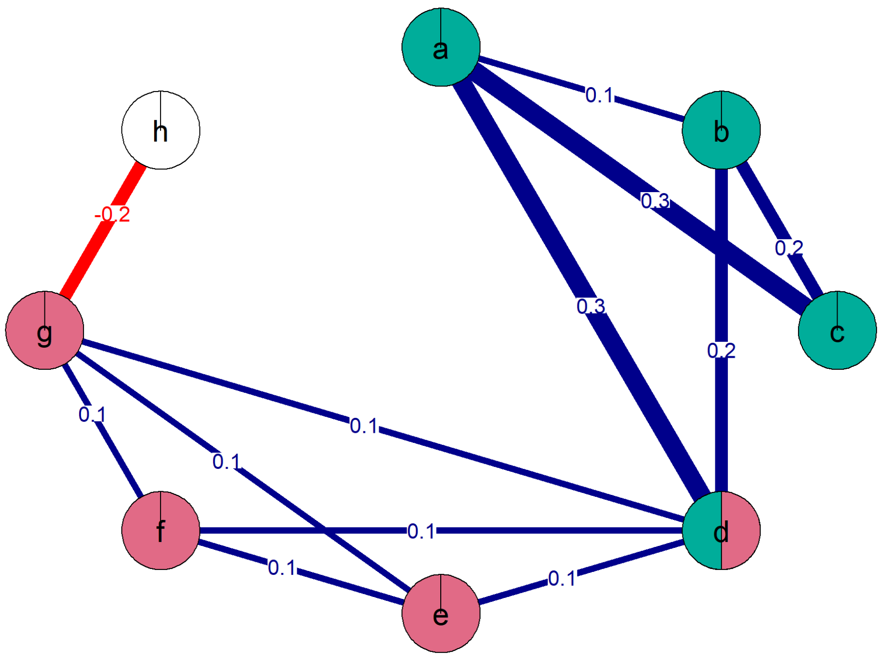

Figure 2: Results of clique percolation algorithm.

Beyond this minimal example, the **CliquePercolation** package provides more functionality for applying the clique percolation algorithm to different kinds of networks and plotting the results. The full suite of possibilities is described in the package vignette, which is available by running `vignette("CliquePercolation")`. Moreover, an elaborate blog post used the package in research on a psychological disorder network and a recent publication applied the package in research on emotions (Lange & Zickfeld, 2021).

Acknowledgements

The development of **CliquePercolation** greatly benefited from discussions with Janis Zickfeld and Tessa Blanken. Moreover, Gergely Palla provided important insights into the clique percolation algorithm as implemented in **CFinder** and recommended the entropy-based method. The development of **CliquePercolation** was also supported by two grants from the German Research Foundation (Deutsche Forschungsgemeinschaft; LA 4029/1-1; LA 4029/2-1).

References

Adamcsek, B., Palla, G., Farkas, I., Derényi, I., & Vicsek, T. (2006). CFinder: Locating cliques and overlapping modules in biological networks. *Bioinformatics*, *22*(8), 1021–1023. https://doi.org/10.1093/bioinformatics/btl039

Barabási, A. L. (2011). The network takeover. *Nature Physics*, *8*(1), 14–16. https://doi.org/10.1038/nphys2188

Epskamp, S., Cramer, A. O. J., Waldorp, L. J., Schmittmann, V. D., & Borsboom, D. (2012). qgraph: Network visualizations of relationships in psychometric data. *Journal of Statistical Software*, *48*(4), 1–18. https://doi.org/10.18637/jss.v048.i04

Farkas, I., Ábel, D., Palla, G., & Vicsek, T. (2007). Weighted network modules. *New Journal of Physics*, *9*(6), 1–18. https://doi.org/10.1088/1367-2630/9/6/180

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, *486*(3-5), 75–174. https://doi.org/10.1016/j.physrep.2009.11.002

Lange, J., & Zickfeld, J. H. (2021). Emotions as overlapping causal networks of emotion components: Implications and methodological approaches. *Emotion Review*, *13*(2), 157–167. https://doi.org/10.1177/1754073920988787

Palla, G., Derényi, I., Farkas, I., & Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, *435*(7043), 814–818. https://doi.org/10.1038/nature03607

R Core Team. (2020). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/