Finding and assessing community structure

Argimiro Arratia & Ramon Ferrer-i-Cancho
(with collaboration by Martí Renedo)

Complex and Social Networks (2022-2023)
Master in Innovation and Research in Informatics (MIRI)

1 Introduction to igraph's community detection algorithms

In this session you will run and compare different community finding algorithms. The following R packages will be used: igraph, igraphdata, clustAnalytics. The package igraphdata contains various graphs in igraph format. After loading the package type in `data(package="igraphdata")` to get a list of data sets included in this package. The package clustAnalytics contains various methods to assess the quality of clusterings.

In the igraph package there are a few clustering algorithms already implemented, including some we have seen in theory class:

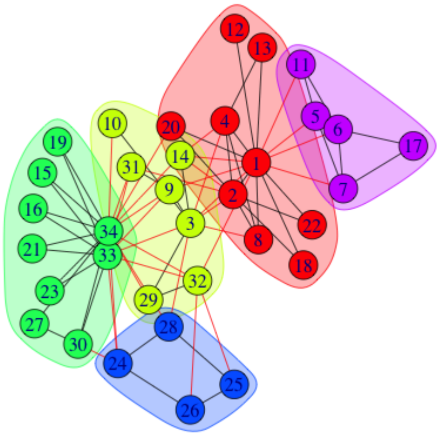
- `edge.betweenness.community` [Newman and Girvan, 2004]
- `fastgreedy.community` [Clauset et al., 2004]: modularity greedy optimization method.
- `label.propagation.community` [Raghavan et al., 2007]: This is a fast, nearly linear time algorithm for detecting community structure in networks. It works by labeling the vertices with unique labels and then updating the labels by majority voting in the neighborhood of the vertex.
- `leading.eigenvector.community` [Newman, 2006]
- `multilevel.community` [Blondel et al., 2008]: This is a multi-level modularity optimization algorithm (the Louvain method).
- `optimal.community` [Brandes et al., 2008]: Works by maximizing the modularity measure over all possible partitions.
- `spinglass.community` [Reichardt and Bornholdt, 2006]: tries to find communities in graphs via a spin-glass model and simulated annealing.

- `walktrap.community` [Pons and Latapy, 2005]: tries to find densely connected subgraphs (communities) in a graph via random walks. The idea is that short random walks tend to stay in the same community.

- `infomap.community` [Rosvall and Bergstrom, 2008]: Find community structure that minimizes the expected description length of a random walker trajectory.

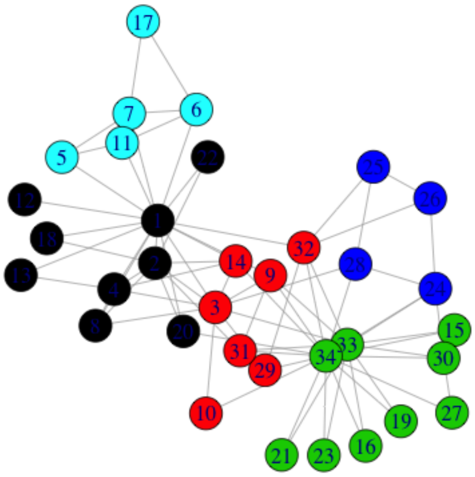
All of these methods return a `communities` object, which you can then use to explore, plot, and compute metrics on. As an example, consider the following snippet of code:

```
> library(igraph)
> library(clustAnalytics)
> data(karate,package="igraphdata")
> wc <- walktrap.community(karate)
> modularity(wc)
[1] 0.3532216
> unname(membership(wc))      ## try membership(wc)
 [1] 1 1 2 1 5 5 5 1 2 2 5 1 1 2 3 3 5 1 3 1 3 1 3 4 4 4 3 4 2 3 2 2 3 3
> plot(wc, karate)
```



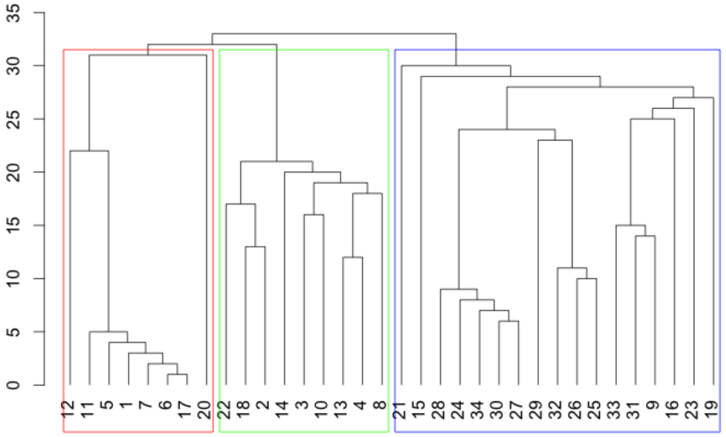
An alternative way of plotting communities without the shaded regions is:

```
> plot(karate, vertex.color=membership(wc))
```



For those algorithms that output communities with hierarchical structure, this information can be visualized using the `dendPlot` function, which displays the corresponding dendogram:

```
> data(karate,package="igraphdata")
> fc <- fastgreedy.community(karate)
> dendPlot(fc)
```



Sometimes it is useful to work with the adjacency matrix associated to the network:

```
> as_adjacency_matrix(as.undirected(karate,mode = "each"))
```

Note that for a multigraph G (a graph with possibly more than one edge between pairs of nodes) its adjacency matrix corresponds to the weighted simple graph with same vertex set $V(G)$ and where weights on each edge corresponds to number of edges in the multigraph. See this by running previous code with multigraph `UKfaculty` in `igraphdata`.

In the package `clustAnalytics` we have the function `evaluate_significance` which takes a graph and a list of clustering algorithms as arguments, applies these algorithms to the graph and evaluates significance of the clusterings through various scoring functions (all seen in class). It returns a table with the scores for each algorithm. If the graph has a known *ground truth* community structure (such as the factions in the karate club), we can set `gt_clustering` as the membership vector to evaluate it and compare it to the results of the clustering algorithms. In our `karate` graph the ground truth is available with `V(karate)$Faction`.

```
> evaluate_significance(karate,alg_list=list(Louvain=cluster_louvain,
+                          "label prop"= cluster_label_prop,
+                           walktrap=cluster_walktrap),
+                        gt_clustering=V(karate)$Faction)
```

Finally, in the analysis of clustering algorithms it is useful to generate controlled examples of networks with communities (synthetic ground truth models). The following creates a graph of order 100 and size 400 with two communities, with the added feature of having scale-free degree distribution, as it is constructed following the Barabasi-Albert preferential attachment method:

```
> B <- matrix(c(1, 0.2, 0.2, 1), ncol=2)
> G <- barabasi_albert_blocks(m=4, p=c(0.5, 0.5), B=B, t_max=100,
                     type="Hajek", sample_with_replacement = FALSE)
> plot(G, vertex.color=(V(G)$label),vertex.label=NA,vertex.size=10)
```

More information on how to use the `clustAnalytics` functions can be found in the Help and in [Arratia and Renedo-Mirambell, 2022].

2 Tasks

We shall design a criteria for evaluating clustering quality based on comparing against a *reference clustering*, that is a clustering for which we have certain guarantees of its significance: either because it is the ground truth or the clustering with best scores from a tandem of significance scoring functions. The criteria will rely on the Jaccard index to quantify two clusterings similarity. Given two sets A and B (in this task these will be parts of two different partitions), their **Jaccard index** is:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

By definition $0 \leq J(A,B) \leq 1$ and the closer to 1 the more similar that A and B are.

1. Write an `R` function that, given two different clusterings of the same network, it outputs a table of the Jaccard index between each of their clusters (i.e. a table that includes the Jaccard index between each cluster of labeling 1 and each cluster of labeling 2).
2. Write a function that for each cluster of labeling 1, identifies which is the cluster of labeling 2 that is more similar according to the Jaccard index, and returns all these indices indicating which clusters they correspond to.
3. Write a function that computes the weighted mean of the vector of values output by previous function (weights given by fraction of number of nodes in each cluster). This quantity we will consider as the (global) Jaccard similarity of the two clusterings. Why is this a reasonable similarity among clusterings (say, as opposed to the mean value)? Can you think of another way of combining the vector of Jaccard indices obtained in 2. to quantify clusterings similarity?

The following snippet of code illustrates what we expect. Assume we have a network and two different clusterings: one is the ground truth (`memb_GT`) with 4 clusters, and the other produced by Louvain algorithm (`memb_louvain`) with 6 clusters:

```
> JS <- jaccard_sim(memb_GT, memb_louvain)
> JS
            1          2          3          4          5          6
3  0.52941176 0.04054054 0.09722222 0.01098901 0.08641975 0.03030303
4  0.02666667 0.49019608 0.13043478 0.10975610 0.00000000 0.01515152
1  0.00000000 0.01123596 0.15000000 0.34615385 0.03061224 0.26562500
2  0.02631579 0.01315789 0.05333333 0.09523810 0.54385965 0.01492537

> MC <- match_clusters(JS, name1="GT", name2="Lv")
> MC
  (GT.3,Lv.1)  (GT.4,Lv.2)  ( GT.1,Lv.4)  (GT.2,Lv.5)
   0.5294118    0.4901961    0.3461538    0.5438596
> Wmean(MC)
   0.4804535
```

4. Use your code to evaluate the significance of clusterings produced by different community detection methods applied to four different networks (all these are specified below). The evaluation will be a combination of the significance scoring functions (select a representative of each class, and rank the clusterings according to best values –recall that for some scoring functions high is best while for others low is best), plus the Jaccard similarity (local and global) with respect to a *good reference clustering*. Take as the reference clustering the ground truth, if it is known; or the best ranked clustering (according to your selected group of scoring functions), otherwise.

   You will evaluate at least the following community detection algorithms: Louvain, Label Propagation, Walktrap and Edge Betweenness. You may add any other(s) you like to the study and found in `igraph`.

   You will apply the community detection algorithms to the following networks (the first two with known ground truth communities):

   - karate (`igraphdata`)
   - a synthetic network with scale-free degree distribution, 200 nodes, 800 edges and 4 communities, which you have to build with `barabasi_albert_blocks` (`clusAnalytics`)
   - ENRON network (`igraphdata`). Note that this is a multigraph so you must convert it to a (undirected) simple weighted graph, where weights are number of edges among two nodes.
   - A network of your choice with no known community structure. You can use networks from network repositories available in the web or from the `igraphdata` package.

3 Deliverables

You have to prepare a report describing your findings and results while solving this lab, report and discuss summary tables of clustering significance evaluation based on scoring functions and the Jaccard similarity criterion for each network, and especially emphasizing any difficulties you encountered and the solution you found to overcome them.

To deliver: You must deliver the report explained above in PDF format. You also have to hand in the source code in `R` (or any other language) that you have used, including some minimal comments that can help the reader. This work can be done in pairs; in that case it is enough that one of you submits the work as long as both names are clearly visible in the report.

Procedure: Submit your work through the raco platform as a single compressed file.

Deadline: Work must be delivered within 2 weeks from the lab session you attend. Late deliveries risk being penalized or not accepted at all. If you anticipate problems with the deadline, please tell us as soon as possible.

References

[Arratia and Renedo-Mirambell, 2022] Arratia, A. and Renedo-Mirambell, M. (2022). clustAnalytics: An R Package for Assessing Stability and Significance of Communities in Networks. *R Journal*, xxx(0):1–11.

[Blondel et al., 2008] Blondel, V. D., Guillaume, J.-l., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of community hierarchies in large networks. *Networks*, pages 1–6.

[Brandes et al., 2008] Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., and Wagner, D. (2008). On Modularity Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20.

[Clauset et al., 2004] Clauset, A., Newman, M. E. J., and Moore, C. (2004). Finding community structure in very large networks. *Physical Review E.*

[Newman, 2006] Newman, M. E. J. (2006). Finding community structure in networks using the eigenvectors of matrices. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 74:036104.

[Newman and Girvan, 2004] Newman, M. E. J. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 69(2 Pt 2):026113.

[Pons and Latapy, 2005] Pons, P. and Latapy, M. (2005). Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10:191–218.

[Raghavan et al., 2007] Raghavan, U. N., Albert, R., and Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 76:036106.

[Reichardt and Bornholdt, 2006] Reichardt, J. and Bornholdt, S. (2006). Statistical mechanics of community detection. *Physical Review E*, 74.

[Rosvall and Bergstrom, 2008] Rosvall, M. and Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences of the United States of America*, 105:1118–1123.