Finding community structure

Marta Arias & Ramon Ferrer-i-Cancho

Version 0.1
Complex and Social Networks (2015-2016)
Master in Innovation and Research in Informatics (MIRI)

1 Introduction to `igraph`'s community detection algorithms

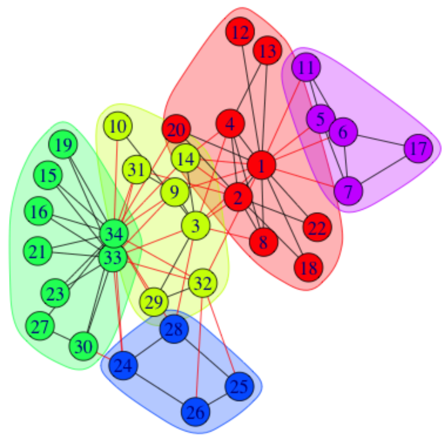
In this session you will run and compare different community finding algorithms. In the `igraph` package there are a few already implemented, including some we have seen in theory class:

- `edge.betweenness.community` [Newman and Girvan, 2004]
- `fastgreedy.community` [Clauset et al., 2004] (modularity optimization method)
- `label.propagation.community` [Raghavan et al., 2007]
- `leading.eigenvector.community` [Newman, 2006]
- `multilevel.community` [Blondel et al., 2008] (the Louvain method)
- `optimal.community` [Brandes et al., 2008]
- `spinglass.community` [Reichardt and Bornholdt, 2006]
- `walktrap.community` [Pons and Latapy, 2005]
- `infomap.community` [Rosvall and Bergstrom, 2008]

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

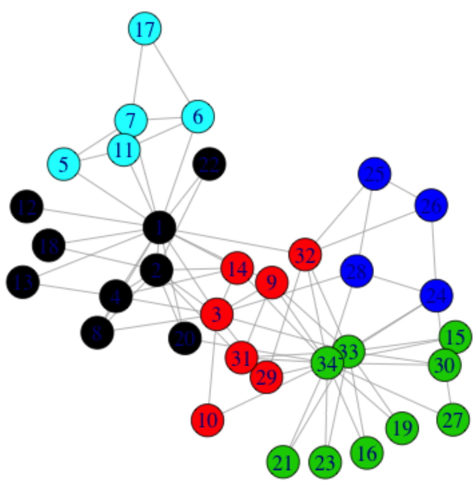
```
> karate <- graph.famous("Zachary")
> wc <- walktrap.community(karate)
> modularity(wc)
```

```
[1] 0.3532216
> 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)
```



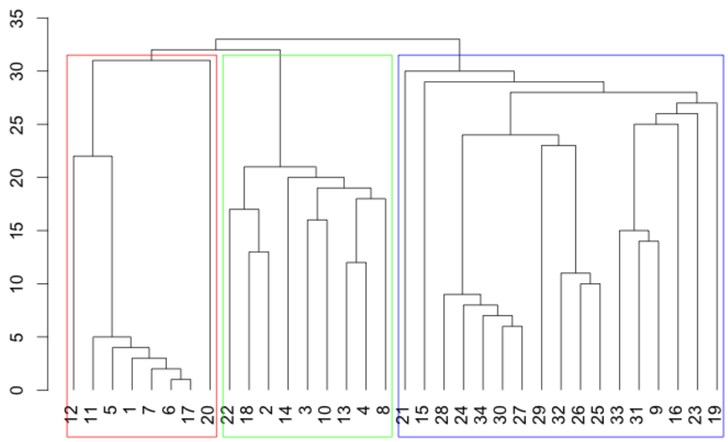
An alternative way of plotting communities without the shaded regions is:

```
> 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:

```
> karate <- graph.famous("Zachary")
> fc <- fastgreedy.community(karate)
> dendPlot(fc)
```



2 Tasks

2.1 Task 1

Write an `R` function that, given an undirected graph, it outputs for each available community finding algorithm the value achieved by the output partition for each of the following criteria: `‘modularity’`, `‘conductance’`, `‘cut ratio’`, and `‘expansion’` (see theory lecture notes for their definitions). Notice also that these definitions are for a single community, in order to compute the metrics for the whole network, you can use a weighted average of each of the communities, where the weight is given by the “volume” (i.e. fraction of number of nodes) in each community.

Use your code to find graphs for which different community detection methods beat on different criteria. In particular, you should build matrices like the following for a minimum three different networks of your choice.

	modularity	*conductance*	*cut ratio*	*expansion*
`edge.betweenness`				
`fastgreedy`				
⋮				
`infomap`				

Beware that community detection algorithms can be very time consuming, so stay away from large networks. Unless you have a *very* powerful computer.

Regarding what networks to consider, you can be as creative as you want. Simple suggestions could be to use the famous *Zachary* karate network, or generate your own using `igraph`'s generation tools. A way to generate a small network with very clear community structure is, for example:

```
g <- graph.full(10) + graph.full(10)
g <- g + edges(sample(V(g), 10, replace=TRUE))
```

However, in such simple networks the outputs of the community detection algorithms are going to be the same so it is not going to be very interesting. However, you can build up on this idea to generate more complex networks.

Alternatively, you can use networks from network repositories available in the web.

2.2 Task 2

Load the network `wikipedia.gml` provided[1]. It is in gml format, which can be imported into `igraph` using the following command

```
read.graph("wikipedia.gml", format="gml")
```

The vertices of this network are wikipedia pages. The label of each vertex is the title of the wikipedia page.

Now use any community detection algorithm. Do you think the communities found make sense? You can use the vertex labels to check this.

3 Deliverables

You have to prepare a report describing your findings and results while solving this lab, especially emphasizing any difficulties you encountered and the solution you found to overcome them.

[1]Thanks to Lada Adamic for providing this in her course *Social Network Analysis*.

To deliver: You must deliver the report explained above. The formats accepted for the report are, in principle, pdf, Word, OpenOffice, and Postscript. 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

[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.