The generation, visualization, and analysis of link communities in arbitrary networks with the R package **linkcomm**

Alex T. Kalinka

April 27, 2011

alex.t.kalinka@gmail.com
Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany.

Abstract

Identifying communities of related nodes in networks is an essential tool enabling researchers to make sense of complex data sets. The `linkcomm` package greatly facilitates this process by extracting communities of links from networks of arbitrary size and type. By clustering links, as opposed to nodes, it is possible for nodes to belong to multiple communities thereby revealing the overlapping and nested structure of the network and uncovering the key nodes that form connections across several communities. In addition to this, `linkcomm` provides extensive functionality for visualizing and analysing the resulting link communities.

1 Introduction

The representation of systems of interacting elements as networks has brought fresh perspectives and insights to the analysis of complex phenomena from the biological to the social sciences. When analysing the structure, function, and dynamics of these networks it is extremely useful to identify sets of related nodes, known as communities (Radicchi *et˜al.*, 2004).

This approach is greatly enhanced by clustering the links between nodes, rather than the nodes themselves (Evans and Lambiotte, 2009; Ahn *et˜al.*, 2010). With this method it is possible for nodes to belong to multiple communities, and this in turn reveals the overlapping and nested structure of the network while simultaneously identifying key nodes with membership across several communities.

The R package `linkcomm` implements the algorithm proposed by Ahn *et˜al.* (2010). Similarities between links that share a node are assigned using the Jaccard coefficient and these similarities are then used to hierarchically cluster the links. The resulting dendrogram is cut at a point that maximises the density of links within the clusters after normalizing against the maximum and minimum numbers of links possible in each cluster.

2 Quick start

To install `linkcomm` run the following command within R:

```
> install.packages("linkcomm")
```

To run an interactive demonstration of linkcomm within R:

```
> demo(topic = "linkcomm", package = "linkcomm")
```

3 Example session

To illustrate the package we will extract communities from a densely-connected sub-network from the yeast protein interactome (Yu *et˜al.*, 2008). This sub-network includes 56 proteins and 449 interactions involved in transcription (see Ahn *et˜al.* (2010), SI, p. 16). The network is undirected and unweighted (see section 3.4 for dealing with directed and weighted networks).

3.1 Extracting link communities

When extracting communities from a network, the input data must be arranged as a simple edge list.

```
> yeast_pp <- read.table("pp_rnapol.txt", header = FALSE)
> head(yeast_pp)
```

```
         V1      V2
290 YBR198C YGL112C
406 YCL010C YDR176W
907 YDR167W YML015C
908 YDR167W YML114C
922 YDR176W YDR448W
928 YDR190C YPL235W
```

In the above edge list, each row corresponds to an interaction between the elements in each of the columns. The interacting elements can be character names, as above, or integer numbers.

```
> lc <- getLinkCommunities(yeast_pp)
```

```
   Removing loops...
   Removing edge duplicates...
   Calculating edge similarities for 449 edges... 100.00%
   Hierarchical clustering of edges...
   Calculating link densities... 100.00%
   Partition density maximum height =  0.3333333
   Finishing up...4/4... 100%
   Plotting...
   Colouring dendrogram... 100%
```

The algorithm returns an object of class `linkcomm` and plots a summary of the results (Fig. 1). Included in this object are the communities that were extracted together with additional data required for plotting and further analysing the communities.

To view a summary of this object:

```
> print(lc)
```

```
   *** Summary ***
   Number of nodes =  56
   Number of edges =  449
   Number of communities =  11
   Link partition density maximum height =  0.3333333
   Number of nodes in largest cluster =  22
   Hclust method:  single
```

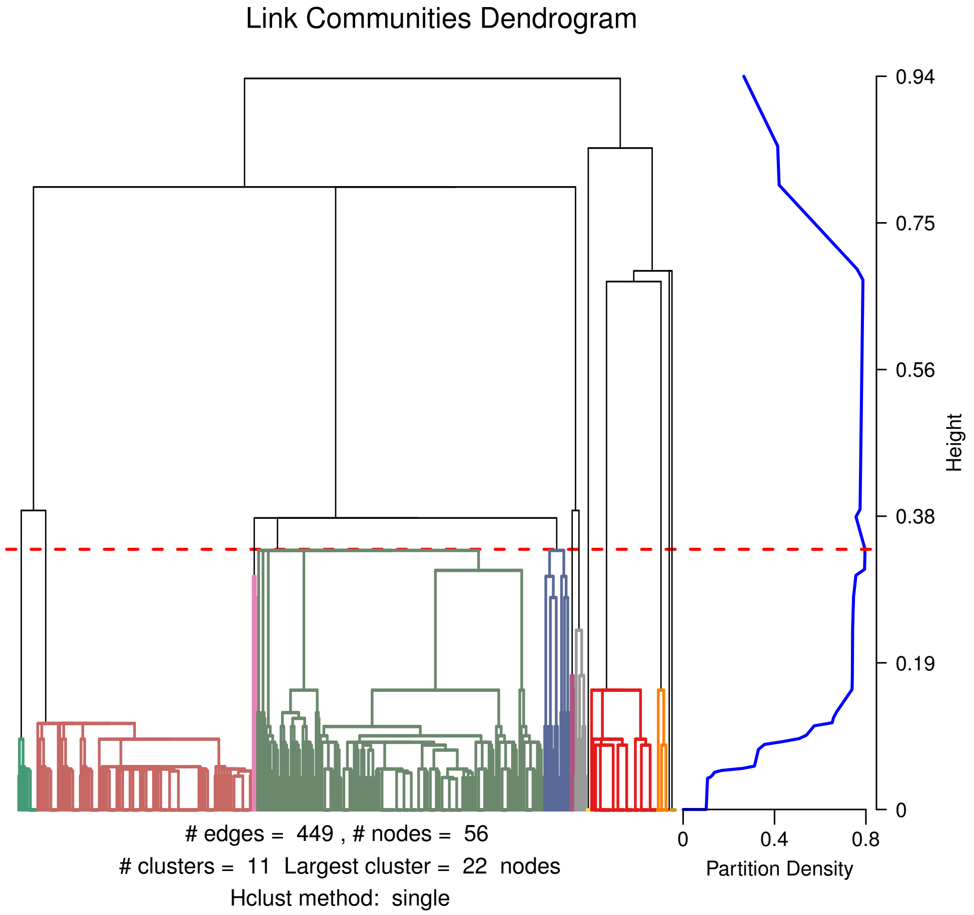

Figure 1: *Example output from extracting link communities from a yeast protein interaction network involved in transcription.*

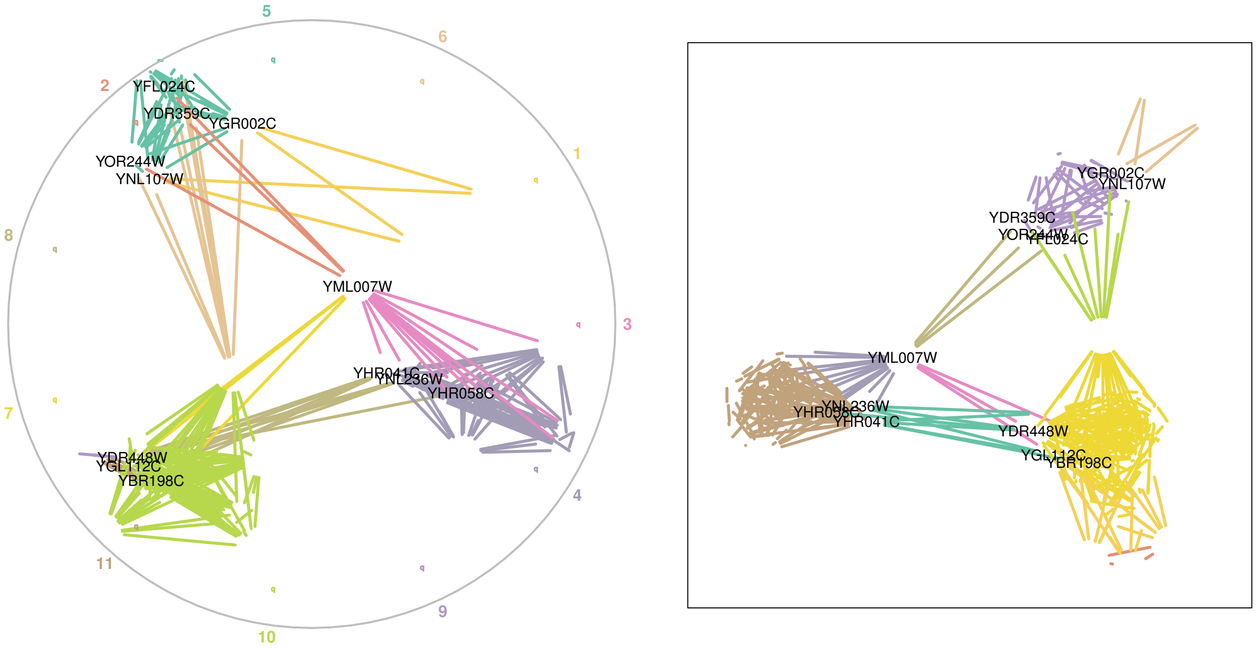

Figure 2: *Visualization of link communities using graphs. The panel on the left shows a Spencer circle layout (Spencer, 2010), while the panel on the right shows a Fruchterman-Reingold layout. In both graphs, we show only the nodes that belong to 3 or more communities. Numbers around the circumference of the circle refer to community IDs.*

3.2 Visualizing link communities

We can visualize link communities using coloured edges in a graph layout.

```
> plot(lc, type = "graph", layout = layout.fruchterman.reingold)
> plot(lc, type = "graph", layout = "spencer.circle")
```

To aid the visualization of key nodes we can limit the display of nodes using the `shownodesin` parameter. The following displays only the nodes that belong to 3 or more communities (Fig. 2):

```
> plot(lc, type = "graph", layout = "spencer.circle", shownodesin = 3)
```

We can also visualize node community membership for the top-connected nodes using a community membership matrix (Fig. 3):

```
> plot(lc, type = "members")
```

We can also display a summary of the results of the link communities algorithm (Fig. 1):

```
> plot(lc, type = "summary")
```

Additionally, we can simply plot the dendrogram on its own with coloured community clusters:

```
> plot(lc, type = "dend")
```

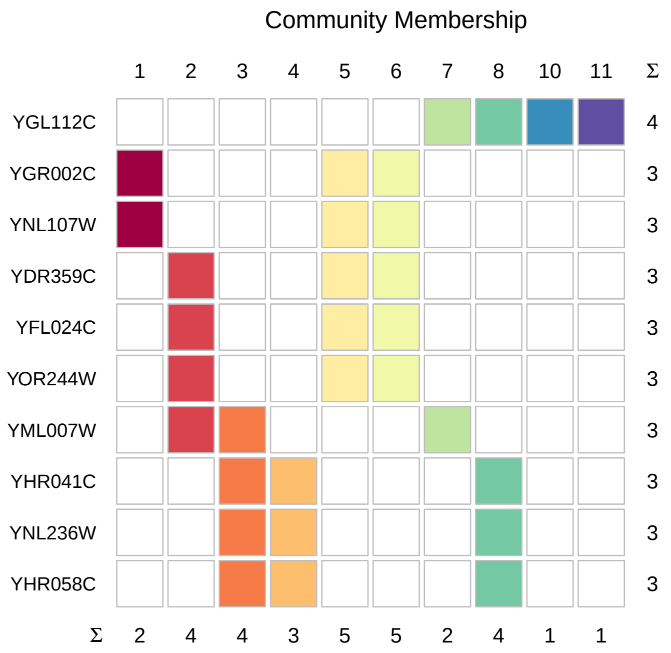

Figure 3: *Visualizing community membership for nodes that belong to the most communities. Colours indicate community-specific membership.*

3.3 Analysing link communities

3.3.1 Nested communities

Nodes can belong to multiple link communities, and so it is possible to discover sets of nodes that belong to a community that is entirely nested within a larger community of nodes:

```
> getAllNestedComm(lc)
```

```
$`9`
[1] 11
```

This result indicates that there is one nested community; the nodes in community 9 are entirely nested within the nodes of community 11. We can verify this by plotting these communities as a graph (Fig. 4):

```
> getNestedHierarchies(lc, clusid = 9)
```

or

```
> plot(lc, type = "graph", clusterids = c(9, 11))
```

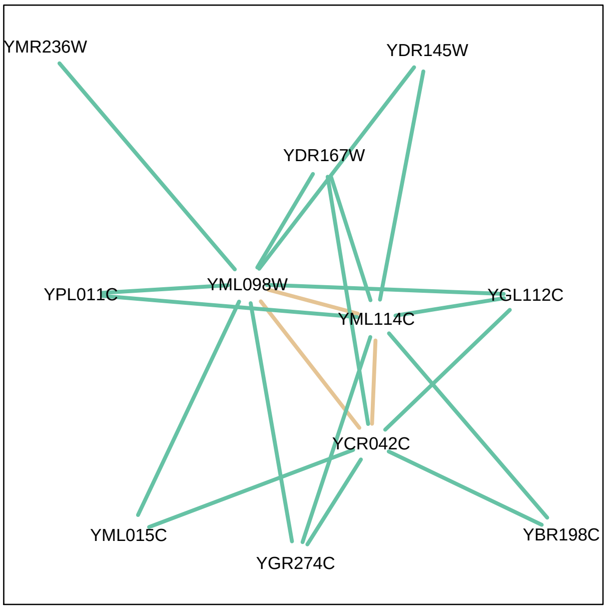

Figure 4: *Nested community structure revealed by* `linkcomm`*. YML098W, YCR042C, and YML114C belong to both communities.*

3.3.2 Relationships between communities

We can also explore the relationships between communities based on the numbers of shared nodes. To do this we can hierarchically cluster the communities after scoring the pair-wise similarities between them using the Jaccard coefficient (based on the number of nodes that they share).

```
> cr <- getClusterRelatedness(lc, hcmethod = "ward")
```

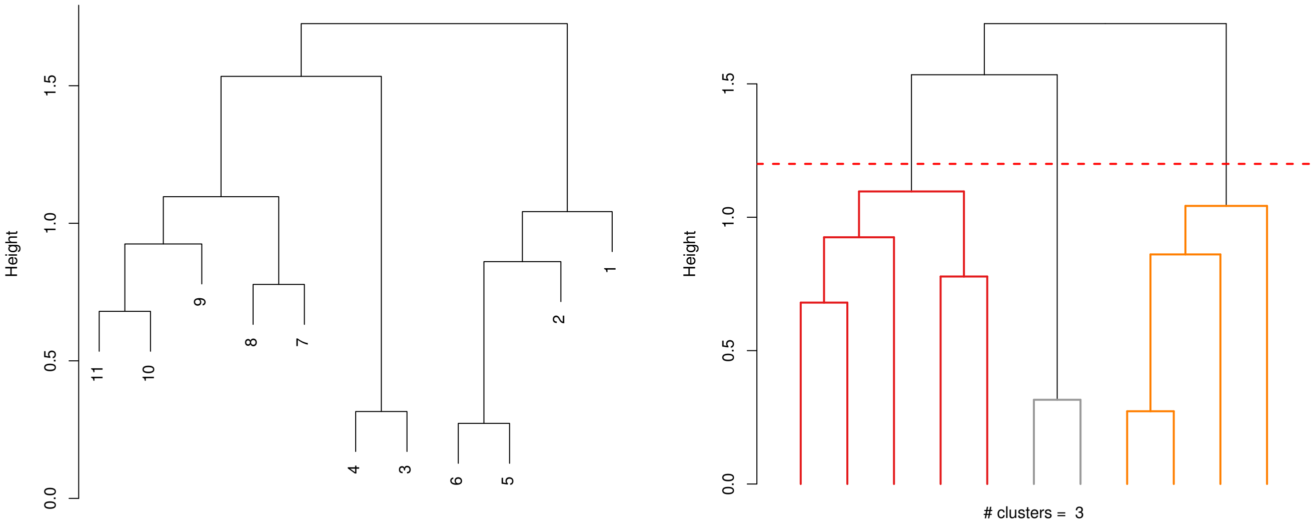

Figure 5: *Dendrograms showing, on the left, the similarity between link communities, and, on the right, clustering between these communities to produce 3 meta-communities.*

This function returns a hierarchical clustering object of class “hclust” and plots the dendrogram (Fig. 5). After inspecting this dendrogram, we can select a height at which to cut the dendrogram and extract the resulting meta-communities.

```
> cutDendrogramAt(cr, cutat = 1.2)

[[1]]
[1] 3 4

[[2]]
[1] 1 2 5 6

[[3]]
[1]  7  8  9 10 11
```

This returns the meta-communities (clusters of community IDs). Alternatively, we could have cut the dendrogram using the `cutat` option in the original clustering function, `getClusterRelatedness`. We can verify that these meta-communities make sense by inspecting the communities in the Spencer circle (Fig. 2). Note that nodes may still belong to multiple meta-communities.

3.3.3 Community centrality

In link communities, nodes may belong to several communities, and so it is possible to measure the importance of a node in a network based on the number of communities to which it belongs. To do this, we weight the membership of a node in a community by how distinct that community is from the other communities to which the same node belongs

$$C_c(i) = \sum_{i \in j}^{N} \left(1 - \frac{1}{m} \sum_{i \in j \cap k}^{m} S(j,k) \right), \tag{1}$$

where the main sum is over the N communities to which node i belongs, and $S(j,k)$ refers to the similarity between community j and k, calculated as the Jaccard coefficient for the number of shared nodes between each community pair, and this is averaged over the m communities paired with community j and in which node i jointly belongs.

Hence, nodes that belong to a lot of different communities will get the largest community centrality scores, whereas nodes that belong to overlapping or nested communities, or to few or no communities, will receive the lowest community centrality scores.

We can calculate this value for a set of nodes that have been assigned to link communities:

```
> cc <- getCommunityCentrality(lc)
```

```
> head(sort(cc, decreasing = TRUE))
```

```
 YGL112C  YML007W  YDR448W  YBR198C  YHR041C  YNL236W
4.313943 3.739496 3.487343 3.395000 3.012759 3.012759
```

Comparing against the unweighted community membership, we find differences:

```
> head(lc$numclusters)
```

```
YGL112C YGR002C YNL107W YDR359C YFL024C YOR244W
      4       3       3       3       3       3
```

3.3.4 Community modularity and connectedness

We can also calculate the modularity of communities - the relative number of links within the community versus links outside of the community - and its inverse, community connectedness. The modularity of community i can be written as

$$M_i = \left(\frac{e_w(i)}{n_i(n_i - 1)/2} \right) \cdot \left(\frac{e_b(i)}{n_i \bar{d}} \right)^{-1}, \tag{2}$$

where $e_w(i)$ and $e_b(i)$ are the number of links within and without community i respectively, n_i is the number of nodes in community i, and $\bar{d}$ is the average degree of nodes in the network.

We can calculate and plot the modularity of the communities in our network (Fig. 6):

```
> cm <- getCommunityConnectedness(lc, conn = "modularity")
> plot(lc, type = "commsumm", summary = "modularity")
```

We can verify that these measures make sense by inspecting the communities in the Spencer circle (Fig. 2).

3.3.5 User-defined link communities

It is also possible for the user to extract link communities that are not defined by the maximisation of the link partition density. Instead, the user can inspect the clustering dendrogram and select a different height at which to extract communities:

```
> lc2 <- newLinkCommsAt(lc, cutat = 0.4)
```

Now the number and composition of the communities has changed, but we use the clustering method from the original clustering:

```
> print(lc2)
```

```
   *** Summary ***
   Number of nodes =  56
   Number of edges =  449
   Number of communities =  7
   Link partition density maximum height =  0.4
   Number of nodes in largest cluster =  25
   Hclust method:  single
```

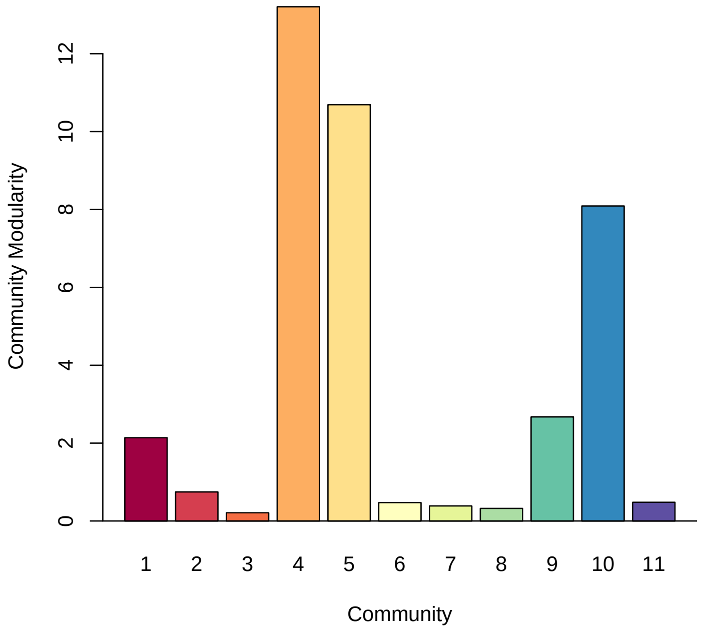

Figure 6: *The modularity of different link communities.*

3.3.6 Community membership of nodes

We can extract the nodes from single or multiple communities:

```
> getNodesIn(lc, clusterids = c(4, 5))
```

```
 [1] "YGL127C" "YOR174W" "YHR058C" "YPR168W" "YNR010W" "YDL005C" "YBL093C"
 [8] "YGR104C" "YHR041C" "YLR071C" "YMR112C" "YNL236W" "YOL051W" "YPL042C"
[15] "YCR081W" "YDR443C" "YER022W" "YDR308C" "YHR090C" "YEL018W" "YDR359C"
[22] "YJR082C" "YNL136W" "YOR244W" "YPR023C" "YGR002C" "YNL107W" "YFL024C"
```

3.4 Directed and weighted networks

When analysing directed networks, we need to specify that the network is directed, and/or choose a weight for links that share nodes yet are in the opposite orientation (the default value is 0.5):

```
> lc <- getLinkCommunities(yeast_pp, directed = TRUE, dirweight = 0.8)
```

For weighted networks, the input data must be an edge list with an additional third column of numerical weights, for example:

```
> head(weighted)
```

```
        node1       node2            weight
1 FBgn0000575 FBgn0037290               0.5
2 FBgn0000575 FBgn0004880               0.5
3 FBgn0000575 FBgn0037375 0.333333333333333
4 FBgn0000575 FBgn0040281              0.25
5 FBgn0000575 FBgn0035101 0.333333333333333
6 FBgn0000575 FBgn0027111               0.4
```

For both directed and weighted networks, the algorithm will score the similarities between links that share a node using the Tanimoto coefficient:

$$S(e_{ik}, e_{jk}) = \frac{\mathbf{a}_i.\mathbf{a}_j}{|\mathbf{a}_i|^2 + |\mathbf{a}_j|^2 - \mathbf{a}_i.\mathbf{a}_j}, \quad (3)$$

where $\mathbf{a}_i$ refers to a vector describing the weights of links between node i and the nodes in the first-order neighbourhoods of both nodes i and j (equal to 0 in the event of an absent link). For directed networks, links to nodes shared by both node i and j are given a user-defined weight below 1 if they are in the opposite orientation.

3.5 Handling large networks

The `linkcomm` package can handle networks of any size. To do so, the upper triangular dissimilarity matrix used by the hierarchical clustering algorithm is compressed and written to disk as a temporary file. This matrix is then read, modified, and re-written (by a compiled C++ function) as clustering proceeds until the file size is 0 bytes. The speed at which large networks are processed will depend on the power of the computer being used. The size of the file that holds the compressed matrix may also be very large, so hard disk space could be a limiting factor for extremely large networks.

We make no assumptions about the computer resources available to end users, and so we provide a parameter, `edglim`, which can be modified by the user and which determines the maximum permissible size of the input network in terms of links for it to be handled in memory - above this size the dissimilarity matrix will be handled on the disk. The default value is 10^4 links, but this can be modified:

```
> lc <- getLinkCommunities(yeast_pp, edglim = 10)
```

As a guide, a network with 10^4 links will require $((10^4)^2) * 8 = 800$ MB to be handled in an uncompressed format in the memory.

3.6 Exporting link communities to Cytoscape

Cytoscape is an open source platform for complex-network analysis and visualization (Cline *et˜al.*, 2007). We can export our link communities into an edge attribute file that can be imported into Cytoscape:

```
> linkcomm2cytoscape(lc, interaction = "pp", ea = "linkcomms.ea")
```

This will save an edge attribute file to “linkcomms.ea” in the current directory.

References

Ahn,Y.Y., Bagrow,J.P., and Lehmann,S. (2010) Link communities reveal multiscale complexity in networks, *Nature*, **466**, 761-764.

Cline,M.S., *et al.* (2007) Integration of biological networks and gene expression data using Cytoscape. *Nat Protoc*, **2**, 2366-2382.

Evans,T.S., and Lambiotte,R. (2009) Line graphs, link partitions and overlapping communities, *Phys. Rev. E.*, **80**, 016105.

Raddichi,F., *et al.* (2004) Defining and identifying communities in networks, *Proc. Natl Acad. Sci USA*, **101**, 2658-2663.

Spencer,R., (2010). http://scaledinnovation.com/analytics/communities/comlinks.html.

Yu,H., *et al.* (2008) High-quality binary protein interaction map of the yeast interactome network, *Science*, **322**, 104-110.