Bioinformatics III

Prof. Dr. Volkhard Helms — Saarland University
Ruslan Akulenko, Ahmad Barghash, Duy Nguyen, Christian Spaniol — Chair for Computational Biology
Winter Semester 2013/2014

Exercise Sheet 3

Due: November 15, 2013 13:15

Submit your solutions on paper, hand-written or printed at the *beginning* of the lecture or in building E2 1, Room 3.01. Alternatively you may send an email with a single PDF attachment, via mail to christian.spaniol@bioinformatik.uni-saarland.de.

Network Communities & Real Interaction Networks

In this assignment you will deal with network communities which are defined as regions of the network that have more internal connections than connections to the rest of the network. Your task is it to use the algorithm of *Radicchi et al.* to identify the communities of a given network.

Exercise 3.1: Network Communities (50pts)

(a) **Edge-clustering coefficient**

The edge-clustering coefficient $\tilde{C}_{i,j}^{(3)}$ of a link between nodes i and j is defined as the ratio of the actual number of triangles $z_{i,j}^{(3)}$ to which the link between i and j contributes and the number of possible triangles, determined by the minimum of the degrees k_i and k_j of the two nodes i and j:

$$\tilde{C}_{i,j}^{(3)} = \frac{z_{i,j}^{(3)} + 1}{\min[k_i - 1, k_j - 1]}$$

If one of the nodes has a degree of 1, then $\tilde{C}_{i,j}^{(3)}$ is infinite. What is the maximal *finite* value that the edge-clustering coefficient can take? For which configuration does this occur?

(b) **Determine communities**

To determine the communities of the supplied artificial network given in `bb.txt` (found in the additional materials), perform the following steps:

(1) **Decomposition of the network**

As explained in the lecture, iteratively delete the links with the smallest $\tilde{C}_{i,j}^{(3)}$:

i. Read in the network file.
ii. Calculate the edge-clustering coefficient $\tilde{C}_{i,j}^{(3)}$ for each link.
iii. Find the link with the smallest $\tilde{C}_{i,j}^{(3)}$ and delete it from the network. Store this link.
Hint: When you encounter multiple links with the same $\tilde{C}_{i,j}^{(3)}$, take the one that occurs last in the network definition file!
iv. Repeat from (ii) until there is no link left.

Give the links that you deleted from the network in (iii) by printing their identifier (line number in the original file), the names of the two nodes, and their current edge-clustering coefficient in the order of their deletion.

(2) **Build communities and the dendrogram**

You may solve this part with pen and paper instead of writing a program.

There are two criteria for a community (see *Radicchi et al., 2004*):

i. In a *community in a strong sense* every single member of the subgraph V has more links to the inside of the community (k^{in}) than to the outside (k^{out}):

$$k_i^{\text{in}}(V) > k_i^{\text{out}}(V) \qquad \forall i \in V$$

ii. In a *community in a weak sense* the total number of links inside the subgraph V is bigger than to the outside:

$$\sum_{i \in V} k_i^{\text{in}} > \sum_{i \in V} k_i^{\text{out}}$$

Now use the links deleted in (1) in reverse order, i.e., the link that was deleted last is now used first to construct the communities. To do so, take one link after the other and check if they have nodes in common with the already included links. During this composition stage you do not need to keep track of the links, but only of the nodes that belong to the same subgraph.

i. If the latest link is disjoint from the already processed links, then start a new subgraph (=list of nodes of this subgraph) from this one.
ii. If the latest link has a single node in common with one of the existing subgraphs, then add the other node of this link to that (list of the nodes of the) subgraph, too.
iii. If the two nodes of the latest link belong to two different subgraphs, then join the two subgraphs to form a single one from them. Highlight the two lists of nodes that are joined in this step.

Finally, when the last link is added, you should end up with a single graph that contains all nodes of the network and a listing of the subgraphs just before they were joined to form bigger ones.

To draw the dendrogram of the network, look at the above choice (iii), the joining of two groups: start from the individual nodes and every time that this happens, connect two subgraphs.

(c) **Visualization of the communities**

In Figure 1 the final positions from a force directed layout of the "Baking Bread" network are given. Use this plot to visually identify the communities. Point out *four* communities that are disjunct. Cover all nodes and specify for each of the communities whether the weak or the strong criterion applies.

Exercise 3.2: Biological Interaction Networks (50pts)

The "Biomolecular Interaction Network Database" (BIND) contains many known interactions between proteins and small molecules for many different species.

Make yourself familiar with the BIND dataset in the supplementary found on our webserver. The archive contains the flat text file with the interactions ("`interactions.txt`"), the list of taxon identifieres ("`taxon.txt`"), and a "`bind_readme.txt`" which explains the format of the interaction file.

Build a data structure that stores the taxon identifiers and helps to retrieve the interactions for a certain organism, i.e., a specific taxon identifier.

(a) Get the BINDData class stub from the supplementaries on our webserver. It contains the implementation as well as a declaration of several methods to parse BIND interactions and taxonomy lists.

(b) Implement the method `__readTaxonIdentifiers__()` that reads the taxon file and maps the taxon identifiers to an organism name.

(c) Implement the method __countInteractions__() that counts all interactions within each organism, i.e., how often each taxon identifier occurs in the interactions registered in BIND. Only count those interactions where both partners have the same taxon id or exactly one of them is a "small molecule". Which are the top five species that have the largest number of these interactions in BIND? Give their taxon identifiers, their scientific names and the respective number of occurrences.

Hints:

- Since the data is non-artificial, there are some hurdles to overcome: there is an issue with whitespacing within nodes in the files. Your program has to be robust against failures.
- The file format and the meaning of the columns of the interaction file are explained in the README.
- There is the class of small molecules with the taxon id of 0. This is not a species!
- Take care when using the dictionary, because a handful of taxon identifiers in the interaction file are not listed in the taxon overview.
- When reading Footnote 5 of the README file, you will notice that it indicates that there may be either missing accession codes or missing molecule ids for the proteins. A workaround is to create a new label for the proteins by concatenating the accession code and the molecule id. This ensures that each protein is labelled uniquely, so these labels represent appropriate node identifiers.
- The AB column does not matter (Footnote 9). Also, the network is not directed!
- Sort the resulting interaction list by number of interactions descending, i.e., the organism with the most interactions is listed first in self.taxonSort. A possible method to sort a python dictionary is as follows:

```
from operator import itemgetter
self.taxonSort = sorted(self.taxonHist.items(), \
                        key=itemgetter(1), \
                        reverse=True)
```

(d) Implement the method getPPIsFor(targetTaxonId) that returns a list of all *protein-protein* interactions within a species, i.e pick those interactions where both partners belong to the chosen species *and* both are a protein.

(e) Back to networks: from AbstractNetwork derive a class BINDNetwork that has a taxon identifier and a BINDData object as constructing argument. Build a network by getting all protein-protein interactions for a species from the BIND dataset. Create a node for each protein (*once*) and link them to each other in order to model an interaction.

(f) Now for each of the top five species build a protein interaction network. Count the total number of proteins that are listed for this species. How many proteins are interacting in each species? Use the python module you implemented in the first assignment to determine the degree distribution of the protein interaction networks. Do you see any difference between the degree distributions of the interaction networks? Are they more like the scale-free network or more like the random graph? Explain your observations.

Hint:

- Plot the integrated $P(k)$ to determine the behaviour (exponent?).

Have fun!

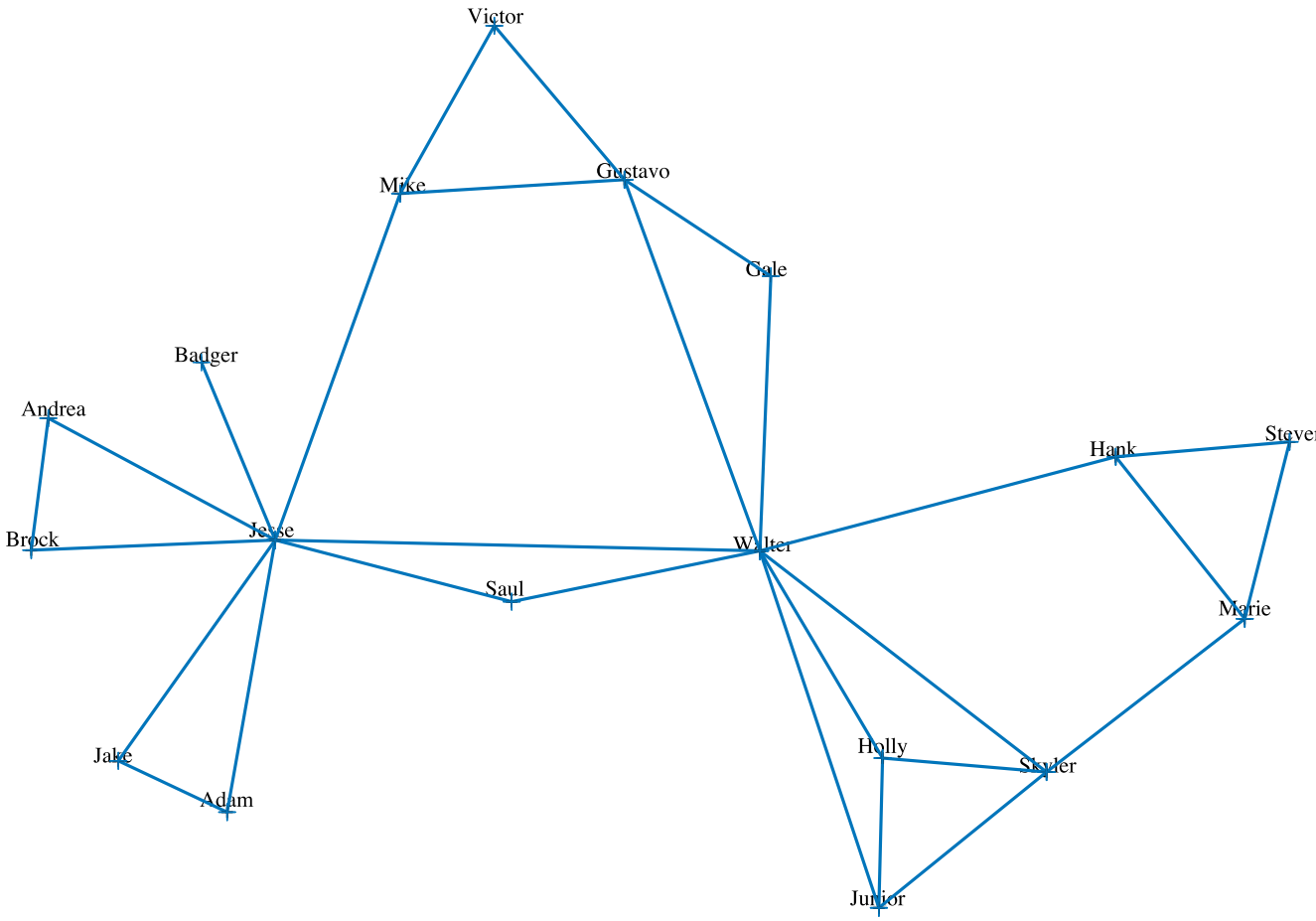

Figure 1: Force-directed layout of the “bb.txt”-network