BIOINF 703: Networks Assessment

Matthew Egbert, with thanks to David Welch

September 20, 2016

Setting up

1. You have already downloaded the network_lab.zip file from canvas. Make sure you have extracted this file and the others into a an appropriate directory (e.g. one in your own file space that won't be deleted when you log out).
2. Open a Python terminal such as `ipython` or `python`.
3. Make sure that `networkx` is installed by typing `import networkx as nx` into the python terminal. I recommend keeping a terminal like this open so that you can try out various python commands to see how they work and to make sure that you are using them correctly before using them in a larger program. In `ipython`, you can write the name of a function followed by a question-mark and it will often give you information about that function.
4. Use a text-editor to open `edge_data.txt`. What is in this file?
5. Use a text-editor to open up `network_helper.py`. Take a look at the various functions in this file.
   - `readFile(fn)` : reads a text file, where each line row is a pair of vertices connected by an edge. Returns a `networkX` graph of the described graph.
   - `localClusteringCoeff(G)` : returns the mean local clustering coefficient for the graph G.
   - `randomGraphFromDegreeDistribution(G)` : takes a graph G as an argument, and returns a random graph with a similar size and degree distribution.
   - `allDegrees(G)` : for a graph G of size N, this function returns a list of N integers, representing the degrees of all of the nodes in the graph
6. Use a text-editor to open up `main.py`. This is the file that you will write your code in. You will submit it along with a PDF write up. Figure out how to run it. Generally you can open up a terminal and once you are in the correct directory, execute `python main.py`. Without any editing, this file should display the number of vertices and edges in the graph described by `edge_data.txt`.

Instructions

Use the script provided to read in the network from the file `edge_data.txt`. Call this network G. The file is a rather messy edge-list that represents an RNA-to-RNA transcript abundance network which is inferred from experimental data around the Rel/NFKB family of transcription factors. The nodes in the networks represent mRNA transcripts (actually, probes on a microarray), and the edges represent indirect relationships between them.

Your task is to to make a thorough comparison of this graph with the Erdös-Renyi (ER) model. Answer all the questions below in a report that includes appropriate plots. There is a `save` button in the python-plot windows that you can use to save your plots, which can then be imported into your preferred word-processor.

Part A

1. Make a plot of the degree distribution of G. I have provided a helper function `allDegrees(G)` that returns a list of integers representing the degree of every node in the graph G. You may find the `numpy` function `histogram` [1] useful.

2. We've seen that networks generated under the ER model have a binomial degree distribution. Choose the parameters of the binomial distribution that corresponds to an ER network with the size and mean degree of G. Don't just eyeball this – you may want to look in the lecture slides.

   Make a plot of a binomial distribution with these parameters (use the `binom` function from `scipy.stats` if you like `http://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.binom.html` functions). Label your axes and include the parameters you chose in the title or in a legend.

3. Does the ER model look like a good model for G? Why or why not?

4. Another model we have looked at is scale-free networks. The degree distribution in these networks has the form $P(k) \propto k^{-\gamma}$ where $\gamma > 1$, and typically, in observed networks, $1 < \gamma \leq 3$.

   You can make a plot of this function using the following Python code, if you first set n to be the number of the vertices in G and choose a value for gamma.

   ```
   k = arange(2.0,n-1)
   title("Unnormalized Density")
   plot(k,k**(-gamma),label="Power law w/gamma=%f"%(gamma))
   ```

   (a) Why did I suggest a title of "**Unnormalized** Density"?
   (b) Which values of $1 < \gamma \leq 3$ best match the degree distribution of G?
   (c) How well does it match?

5. Both of these models are poor, but let's push on with the ER model. Using `networkx`'s function `erdos_renyi_graph(n,p)` — `nx.erdos_renyi_graph`, generate 200 Erdös-Renyi graphs to compare with G. For each graph, measure the following statistics.

   - **mean betweeness** – use the networkx function `nx.betweenness_centrality` as part of your solution.

     This function returns the data you need, but in a slightly awkward format. To transform it into a simple list of values, you can do the following.. Note the addition of the `.values()` and the call of the `list()` function.

     ```
     x = list(nx.betweenness_centrality(G).values())
     ```

   - **mean local clustering coefficient** – use the networkx function `nx.average_clustering(G)` as part of your solution

[1] `http://docs.scipy.org/doc/numpy/reference/generated/numpy.histogram.html`

Make histograms of these statistics and mark on them where the corresponding values of G would lie. A good function to use to do this marking is `axvline` – google it!

6. Give a brief description of what each statistic means (not just the mathematical definition).

7. Explain how the above simulations (comprehensively) demonstrate that the graph G does not fit the ER model.

8. We can build a graph with a degree distribution based on the empirical distribution of G by resampling from the degree distribution of G, assigning the sampled degrees to vertices and then uniformly at random connecting these vertices to one another. The function `randomGraphFromDegreeDistribution` allows you to do this. Repeat the analysis in Step 5, replacing the ER graphs with graphs built under this "boot-strapping" model.

9. Comment on how well the model in the previous question fits G, making reference to the fact that G and graphs sampled from the bootstrapping model have (roughly) the same degree distribution.

10. Supposing we had a good network model for G, what would we expect the plots in Step 5 to look like when generated under this good model?

Part B

Most of what we have talked about assumes we have been given a network. But the large part of the work in this area involves actually inferring networks from various sources of data. Read the 2010 review article by Riet De Smet and Kathleen Marchal, *Advantages and limitations of current network inference methods*. Using the paper as your main reference, describe the network inference problem and define the following terms in this context:

- Bipartite network
- Gene co-expression network
- Transcription regulation network
- Underdetermined problem
- Clustering/biclustering techniques
- Supervised/unsupervised learning
- Combinatorial regulation
- query-driven inference

Ignore the details of particular programs or software mentioned in the paper. Write approximately 1000–1500 words.