lab notes

See Jupyter notebook / Slides used during the lab uploaded to moodle in pdf and ipynb format. The following is to add commentary, i.e. what we discussed during the lab.

Notes about the homework

always submit code as well

When handing in homework, make sure to always include the scripts you used, since this makes it much easier to review your results and also allows me to offer comments and suggestions about your code.

correctness vs. style comments

Comments about code can be divided into two groups: those about code correctness, and those about code style. Style doesn't directly influence the correctness of the code, and thus won't directly affect grading. However, style does indirectly affect correctness in that messy code is harder to extend and debug, and thus it will indirectly impact grading as well.

Plotting the histograms

Selecting the number of bins

When plotting a binned histogram, one important task is to select the correct number of bins to use: too few, and we lose too much information because we are averaging too much: the resolution of the plot won't be very good, it will be too "blocky". Too many, and we will get either a noisy histogram (because we are not averaging enough), or even empty bins between non-empty one (where we had no datapoints falling into a given bin)

Using logarithmic axis for plots

The in-degree distribution has values over a large range. Plotting it with linear axis makes all the datapoints get squeezed in the lower left corner of the plot. In such cases, allways use logarithmic axis to make the data more visible. For plt.hist(), use the "log=True" to set the vertical axis to logarithmic. To set the horizontal axis to logarithmic as well, use plt.xscale('log'). (For plt.plot(), use plt.yscale('log') and plt.xscale('log'), respectively.)

When describing the plots, consider additional checks you can make

We have mentioned that the spike on the left-hand side of the out-degree distribution is due to nodes with zero out-degree. It is very important that we made this statement not just by looking at the histogram, but actually by counting the number of nodes with zero degree. Looking at the in-degree and in-strength histograms, we might be tempted to also attribute the biggest spike to zero degree nodes, but in that case we would be wrong: there are (comparatively) few nodes with 0 in-degree, the very large spike on thos histograms is due to nodes with low, but non-zero in-degree and in-strength values.

Similarly to what we have seen when doing network visualizations (the first homework assignment), the visualization or evidence for a given observation might not be the one you are looking at when you notice the given feature. Whenever this is simple to do, make sure to look at a more approriate one, as an additional check.

Testing for powerlaws

A very good paper on testing for powerlaws in data is available at https://arxiv.org/abs/0706.1062 Careful statistical analysis is needed because many distributions can appear powerlaw-like. Also consider whether you actually need to make a statement about the data being a powerlaw, or a much weaker (and thus easier to prove) statement like fat-tailedness is enough.

Use the "powerlaw" python module to quickly run the tests described in this paper. The important aspects:

1. Run the fitting procedure with "fit = powerlaw.Fit(...data...)"
2. Compare different distributions by using fit.distribution_compare() -- the result: R: how big a difference is there between the fit of the two distributions, and in which direction (positive: first one is better, negative: second one is better). p: the p-value showing statistical reliability of the result. The lower the better, usual threshold is that if p is larger than 0.05, the result is inconclusive (due to not being reliable enough).
3. Plot the data, including the fitted functions, to check the fit by eye.

Explaining the shape of the out-degree and out-strength distributions

Note that this is going beyond describing the shape of the distribution. We are going to propose hypotheses for mechanisms that might produce the features we see in the data. Like we mentioned for interpreting observations for network visualizations (the first homework), we should be open to alternative explanations, and we should be aware that we only have partial information about the system the data is coming from.

Comparing the two out- distributions (out-degree and out-strength), we can see that the spike at zero is appears in both, as it must: if a node has zero degree, its strength will be zero as well. The rest of the distribution is more skewed on the strength distribution, and that part ending at 1.0 is also interesting -- we didn't look into where the degree distribution ends, but the strength distribution ending at 1.0 might have some meaning. Looking at the top of the datafile might provide some explanation:

Thus our explanation: data is collected by showing people a word, and having them reply with another one. Words that were not used as prompt words (of which there will be many, since the experiment subjects can reply with any word, not just those on the list of prompt words) will have zero out-degree (and thus zero out-strength as well). Calculating how frequently certain replies were given for a given prompt word results in normalizing the weights of the edges such that all (non-zero) out-strength values are exactly 1.0. Some process then removes some of the weights, resulting in the peak at 1.0 to be "smeared" to the left (but explaining the skewed shape of that part of the distribution). This process might be either removal of the self-edges (if that is done after the edge weights are normalized), or removal of weak edges (edges with very low weights).

Note that although this very nicely explains almost all of the out-strenght distribution, there is an outlier datapoint: a single node has an out-strength larger than 1.0. (This is more prominent when plotting the data using log scaling the vertical axis, see below.) This shouldn't happen according to the explanation above, since those processes lead to out-strengths not larger than 1.0. We don't really have a good explanation here, it might be due to rounding error, data corruption, or something similar. Although this single datapoint would invalidate our explanation, it is often more useful to stick to our explanation and consider this datapoint an un-explained outlier, since our explanation does work for the rest of the data and provides a good description there.

Theoretical estimate for the average clustering coefficient in the ring & shortcuts network:

The problem: Take N nodes on a ring, the links connect first and second neighbours. In addition, randomly distribute N/2 shortcuts among the nodes (i.e. on average one shortcut for each node), connecting random pairs. Calculate the average clustering coefficient using both a theoretical calculation, and also computationally by generating such networks.

Theoretical estimate

We can get an estimate using the following: Assume that the number of triangles formed by the shortcuts is negligible. (Since for a shortcut starting from a given node to form a triangle it has to end at one of a handful of nodes, while there are almost N other nodes it could connect to. This means as N increases, the probability of triangles being formed decreases). Thus, for each node we will have three triangles before the shortcuts are introduced. Since the average degree is <k> = 5 (4 from the original, and 1 from shortcuts), the average clustering coefficient is: <c> = 2* (number of triangles) / (<k>(<k>-1)) = 2*3/(5 * 4) = 0.3