Alexander Looi
Data Science II
March 28th, 2017
Write Up

1. There were a couple of things I changed/would change with the set up for the nets.py script. First, I moved the degree_distribution() and clustering_coef() functions into the graph() class. Additionally, I would move connected_nodes(), relabel_nodes(), and subgraph() all into the graph class as well since these functions all need to take in a graph() object. It'll reduce the amount of potential user error if you could reduce the amount of inputs needed.
2. From my specific script, there is no way to handle Directed graphs, though as far as I can tell, most of the code I think can handle a directed graph.
    a. Direct paths between two specific nodes are not made
    b. The type of degree distribution of the graph is not determined (is the degree distribution normal or scale free?).
        i.
    c. The module does nothing really with "assortivity"
        i. Homophility of a node is not calculated (so do "like" nodes connect with each other?).
        ii. Could calculate the assortivity coef.
        iii. Potentially useful when trying to sort out communities in your graph
    d. Betweenness Centrality and Degree centrality are not calculated
        i. Degree centrality—how nodes are ranked according to degree
        ii. Betweenness Centrality—Ranks of nodes by the total number of shortests paths that go through it.
        iii. Useful to know to for failure rates when links or nodes are removed.
    e. The calculation of a Null Model using the node number of the graph
        i. Should also be able to not self link if needed. Or Erdos Remy Model
        ii. Configuration Model
        iii. Useful conducting simple hypothesis testing of your graphs.
3. The code seems pretty fast, nosetests can run all 28 of my test in 0.013 seconds which seems fast. I suppose it could be faster if I used more implicit loops. But I feel that optimizing my code at this point will be an exercise in diminishing returns since essentially using the entire module 28 times and using 0.013 seconds is a huge chunk out of my day.
4. I think my code is decently tested. However, I've lost track of potential good bugs tests since I didn't write a test for a bug that would crop in coding. I think I could reduce the number of tests needed by moving more functions like connected_nodes() and relabel_nodes() into the Graph() class since this would lower the number of inputs a user would have to enter.
5. Initially getting started was probably the most difficult part, followed by testing, and trying to figure out the syntax and tricks of python in general. I usually had a pretty good idea of what I thought I had to do. But sometimes the data structure of a list or how the return value or data structure would force me to rethink things. Though it was not really frustrating since it made me realize that there are some parts of programming that programs like R made me lazy or did not train me well in.