27-networkx

April 24, 2016

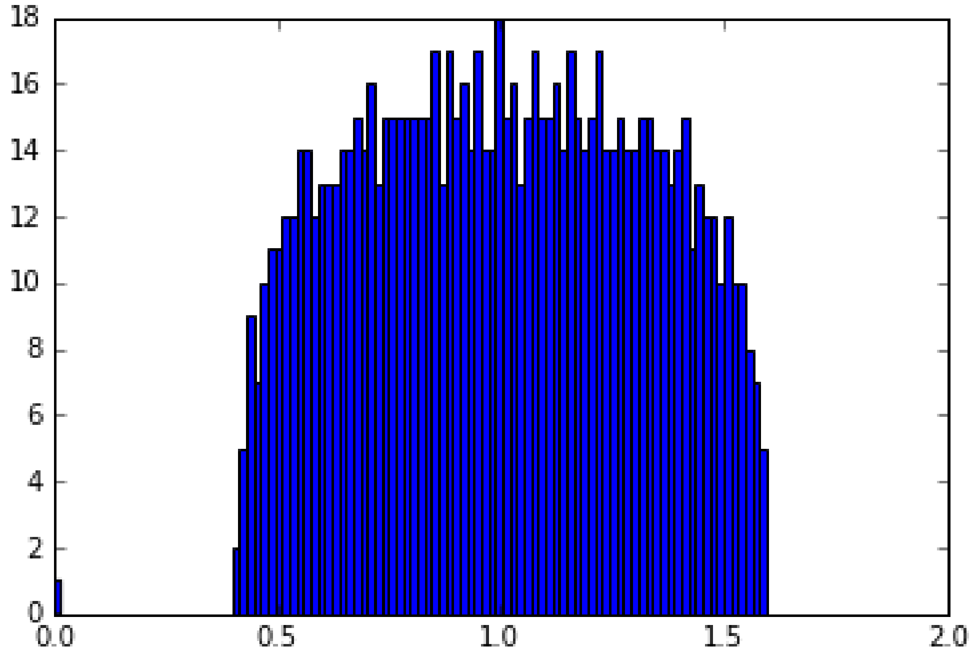
1 Networkx

- comprehensive graph package
- analyse social networks
- algorithms
- drawing techniques
- Note - must install graphviz to draw graphs(it does the layout)

```
In [1]: # Graph types networkx knows about

        import networkx as nx

        [s for s in dir(nx) if s.endswith('graph')]

Out[1]: ['LCF_graph',
         'barabasi_albert_graph',
         'barbell_graph',
         'binomial_graph',
         'bull_graph',
         'caveman_graph',
         'chordal_cycle_graph',
         'chvatal_graph',
         'circulant_graph',
         'circular_ladder_graph',
         'complete_bipartite_graph',
         'complete_graph',
         'complete_multipartite_graph',
         'connected_caveman_graph',
         'connected_watts_strogatz_graph',
         'cubical_graph',
         'cycle_graph',
         'davis_southern_women_graph',
         'dense_gnm_random_graph',
         'desargues_graph',
         'diamond_graph',
         'digraph',
         'directed_havel_hakimi_graph',
         'dodecahedral_graph',
         'dorogovtsev_goltsev_mendes_graph',
         'duplication_divergence_graph',
         'ego_graph',
         'empty_graph',
         'erdos_renyi_graph',
’expected_degree_graph’,
’fast_gnp_random_graph’,
’florentine_families_graph’,
’from_agraph’,
’frucht_graph’,
’gaussian_random_partition_graph’,
’general_random_intersection_graph’,
’geographical_threshold_graph’,
’gn_graph’,
’gnc_graph’,
’gnm_random_graph’,
’gnp_random_graph’,
’gnr_graph’,
’graph’,
’grid_2d_graph’,
’grid_graph’,
’havel_hakimi_graph’,
’heawood_graph’,
’house_graph’,
’house_x_graph’,
’hypercube_graph’,
’icosahedral_graph’,
’is_directed_acyclic_graph’,
’k_random_intersection_graph’,
’karate_club_graph’,
’kl_connected_subgraph’,
’krackhardt_kite_graph’,
’ladder_graph’,
’line_graph’,
’lollipop_graph’,
’make_max_clique_graph’,
’make_small_graph’,
’margulis_gabber_galil_graph’,
’moebius_kantor_graph’,
’multidigraph’,
’multigraph’,
’navigable_small_world_graph’,
’newman_watts_strogatz_graph’,
’null_graph’,
’nx_agraph’,
’octahedral_graph’,
’pappus_graph’,
’path_graph’,
’petersen_graph’,
’planted_partition_graph’,
’powerlaw_cluster_graph’,
’projected_graph’,
’quotient_graph’,
’random_clustered_graph’,
’random_degree_sequence_graph’,
’random_geometric_graph’,
’random_partition_graph’,
’random_regular_graph’,
’random_shell_graph’,
’relabel_gexf_graph’,
’relaxed_caveman_graph’,
’scale_free_graph’,
’sedgewick_maze_graph’,
’star_graph’,
’stochastic_graph’,
’subgraph’,
’tetrahedral_graph’,
’to_agraph’,
’to_networkx_graph’,
’trivial_graph’,
’truncated_cube_graph’,
’truncated_tetrahedron_graph’,
’tutte_graph’,
’uniform_random_intersection_graph’,
’watts_strogatz_graph’,
’waxman_graph’,
’wheel_graph’]
```

```
In [2]: %matplotlib inline
        """
        Create an G{n,m} random graph and compute the eigenvalues.
        Requires numpy and matplotlib.
        """
        import networkx as nx
        import numpy.linalg
        import matplotlib.pyplot as plt

        n = 1000 # 1000 nodes
        m = 5000 # 5000 edges
        G = nx.gnm_random_graph(n,m)

        L = nx.normalized_laplacian_matrix(G)
        e = numpy.linalg.eigvals(L.A)
        print("Largest eigenvalue:", max(e))
        print("Smallest eigenvalue:", min(e))
        plt.hist(e,bins=100) # histogram with 100 bins
        plt.xlim(0,2)  # eigenvalues between 0 and 2
        plt.show()
```

```
Largest eigenvalue: 1.59460347656
Smallest eigenvalue: -1.84079509604e-16
```



```
In [ ]: """
        Example using unicode strings as graph labels.

        Also shows creative use of the Heavy Metal Umlaut:
        http://en.wikipedia.org/wiki/Heavy_metal_umlaut

        """
        __author__ = """Aric Hagberg (hagberg@lanl.gov)"""
        __date__ = ""
        __credits__ = """"""
        __revision__ = ""
        #    Copyright (C) 2006 by
        #    Aric Hagberg <hagberg@lanl.gov>
        #    Dan Schult <dschult@colgate.edu>
        #    Pieter Swart <swart@lanl.gov>
        #    All rights reserved.
        #    BSD license.

        import networkx as NX
        try:
            import pylab as P
        except ImportError:
            pass

        try:
            hd=’H’ + unichr(252) + ’sker D’ + unichr(252)
            mh=’Mot’ + unichr(246) + ’rhead’
            mc=’M’ + unichr(246) + ’tley Cr’ + unichr(252) + ’e’
            st=’Sp’ + unichr(305) + ’n’ + unichr(776) + ’al Tap’
```

```
    q='Queensr' + unichr(255) + 'che'
    boc='Blue ' + unichr(214) +'yster Cult'
    dt='Deatht' + unichr(246) + 'ngue'
except NameError:
    hd='H' + chr(252) + 'sker D' + chr(252)
    mh='Mot' + chr(246) + 'rhead'
    mc='M' + chr(246) + 'tley Cr' + chr(252) + 'e'
    st='Sp' + chr(305) + 'n' + chr(776) + 'al Tap'
    q='Queensr' + chr(255) + 'che'
    boc='Blue ' + chr(214) +'yster Cult'
    dt='Deatht' + chr(246) + 'ngue'

G=NX.Graph()
G.add_edge(hd,mh)
G.add_edge(mc,st)
G.add_edge(boc,mc)
G.add_edge(boc,dt)
G.add_edge(st,dt)
G.add_edge(q,st)
G.add_edge(dt,mh)
G.add_edge(st,mh)

# write in UTF-8 encoding
fh=open('edgelist.utf-8','wb')
fh.write('# -*- coding: utf-8 -*-\n'.encode('utf-8')) # encoding hint for emacs
NX.write_multiline_adjlist(G,fh,delimiter='\t', encoding = 'utf-8')

# read and store in UTF-8
fh=open('edgelist.utf-8','rb')
H=NX.read_multiline_adjlist(fh,delimiter='\t', encoding = 'utf-8')

for n in G.nodes():
    if n not in H:
        print(False)

print(G.nodes())

try:
    pos=NX.spring_layout(G)
    NX.draw(G,pos,font_size=16,with_labels=False)
    for p in pos: # raise text positions
        pos[p][1]+=0.07
    NX.draw_networkx_labels(G,pos)
    P.show()
except:
    pass
```