Graphs, Networks and Python: The Power of Interconnection

Lachlan Blackhall - lachlan@repositpower.com
A little about me...
Graphs

- Graph, $G = (V, E)$
- $V = \text{Vertices} / \text{Nodes}$
- $E = \text{Edges}$
NetworkX

- Native graph structures for Python.
  - Nodes can be any hashable object.
  - Edges are tuples of nodes with optional edge data which is stored in a dictionary.
  - Well maintained package
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
Let's add some nodes

```python
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_node('1')
G.add_node('2')
G.add_node('3')
G.add_node('4')
G.add_node('5')
```
Let's add some edges

```python
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_node('1')
G.add_node('2')
G.add_node('3')
G.add_node('4')
G.add_node('5')
G.add_edge('1', '2')
G.add_edge('2', '3')
G.add_edge('3', '4')
G.add_edge('4', '1')
G.add_edge('4', '5')
```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_node('1')
G.add_node('2')
G.add_node('3')
G.add_node('4')
G.add_node('5')
G.add_edge('1', '2')
G.add_edge('2', '3')
G.add_edge('3', '4')
G.add_edge('4', '1')
G.add_edge('4', '5')
nx.draw_spectral(G)
plt.show()
Now a directed graph

```python
import networkx as nx
import matplotlib.pyplot as plt

G = nx.DiGraph()
G.add_nodes_from(['1', '2', '3', '4', '5'])
G.add_edge('1', '2')
G.add_edge('2', '3')
G.add_edge('3', '4')
G.add_edge('4', '1')
G.add_edge('4', '5')
nx.draw_spectral(G)
plt.show()
```
import networkx as nx

G = nx.DiGraph()
G.add_edge('1', '2')
G.add_edge('2', '3')
G.add_edge('3', '4')
G.add_edge('4', '1')
G.add_edge('4', '5')
More Graphs

• Networkx can generate lots of interesting graphs to experiment with.

• Lets have a look at a few of them.
import networkx as nx
import matplotlib.pyplot as plt

K_5 = nx.complete_graph(5)
nx.draw(K_5)
plt.show()

barbell = nx.barbell_graph(10, 10)
nx.draw(barbell)
plt.show()
Erdos-Renyi

- Named for Paul Erdős and Alfréd Rényi.
- The Erdős–Rényi model is a model for generating random graphs.
- The model sets an edge between each pair of nodes with equal probability.
Erdos-Renyi

```python
import networkx as nx
import matplotlib.pyplot as plt

er = nx.erdos_renyi_graph(100, 0.15)
nx.draw(er)
plt.show()
```
Watts-Strogatz

• Named for Duncan J. Watts and Steven Strogatz.

• The Watts–Strogatz model produces graphs with small-world properties, including short average path lengths and high clustering.

• Most nodes are not neighbours but can be reached from every other node by a small number of hops or steps.
import networkx as nx
import matplotlib.pyplot as plt

ws = nx.watts_strogatz_graph(100, 15, 0.1)
nx.draw(ws)
plt.show()
Barabasi - Albert

- Named for Albert-László Barabási and Réka Albert.
- The Barabási–Albert model is an algorithm for generating random scale-free networks.
- Scale free describes the distribution of node degrees of the network.
import networkx as nx
import matplotlib.pyplot as plt

ba = nx.barabasi_albert_graph(100, 5)
xn.draw(ba)
plt.show()
Social Network Analysis

- One major area of interest in network analysis.
- Networkx is well suited to this type of analysis.
- Interested in understanding graph properties that explain the social interaction.
Enron Data

- Enron declared bankruptcy in 2001
- Enron email corpus contains data from about 150 users, mostly senior management of Enron.
- Converted the corpus to a TSV file of the form

| SENDER | RECIPIENTS | EMAIL DATA ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>str</td>
<td>str,...,str</td>
<td></td>
</tr>
</tbody>
</table>
import csv
import networkx as nx

G = nx.Graph()
Load the email data

```python
import csv
import networkx as nx

G = nx.Graph()

with open('enron.tsv', 'rb') as tsvin:
    tsvin = csv.reader(tsvin, delimiter='\t')
```
import csv
import networkx as nx

G = nx.Graph()

with open('enron.tsv', 'rb') as tsvin:
    tsvin = csv.reader(tsvin, delimiter='\t')

    for row in tsvin:
        sender = row[0]
        recipients = row[1].split(',
')
        for recipient in recipients:
            G.add_edge(sender, recipient)
And visualise!!!

```python
import csv
import networkx as nx

G = nx.Graph()

with open('enron.tsv', 'rb') as tsvin:
    tsvin = csv.reader(tsvin, delimiter='\t')

    for row in tsvin:
        sender = row[0]
        recipients = row[1].split(',

        for recipient in recipients:
            G.add_edge(sender, recipient)

nx.write_gexf(G, 'enron.gexf')
```
So what?

• Visualisations are pretty but...
  • Networkx can help us do better
  • Lets dive into some graph analysis
Node Degree

```python
node_degree = nx.degree(G).items()
sorted_degrees = sorted(node_degree, key=lambda tup: tup[1])
poi = [email for (email, degree) in sorted_degrees[-10:]]

>>> ['tana.jones@enron.com', 'technology.enron@enron.com', '', 'klay@enron.com', 'jeff.skilling@enron.com', 'david.forster@enron.com', 'jeff.dasovich@enron.com', 'outlook.team@enron.com', 'sally.beck@enron.com', 'kenneth.lay@enron.com']
```
The People

• Kenneth Lay - Chairman
• Jeff Skilling - CEO
• Tana Jones - Senior Legal Specialist
• Sally Beck - COO
• David Forster - Vice President
• Jeff Dasovich - Government Relation Executive
Subgraphs

real.poi = ['tana.jones@enron.com', 'klay@enron.com', 'jeff.skilling@enron.com', 'david.forster@enron.com', 'jeff.dasovich@enron.com', 'sally.beck@enron.com', 'kenneth.lay@enron.com']

sub_G = G.subgraph(real.poi)
nx.draw(sub_G)
plt.show()
MultiGraphs

- Allows you to have multiple edges between nodes. i.e. one edge per email.

- Instead of
  \[
  G = nx.Graph()
  \]

- Use
  \[
  G = nx.MultiGraph()
  \]
Relationships?

- **Neighbours**
  
  \[
  G\text{.neighbors('kenneth.lay@enron.com')}
  \]
  
  \[
  G\text{.neighbors_iter('kenneth.lay@enron.com')}
  \]

- **Cliques**
  
  \[
  nx\text{.find_cliques}(G)
  \]
NetworkX Functionality

NetworkX is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

Features
- Python language data structures for graphs, digraphs, and multigraphs.
- Nodes can be "anything" (e.g. text, images, XML records)
Other useful graph Packages

- Gephi (http://gephi.github.io/)
- Pajek (http://pajek.imfm.si/doku.php)
Conclusion

- Graphs are good
- Networkx is awesome
- Have fun :}

Questions

???