COMP2200/COMP6200 Practical Exercise (Week 11): Network Analysis

School of Computing

Preparation

Choose your environment: either run locally with `uv` or use Google Colab.

1. **Local with `uv`**:
   (a) Install `uv`.
   MacOS/Linux: `curl -LsSf https://astral.sh/uv/install.sh | sh`
   Windows: `powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"`
   (b) Create a folder for this practical (for example, `week11-prac`) and open a terminal in it.
   (c) Set up your environment:

   ```
   uv init
   uv add networkx matplotlib pandas openpyxl
   ```

   (d) Launch Jupyter with `uv run --with jupyter jupyter lab` (or `... notebook`).
2. **Google Colab**:
   (a) Open Colab and create a new notebook.
   (b) Upload `TopRoutesJuly2014Jul2025.xlsx` from iLearn or this week's zip file.
   (c) Install packages with `!pip install networkx matplotlib pandas openpyxl`.

Part A — Class network exercise (30 min)

In this practical, we will form a small network using the connections between students in this prac, and then we'll look at a much bigger network (passenger routes between Australian airports).

A1. Import required libraries

First import the Python modules we need: we'll need `math`, `networkx`, `pandas` and `matplotlib`.

```
import networkx as nx
import matplotlib.pyplot as plt
import pandas as pd
import math
```

A2. Collect class network data

Unless your SGTA leader has created another spreadsheet for you, use the following spreadsheet.

`https://mqoutlook-my.sharepoint.com/:x:/g/personal/greg_baker_mq_edu_au/EZ0yJXjL_X5LjDXzugsItLYB0qkxoyNFC9exHXFyYyg?e=dOFU3t`

Look for the worksheet tab with your prac group. Add a row with a pseudonym and the unit codes you are studying this semester (not including COMP2200/COMP6200).

When everyone has finished entering data, download the spreadsheet (File → Create a copy → Download).

Using `pandas`, load the spreadsheet and build a graph where each student is a node and an edge exists when two students share a unit other than COMP2200/COMP6200.

```
# Load the spreadsheet (adjust filename and sheet name as needed)
class_df = pd.read_excel('class_network.xlsx', sheet_name='Wed 11am prac')
```

If you get an error about `openpyxl` not being available, fix that with whatever tool you use to manage packages (`pip install openpyxl` or `uv add openpyxl` or `conda install openpyxl`).

A3. Build the class network graph

These are the columns that contain each student's units:

```
unit_cols = ['Unit1', 'Unit2', 'Unit3', 'Unit4']
```

Create an empty graph:

```
G_classes = nx.Graph()
```

Add an edge between a student and each unit they study. You can do this by iterating over every element in `unit_cols` and creating an edge between the `'Name'` and that unit column.

```
for unit in unit_cols:
    for name, u in zip(class_df['Name'], class_df[unit]):
        G_classes.add_edge(name, u)
```

A4. Find paths in the network

Use `nx.shortest_path` to find the path between yourself and another student.

```
path = nx.shortest_path(G_classes, 'YourName', 'TheirName')
print('Shortest path:', path)
```

A5. Visualise the class network

Create a layout for the class network and draw it using Matplotlib.

```
pos_class = nx.spring_layout(G_classes)
plt.figure(figsize=(8,8))
nx.draw_networkx(G_classes, pos=pos_class, node_size=20, alpha=0.7)
plt.title('Class network')
plt.show()
```

A6. Characterise the network

Calculate the average clustering coefficient and the average shortest path length in the network, and calculate what N and $\log(N)$ are – N is the number of students and units (in total).

```
avg_clust_class = nx.average_clustering(G_classes)
print('Class network average clustering coefficient:', avg_clust_class)

if nx.is_connected(G_classes):
    avg_shortest_class = nx.average_shortest_path_length(G_classes)
else:
    subgraph = G_classes.subgraph(max(nx.connected_components(G_classes), key=len))
    avg_shortest_class = nx.average_shortest_path_length(subgraph)
print('Class network average shortest path length:', avg_shortest_class)

print('N: ', G_classes.number_of_nodes())
print('log(N): ', math.log(G_classes.number_of_nodes()))
```

Based on these metrics for your class network, does it look more like:

- a **random graph** (low clustering, average path around $\log(N)$)
- a **small world** network (high clustering, average path around $\log(N)$)

- a **chain** (average path grows with N, low clustering)
- or **other**?

Part B — Airport network analysis (50 min)

Downloaded from `https://www.bitre.gov.au/publications/ongoing/domestic_airline_activity-time_series`

B1. Load the airport data

Read the Excel file (`TopRoutesJuly2014Jul2025.xlsx`) that lists the top airline routes. You'll need to specify the sheet name as "Top Routes" and skip the first 12 rows.

```
raw_route_information = pd.read_excel('TopRoutesJuly2014Jul2025.xlsx',
                                      sheet_name='Top Routes',
                                      skiprows=12)
raw_route_information
```

B2. Clean the flight data

There is a column called "City Pair Route.1" which has the origin and destination airports.

- Drop the nulls
- Occasionally it has a newline character `'\n'` instead of a space, so replace those
- Drop the duplicates

```
flights = raw_route_information['City Pair Route.1'].str.replace('\n', ' ').drop_duplicates().dropna()
flights
```

B3. Build the airport network

Create a graph for the airport network.

```
airport_network = nx.Graph()
```

Add an edge for each flight. You will need to iterate over the values in the flights column and split the string on `' - '`

```
for flight in list(flights):
    source, dest = flight.split(' - ')
    airport_network.add_edge(source, dest)
```

Check how many airports and routes are in the network:

```
airport_network.number_of_nodes(), airport_network.number_of_edges()
```

B4. Visualise the airport network

Create a layout for the network visualization.

```
pos = nx.spring_layout(airport_network)
```

Draw the nodes and edges using Matplotlib.

```
plt.figure(figsize=(12,12))
nx.draw_networkx(airport_network, pos=pos, node_size=10,
                 alpha=0.5, with_labels=True)
plt.title('Australian Airport Network')
plt.show()
```

B5. Average clustering and path length

Calculate the average clustering coefficient.

```
avg_clust = nx.average_clustering(airport_network)
print('Average clustering:', avg_clust)
```

Find the average shortest path length. For disconnected graphs, use the largest connected component.

```
if nx.is_connected(airport_network):
    avg_shortest = nx.average_shortest_path_length(airport_network)
else:
    subgraph = airport_network.subgraph(
        max(nx.connected_components(airport_network), key=len)
    )
    avg_shortest = nx.average_shortest_path_length(subgraph)
print('Average shortest path length:', avg_shortest)
```

Calculate N and $\log(N)$:

```
print(airport_network.number_of_nodes())
print(math.log(airport_network.number_of_nodes()))
```

Based on the last two outputs, does this airport network look more like:

- a **random graph** (low clustering, average path around $\log(N)$)
- a **small world** network (high clustering, average path around $\log(N)$)
- a **chain** (average path grows with N, low clustering)
- or **other**?

B6. Is it a scale-free (ultra small world) distribution?

Calculate the degree of each airport and plot the distribution on a log scale.

```
degrees = [d for _, d in airport_network.degree()]
degrees_sorted = sorted(degrees, reverse=True)
plt.figure()
plt.loglog(degrees_sorted, marker='o', linestyle='None')
plt.xlabel('Rank')
plt.ylabel('Degree')
plt.title('Degree Distribution (log-log)')
plt.show()
```

Does the log-log graph look linear? If so, this suggests a scale-free network with a power-law degree distribution.

Part C — Centrality measures (30 min)

C1. Degree centrality

Compute degree centrality. It returns a dictionary; it is often convenient to turn it into a Series and only show the top few.

```
pd.Series(nx.degree_centrality(airport_network)).nlargest(5)
```

C2. Betweenness centrality

Compute betweenness centrality.

```
pd.Series(nx.betweenness_centrality(airport_network)).nlargest(5)
```

C3. Closeness centrality

Compute closeness centrality.

```
pd.Series(nx.closeness_centrality(airport_network)).nlargest(5)
```

C4. PageRank

Compute PageRank.

```
pd.Series(nx.pagerank(airport_network)).nlargest(5)
```

C5. Eigenvector centrality

Compute eigenvector centrality.

```
pd.Series(nx.eigenvector_centrality(airport_network)).nlargest(5)
```

C6. Discussion questions

1. Based on the centrality measures above, which airport is the most **central** or influential in this network?
2. There are differences in the **second** most central node. What is that reflecting?

Part D — Ego networks and communities (20 min)

D1. Ego network of the center of the network

We can get a bit of an understanding of the core of the network by picking a very central node and then creating an ego network from it.

```
ego_net = nx.ego_graph(airport_network, 'Sydney')
plt.figure(figsize=(6,6))
pos = nx.spring_layout(ego_net)
nx.draw_networkx(ego_net, pos=pos, node_size=20,
                 alpha=0.7, with_labels=True)
plt.title(f'Ego network for Sydney')
plt.show()
```

D2. Try an alternate visualisation

Draw the same ego network using a shell layout.

```
nx.draw_shell(ego_net, node_size=10)
```

D3. Community detection

Sometimes we don't want to focus on the core of the network – sometimes we want to get an overview of the whole network.

Run greedy modularity to find communities.

```
from networkx.algorithms import community
communities = community.greedy_modularity_communities(airport_network)
```

Print the number of communities and their members.

```
print('Number of communities:', len(communities))
for c in communities:
    print(c)
```

Do the detected communities make sense geographically? Do they cluster by region (e.g., Queensland routes, NSW routes, WA routes)?

Discussion Questions

1. How do the class network and airport network differ in their structure?
2. Which centrality measure do you think best captures "importance" for airports? Why?
3. What does the community structure tell you about Australian domestic aviation?
4. If you were to remove the most central airport from the network, how would that affect connectivity?

Quiz questions

As usual, carry on with some quiz questions.