Social Network Analysis

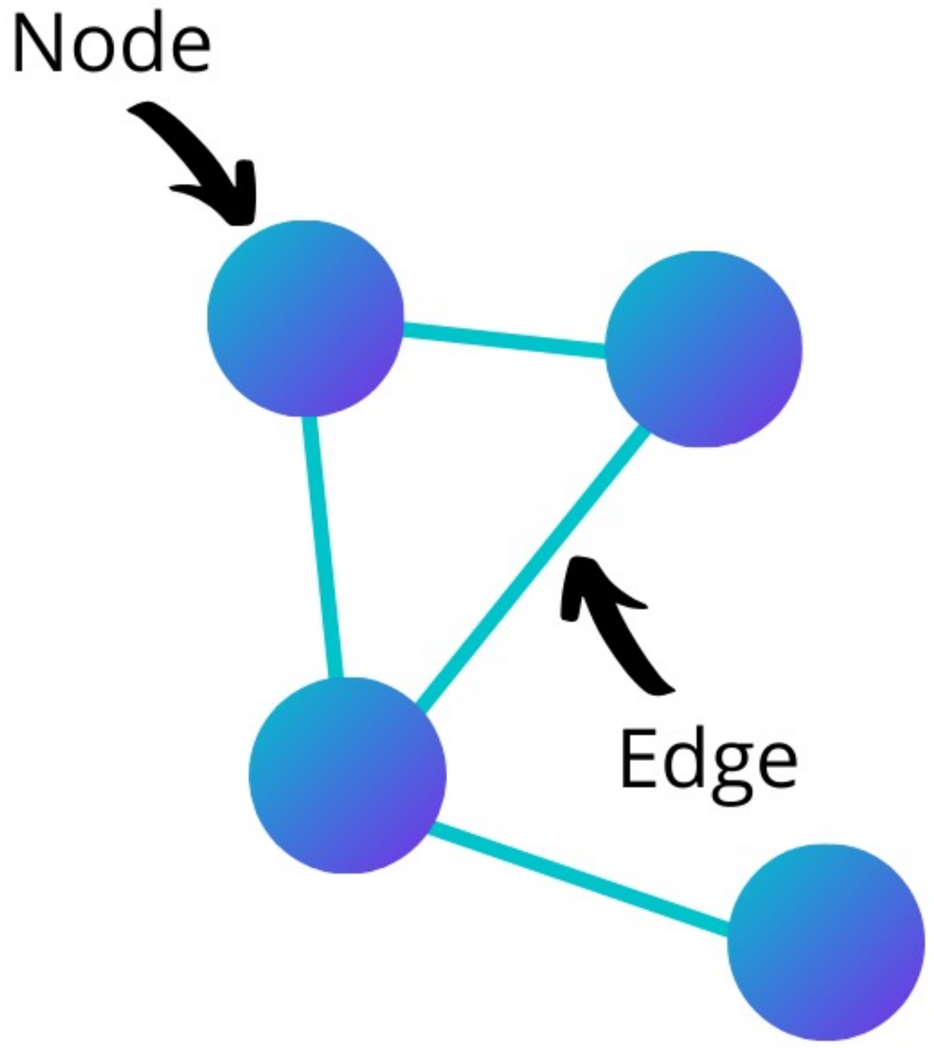
CLASS 2: NETWORK ANALYSIS CONCEPTS

Network Analysis Concepts

- Before we start working with networks in R (which we'll do next class), we should look at some of the core concepts and terminology of network analysis.
- These are mostly ideas which come from the mathematical study of graph theory, but some of them are also specific to social science uses of networks.

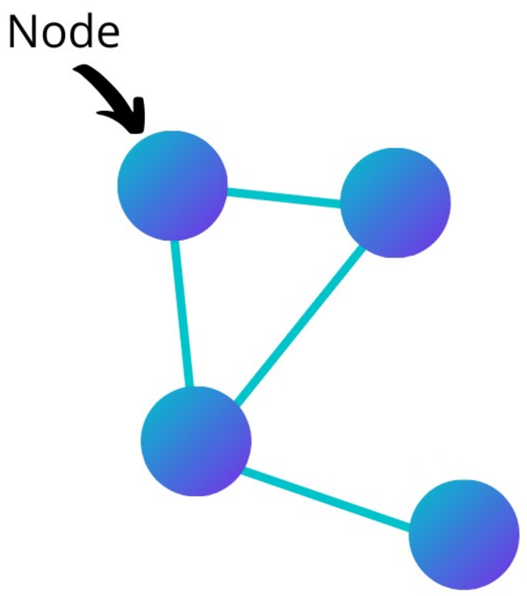
Recap: Nodes and Edges

- As we saw yesterday, networks are made up of **nodes** and **edges**.
- Nodes (or *vertices*) are the points on a network.
- Edges (or *ties*) are the connections between points.



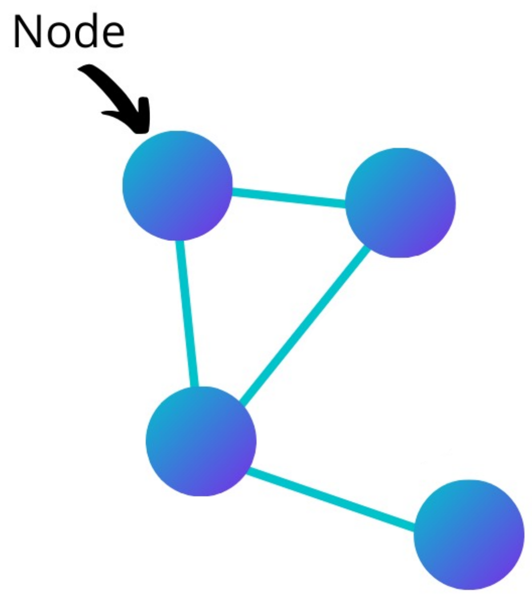
Nodes

- A **node** can represent just about anything. Common examples are…
  - People
  - Internet accounts
  - Bank accounts
  - Companies
  - Countries
  - Academic papers
  - Words



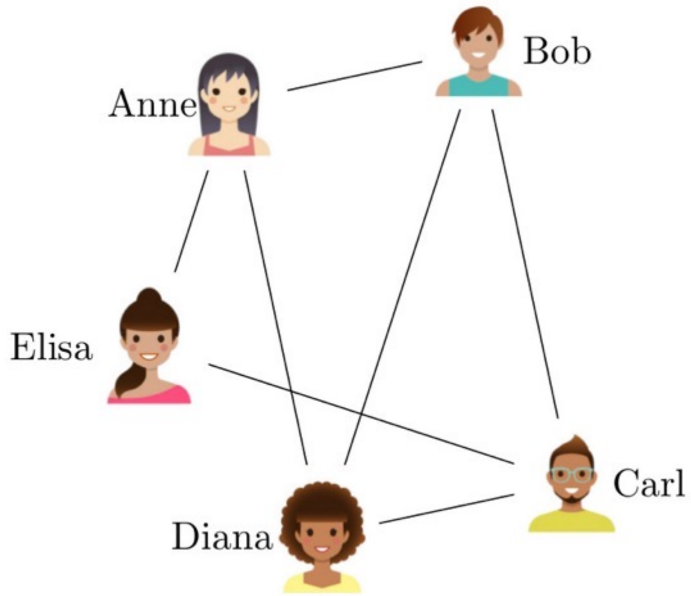
Edges

- An **edge** is a connection or interaction between nodes. For example…
  - A friendship
  - A transaction (financial etc.)
  - Shared membership in a group
  - A family relationship
  - A paper citing another paper
  - A treaty or contract
  - Words co-occurring in a text



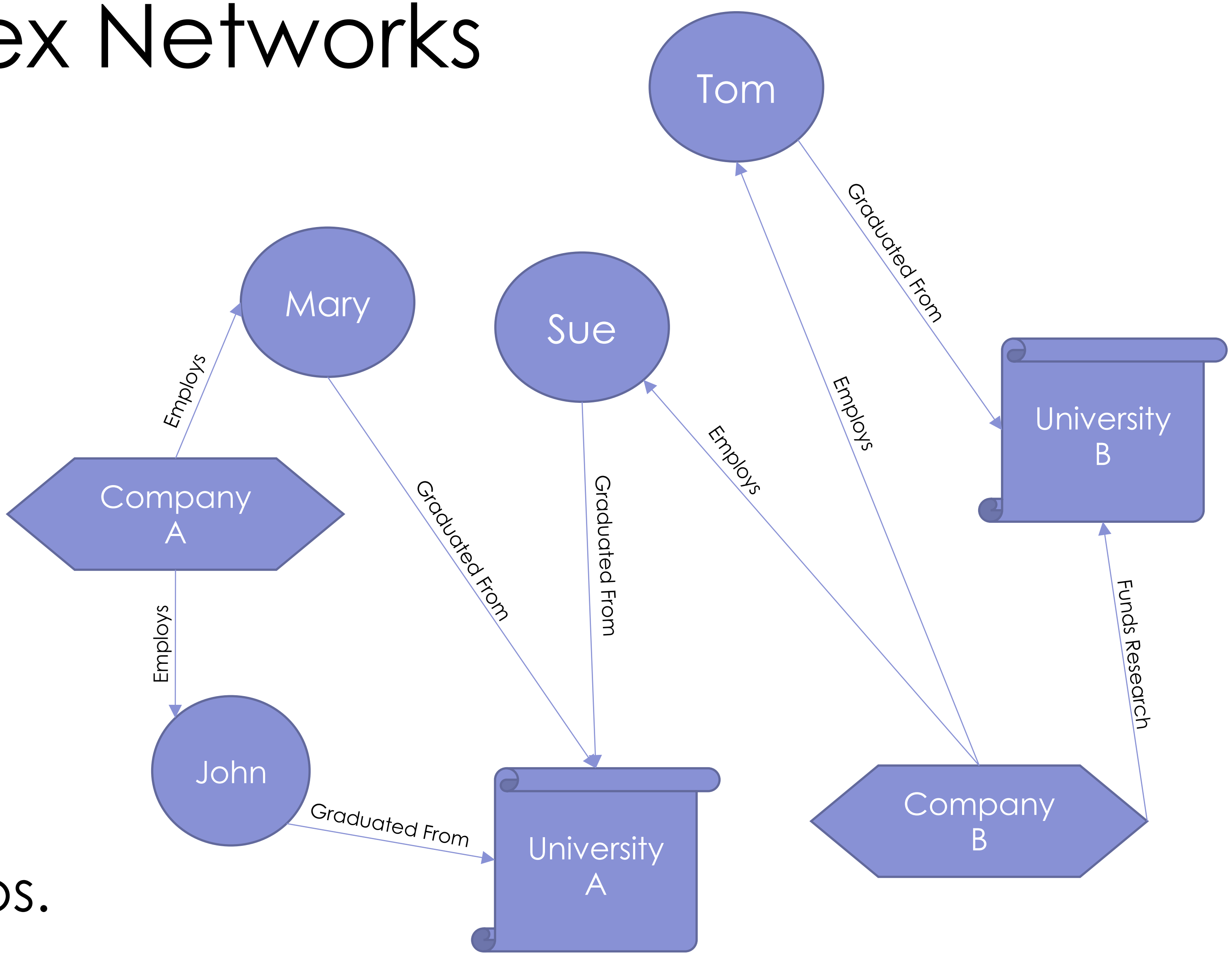
Simple Networks

- In a simple network, every **node** and **edge** is of the same type.
  - In other words, there's just one type of object, and only one type of connection/interaction between them.
- For example, in this simple network, every node is a person, and every edge is a friendship.



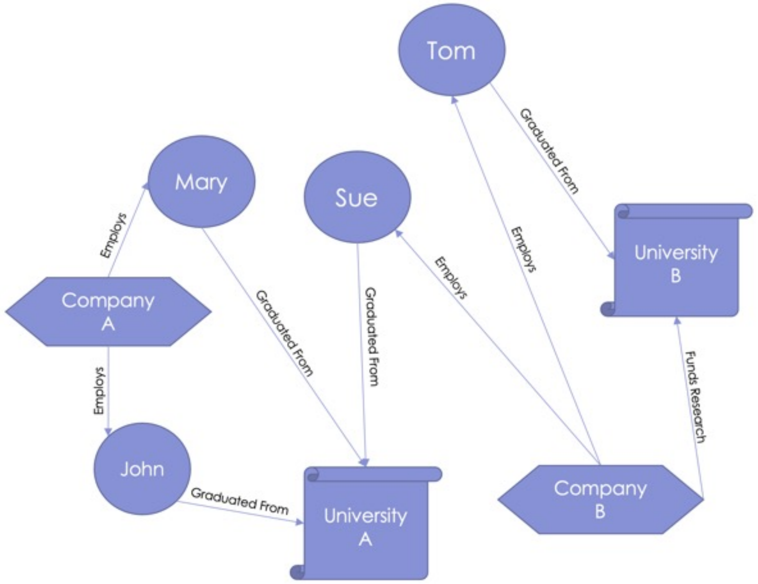
More Complex Networks

- However, it’s possible to create networks with multiple different kinds of node, and with edges that represent different kinds of relationships.



Complex Networks

- Every every **node** and **edge** in the network can have some data associated with it – called **properties** or **attributes**.
- For example, you might store properties in each node which tell you what kind of object it represents (e.g. a person, a company or a university).
- Similarly, an edge could contain properties telling you what kind of relationship it represents (e.g. employment, graduation or funding ties).

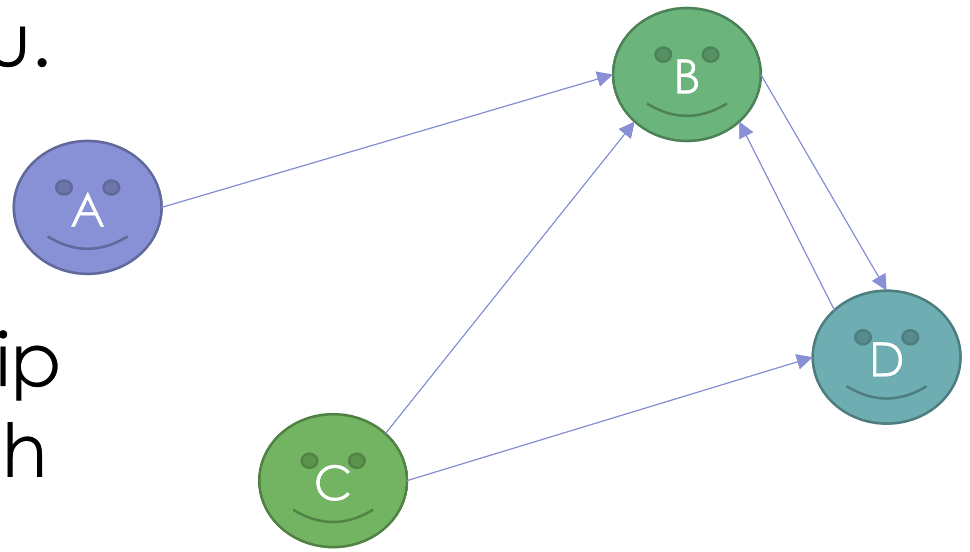


Edge Properties: Directionality

- Edges have two notable properties which can have a major impact on the structure of the network.
- **Directionality** tells you whether the edge goes in a specific direction, or is bi-directional.
- Some networks are **undirected** – meaning all edges just link two nodes together.
- Others are **directed** – so the relationship between nodes may not be reciprocal.

Edge Directionality

- Twitter is a good example of a directed network.
- Follow relationships are **unidirectional** – you don't have to follow everyone who follows you.
- In this graph, A, C and D all follow B.
- The only **bidirectional** relationship is B and D, who both follow each other.

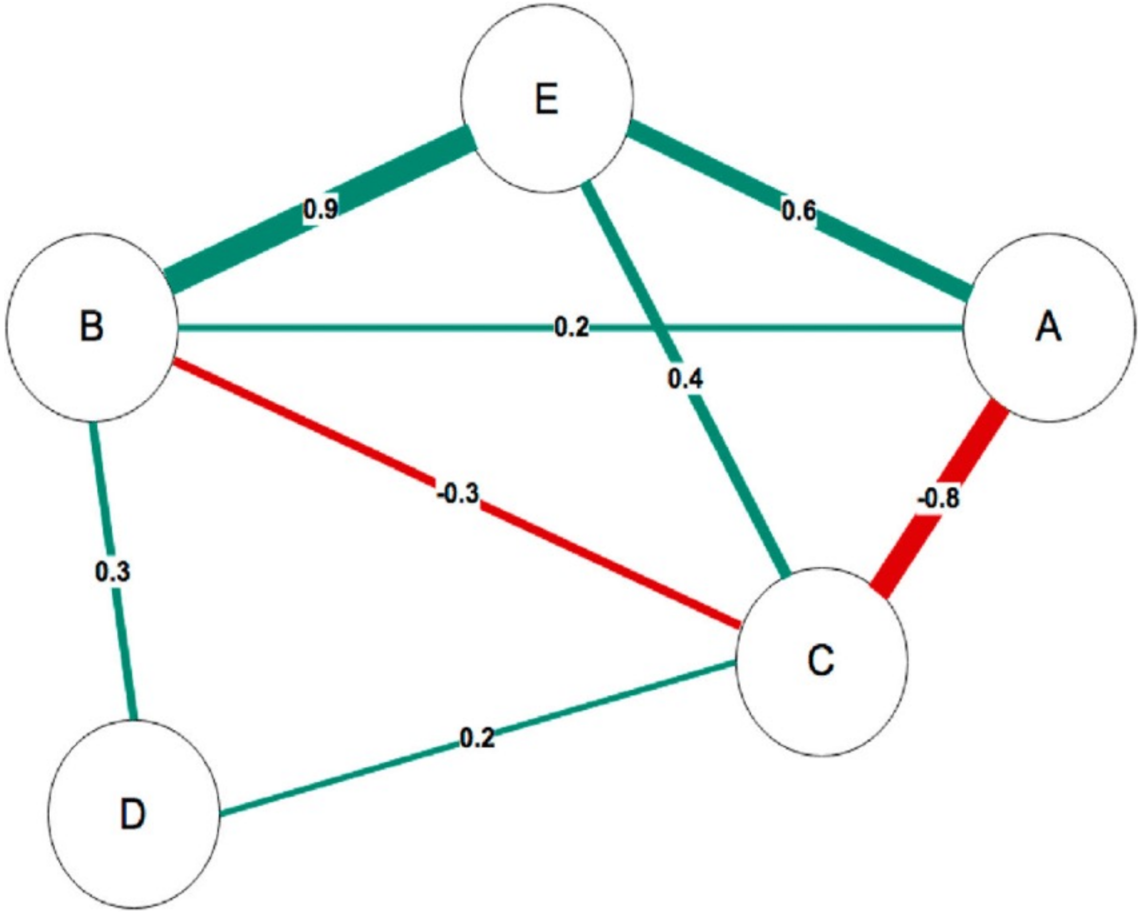


Edge Weight

- The other important property of an edge is its **weight**.
- Some graphs have **uniform edges**, which means every edge is equal to every other edge – for example, Twitter following.
- Others have **weighted edges**.
- For example, in a graph of interactions in a company, the number of times people emailed each other might be the **weight** of the edge between them.

Edge Weight

- In this network, edges have been given a **weight** indicating the strength or magnitude of the relationship.
- Representing these with the thickness of the line is common.
- These weights can be very important for analysis – they can significantly change the **network topography**.



Network Topography

- The **topography** of a network is essentially its **physical structure** – the virtual shapes and geography created by the links in the network.
- We often talk about network analysis in physical terms.
  - We might refer to **closeness**, saying some nodes are “closer together” and others are “further apart”;
  - We may discuss “**clusters**” of nodes, which are groups that are close together in the graph.
  - Sometimes, **visual examination** of a network graph is a key part of analysis.

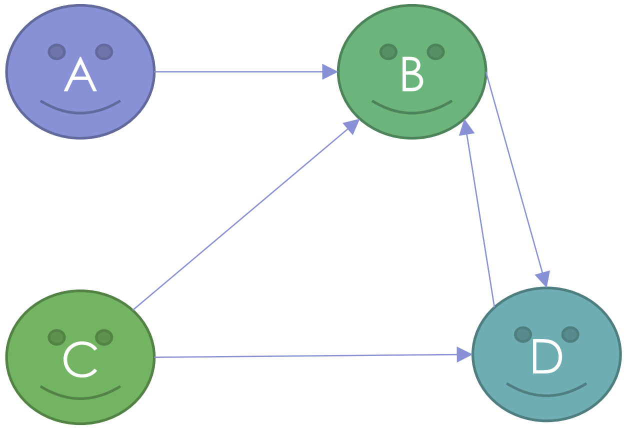
Representing Networks

- R and other statistical analysis software usually handles *variables*, *lists* and *dataframes* – none of which seem like a particularly good fit for network data.

- In general, therefore, we need to use a more “tabular” form to store and process network data. There are two very common solutions to this problem:
  - An **Adjacency Matrix**
  - An **Edgelist**

Adjacency Matrix

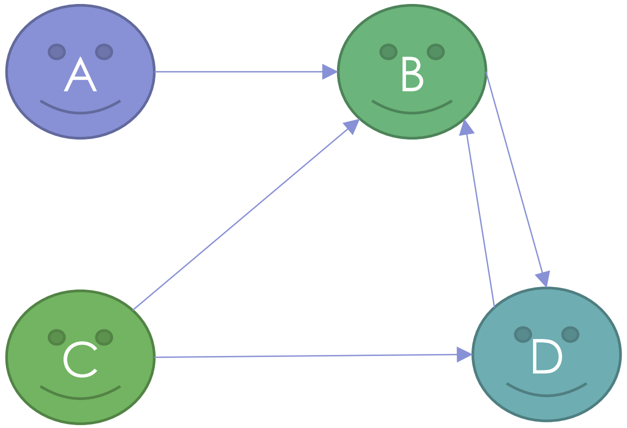
- An adjacency matrix is a two-dimensional table which holds the connections between nodes.
- **Adjacent** nodes – those which are connected to each other – have a non-zero value.
- This kind of data structure can store **edge weights** (by using values other than 1 and 0) and represent both **directed** and **undirected** edges.
  - For example, in this **directed graph**, A is connected to B but B is not reciprocally connected to A. Can you see how that's represented in the adjacency matrix?



	A	B	C	D
A	0	1	0	0
B	0	0	0	1
C	0	1	0	1
D	0	1	0	0

Edgelists

- An **edgelist** is another way of representing network data.
- It consists of a list of edges – one per row – defined by their starting and ending nodes.
- To store **edge weights** or other information, you just put extra columns in the table.
- This can also be used for both directed and undirected graphs.



1	A	B
2	B	D
3	C	B
4	C	D
5	D	B

Which approach to use?

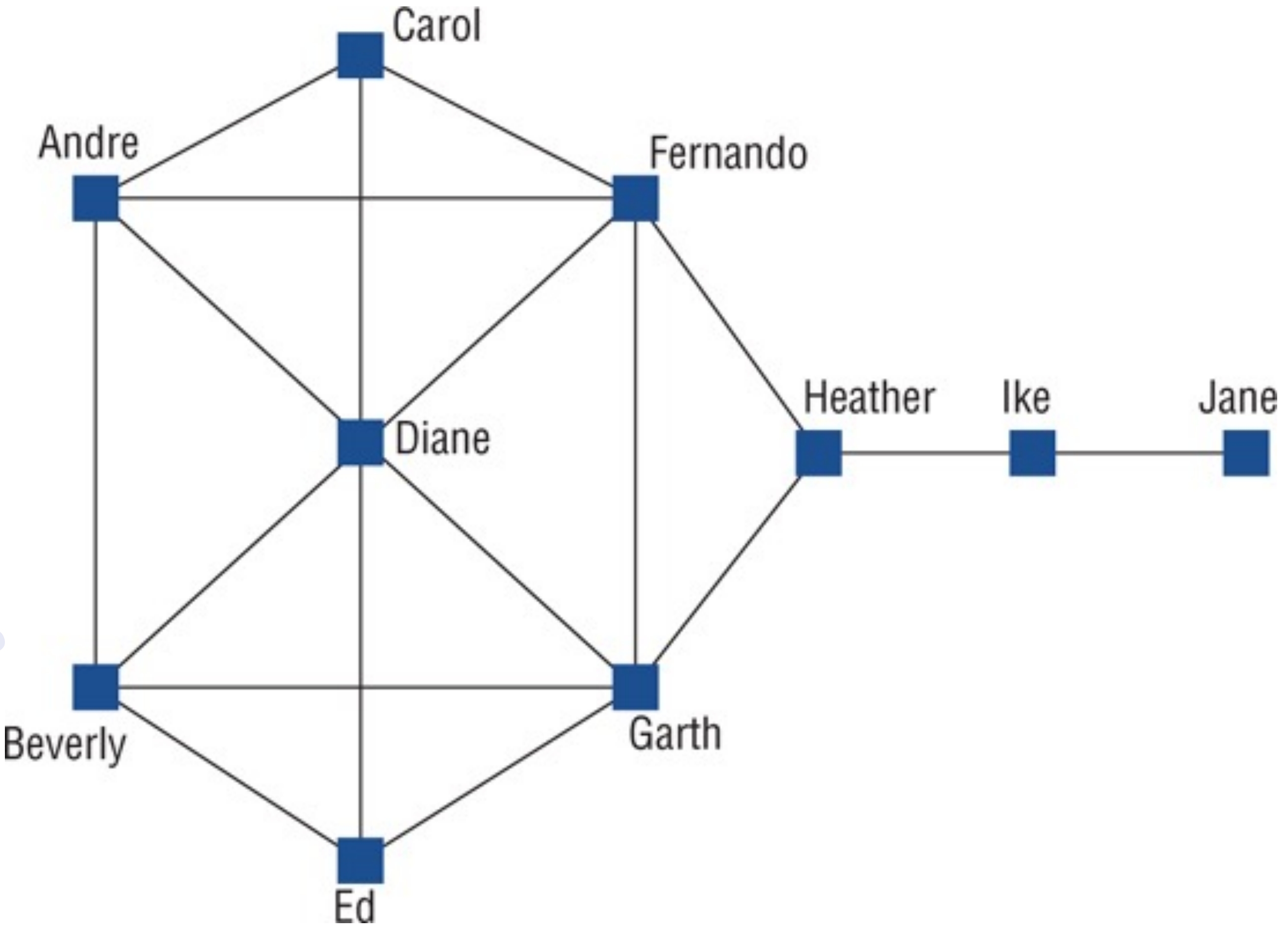
- There are times when an **adjacency matrix** can be useful, but in general, the **edgelist** is the easiest format to work with.
  - It’s also much easier to store in a database – which can be a must for network analysis if you’re working with big data.
- It’s now common for network data to be distributed as two CSV files – an **edgelist** and a **nodelist**.
  - Sometimes you only get an edgelist, which means you’ll have to use R to find all the unique nodes it contains and generate your own nodelist.

Analysing Networks

- Once we've constructed our network, the most basic kind of analysis we can carry out involves looking at the properties of the **nodes**.
- There are a number of measurements which allow us to see what **role** each node plays in the network.
  - These are all essentially measurements of different aspects of a node's **importance**, or **centrality**, to the network structure.

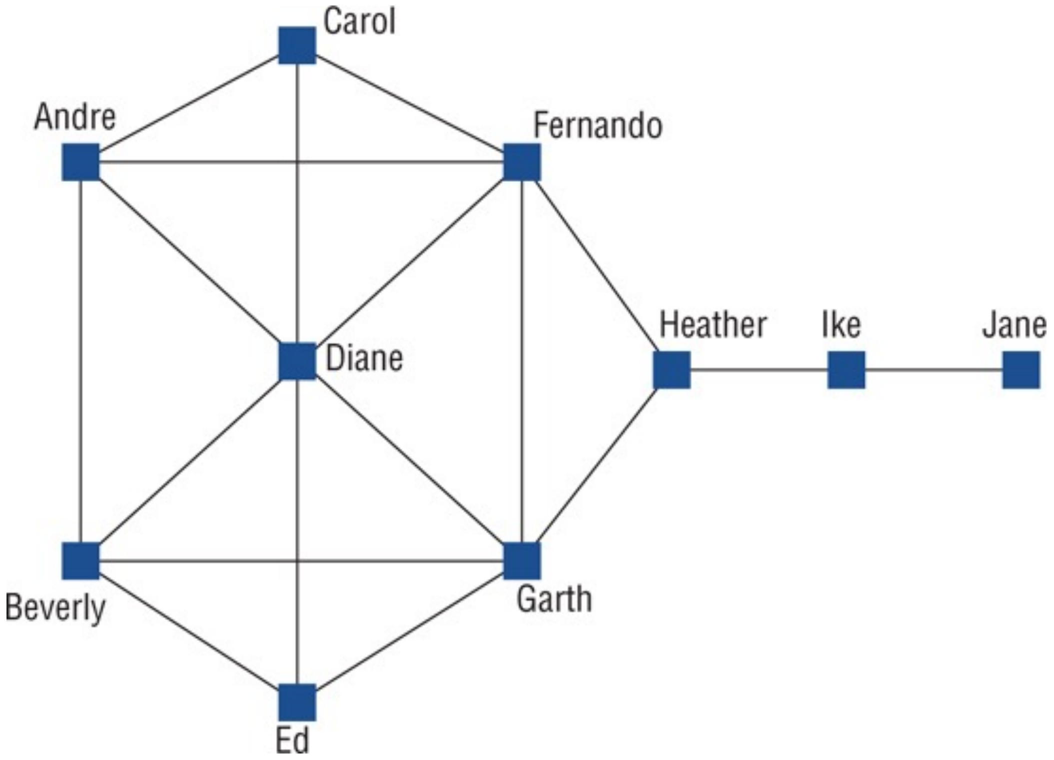
The Kite Network

- Originally created by social networks researcher David Krackhardt, the Kite Network is a famous example of a network that demonstrates all of the different measures of **centrality**.



Centrality

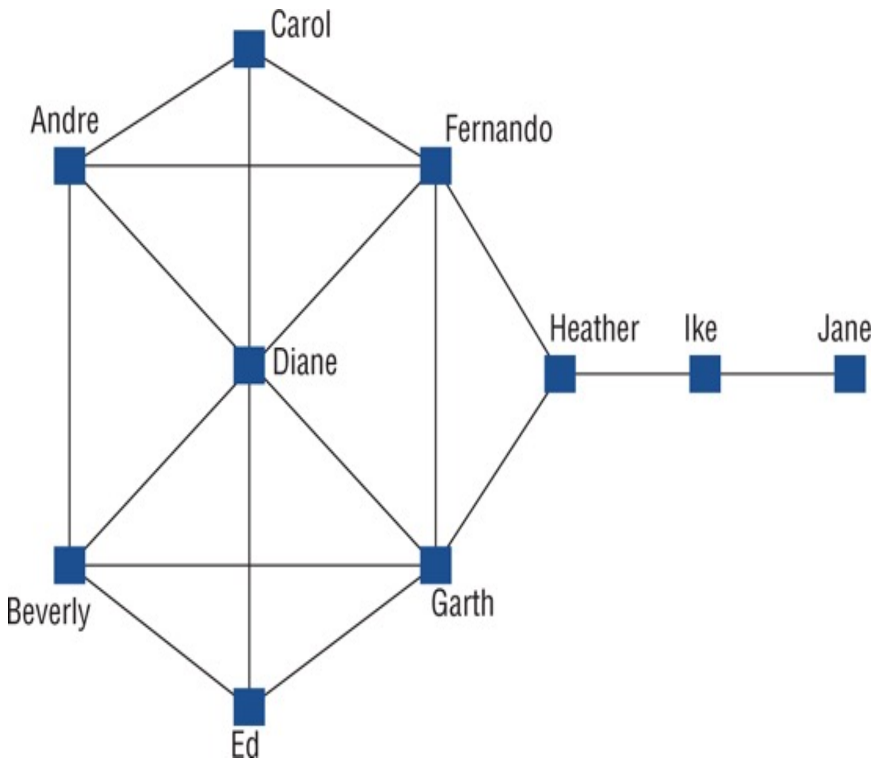
- For each node, we can calculate four different types of **centrality:**
  - **Degree Centrality**
  - **Betweenness Centrality**
  - **Closeness Centrality**
  - **Eigenvector Centrality**



- Each of these has a different meaning for the type of role which the node plays in the network.

Degree

- **Degree** is a measure of how many direct connections a node has to other nodes.
- In the Kite Network, **Diane** has the highest degree centrality (6), because she has the most connections to others.
  - However, note that the nodes Diane is connected to are mostly also connected to one another. She's central only within her community.

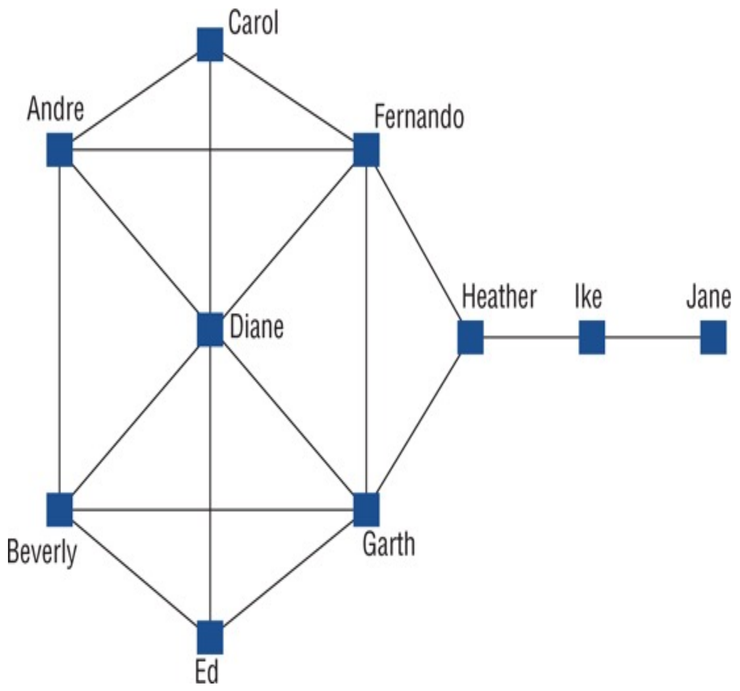


Note: Degree vs. Strength

- You may also sometimes see a reference to a node's **strength**.
- This is a measurement similar to **degree**, but which takes into account the **weights** of the edges that connect to the node.
  - In an unweighted graph, there's no such thing as "strength".

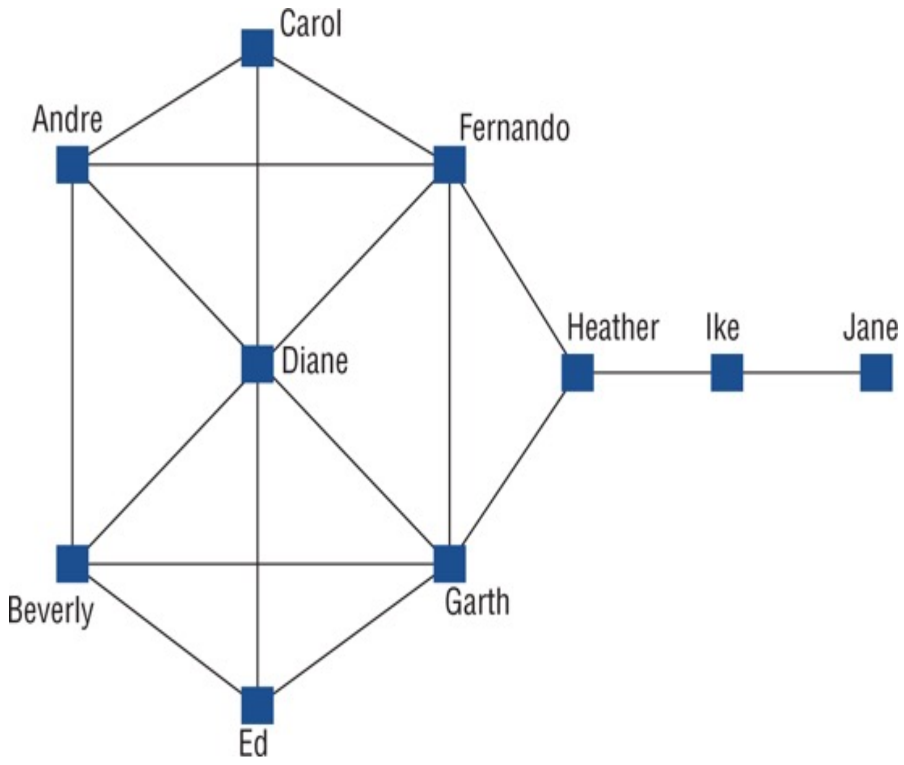
Betweenness

- **Betweenness** is the measurement of the extent to which a node forms the connection between other nodes in the network.
  - It's measured by calculating all the **shortest paths** between nodes and counting how many of them pass through a given node.
- **Heather** has fewer connections than Diane – but she connects all parts of the graph and has the highest **betweenness**.
  - **Betweenness** is important in the study of information flows. Nodes with high betweenness are crucial to spreading information across borders between different groups.



Closeness

- **Closeness** is a measure of the average length of the path between a given node and every other node on the graph.
- **Fernando** and **Garth** can access every other node in only two "hops" – lower than any other node on the network.
    - Closeness can also be useful in understanding information flows; nodes with high closeness have access to information from many different groups.

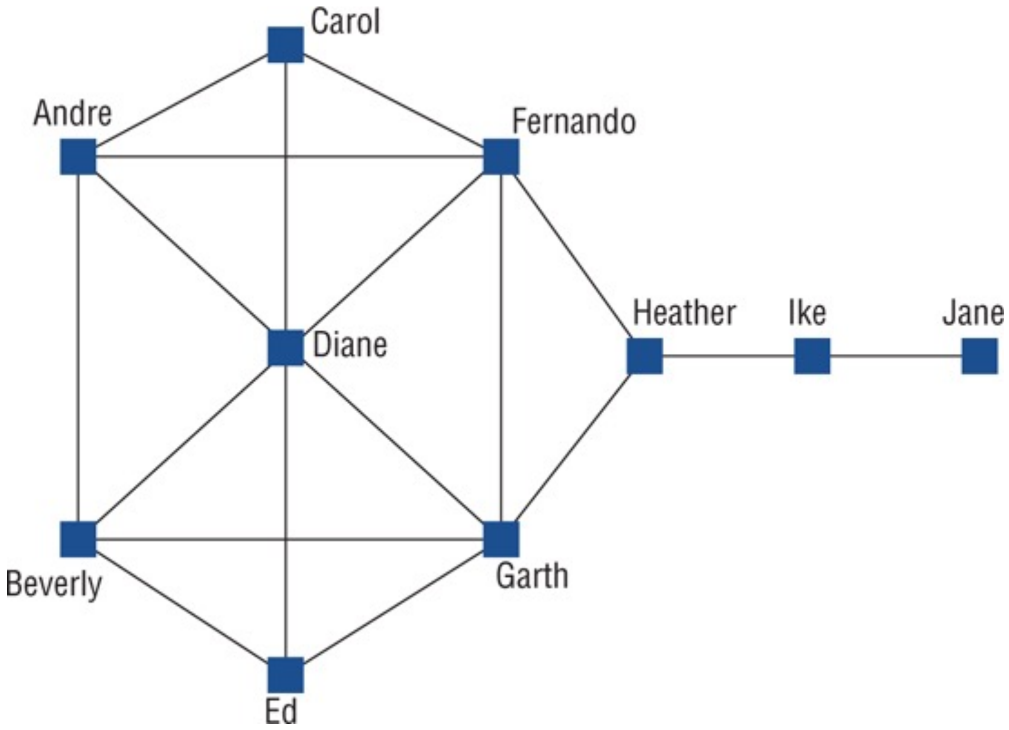


Eigenvector Centrality / Eigencentrality

- **Eigenvector Centrality** (sometimes called **prestige**) is a more complex way to measure centrality.
- It's based on the idea that nodes should score higher for being connected to other nodes with high scores.
- A similar concept is behind a measurement called **Page Rank**, which was developed by Google and estimates the probability that information sent across the network will pass through a given node.

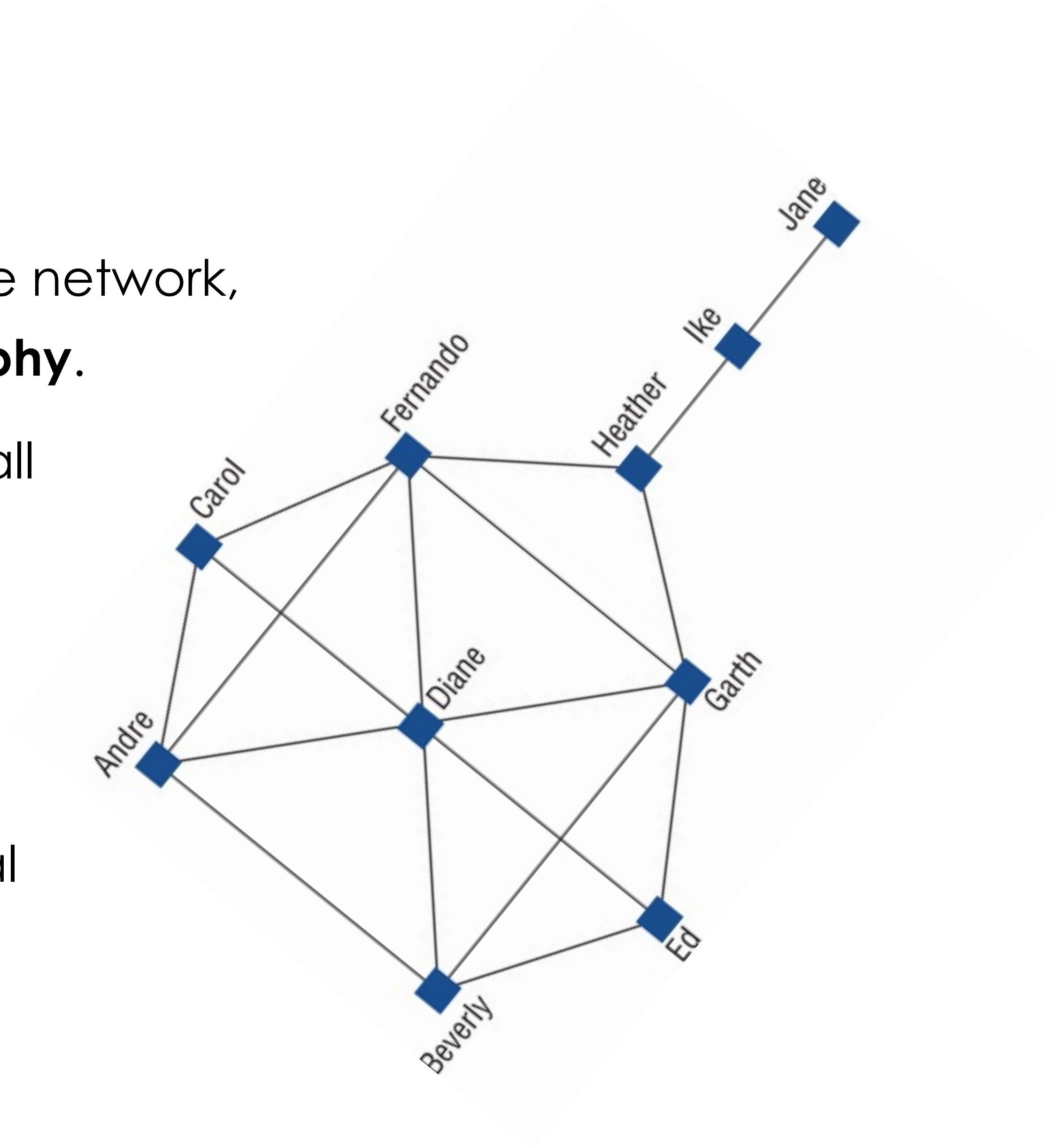
Comparing Centrality Measurements

- If you calculate centrality for each node in the Kite Network, **Diane**, **Heather**, **Fernando** and **Garth** would all have very high scores depending on what measurement you use.
- There is no "right" measure of centrality – it depends on what you're trying to find out!



Network Centrality

- By looking at centrality scores across the whole network, we can start to describe the **network topography**.
- A **highly centralised** network is reliant on a small number of central nodes for its connections.
  - In this case, if Heather disappeared, the network would effectively be split in two.
- A **decentralised** network is one that is not dominated by a small number of highly central nodes. Information flows more widely over it.



Size and Density

- **Network Size** is a simple measurement of how many nodes there are in a network.
    - It doesn't take into account the number of edges.
- **Network Density** is the number of edges divided by the number of *possible* edges (i.e. the number of edges if every node was connected to every other node).
    - Let n = the number of nodes in the graph.
      You can calculate the number of possible edges as:

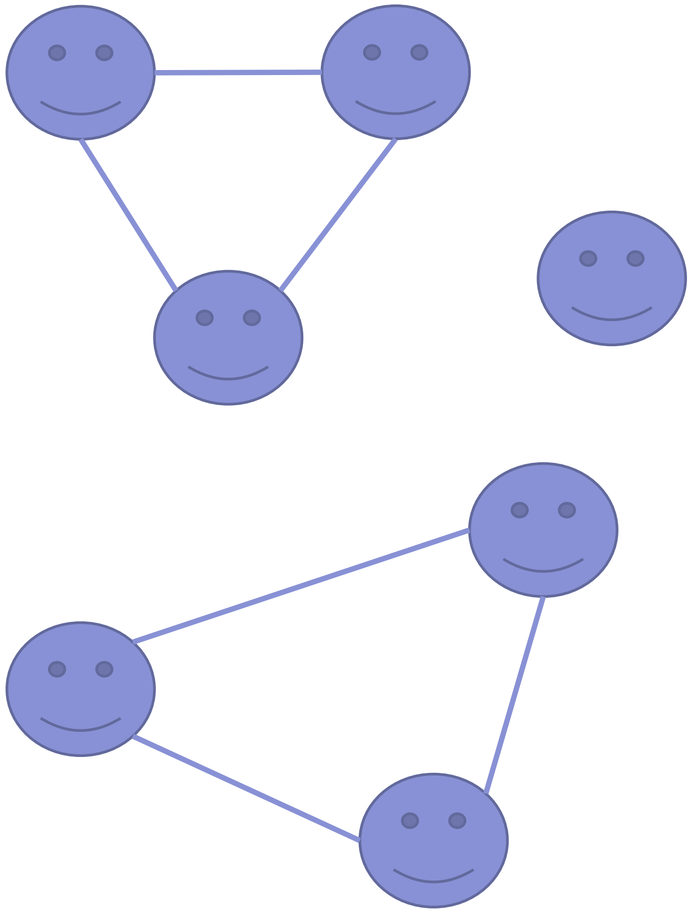
$$\frac{n(n-1)}{2}$$

Diameter and Mean Distance

- **Diameter** is another measure of the network's size – it's the length of the longest path between two nodes on the network.
- **Mean Distance** measures the average length of the paths between nodes.
- Taken together with **size** and **density**, these can help to understand what kind of network you're analysing.

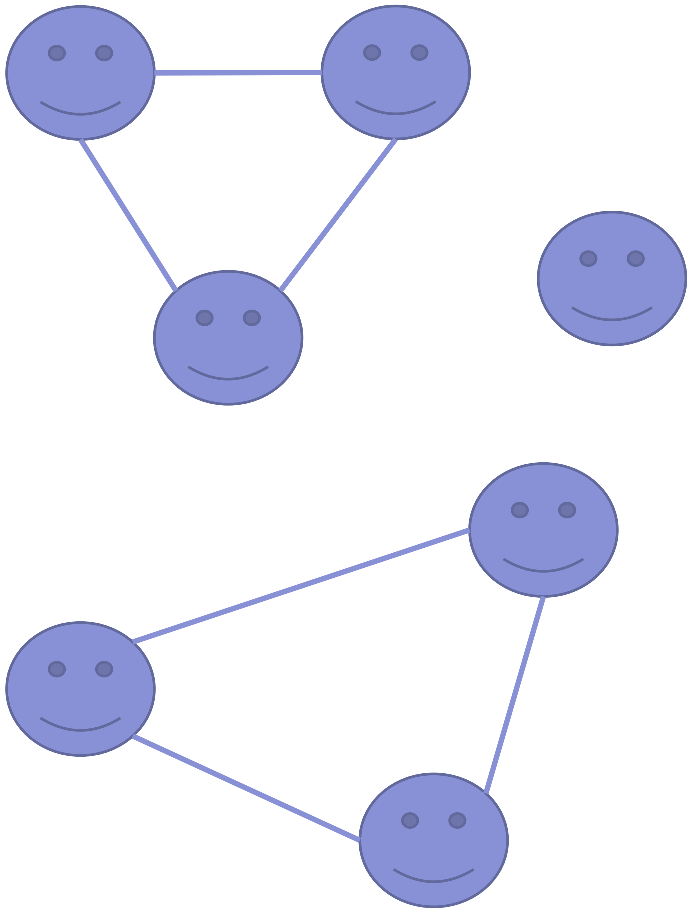
Connected Components

- Look at the network on the right.
- All the networks we saw thus far were all joined together, but this one has two **connected components** - groups of nodes that aren't joined to other groups.
- It also has one **orphan**, or **isolated**, node, with no connections to any other node.



Dealing with Disconnection

- It's common when studying networks to find these kind of problems.
  - It's especially common to find one **giant connected component** which makes up most of the network, and a number of other smaller groups around it.
- Often we choose to **prune** the network – removing orphaned nodes and small connected components – to make analysis easier.
- Of course, your analysis might choose to focus on those more isolated groups instead!

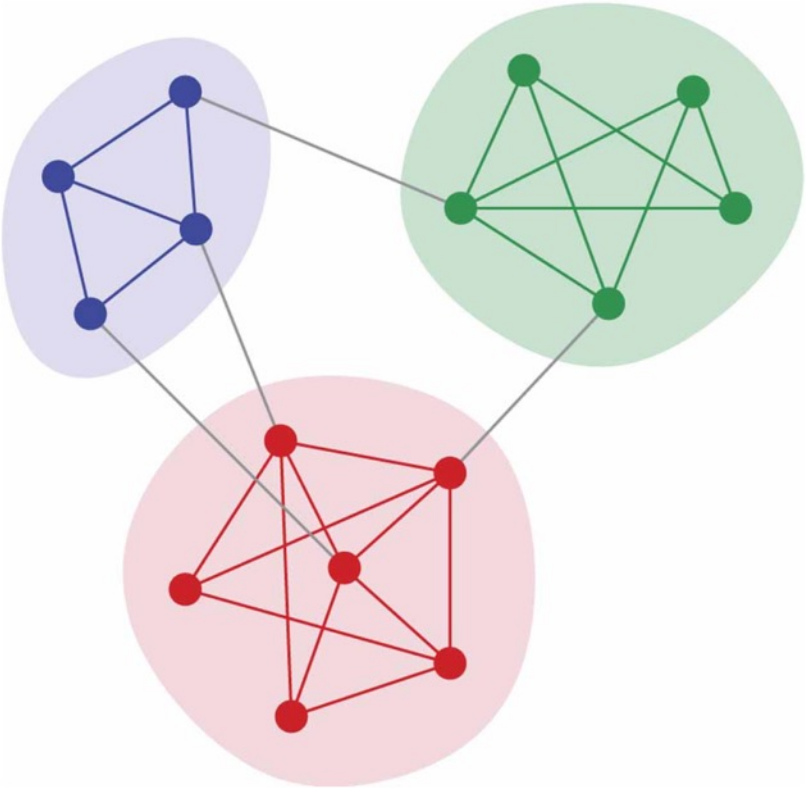


Network Communities

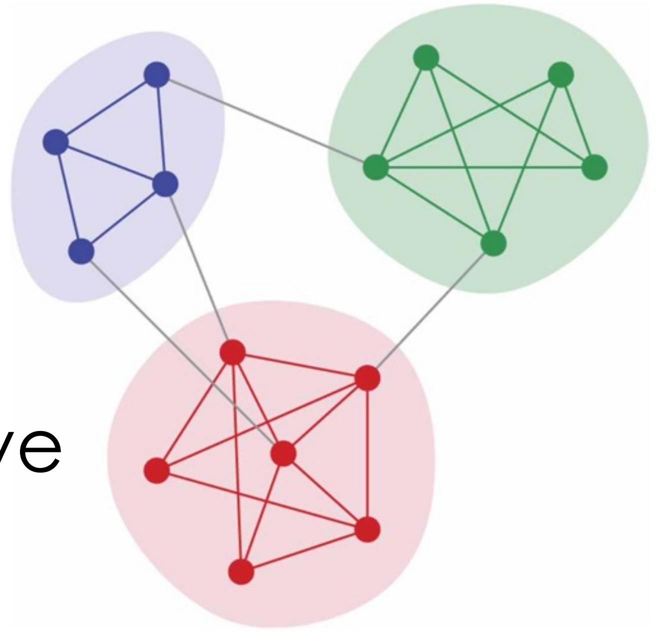
- One of the most powerful features of network analysis is **community detection**.
  - The simplest form of community detection is just looking at **connected components**, of course.
- However, even within a single connected group, there may be some sub-groups that are more closely connected, forming a distinct community.

Community Detection

- In this network, there are no **connected components** – everything is interconnected.
- However, there are three distinct **communities.**
- Within communities, the connections are **dense** – while the connections *between* communities are **sparse**.



Finding Communities



- There are many different algorithms that have been developed to detect communities in network data.
- We will discuss the differences between them and how they work in detail in a future class.

Today's Assignment

- From the graph shown here (which is also in today's assignment PDF), estimate:
  - Network Size
  - Network Density
  - Nodes with highest Degree
  - Nodes with highest Betweenness
  - Nodes with highest Closeness