Visual Analytics–Networks

Dr. Ab Mosca (they/them)

Slides based off slides courtesy of Ashley Suh

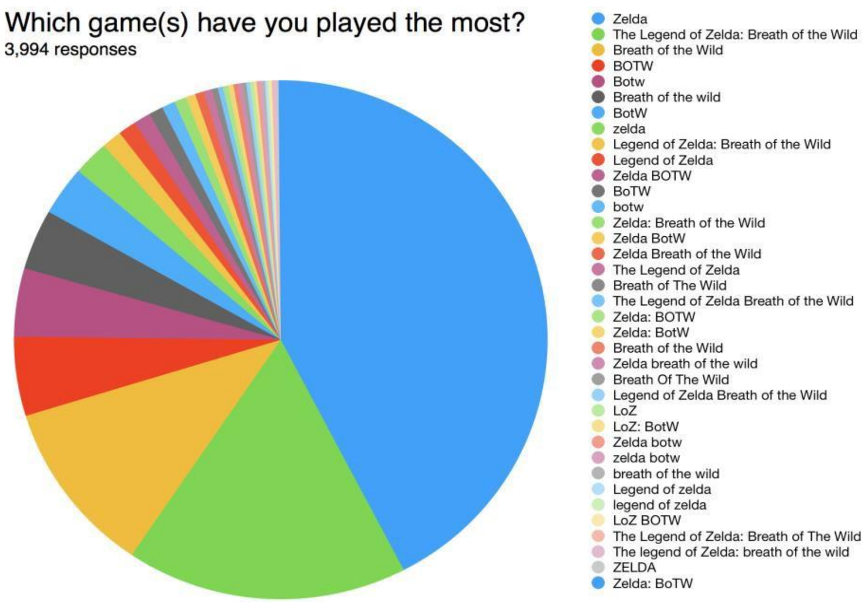
Notes

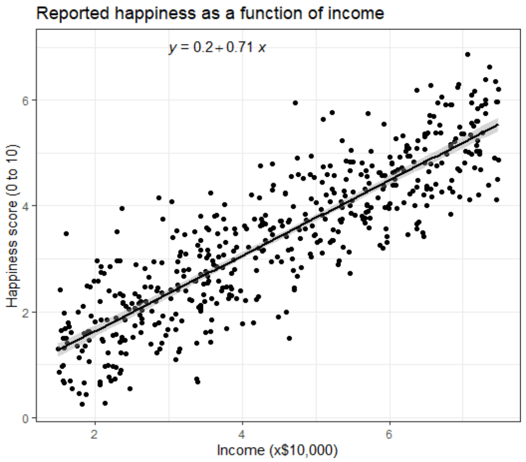
- Hw04
  - Do we need to push the due date back?
- Next week…
  - I will be here, but class will still be reserved for you to start working on midsemester projects

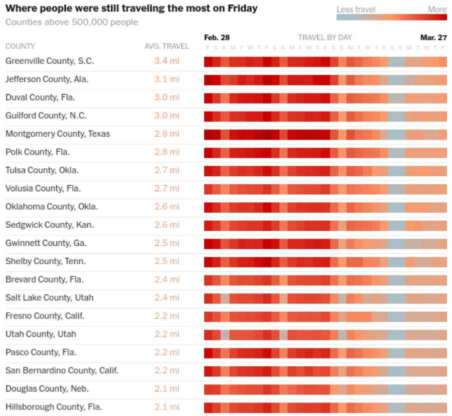
Plan for Today

- What is graph data
- Example visualizations
- Common vis problems & solutions

Graph (defn.)







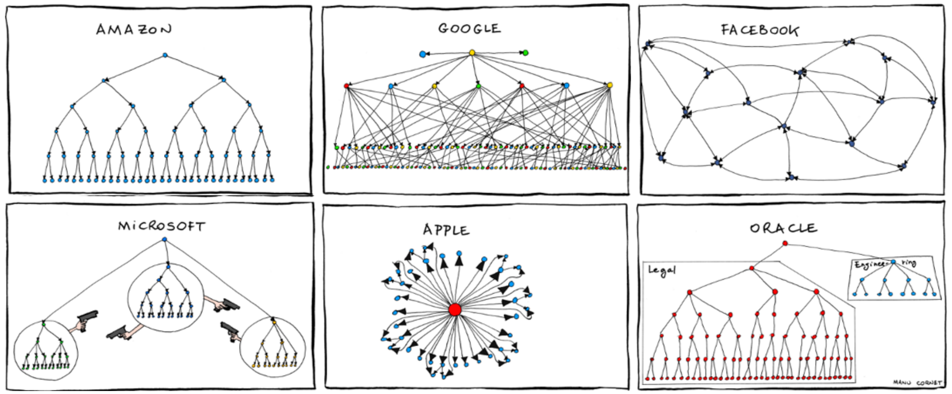
Common charts that represent data are often referred to as "graphs", or "graph visualizations"

- Bar charts
- Line charts
- Pie charts
- Etc.

However, graph is also a type of data.

Graph (defn.)

In the mathematical sense a graph is ***a model for representing items and the relationships between those items***



- Social / friendship networks
- Computer networks
- Energy or transportation grids
- Organizational structures
- Etc.

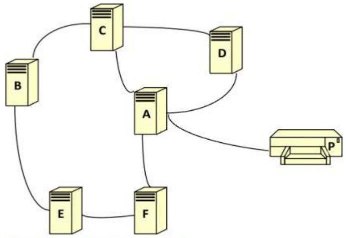
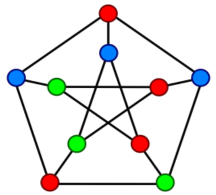
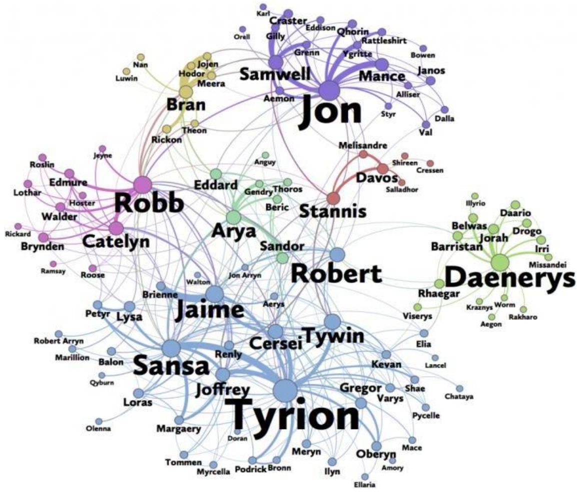


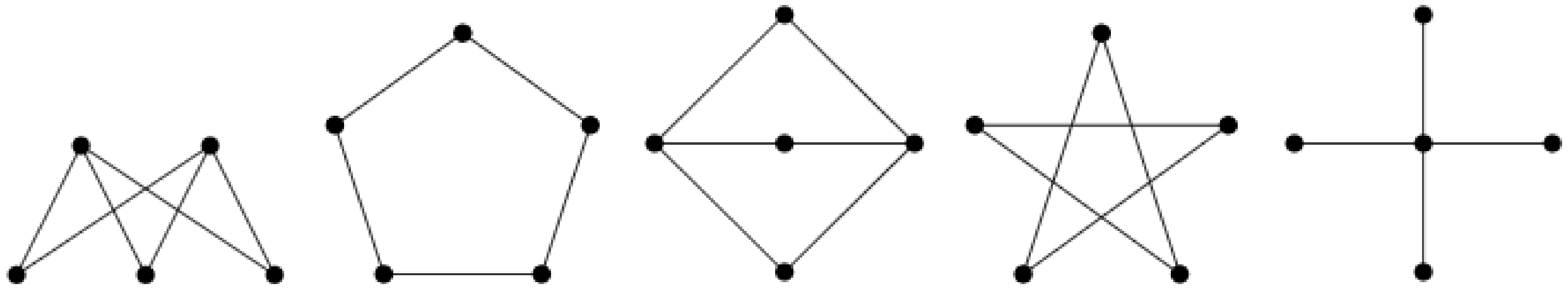
Figure 1- A simple computer network





Building Intuition

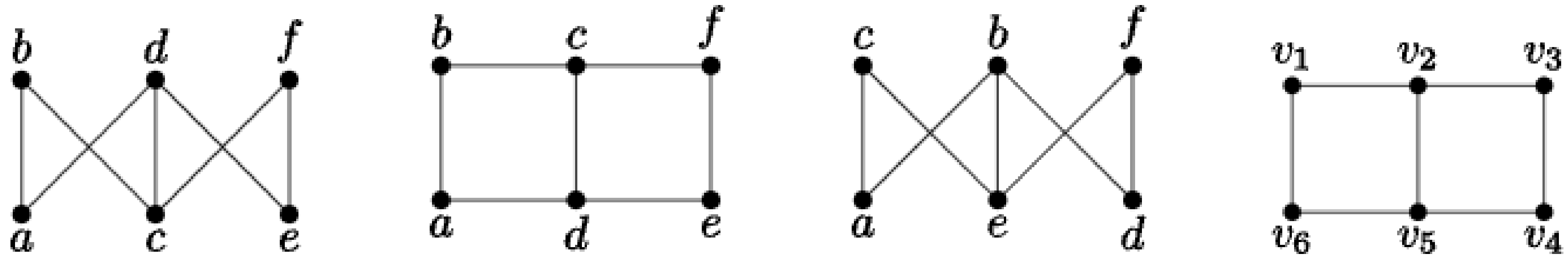
Are any of the graphs below the same? If you said yes, which ones and why?



How about these with labels?

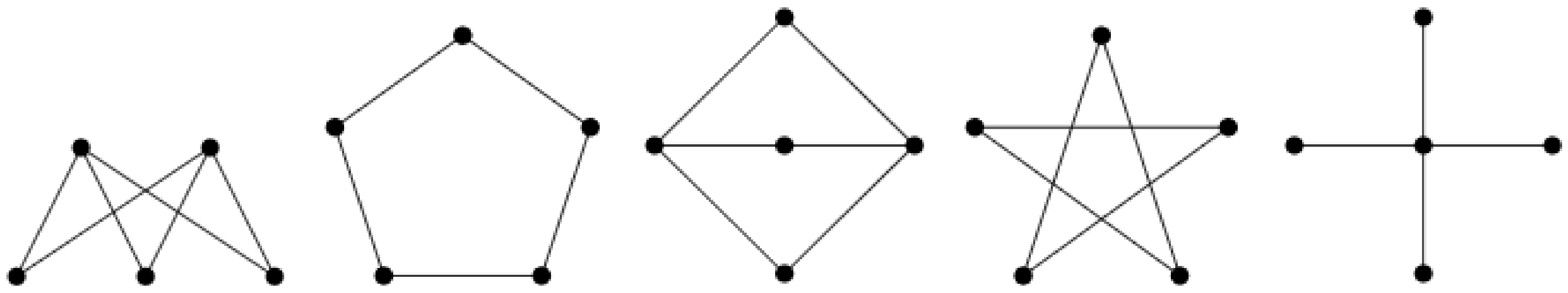
Building Intuition

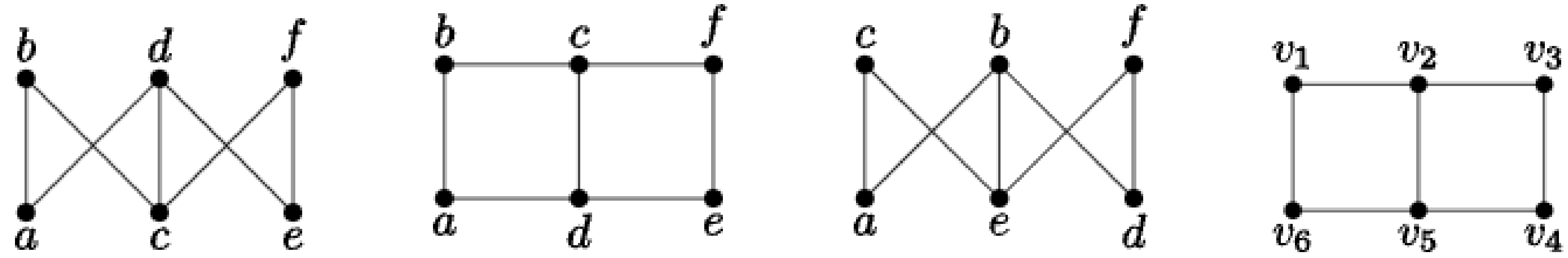
Are any of the graphs below the same? If you said yes, which ones and why?



Graph (defn.)

These examples are ***drawings*** of graphs.

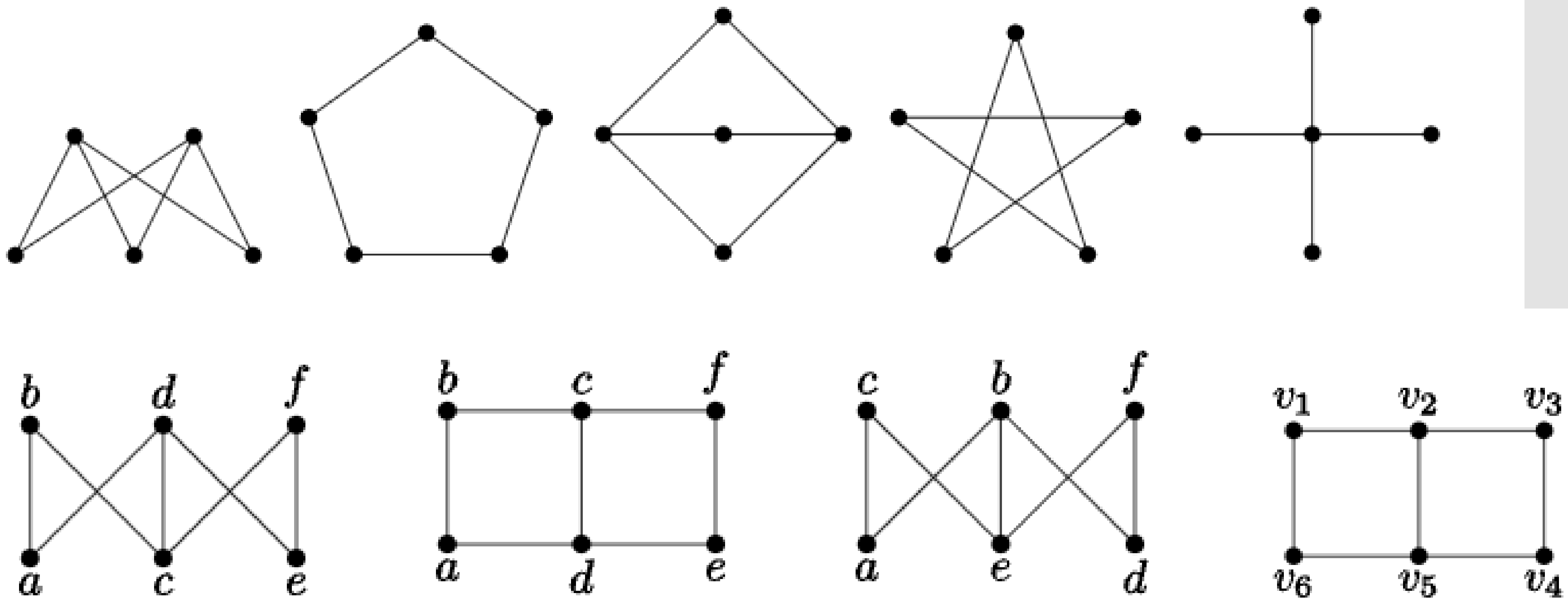




Mathematically, a ***graph*** is an ordered pair, $G = (V, E)$, consisting of a nonempty set, V (called ***vertices***), and a set E (called ***edges***) of two-elements subsets of V.

Graph (defn.)

These examples are ***drawings*** of graphs.



Mathematically, a ***graph*** is an ordered pair, $G = (V, E)$, consisting of a nonempty set, V (called ***vertices***), and a set E (called ***edges***) of two-elements subsets of V.

Note: you may also hear vertices called ***nodes***.

Graph (defn.)

Mathematically, a ***graph*** is an ordered pair, $G = (V, E)$, consisting of a nonempty set, V (called ***vertices***), and a set E (called ***edges***) of two-elements subsets of V.

Ex. Graph 1:

$$V = \{a, b, c, d, e\},$$

$$E = \{\{a, b\}, \{a, c\}, \{a, d\}, \{a, e\}, \{b, c\}, \{d, e\}\}$$

Graph (defn.)

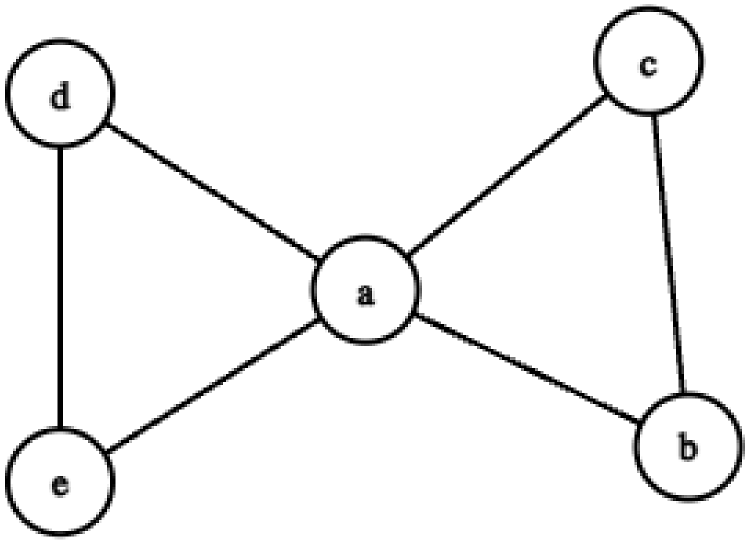
Mathematically, a ***graph*** is an ordered pair, $G = (V, E)$, consisting of a nonempty set, V (called ***vertices***), and a set E (called ***edges***) of two-elements subsets of V.

Ex. Graph 1:

$$V = \{a, b, c, d, e\},$$

$$E = \{\{a, b\}, \{a, c\}, \{a, d\}, \{a, e\}, \{b, c\}, \{d, e\}\}$$

To visualize this graph we might draw it:



Graph (defn.)

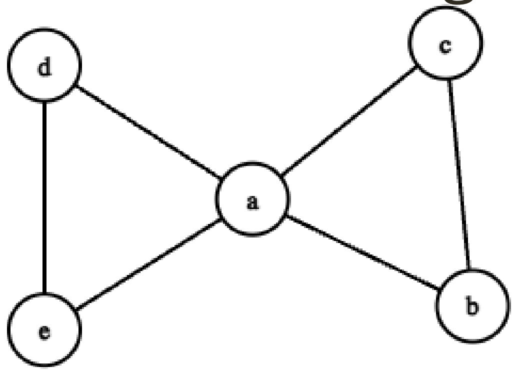
Mathematically, a ***graph*** is an ordered pair, $G = (V, E)$, consisting of a nonempty set, V (called ***vertices***), and a set E (called ***edges***) of two-elements subsets of V.

Ex. Graph 1:

$$V = \{a, b, c, d, e\},$$

$$E = \{\{a, b\}, \{a, c\}, \{a, d\}, \{a, e\}, \{b, c\}, \{d, e\}\}$$

To visualize this graph we might draw it:



Draw this graph:

$$V = \{v_1, v_2, v_3, v_4, v_5\}$$

$$E = \{\{v_1, v_3\}, \{v_1, v_5\}, \{v_2, v_4\}, \{v_2, v_5\}, \{v_3, v_5\}, \{v_4, v_5\}\}$$

Graph (defn.)

Mathematically, a ***graph*** is an ordered pair, $G = (V, E)$, consisting of a nonempty set, V (called ***vertices***), and a set E (called ***edges***) of two-elements subsets of V.

Ex. Graph 1:

$$V = \{a, b, c, d, e\},$$

$$E = \{\{a, b\}, \{a, c\}, \{a, d\}, \{a, e\}, \{b, c\}, \{d, e\}\}$$

We can also represent a graph using an adjacency matrix:

$$\begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Graph (defn.)

Mathematically, a ***graph*** is an ordered pair, $G = (V, E)$, consisting of a nonempty set, V (called ***vertices***), and a set E (called ***edges***) of two-elements subsets of V.

What is the adjacency matrix for this graph?

$$V = \{v_1, v_2, v_3, v_4, v_5\}$$

$$E = \{\{v_1, v_3\}, \{v_1, v_5\}, \{v_2, v_4\}, \{v_2, v_5\}, \{v_3, v_5\}, \{v_4, v_5\}\}$$

We can also represent a graph using an adjacency matrix:

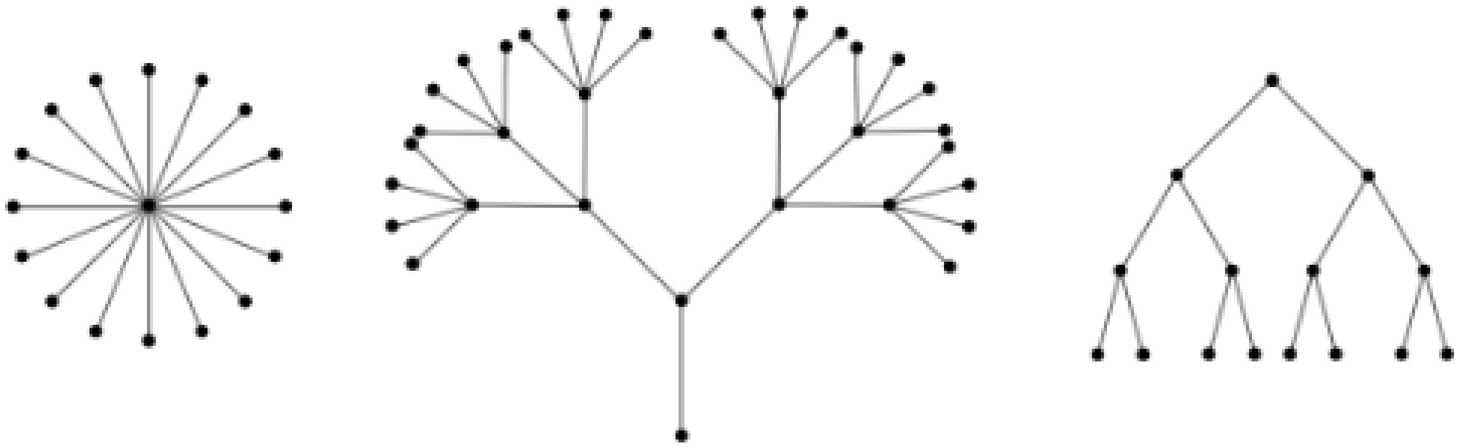
$$\begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Tree (defn.)

A ***tree*** is a connected graph containing no cycles.

A ***forest*** is a graph containing no cycles.

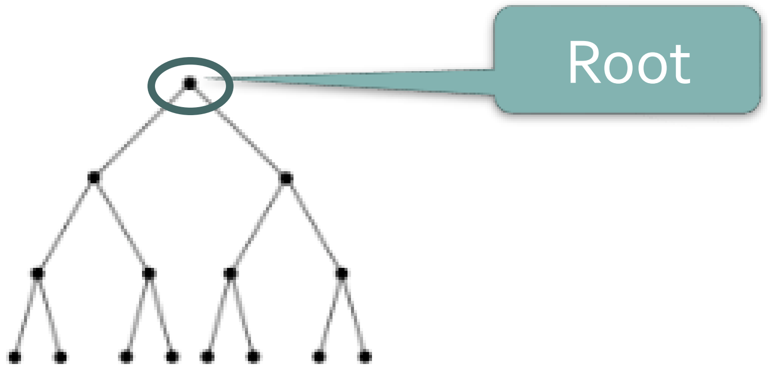
Note: this means a connected forest is a tree.



https://ptwiddle.github.io/Graph-Theory-Notes/s_intro_trees.html

Rooted Trees

We can identify one vertex in a tree as the ***root***. Then, every other vertex on the tree can be characterized by its position relative to the root.

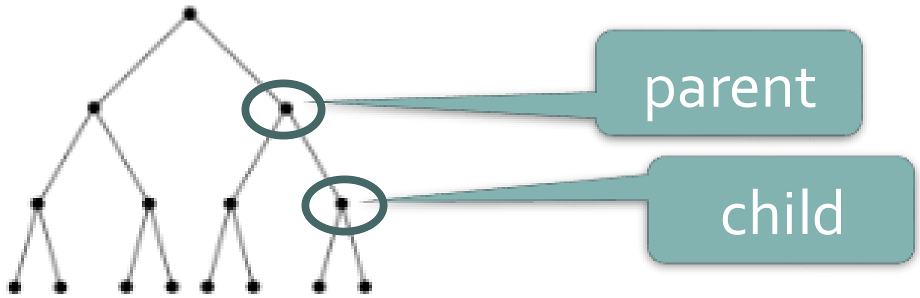


Rooted Trees

We can identify one vertex in a tree as the ***root***. Then, every other vertex on the tree can be characterized by its position relative to the root.

If two vertices are adjacent, we say the one closer to the root is the ***parent***, and the other is the ***child***.

The root of a tree is a parent, but not a child of any vertex. All non-root vertices have exactly one parent.

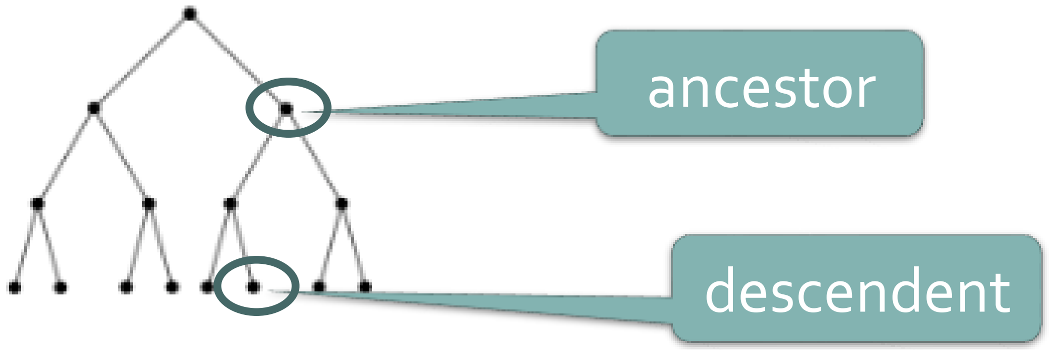


Rooted Trees

We can identify one vertex in a tree as the ***root***. Then, every other vertex on the tree can be characterized by its position relative to the root.

If two vertices are adjacent, we say the one closer to the root is the ***parent***, and the other is the ***child***.

In general, we say a vertex, v, is a ***descendent*** of a vertex, u, provided u is a vertex on the path from v to the root. Then, we would call u an ***ancestor*** of v.



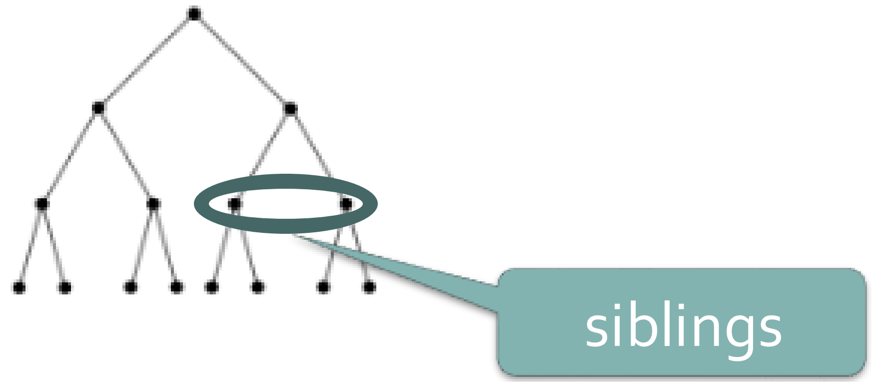
Rooted Trees

We can identify one vertex in a tree as the ***root***. Then, every other vertex on the tree can be characterized by its position relative to the root.

If two vertices are adjacent, we say the one closer to the root is the ***parent***, and the other is the ***child***.

In general, we say a vertex, v, is a ***descendent*** of a vertex, u, provided u is a vertex on the path from v to the root. Then, we would call u an ***ancestor*** of v.

Vertices with the same parent are called ***siblings***.



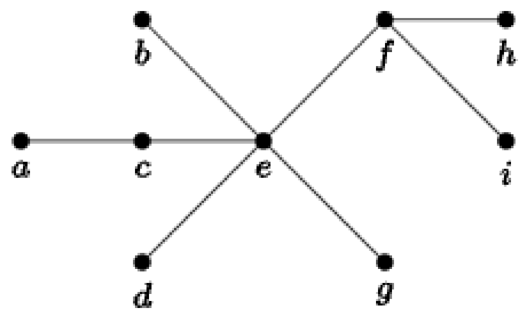
Rooted Trees

We can identify one vertex in a tree as the ***root***. Then, every other vertex on the tree can be characterized by its position relative to the root.

If two vertices are adjacent, we say the one closer to the root is the ***parent***, and the other is the ***child***.

In general, we say a vertex, v, is a ***descendent*** of a vertex, u, provided u is a vertex on the path from v to the root. Then, we would call u an ***ancestor*** of v.

Vertices with the same parent are called ***siblings***.

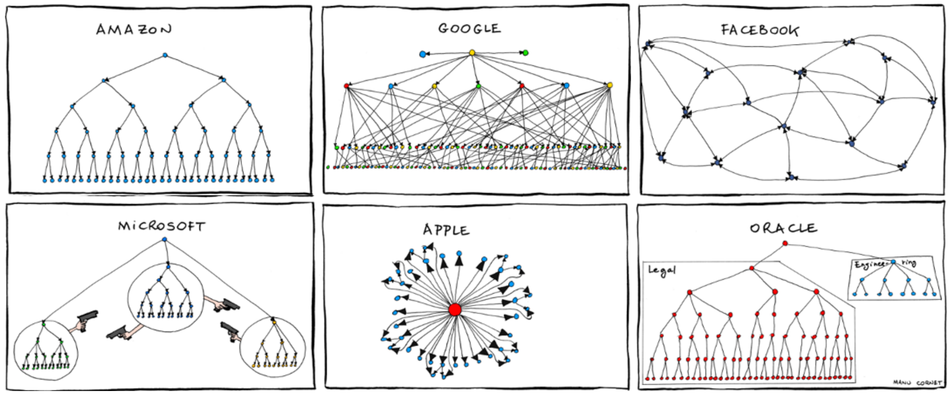


Let f be the root. Label the other vertices.

Flashback: Graph (defn.)

In the mathematical sense a graph is ***a model for representing items and the relationships between those items***

- Social / friendship networks
- Computer networks
- Energy or transportation grids
- Organizational structures
- Etc.



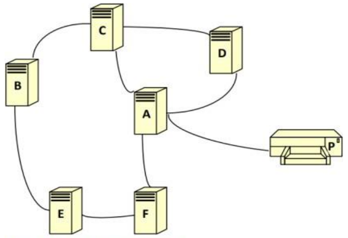
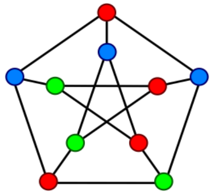
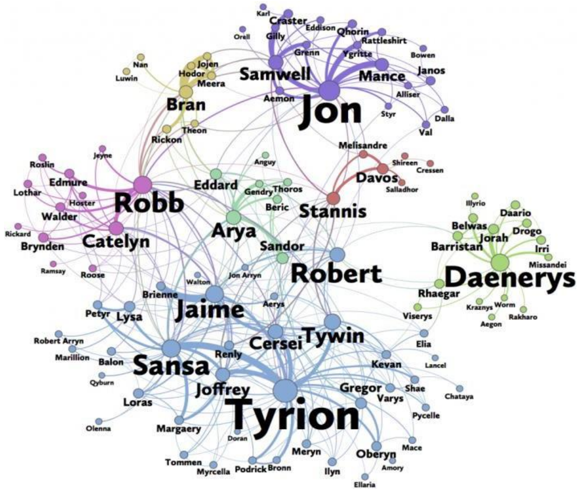


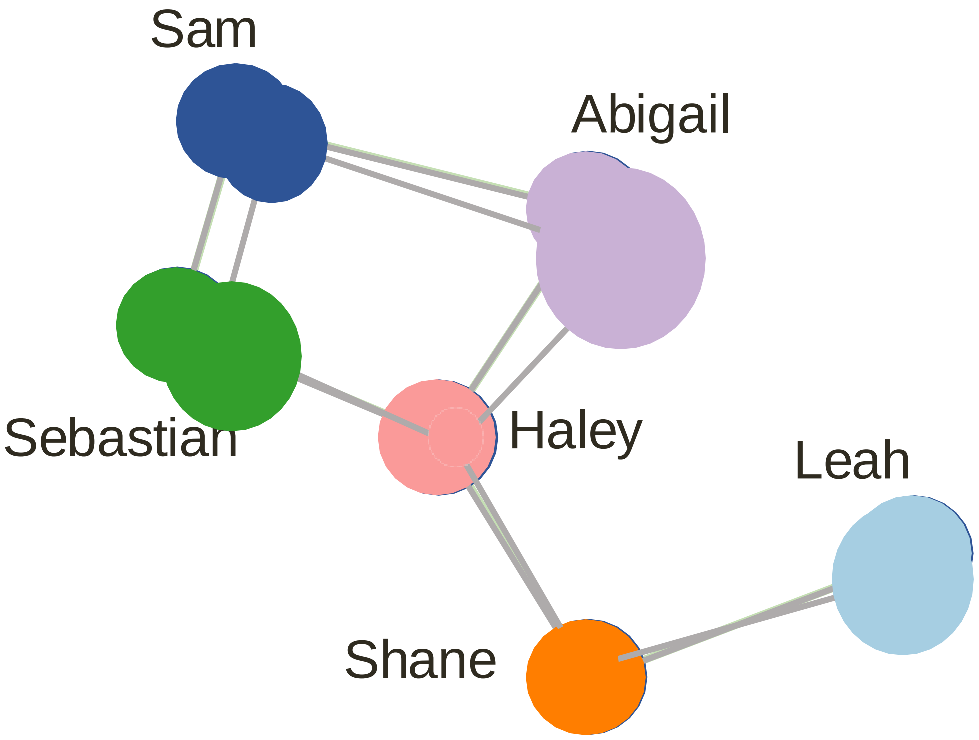
Figure 1- A simple computer network





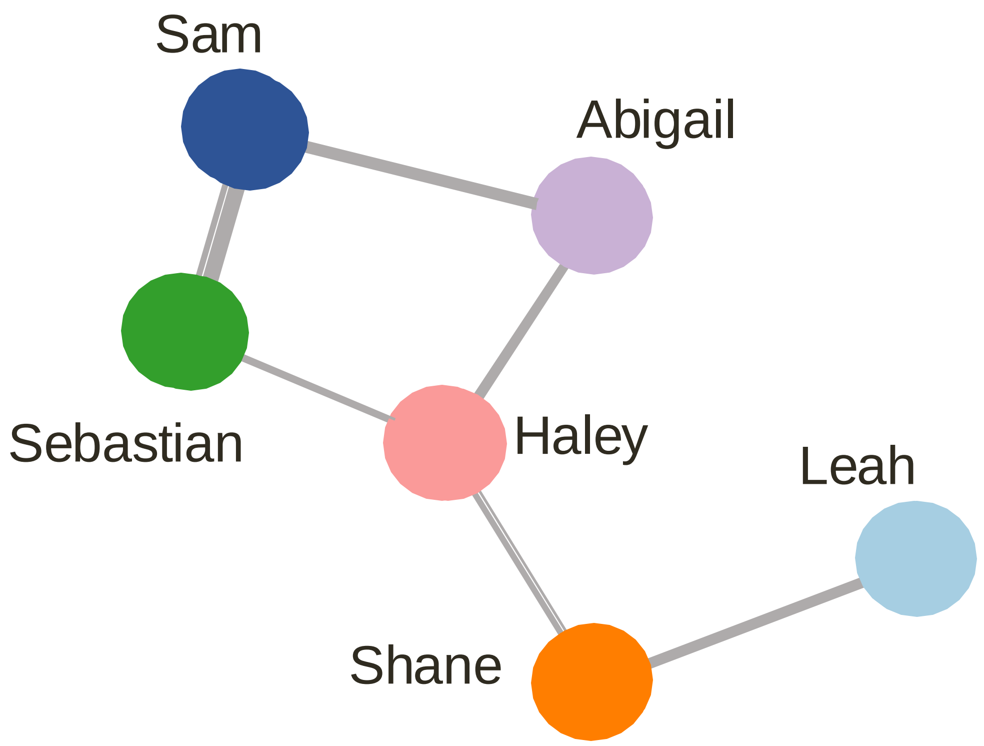
Additional Node Data

We may have additional data (attributes) about the nodes in the graph (i.e. the items in the graph)



Additional Edge Data

We may have additional data (attributes) about the edges in the graph (i.e. the connections between items)

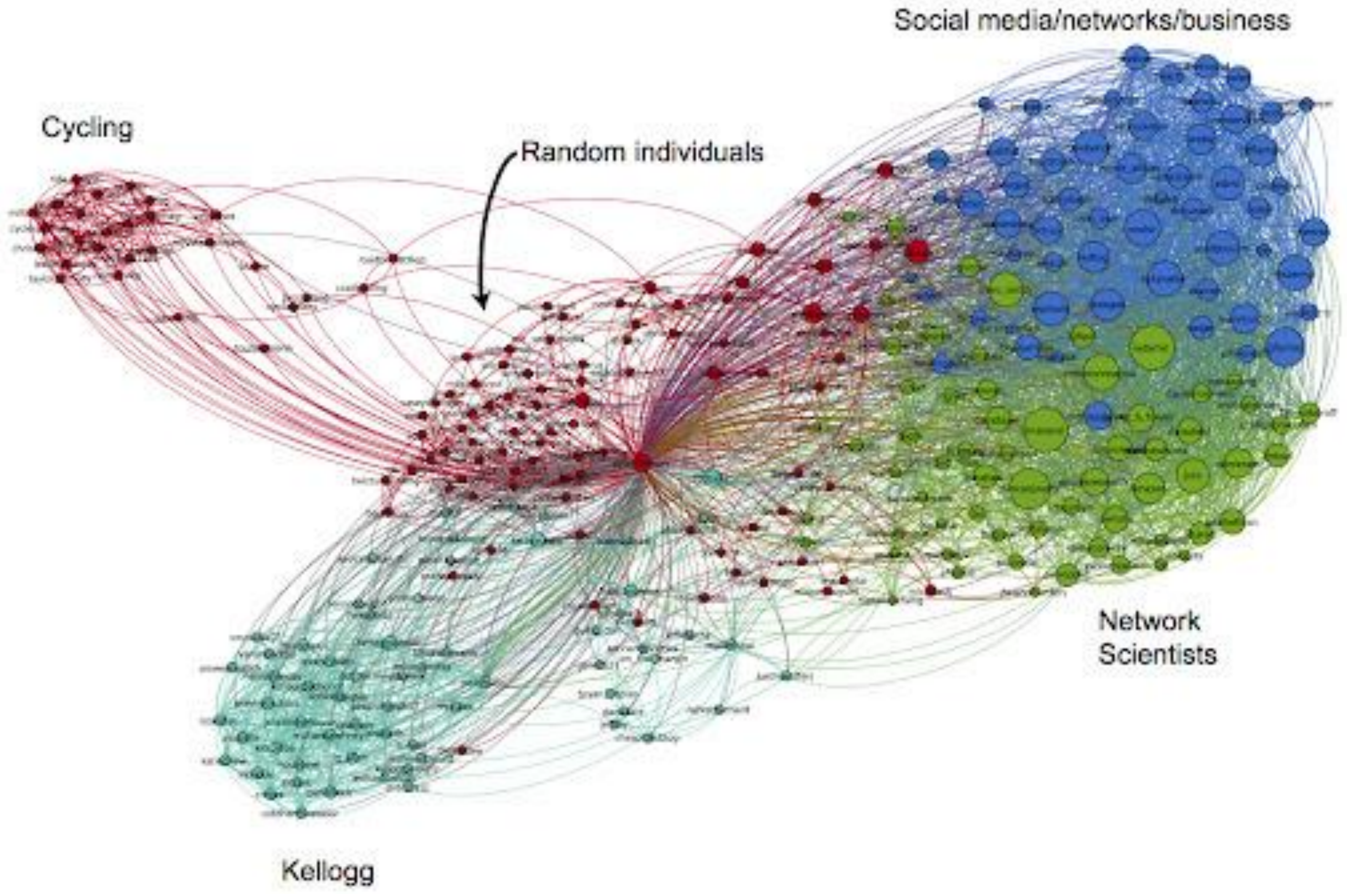


Visualizing

What might be the benefits of visualizing graphs?

Visualizing

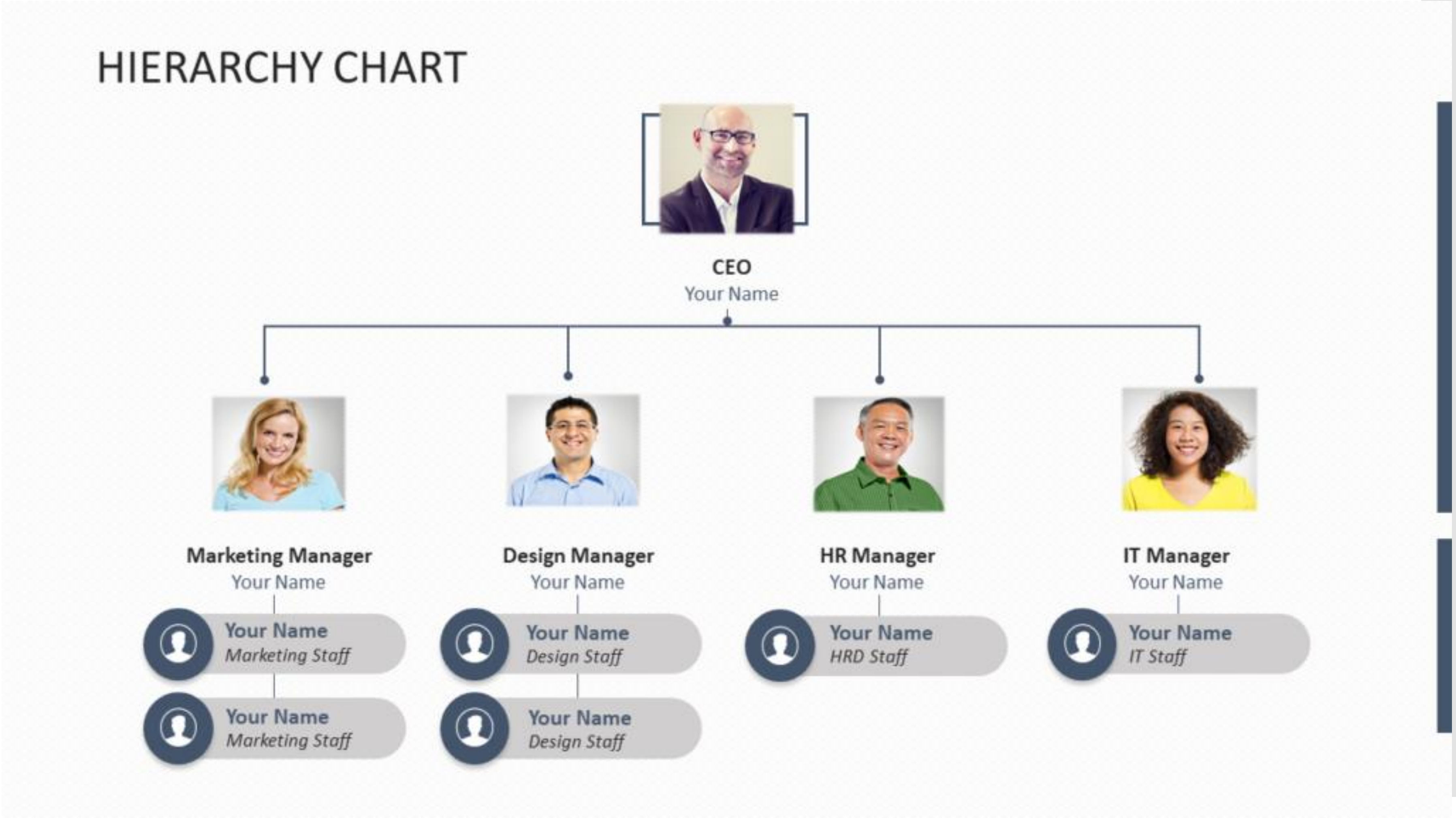
Highlight communities



https://www.google.com/url?sa=i&url=http%3A%2F%2Fsocial-dynamics.org%2Ftag%2Fnetwork-visualization%2F&psig=AOvVaw1csnV4heoey3cU3srQTMiE&ust=1724861687547000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCOij7NPIlYgDFQAAAAAdAAAAABA5

Visualizing

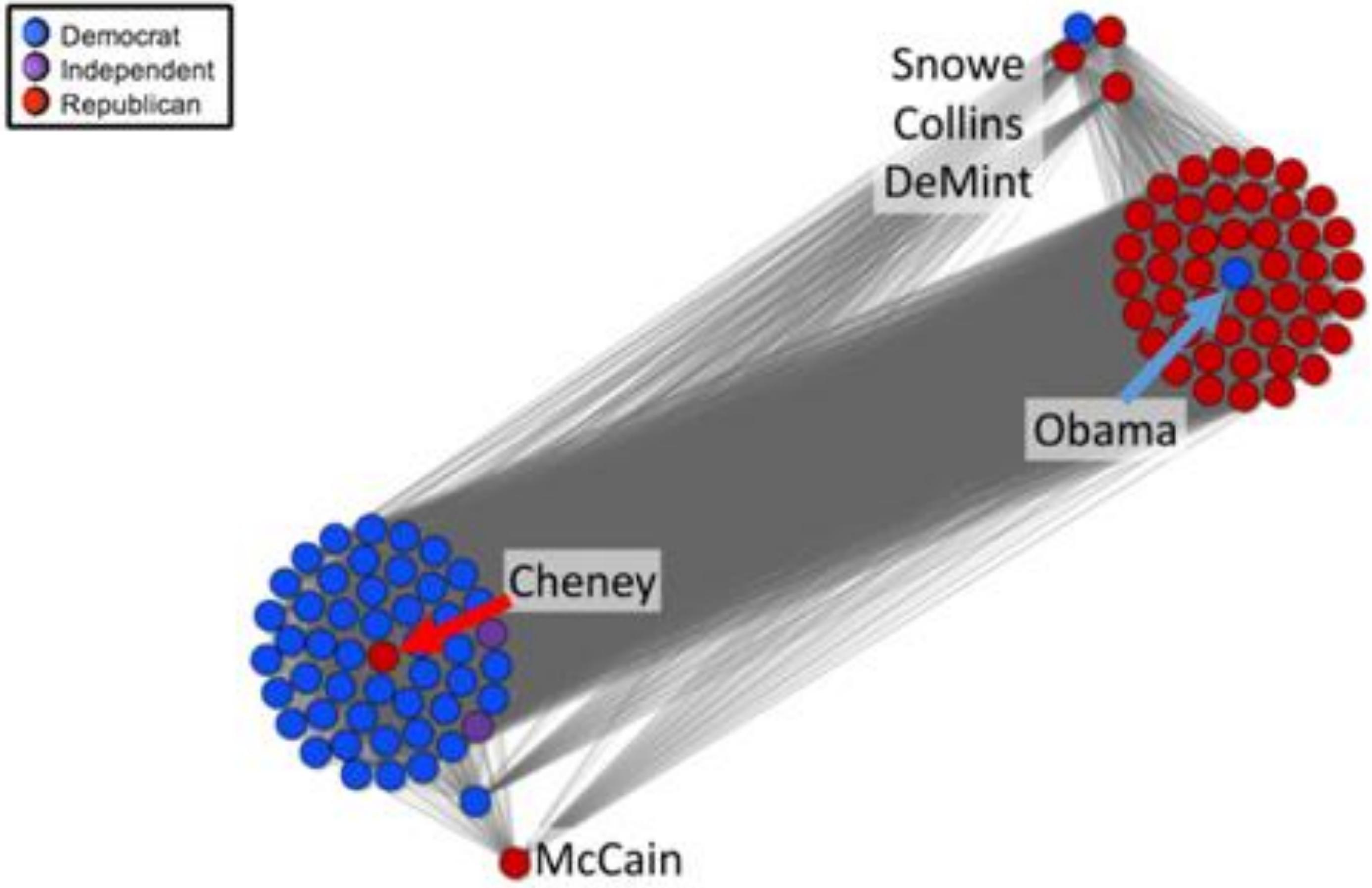
Highlight hierarchy



https://www.google.com/url?sa=i&url=https%3A%2F%2Fslideuplifts.medium.com%2Fmust-have-org-charts-to-visualize-your-project-team-structure-d61df5eb00f5&psig=AOvVaw2MFNBDY9jaG6RGpyqTDuKD&ust=1724861970547000&source=images&cd=vfe&opi=89978449&ved=0CBgQ3YkBahcKEwiYpfLayZWIAxUAAAAAHQAAAAAQCQ

Visualizing

Show outliers and anomalies



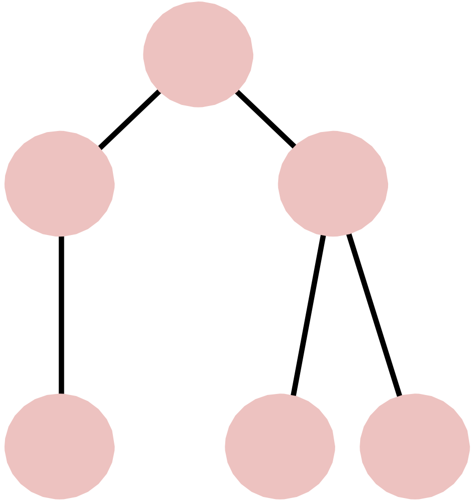
Visualization Techniques

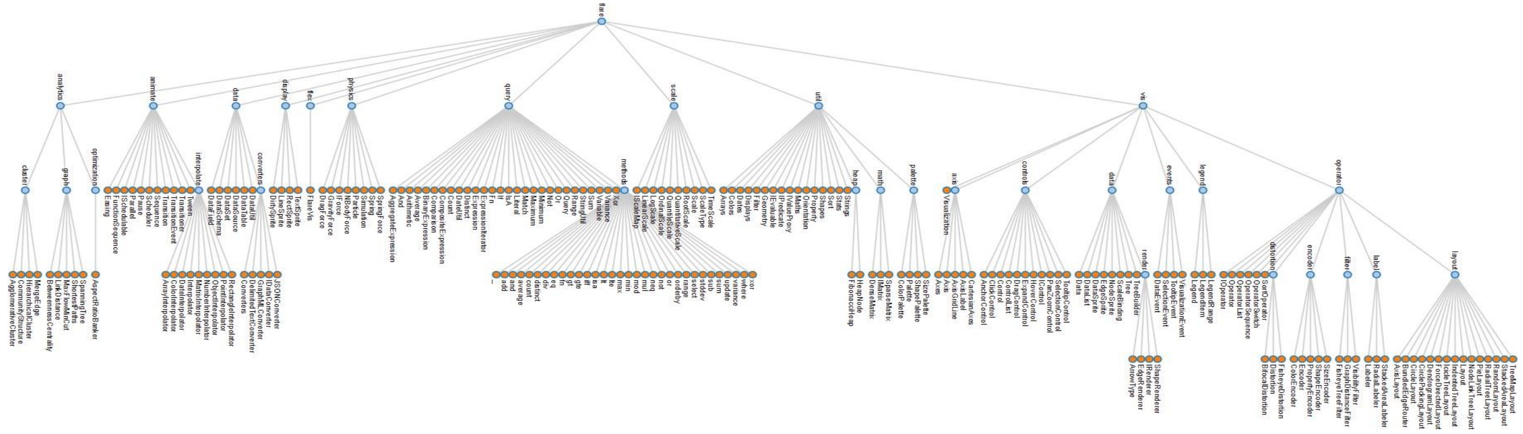
There are many different ways to encode and layout graph visualizations.

The correct one depends on what tasks you need to support or what features of the data need to be highlighted.

Node-link tree diagrams

- Nodes are distributed in space, connected by straight or curved lines
- Typical approach is to use 2D space to break apart breadth and depth
- Often, space is used to communicate hierarchical orientation



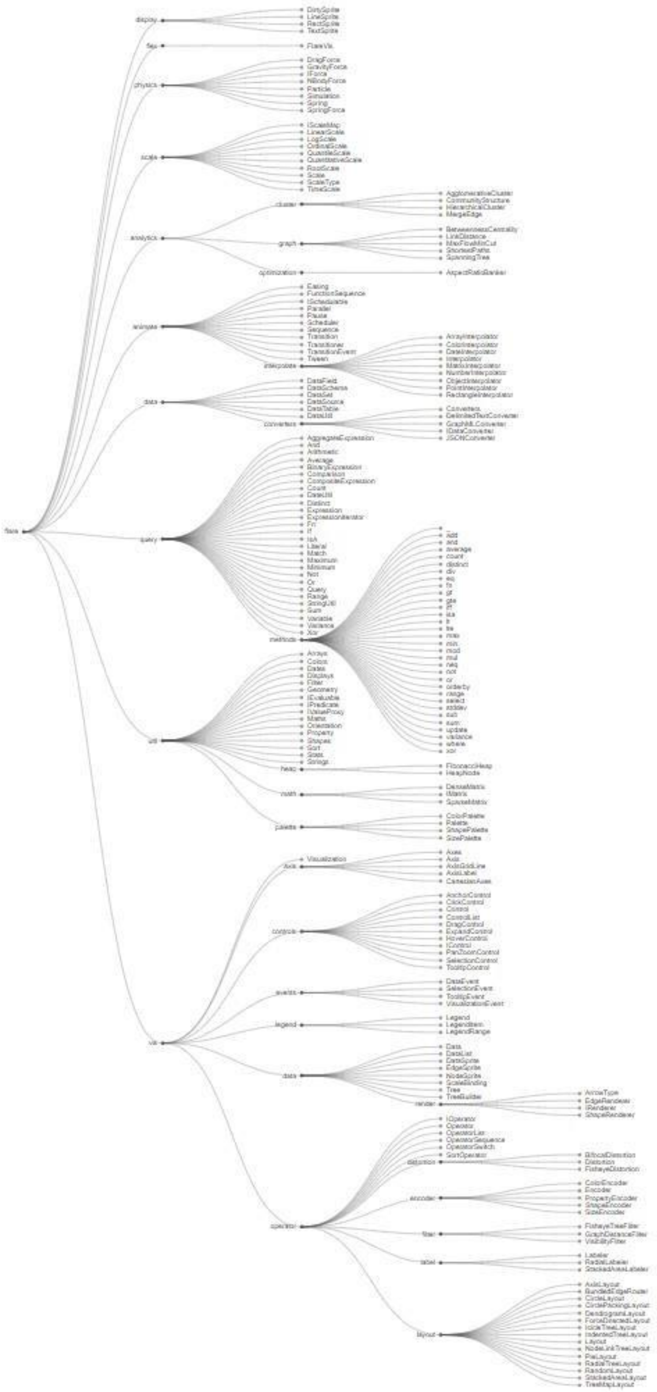


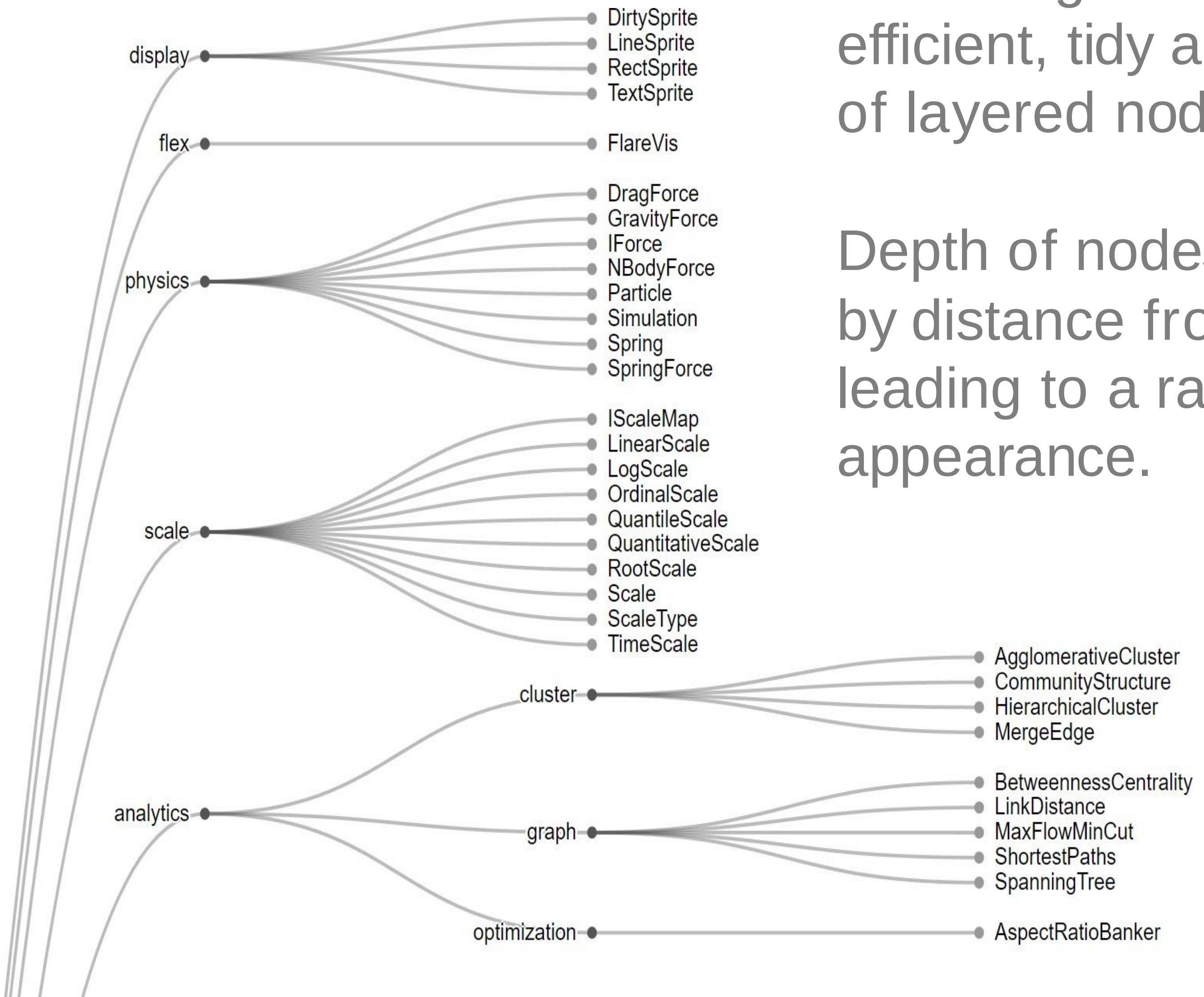
Tidy Tree

https://bl.ocks.org/mbostock/4339184

Implements the Reingold-Tilford algorithm for efficient, tidy arrangement of layered nodes

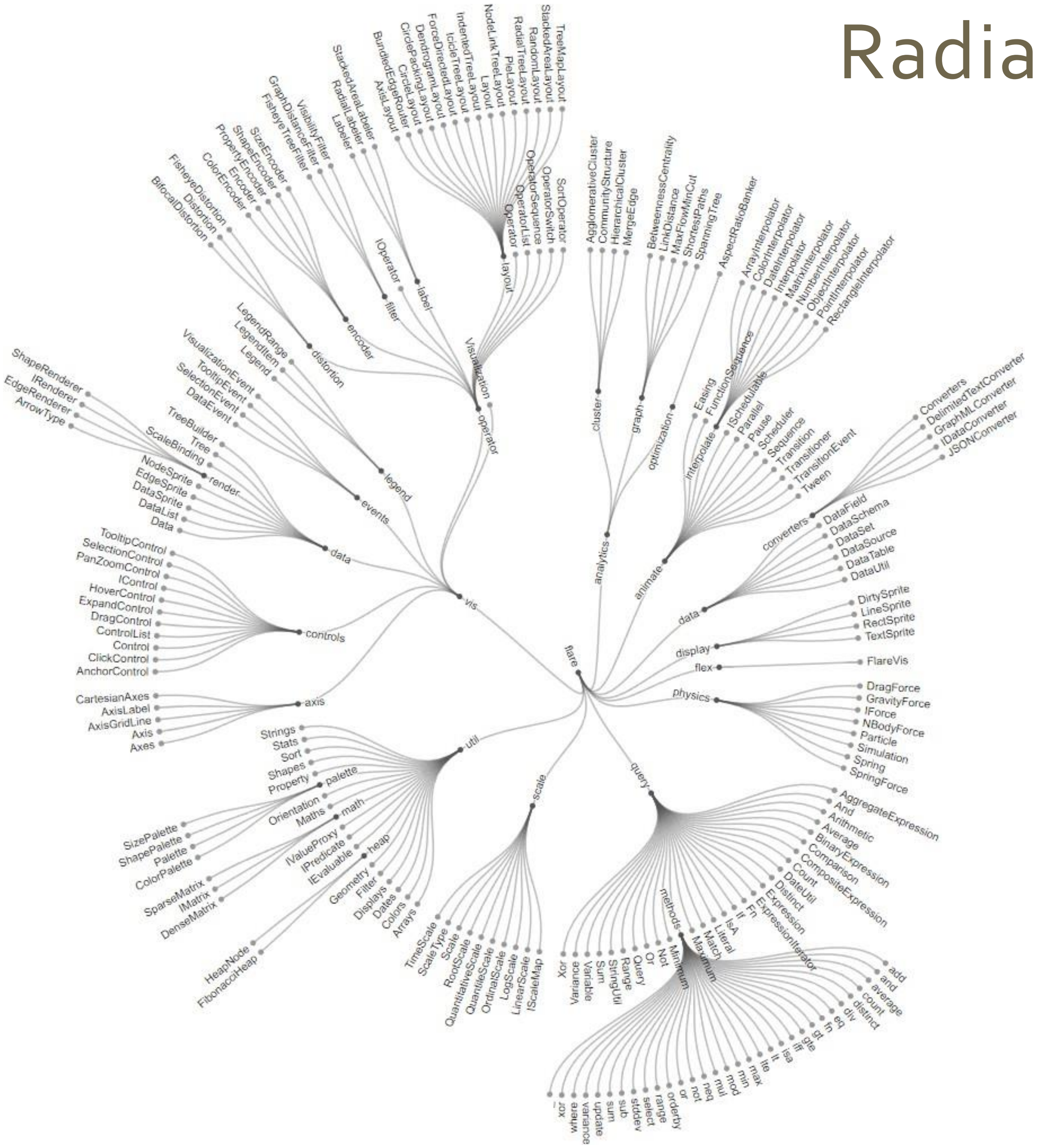
Depth of nodes computed by distance from the root, leading to a ragged appearance.

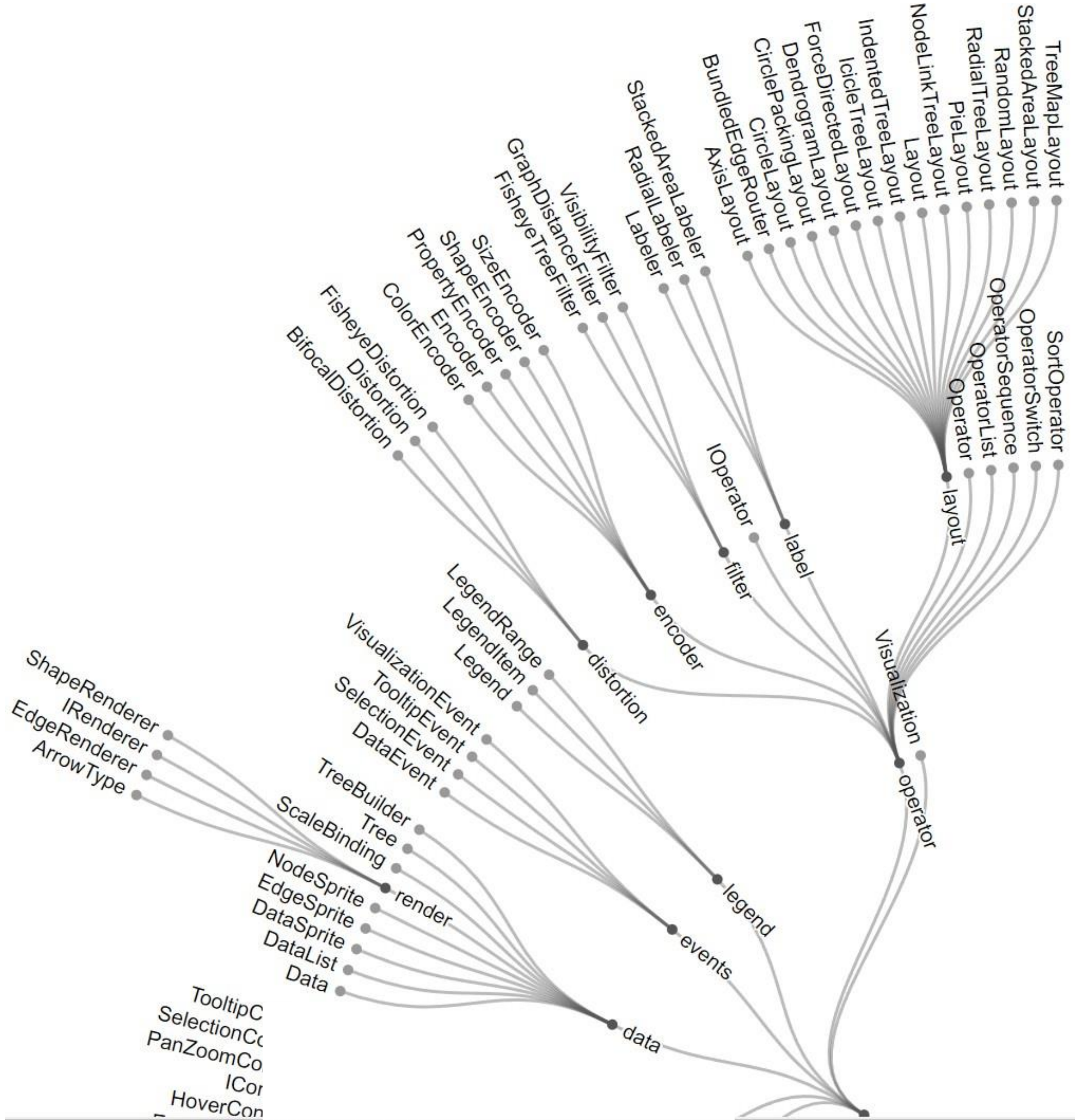




https://bl.ocks.org/mbostock/4063550

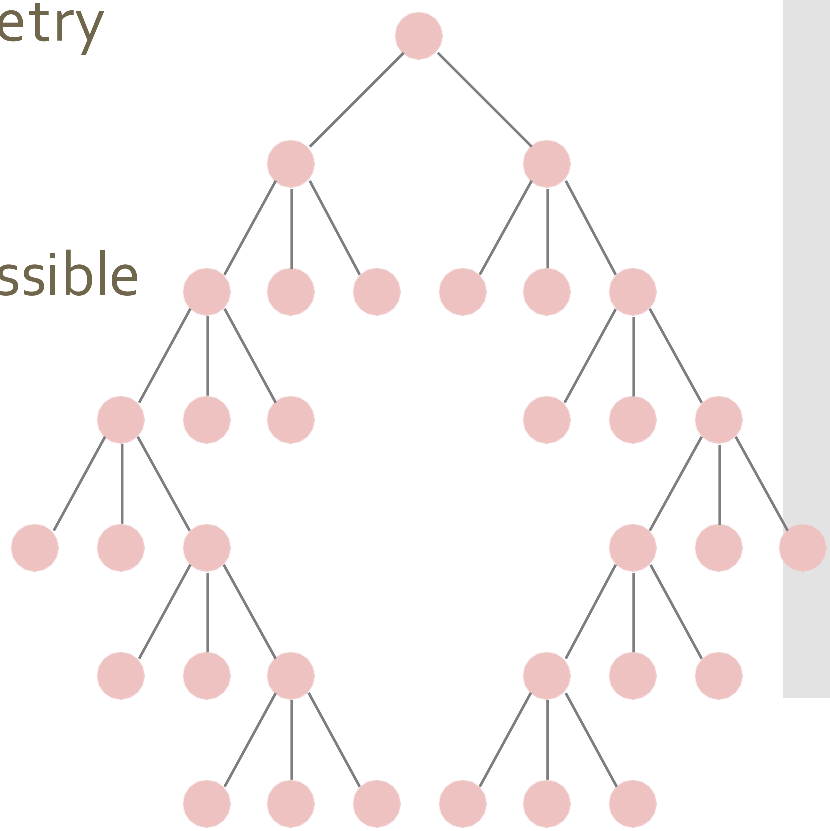
Radial Tidy Tree





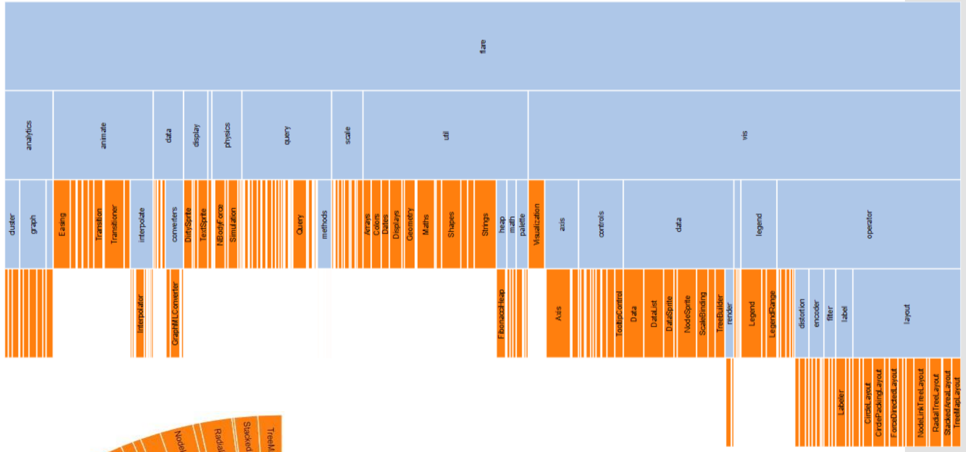
Reingold-Tilford algorithm

- Bottom-up recursive approach
  - Repeatedly divide space by leaf count
- For each parent, make sure subtrees are drawn
- Make smarter use of space
  - \+ Maximize density and symmetry
  - \+ Clearly encode depth level
  - \+ No edge crossings
  - \+ Pack subtrees as closely as possible
  - \+ Centers parent over subtrees

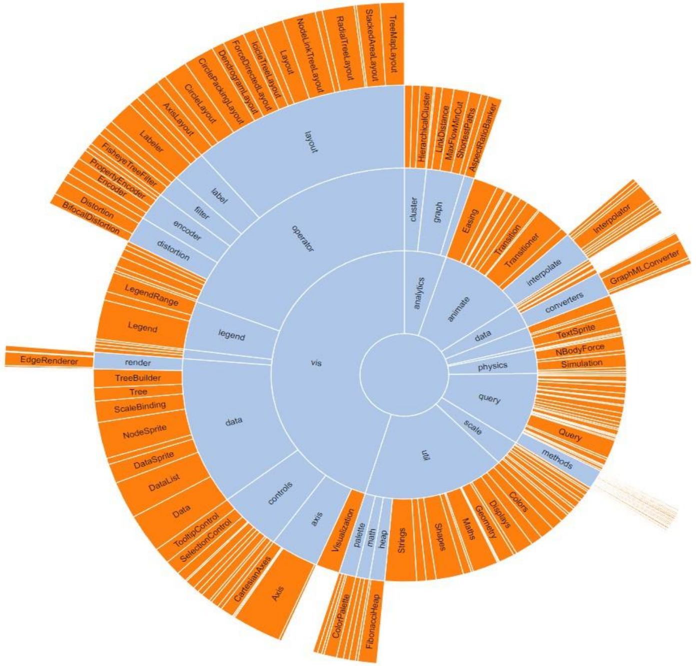


Layered (adjacency) diagrams

- Space-filling variant of node-link diagrams
- Nodes drawn as solid areas (arcs or bars)
- Placement relative to adjacent nodes reveals place in hierarchy
  - Root node at top / center
  - Leaf nodes at botto



Icicle layout (above)

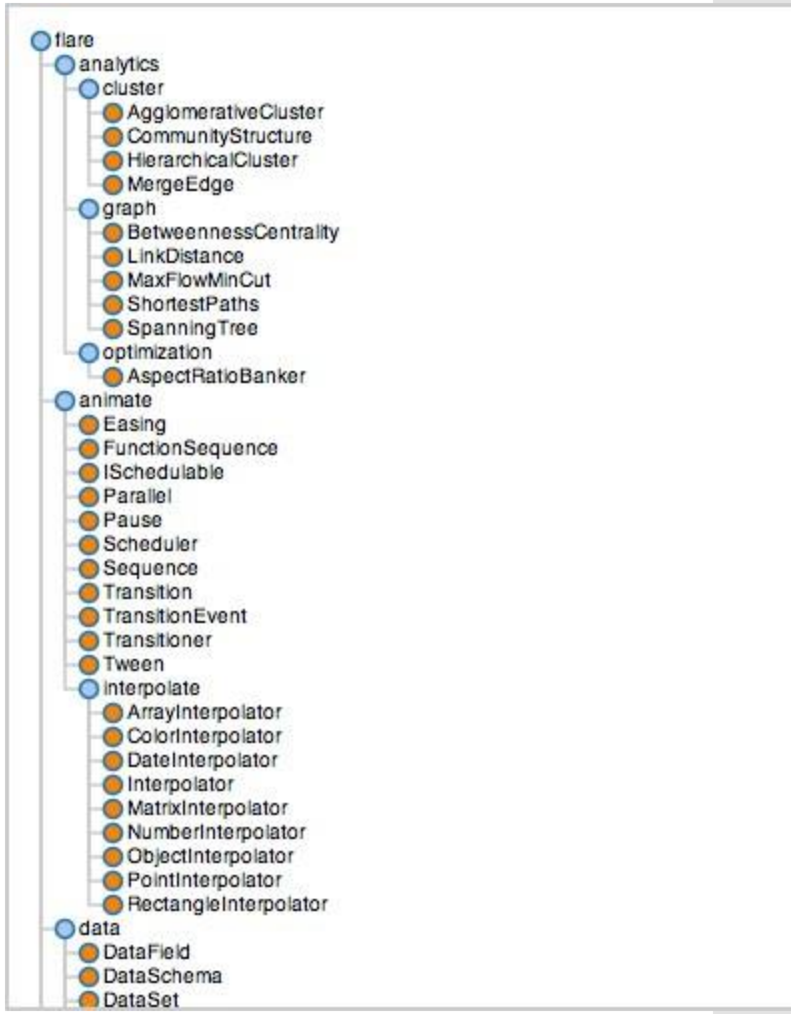


Sunburst layout (left)

Potential problem: can we run out of (screen) space?

Indentation

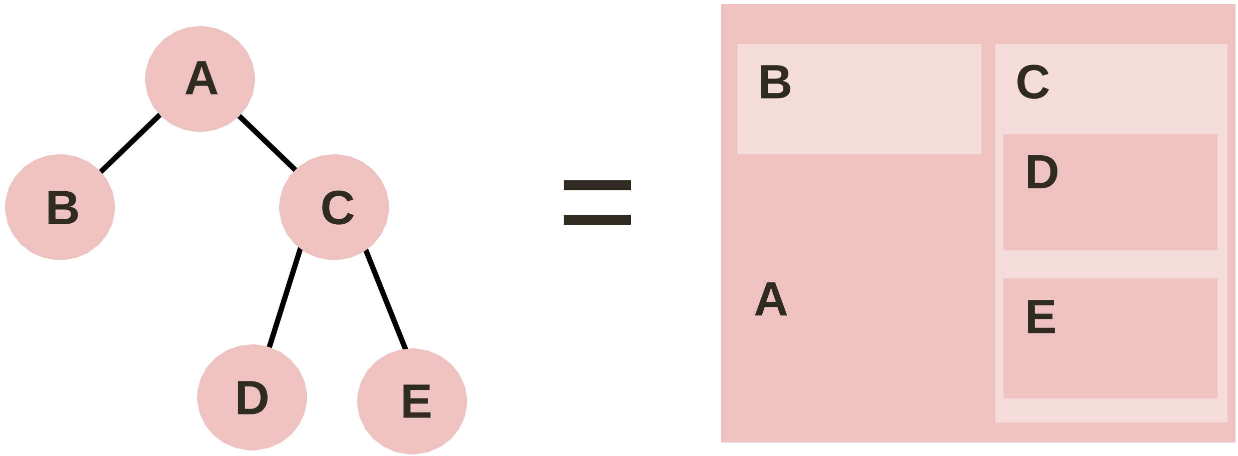
- Used to show parent / child relationships
- Potentially a lot of scrolling!



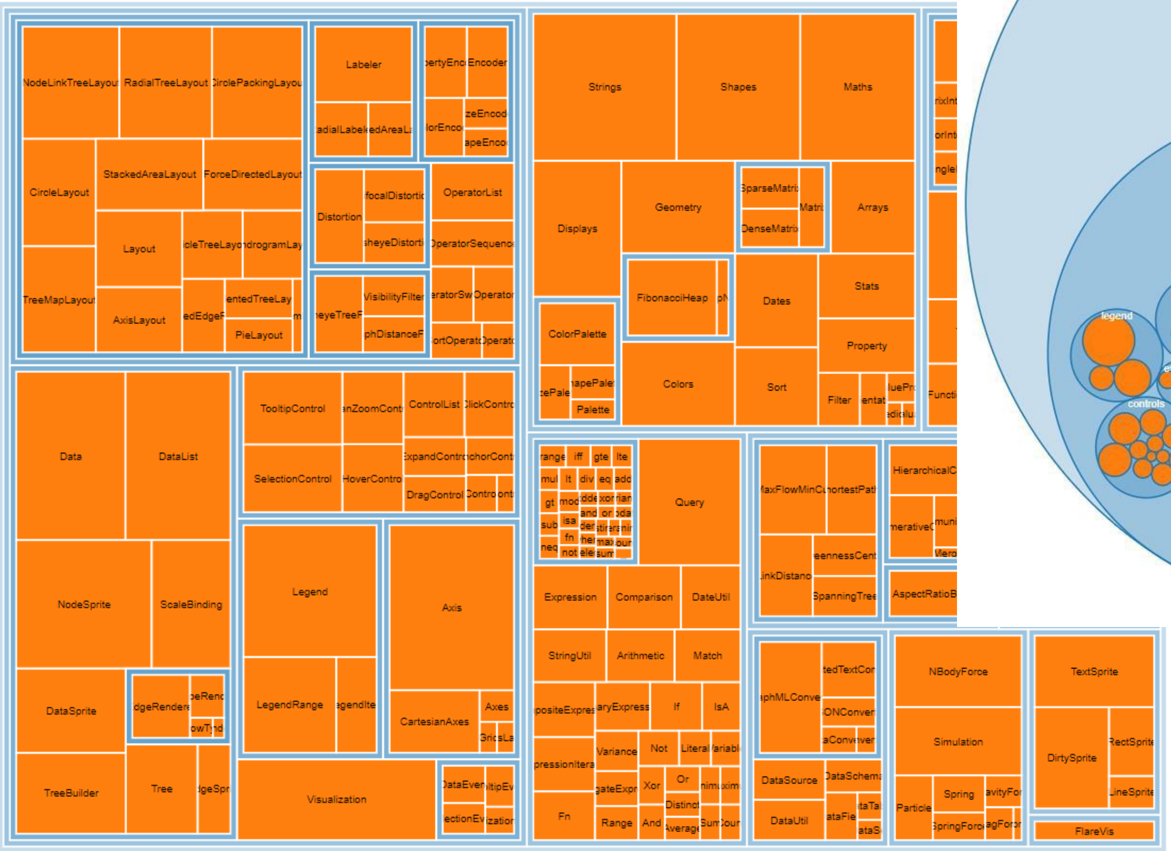
Question: where does this indented tree representation appear often?

Enclosure (treemap) diagrams

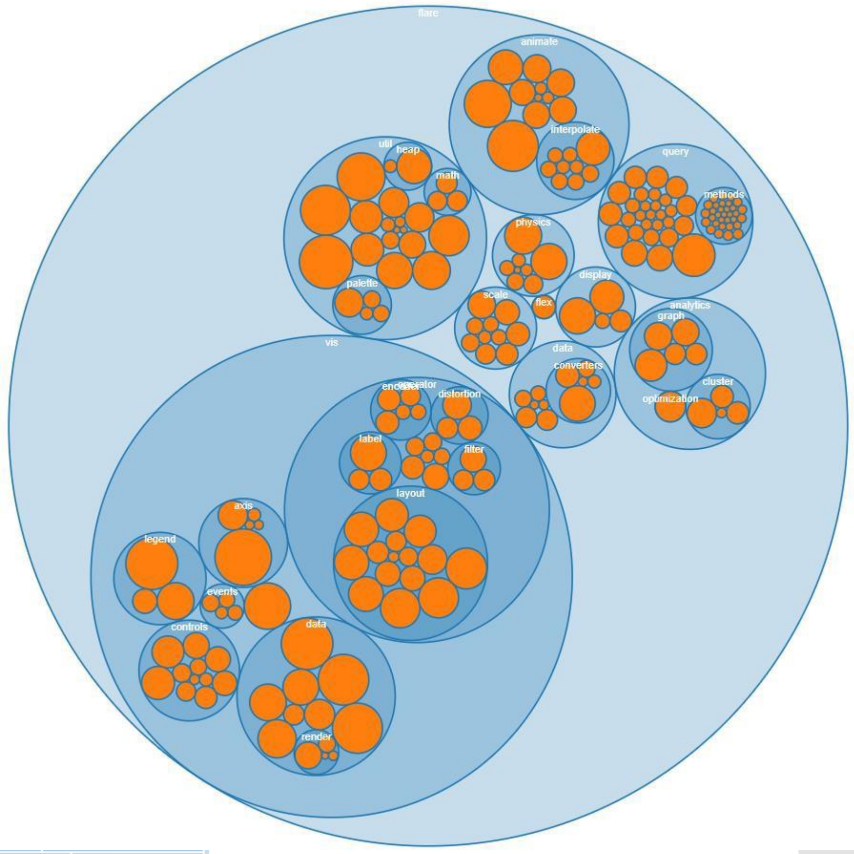
- Encodes tree structure using *spatial enclosure*
  - Enclosure indicates hierarchy
- Benefits:
  - Provides single view of entire tree
  - Easier to spot small / large nodes



Enclosure (treemap) diagrams



Enclosure diagram (treemap)



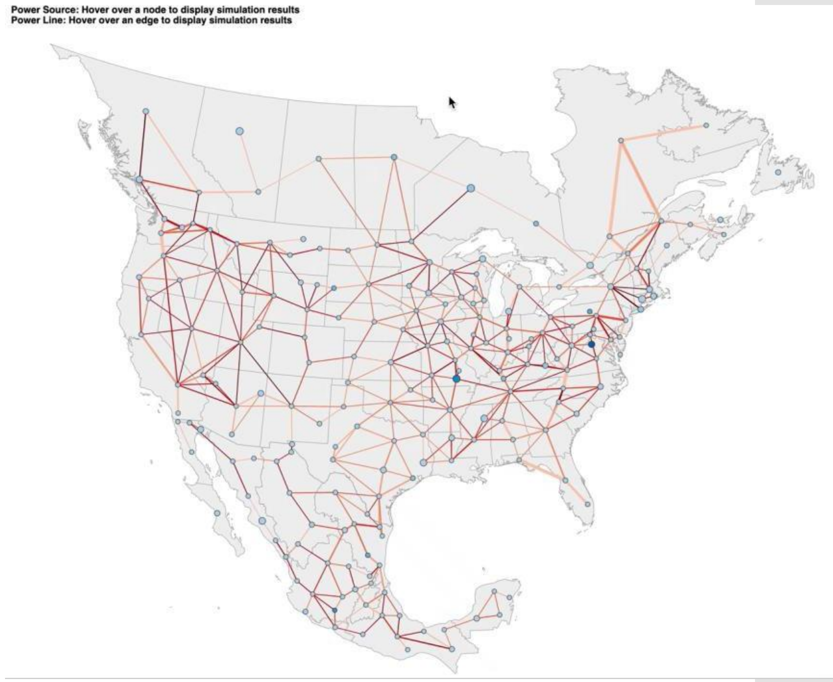
Circle-packing layout

Potential problem: is it easy for you to visually discern the *depth* of the tree?

How do we draw graphs *effectively?*

- Primary concern: the **spatial layout** of vertices and edges

- Often (but not always) the goal is to effectively depict the **graph structure**
  - Connectivity, path-following
  - Network distance
  - Clustering
  - Ordering (e.g., hierarchy level)



Visualizing the Reliability and Security of the North American Power Grid System in 2050
Work done for the National Renewable Energy Laboratory
code can be found @ https://github.com/ashleysuh/nerc-visualization

draw these graphs using https://gephi.org/

Node-link diagrams (again)

How do we draw graphs *effectively?*

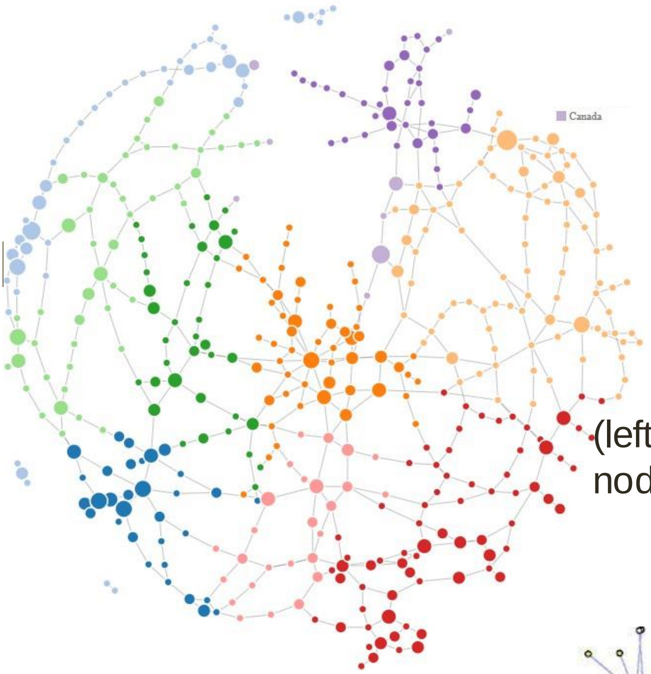
Reingold-Tilford algorithm

PROS:

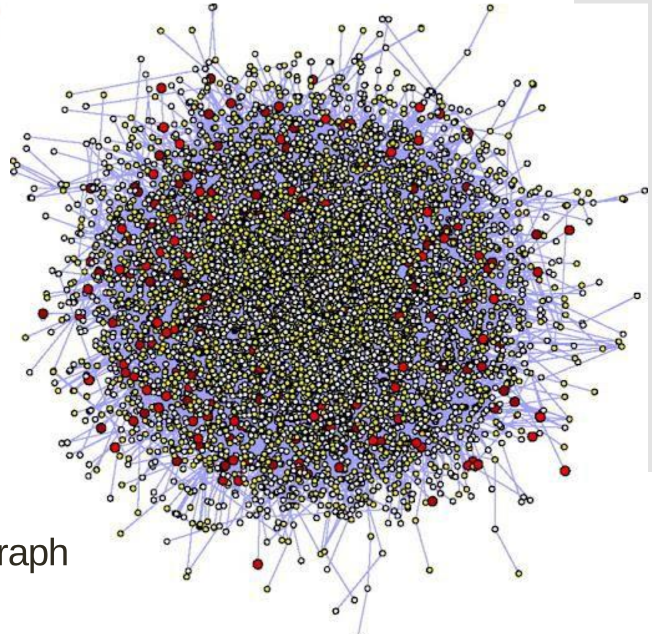
- understandable visual mapping
- shows overall structure, clusters, paths
- flexible, many variations / layouts

CONS:

- most trivial algorithms are slow
- not good for dense (very connected) graphs



(left) a nice, decluttered node-link diagram



(right) a dense, “hairball”-like graph

try this interactively using https://bost.ocks.org/mike/miserables/

How do we draw graphs *effectively?*

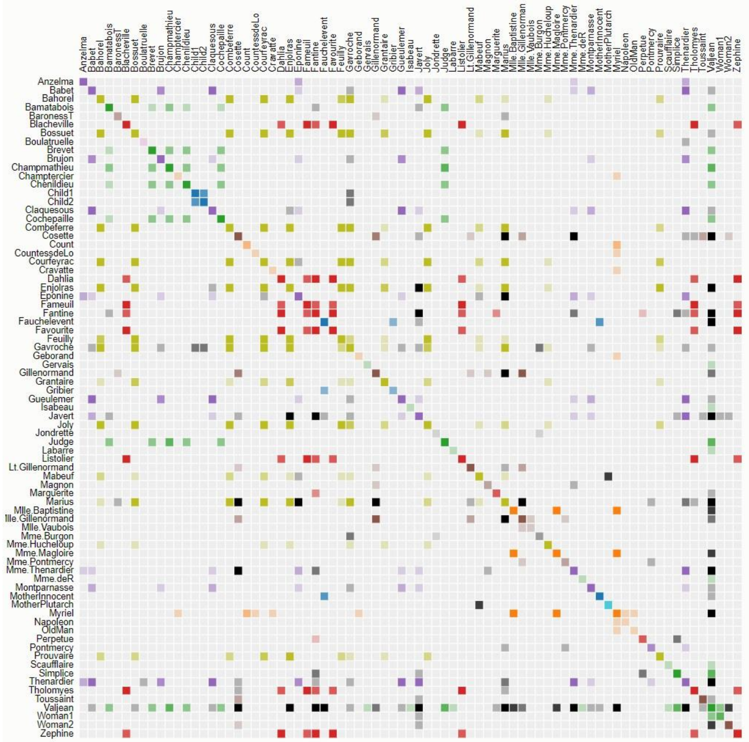
Adjacency Diagram

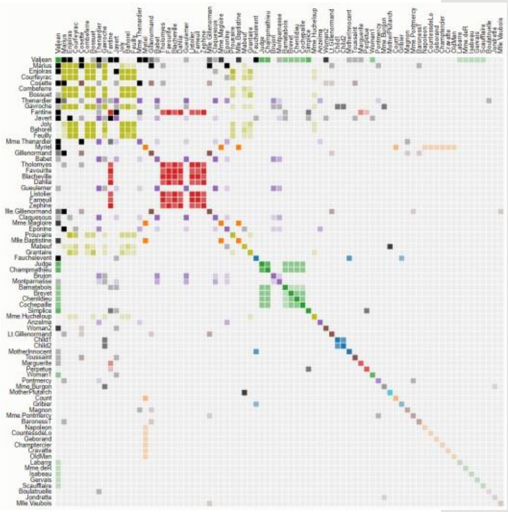
PROS:

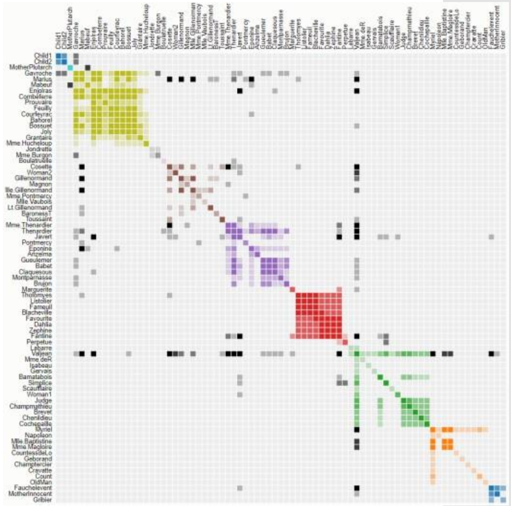
- great for dense graphs
- visually scalable
- can spot clusters

CONS:

- row order affects what you can see
- abstract visualization
- hard to follow paths







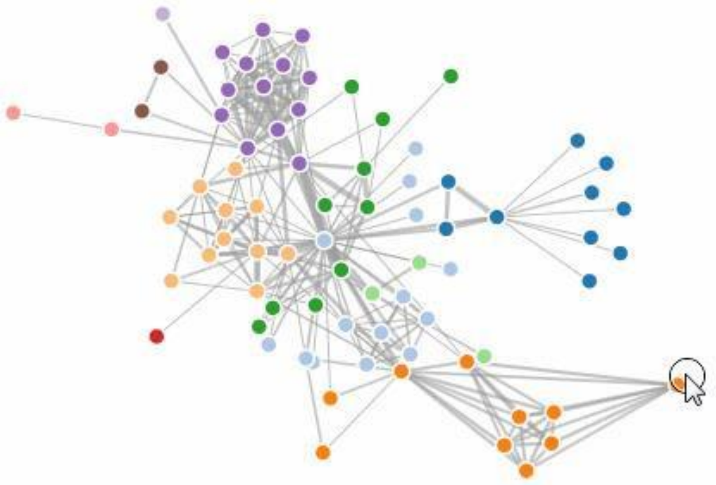
Why does ordering matter?
(above) layout ordered by Name
(left, top) layout ordered by Frequency
(left, bot) layout ordered by Cluster

try it interactively at https://observablehq.com/@d3/force-directed-graph

How do we draw graphs *effectively?*

Force-directed graph drawing

Physical-based model (attractive & repulsive forces)



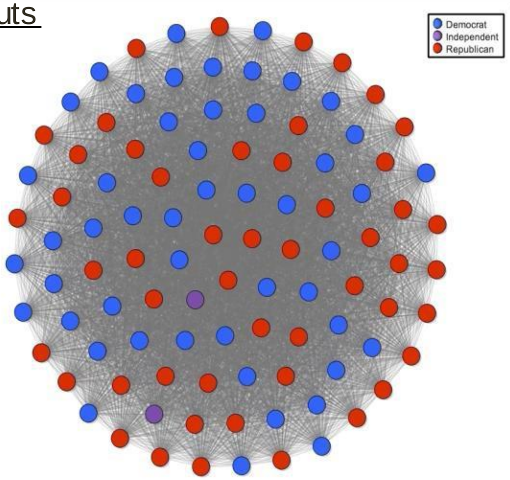
PROS:

- aesthetically-pleasing layout
- interactive (pull & drag!)
- automatic & flexible layout

CONS:

- forces are computationally expensive ~ $O(n^2)$
- doesn't work well on dense graphs

Interactive force-directed layouts
(above) Les Mis dataset
(below) Voting network



Potential problem: how can we interact with a force-directed graph if it's highly connected?

Practice

Exercise: draw the following graph data as an **adjacency matrix** & **node-link diagram**

V = {1, 2, 3, 4, 5}

E = {(1,2), (1,3), (2,3), (3,4), (4,1), (5,1)}

Edge weights = {(1), (7), (4), (2), (2), (1)}

Node Classes = {A, B, A, A, B}

- Draw a node-link diagram in two ways:
    1. Bad layout
    2. Aesthetically-pleasing layout
- Be creative with your attribute encodings! Think about:
    1. What are the advantages / disadvantages to these methods? Which do you prefer?
    2. Why is your bad layout ‘bad’, and your good layout ‘good’?

Tools for Graph Analysis

Network Analysis Tools

- Gephi - an interactive graph analysis application
- NodeXL - a graph analysis plug-in for Excel
- GUESS - a combined visual/scripting interface for graph analysis
- Pajek - another popular network analysis tool
- NetworkX - graph analysis library for Python
- SNAP - graph analysis library for C++

Network Visualization Coding References

- Python
  - https://plotly.com/python/network-graphs/
  - https://towardsdatascience.com/visualizing-networks-in-python-d70f4cbeb259
  - https://pyvis.readthedocs.io/en/latest/
- R
  - https://schochastics.github.io/netVizR/
  - https://r-graph-gallery.com/network.html
  - https://yunranchen.github.io/intro-net-r/advanced-network-visualization.html