Universitat de Girona
Fundació UdG: Innovació i Formació

ViT Visualization for Transparency Foundation

# MVTEC

Data Visualization Programming Anton Bardera Data Visualization Programming Anton Bardera Data Visualization Programming Anton Bardera **Data Visualization Programming** Anton Bardera Data Visualization Programming Anton Bardera Data Visualization Programming Anton Bardera Data Visualization Programming Anton Bardera Data Visualization Programming Anton Bardera Data Visualization

# Visual Encoding

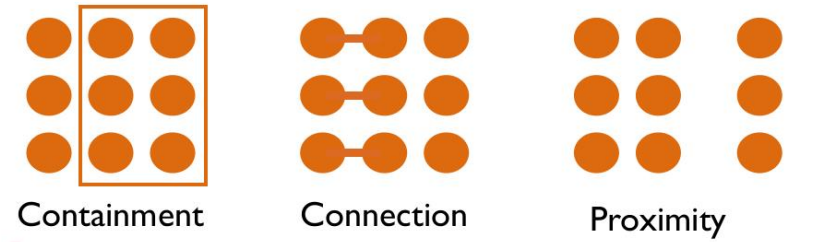
- The way in which data is mapped into visual structures, upon which we build the images on a screen
- Items or links are represented using **marks** or geometrical primitives
- The changes on the mark's appearance based on a data attribute are called **channels**

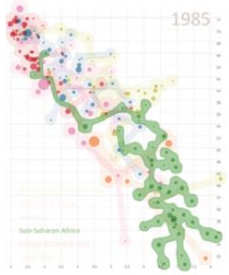
*Channel = Visual Variable*

# Marks

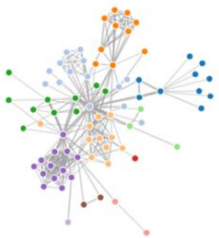
- Marks for Items: basic geometric elements
  - Points
  - Lines
  - Areas
  - Volume: rarely used


- Marks for Links
  - Containment
  - Connection
  - Proximity

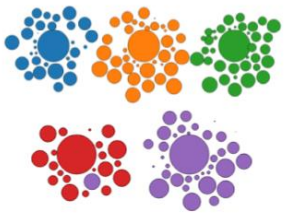



Collins et al. 2009



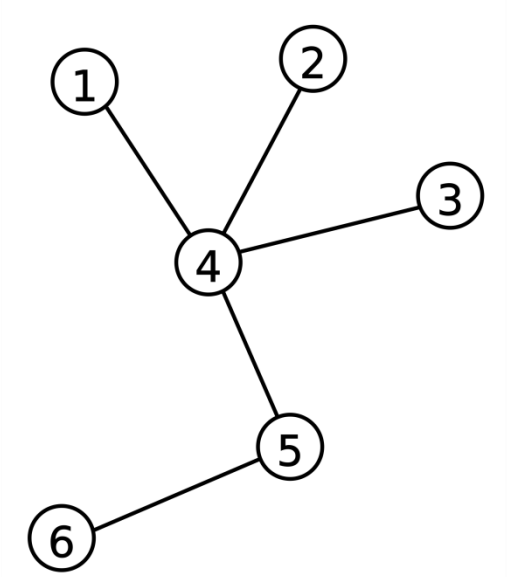
D3.js Example



D3.js Example

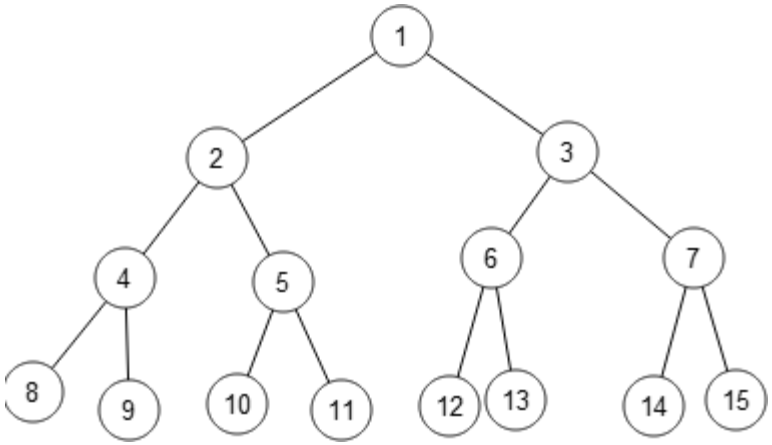
# Graphs and Trees

- A graph *G* consists of a collection of vertices or nodes *V* and a set of edges *E*
- Each edge *e* connects two vertices *x* and *y*
- For example: V={1,2,3,4,5,6}, E={(1,4),(2,4),(3,4),(4,5),(4,6)}



# Graphs and Trees

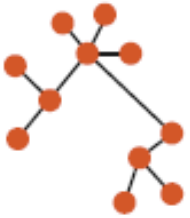
- **Trees:** graphs with no cycles
- *Connected graph with N-1 edges*
- Represent hierarchical structure
- Nodes are related as *parent, children, sibiling (family tree)*
- Root node: highest hierarchy



# Graphs representations

- Three types:

**Node–Link Diagrams**
Connection Marks
✔ NETWORKS ✔ TREES



**Adjacency Matrix**
Derived Table
✔ NETWORKS ✔ TREES

**Enclosure**
Containment Marks
✖ NETWORKS ✔ TREES



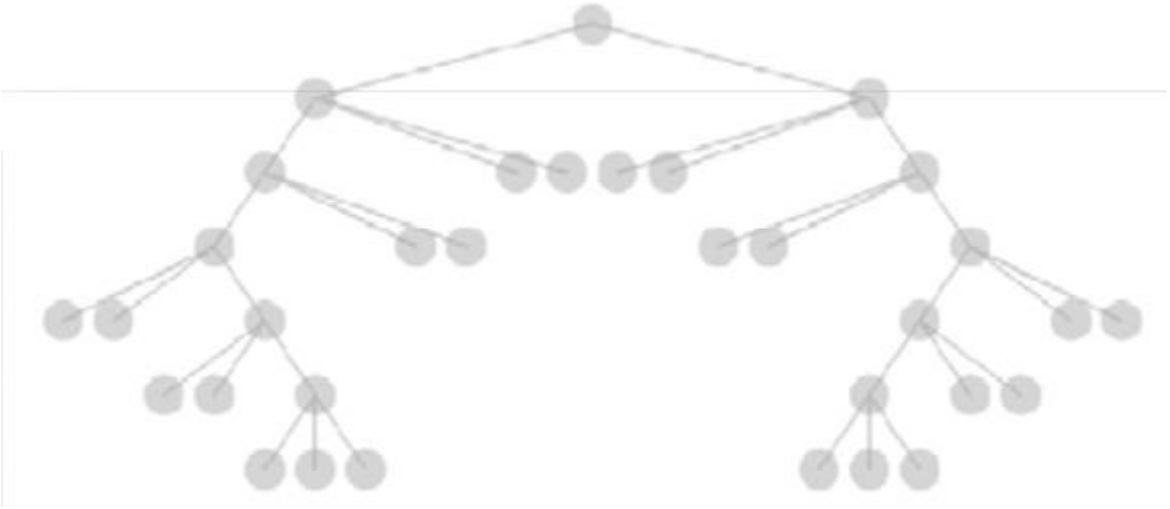
From T. Munzer, 2014

# Tree visualization

- Node-link approach
  - Nodes are distributed in space
  - Edges represented by straight or curved lines
  - Typical approach in tree visualization is to use X for breadth and Y for depth (or viceversa)
  - Often space is used to communicate hierarchical orientation
  - Recursion makes it elegant and fast to draw trees

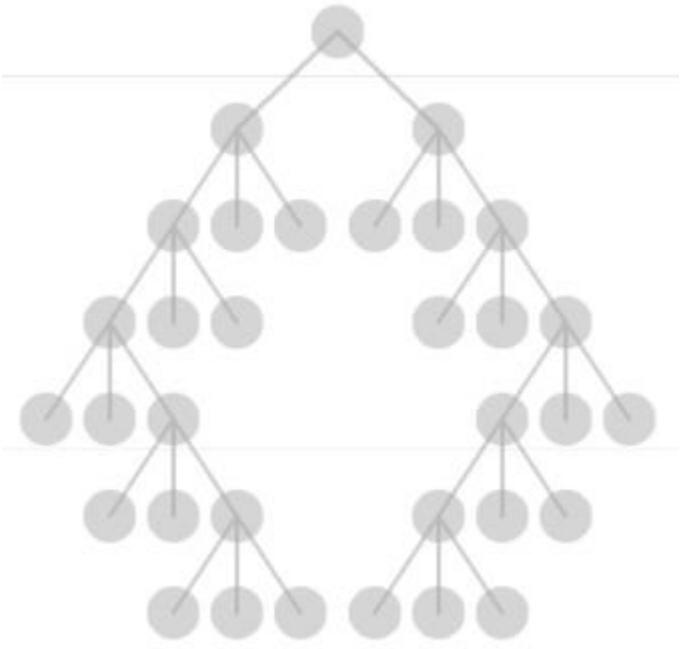
# Tree visualization

- **Naïve approach**: repeatedly divide space for subtrees by leaf count
- Problems:
  - Exponential growth of breadth
  - No efficient use of the whole canvas



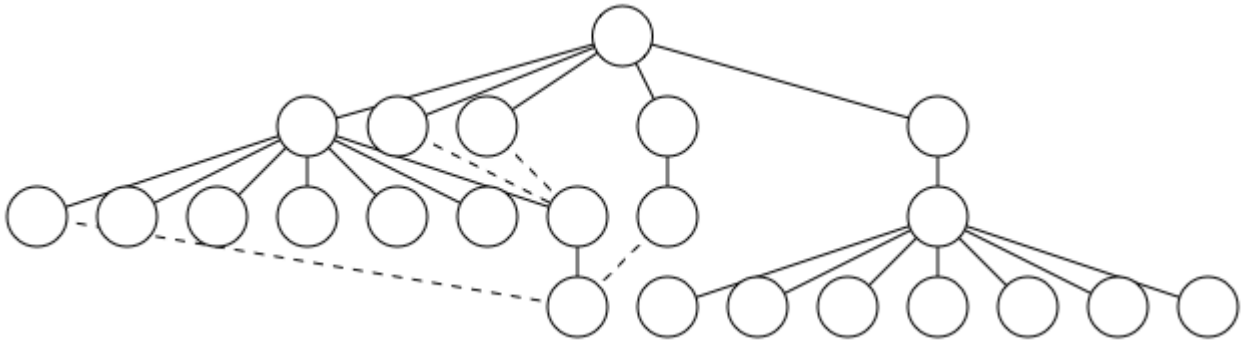
# Tree visualization: Reingold-Tilford

- **Goal**
  - make smarter use of space
  - maximize density and symmetry
- **Design concerns**
  - clearly encode depth level
  - no edge crossings
  - isomorphic subtrees drawn identically
  - compact
- **Approach**
  - bottom-up recursive approach
  - for each parent make sure every subtree is drawn
  - pack subtrees as closely as possible
  - center parent over subtrees



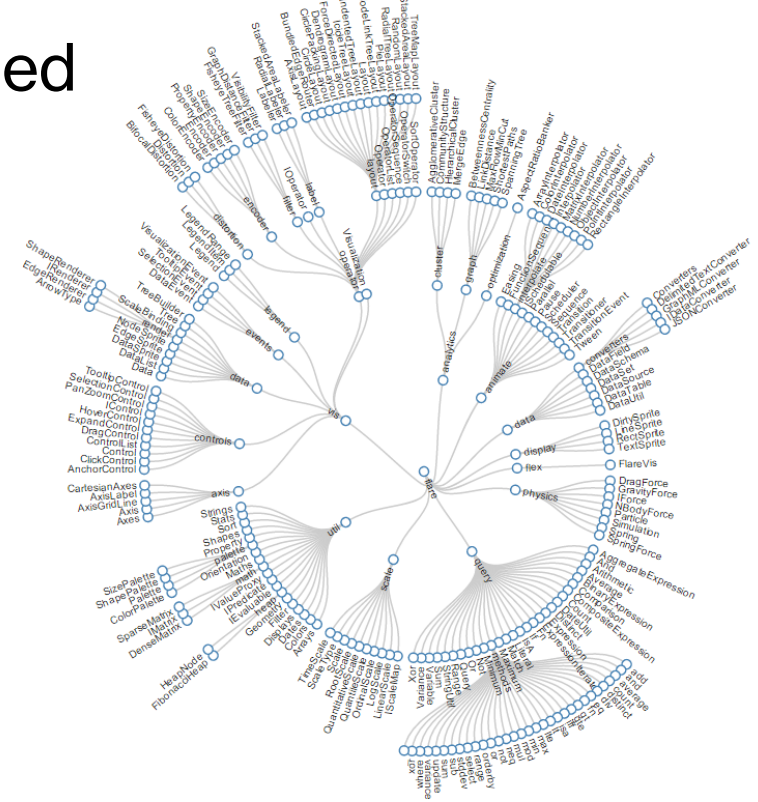
# Tree visualization: Reingold-Tilford

- Algorithm: original paper
  - Do a post-order traversal of the tree
  - if the node is a leaf, give it an x coordinate of 0
  - otherwise, for each of its children, place the child as close to its left sibling as possible
  - place the parent node halfway between its leftmost and rightmost children



# Tree visualization: Radial layout

- Node-link diagram in polar coordinates
- Radius encodes depth with root in center
- Angular sectors assigned to subtrees
- Reingold-Tilford can be applied



# Tree visualization: Node-link problems

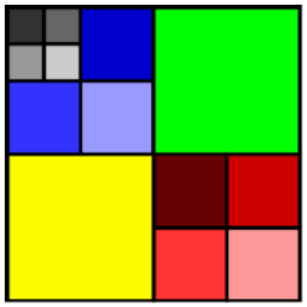
- **Scale**
  - tree breadth often grows exponentially
  - quickly run out of space!
- **Solutions**
  - hyperbolic layout
  - filtering
  - scrolling or panning
  - zooming

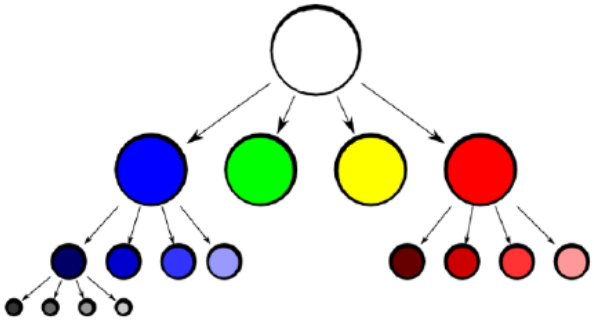
# Tree visualization: Enclosure diagrams

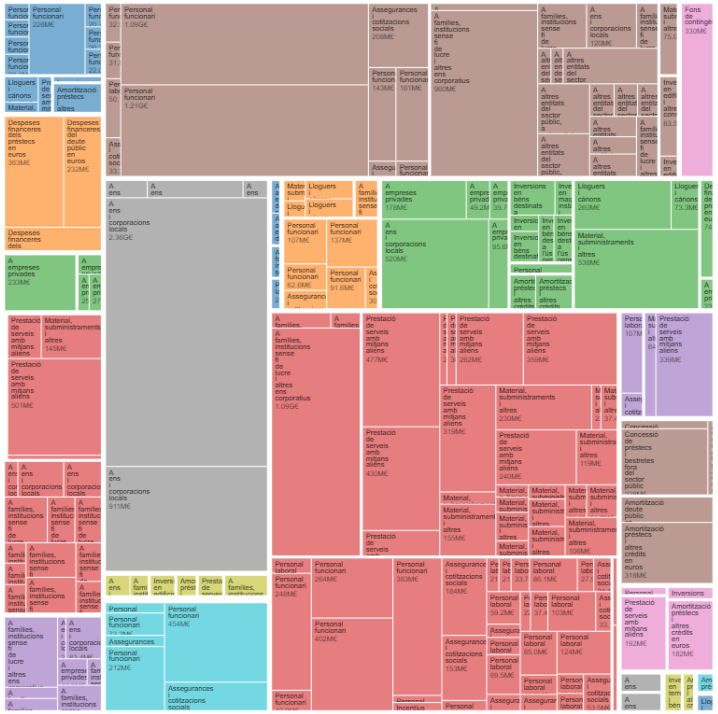
- **Encode structure using spatial enclosure**
  - often referred to as *treemaps*
- **Benefits**
  - provides single view of entire tree
  - easier to spot small / large nodes
- **Problems**
  - difficult to accurately read depth

# Tree visualization: Treemaps

- recursively fill space based on a size metric for nodes
- enclosure indicates hierarchy
- additional measures can control aspect ratio of cells

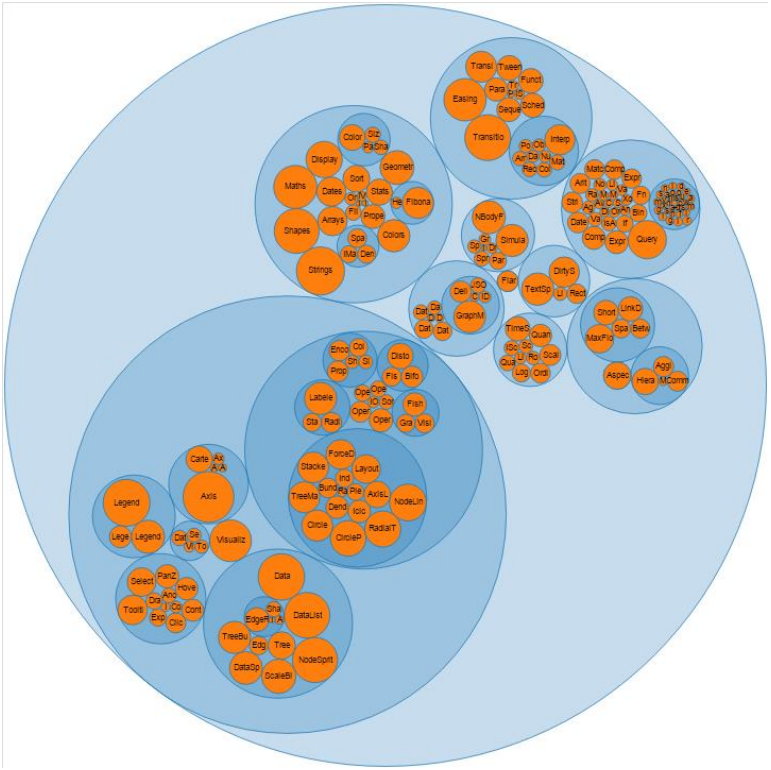


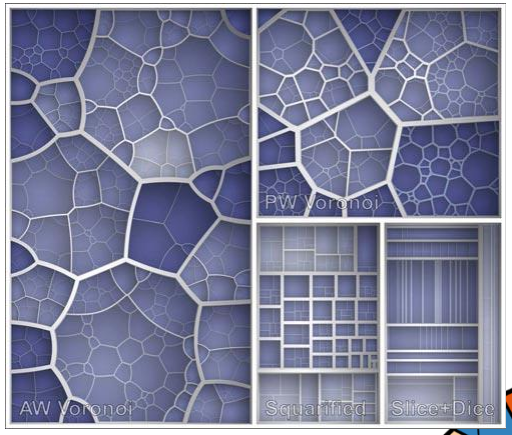


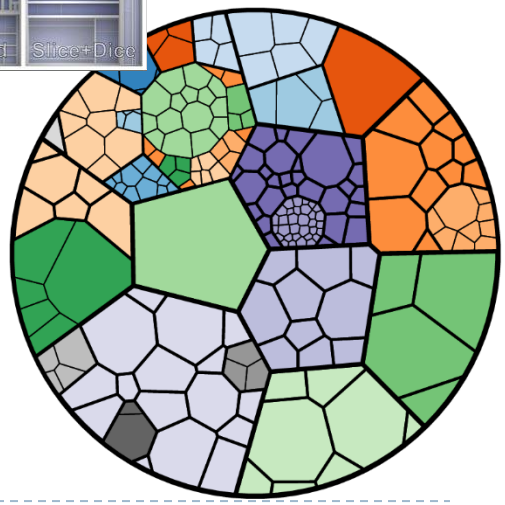


# Tree visualization: Treemaps

- Most often use rectangles, but other shapes are possible: circles, Voronoi tessellation,...

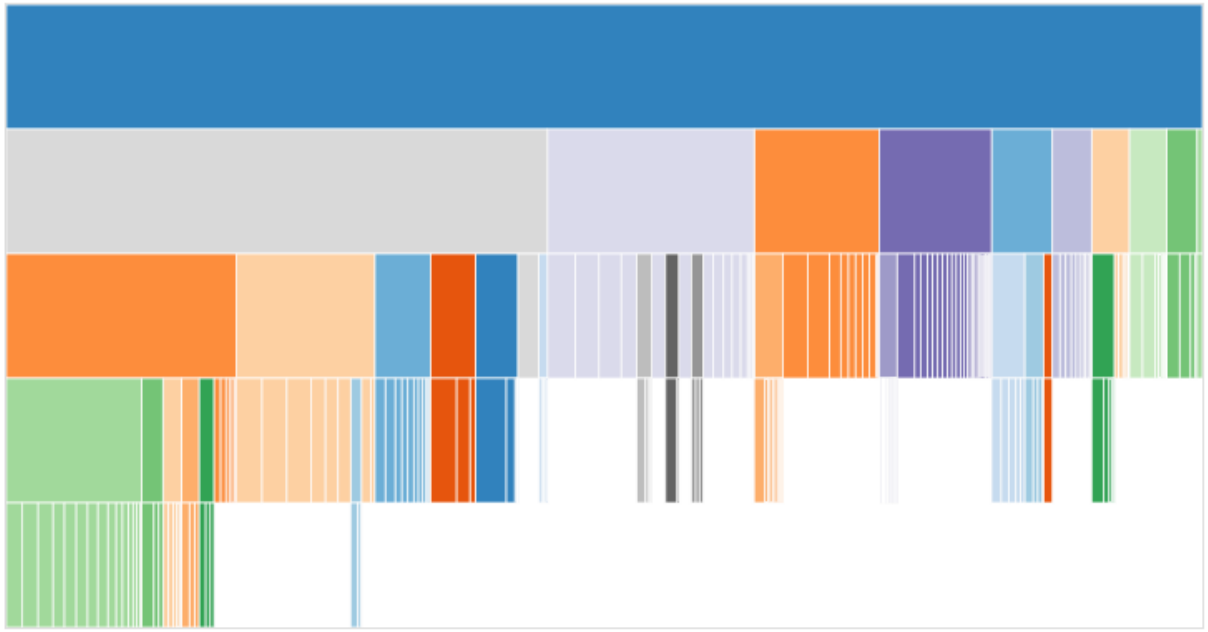




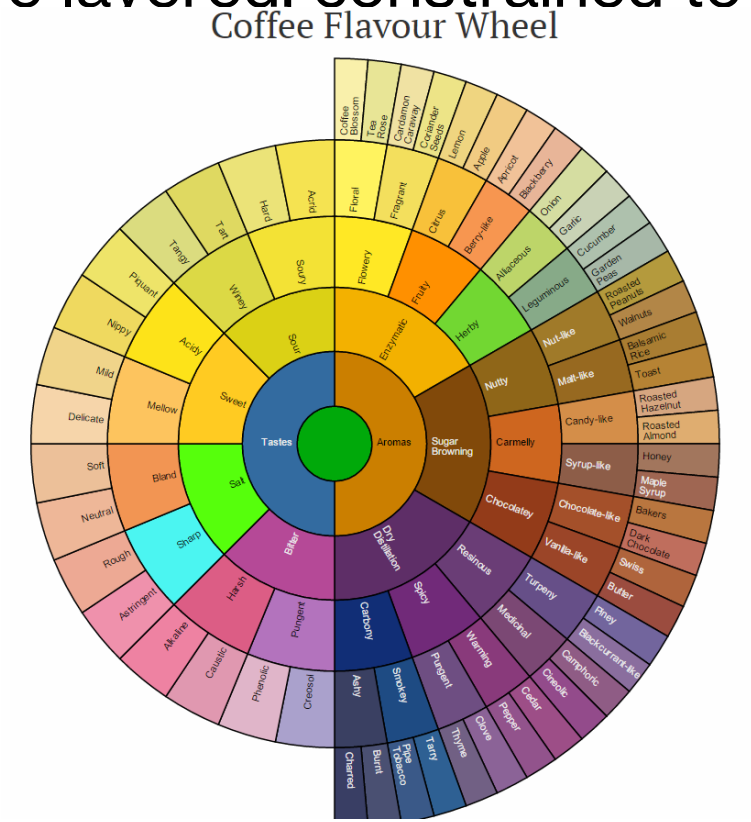
# Tree visualization: Icicle

- Higher-level nodes get a larger layer area, whether that is horizontal extent.
- Child levels are layered, constrained to parent's extent



# Tree visualization: Sunburst Trees

- Similar than icicle using circular layout and angular extent
- Child levels are lavered. constrained to parent's extent



# Tree visualization survey

- http://treevis.net/