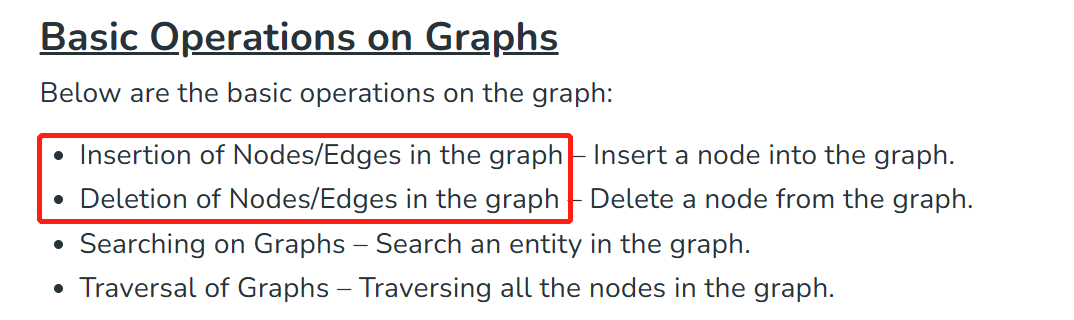
1. **Single causal graph**

**Design requirements:**

1. Able to present **large causal graphs**, nodes and directed edges—computational efficiency; [Adding edge or removing edge or initializing a graph all require computational power, so for large graphs with multiple nodes and edges, computational efficiency is very important.]



1. Show how a factor affects an outcome, **quantitatively** **estimate** the influence of each dimension;
2. Show the **uncertainty** (by computing the value of OIN/OCF (d), mentioned in step 3 below) of the detected causal relations (computed in step 2) [refer to doi: [10.1109/TVCG.2021.3114779](https://doi.org/10.1109/TVCG.2021.3114779) ] to support identification of **spurious causalities**;
3. Incorporate **human knowledge** in the causal model refinement;
4. Provide **diagnostic measures on model quality** [refer to 10.1109/TVCG.2020.3030465 ];
5. Support **real-time graph update**.

**1. Measure how strongly a feature is associated with the chosen outcome**

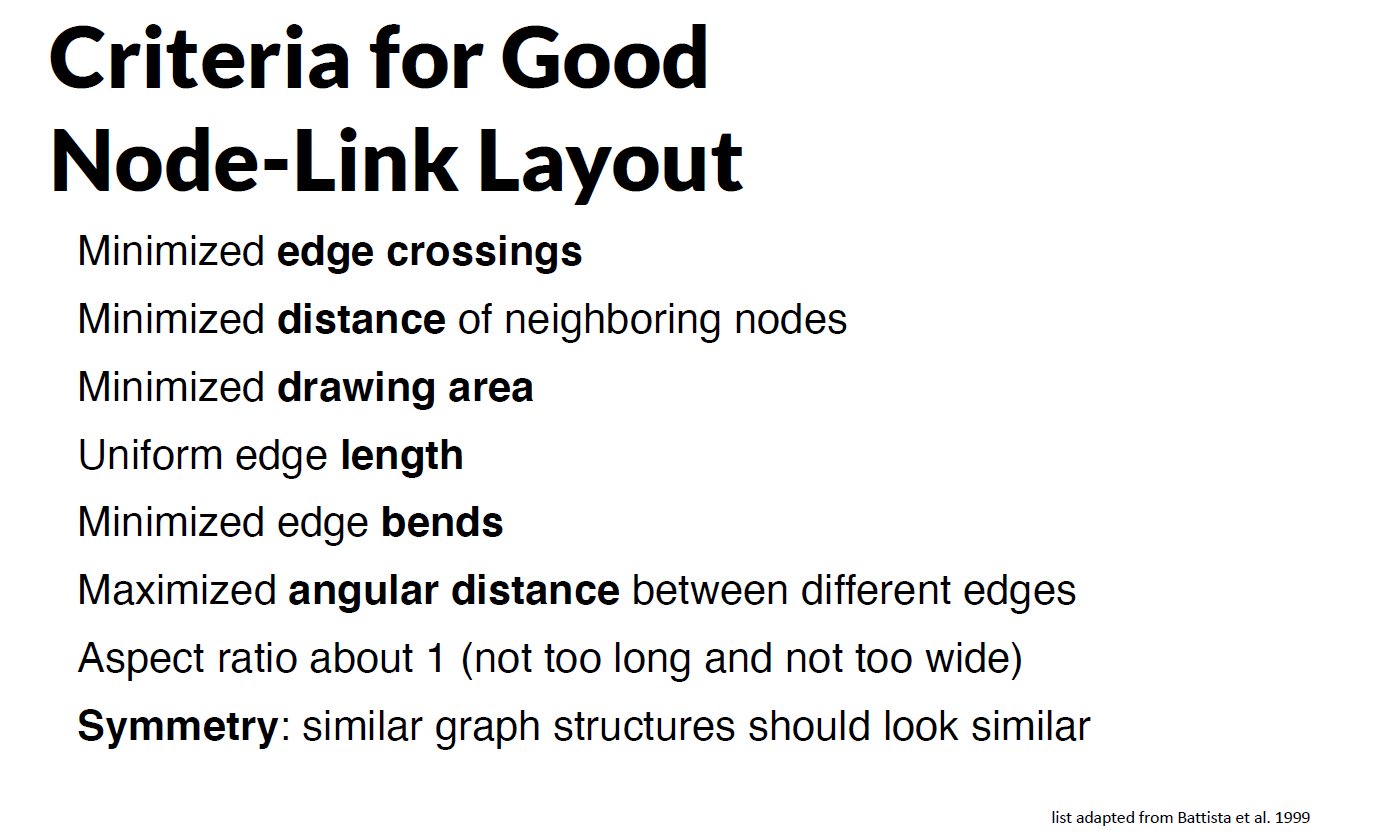
* Both numerical: Pearson Correlation Coefficient
* Categorical and numerical: multiple regression R2
* Both categorical: Cramer’s V

Select the top n highly relative features and show the distributions of the selected features.

**2. Causal discovery algorithm (PC or F-GES) to draw a graph**

**2.1 Graph layout**

Consider the criteria mentioned below, but be cautious. **[proposed by Professor Daniel]**

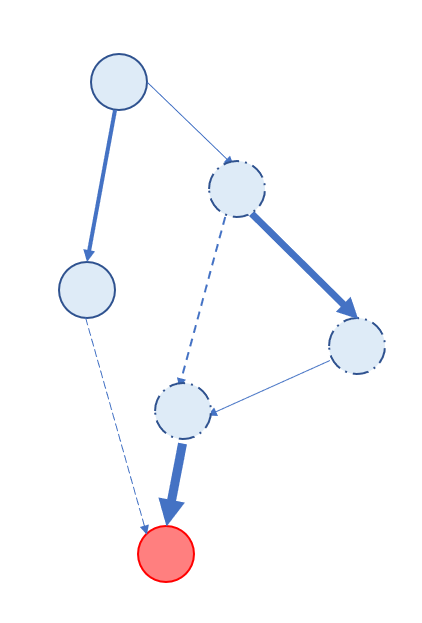


We adapt existing **layered graph layouts** [4] and techniques for reducing edge-crossings [19] to the causal graph visualization.【X. Xie, F. Du and Y. Wu, "A Visual Analytics Approach for Exploratory Causal Analysis: Exploration, Validation, and Applications," in IEEE Transactions on Visualization and Computer Graphics, vol. 27, no. 2, pp. 1448-1458, Feb. 2021, doi: 10.1109/TVCG.2020.3028957. ----finding topological order of nodes】

**Sagiyama layout**

**2.2 Map:**

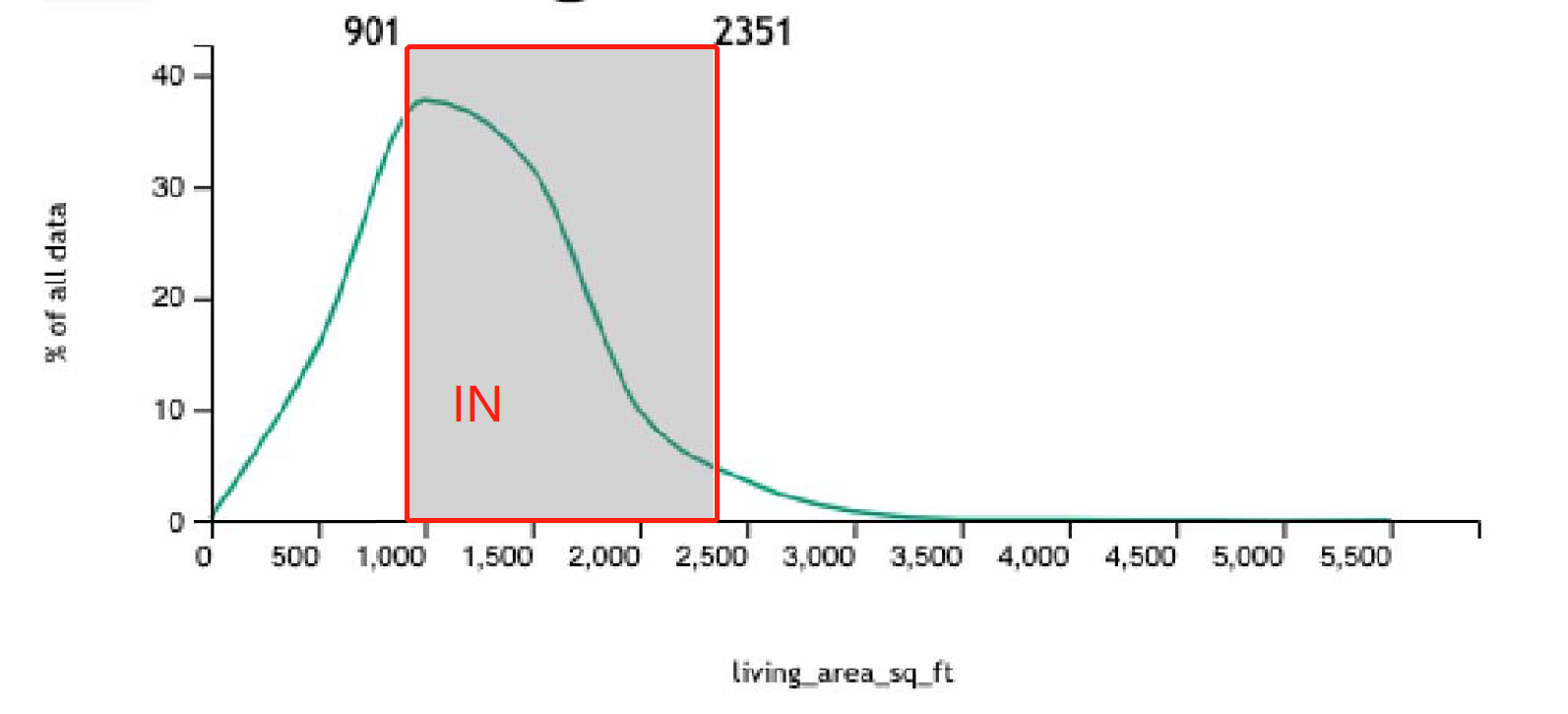
1. variable + label: node
   1. variable type: solid node (numerical), dashed node (categorical)
2. causal direction: direction of edge arrow
3. causal effect size（**only counts those with edges**）: edge weight
   1. positive effect size: solid edge
   2. negative effect size: dashed edge
4. Outcome is highlighted.

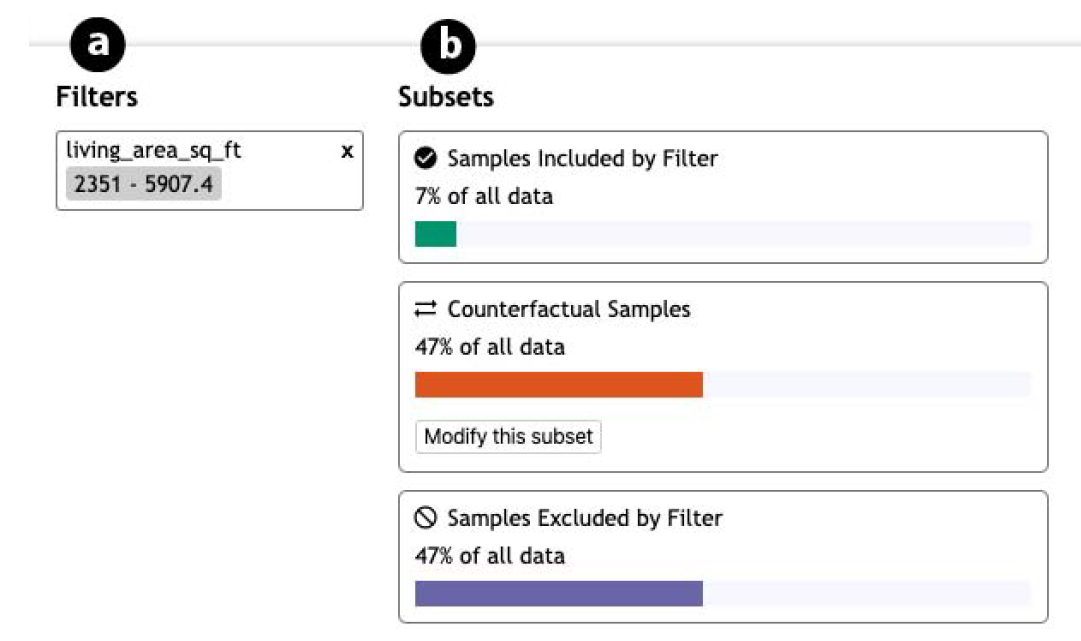


**3. counterfactual approach to identify potential confounding**

***[ S. Kaul, D. Borland, N. Cao and D. Gotz, "Improving Visualization Interpretation Using Counterfactuals," in*IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 998-1008, Jan. 2022, doi: 10.1109/TVCG.2021.3114779.]***

* Applying a filter constraint (f) based on one of the features selected in step 1 and create three subsets----the included subset (IN)、counterfactual subset (CF)、excluded subset (EX)





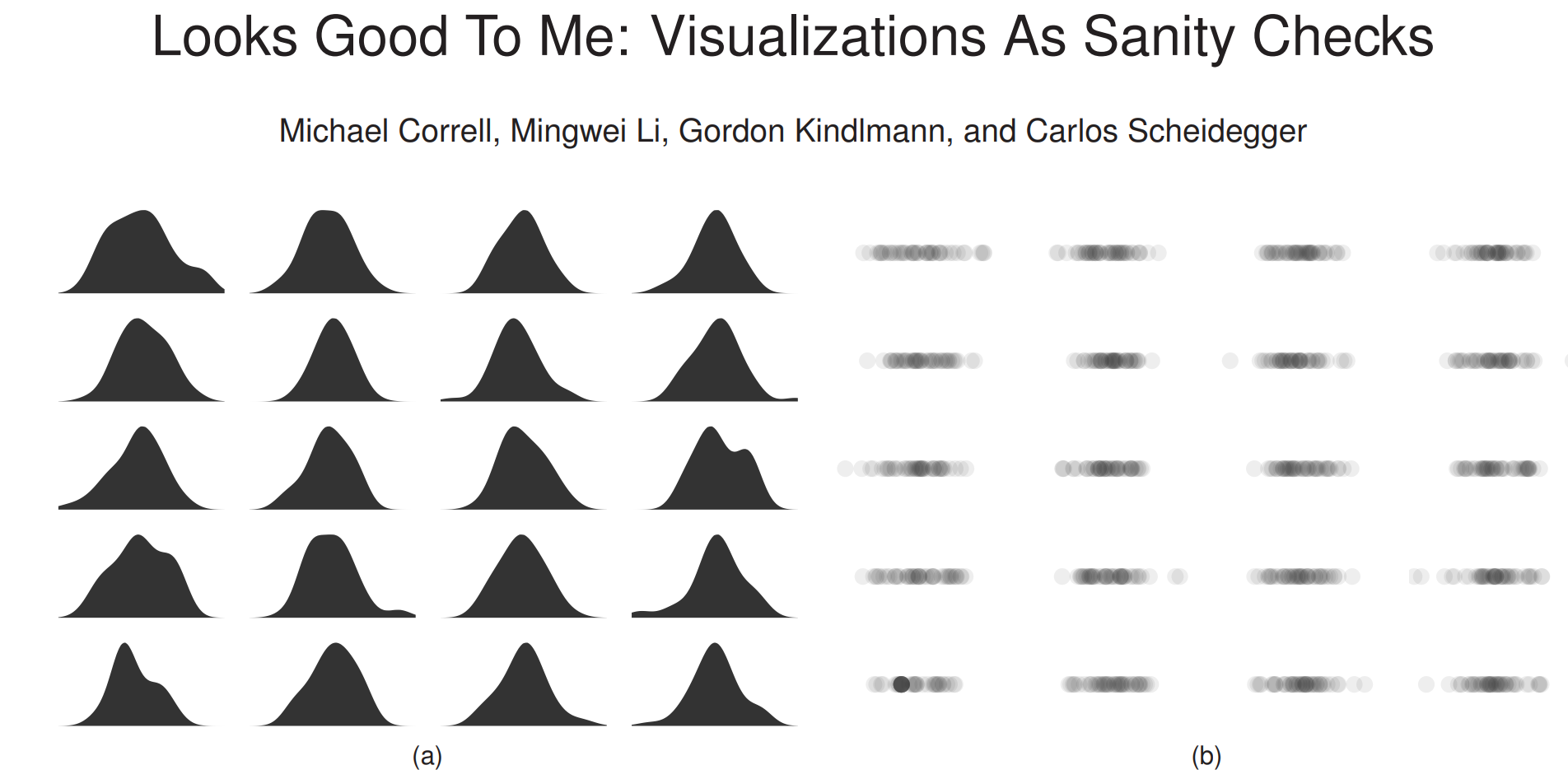
* Observe impact on the **target node** distributions of the IN, CF, and EX subsets of the **start node** ---- OIN, OCF, OEX

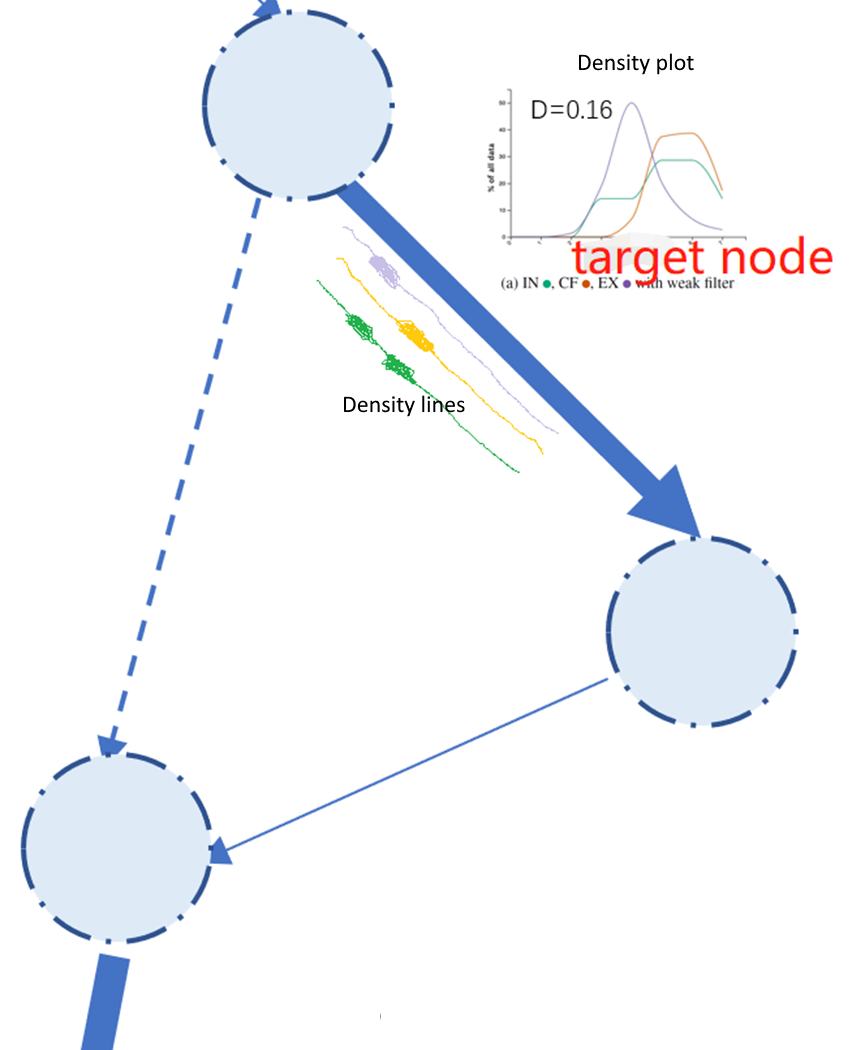
Compute the value of OIN/OCF (d) to measure the difference between OIN and OCF,

the **Hellinger distance** [62] is used for categorical outcomes and the **Kolmogorov-Smirnov test** [45] for numerical outcomes

D ranks between 0-1, **0 corresponds to no impact by f (i.e., OIN = OCF) [likely to be a confounding variable]** and 1 corresponds to a large impact (i.e., a maximal difference between OIN and OCF).

Example lineups refer to 10.1109/TVCG.2018.2864907

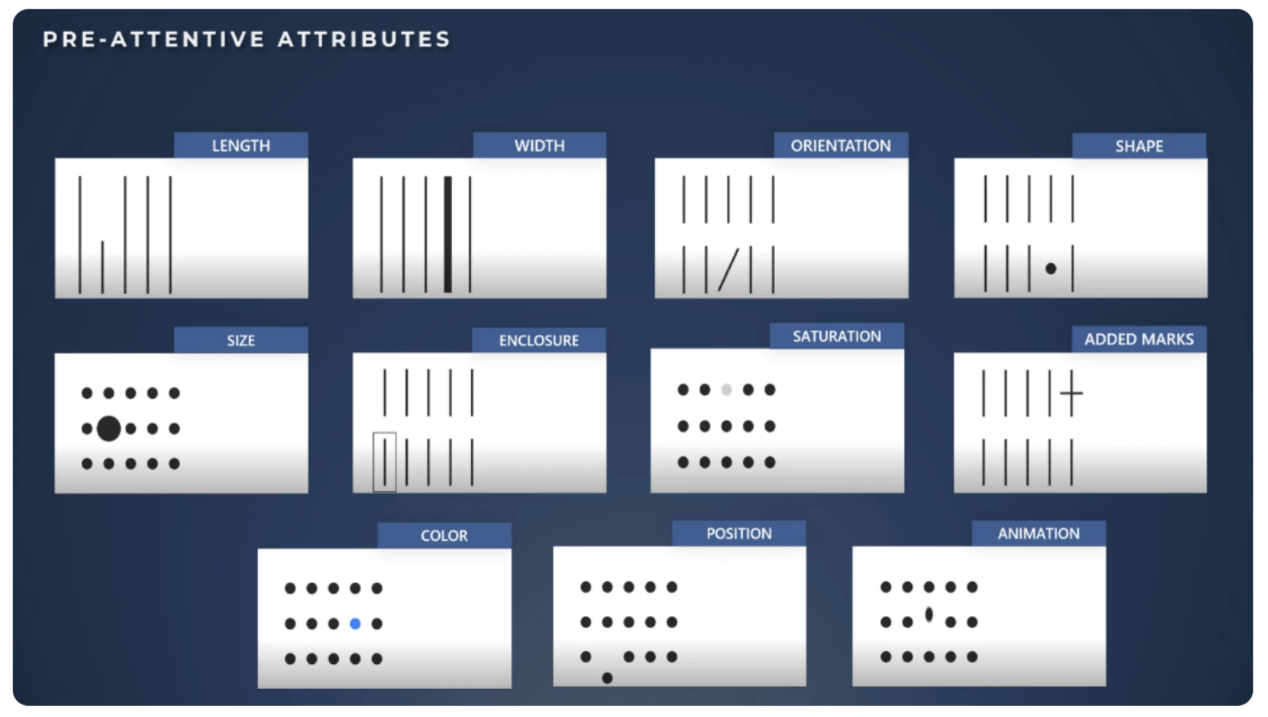




For **each edge**, there should be three density lines for the distributions of all subsets (IN, CF, EX) and a distribution plot containing three subsets, illustrating the end node’s distribution based on the start node’s filter, respectively. Also, a directed weighted line illustrates the causal effect and direction.

**4. manually delete edges based on domain knowledge and potential confounding detected in step 3**

May also consider **pre-attentive mechanism** to highlight some different directions, which has the potential to hint possibly wrong causal relations.



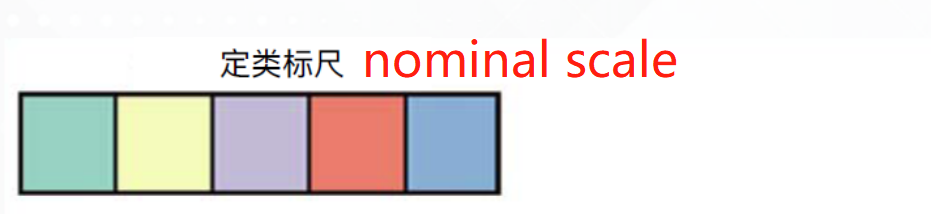
Supporting real-time update → save specific single graph to history

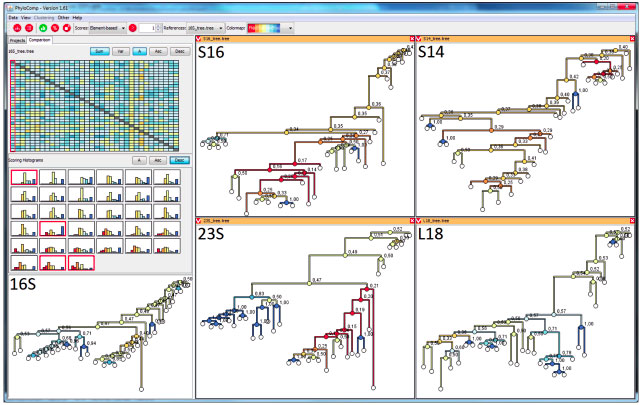
1. **Multiple causal graphs**

**Design requirements:**

Allow **comparison** of causalities for different outcome groups;

Clearly **distinguish between** multiple groups (color);

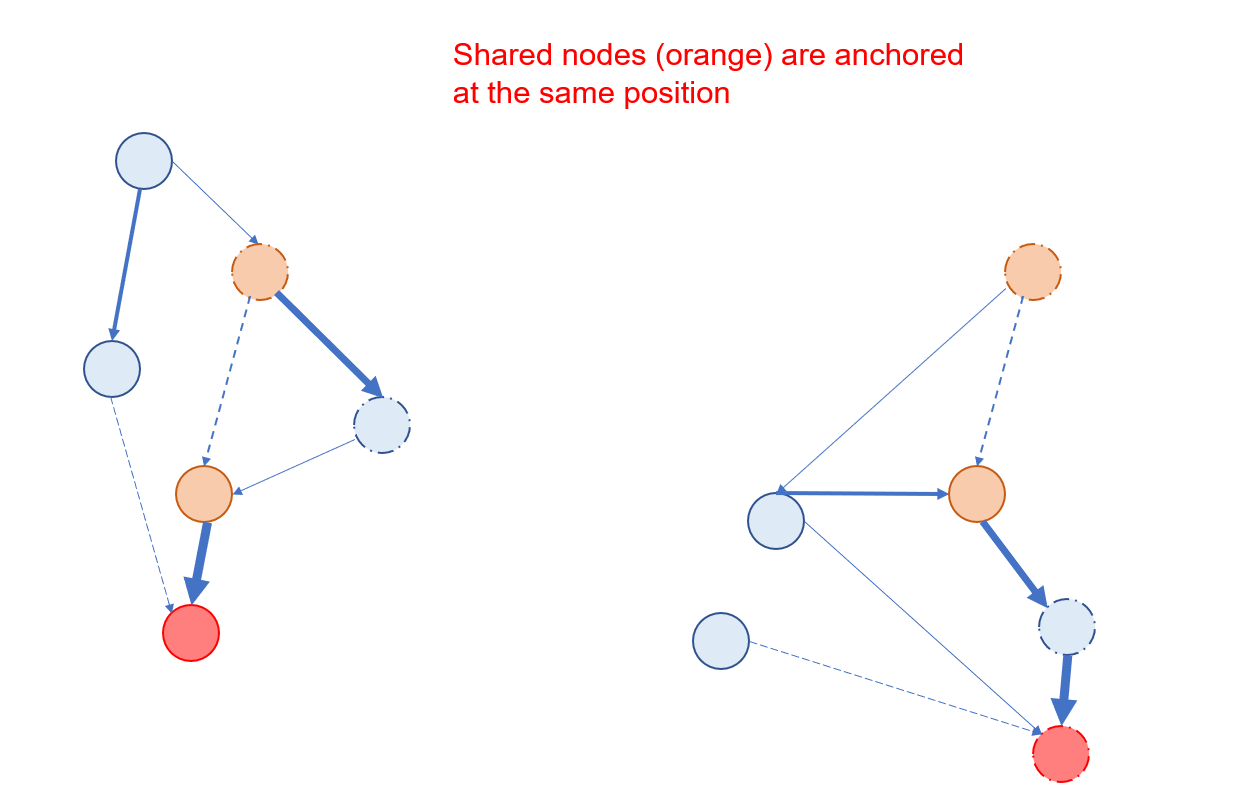




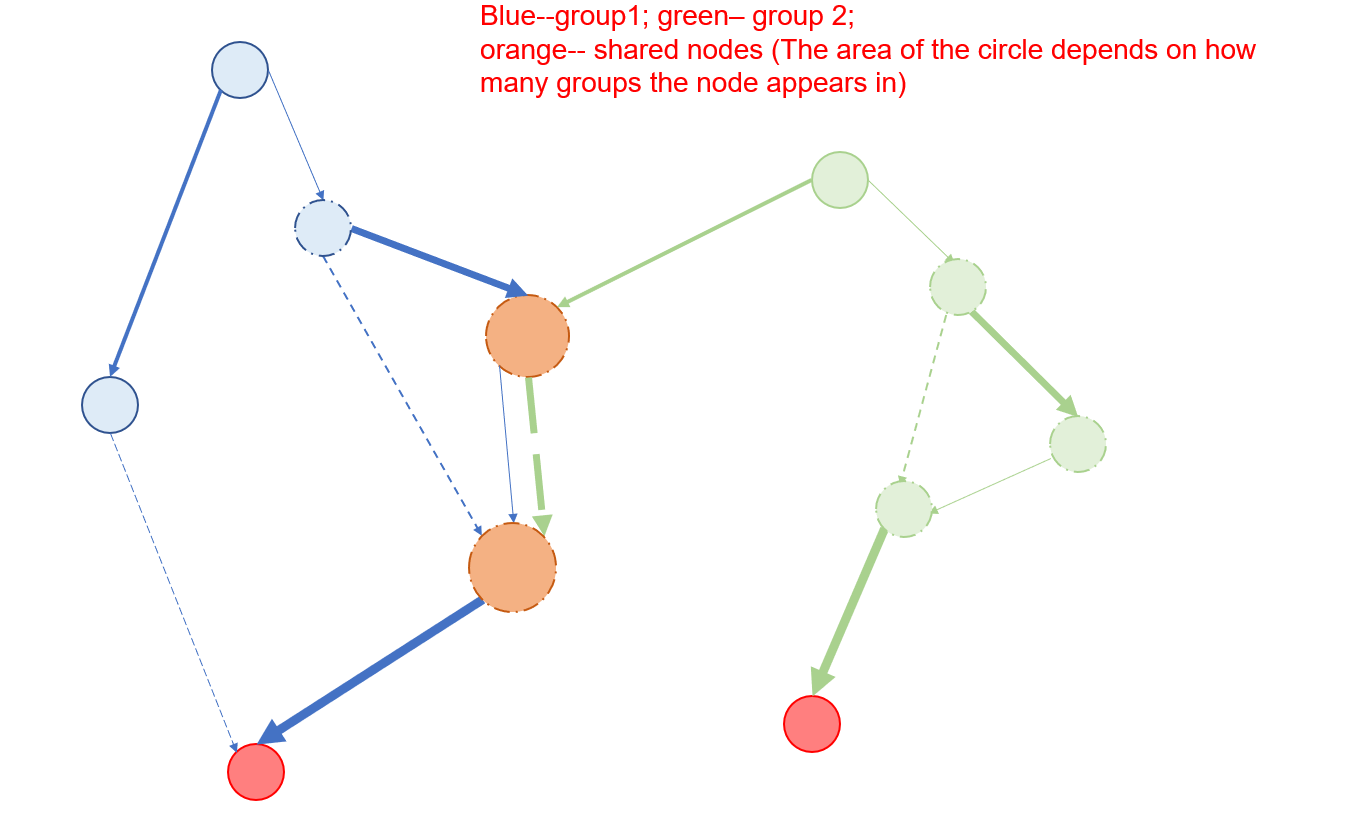
Use the **matrix** as an **overview diagram** to list the similarities between any two graphs, thus helping the user select a portion of the graph for comparison.

**method 1:**

juxtaposition ---- anchor and highlight the same nodes in different groups.



**method 2:** merge same nodes



[TensorBoard 分层复合图可视化布局技术介绍 (yuque.com)](https://www.yuque.com/antv/g6-blog/gzu4hu)

**Hierarchical graph/ layered graph** is a graph with hierarchical structure obtained by layering nodes in a directed graph according to the direction of edges. [**single causal graph**]

**Compound graph** refers to the directed graph with grouping relation of nodes. The existence of grouping relationship leads to the need to give priority to more similar positions of nodes in the same group in the layout process.

**Hierarchical composite diagrams** are diagrams that visualize both grouping and hierarchical relationships. The sequence and combination relationships between nodes can be displayed more intuitively. [**multiple causal graphs**]

Many existing Graph visualization libraries support hierarchical composite graph visualization, such as Dagre, Tensorboard Graph ([Examining the TensorFlow Graph  |  TensorBoard](https://www.tensorflow.org/tensorboard/graphs)), ccNetViz, etc.

**Dagre.js** (<https://github.com/dagrejs/dagre-d3>)is a classic hierarchical graph visualization library. But the underlying Graphlib doesn't deal with grouping relationships. Contributors to the Dagre library join the Cluster later, but the space utilization is not high (<https://github.com/dagrejs/dagre/issues/13>).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| technique | performance | layout | Space efficiency | Independent component |
| Dagre Cluster | fine | fine | low | yes |
| ccNetViz | excellent | ordinary | low | yes |
| JointJS | fine | ordinary | low | yes |
| TensorBoard | excellent | excellent | high | no |