VAD Chapter 4 – Analysis Supplemental Slides

CS 725/825

Michele Weigle

Nested Model

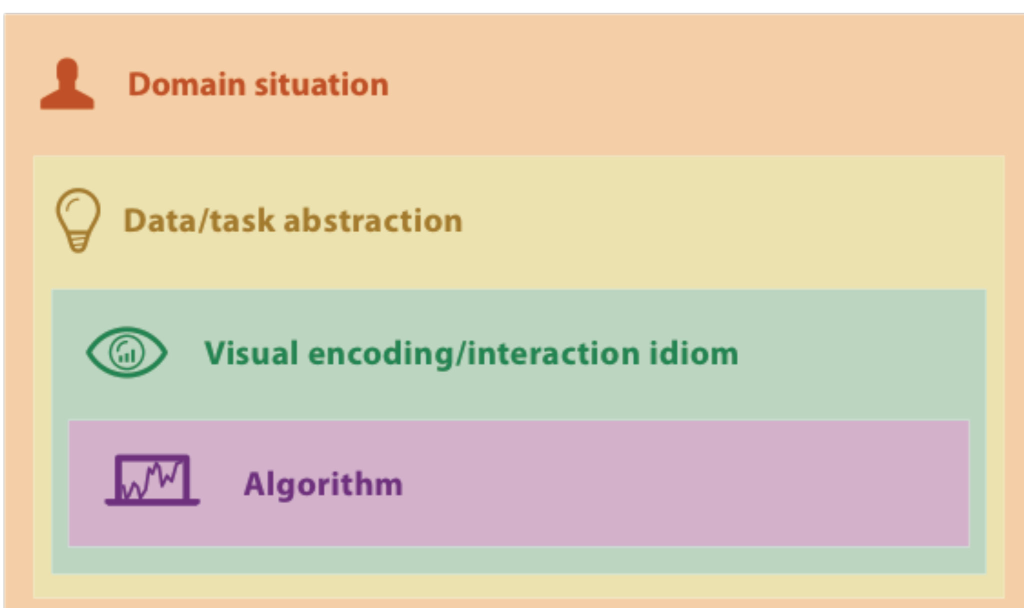
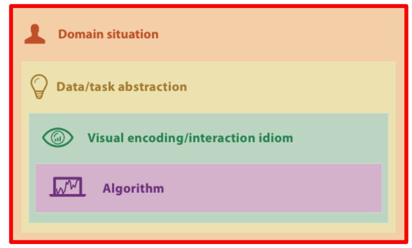


Fig 4.2, VAD, Munzner

Domain Situation

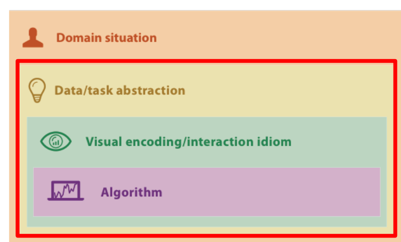


- Each domain has its own vocabulary for describing data and problems

- Common pitfall: Cutting corners by making assumptions rather than actually talking to target users

- Outcome: A detailed set of questions asked about or actions carried out by target users

Data/Task Abstraction



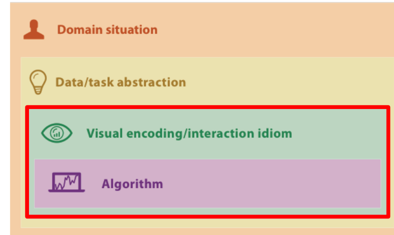
- Abstracting specific domain questions and data into a generic representation

- Task blocks are identified by the designer
  - browsing, comparing, summarizing, ...

- Abstract data blocks are designed
  - use original data, transform data, derive data, ...

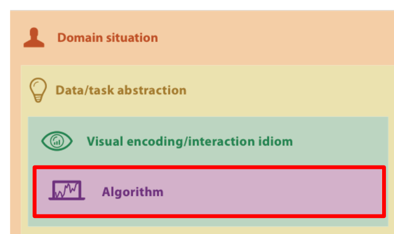
- Many vis idioms are specific to a particular data type
  - which data type would support a visual representation of the data that addresses the user's problem

Visual Encoding/Interaction



- Decide on the specific way to create and manipulate the visual representation of the abstract data block
- Visual encoding idiom
  - controls exactly what users see
- Interaction idiom
  - controls how users change what they see

Algorithm



- Algorithm used to instantiate the idiom
- Deals mainly with computer graphics implementation details
- In most cases, we're using d3 constructs to build the idioms, so we have no control over the underlying rendering algorithm
- Some may use this level in efficiently loading/pre-filtering data to reduce interaction delay

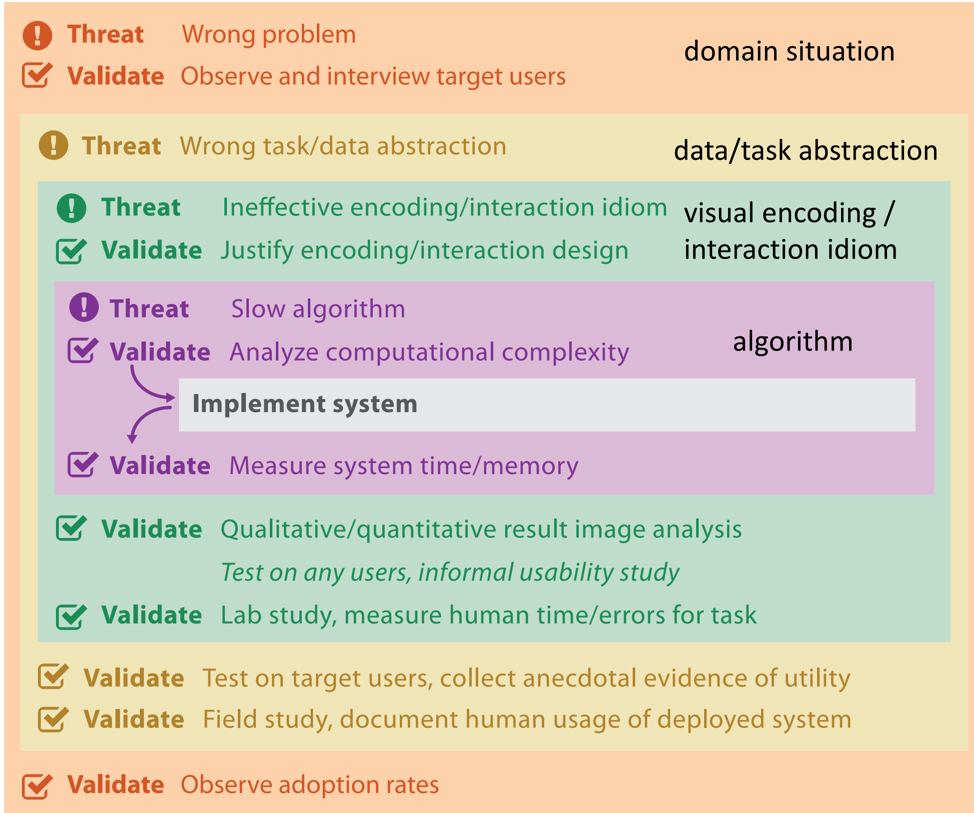
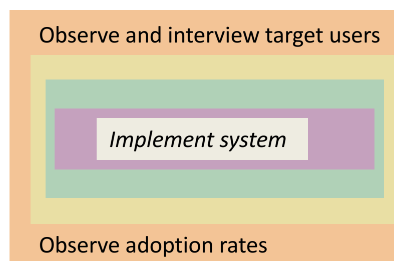


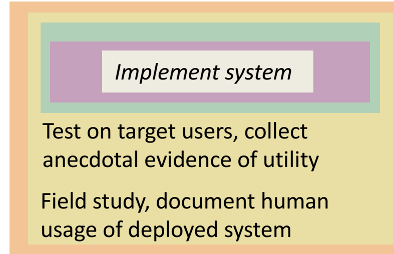
Fig 4.5, VAD, Munzner

Domain Validation



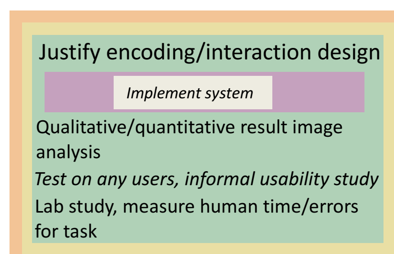
- Threat: Problem is mis-characterized
- Immediate Validation: Field Study
  - investigator observes how people act in real-world settings
- Downstream Validation: Adoption Rates
  - how many people are actually using the system

Abstraction Validation



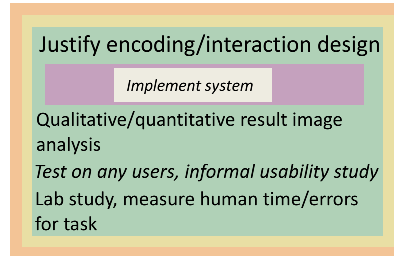
- Threat: Wrong task/data abstraction
- Immediate Validation: *None*
- Downstream Validation: Test the system
  – by target users doing their own work

Idiom Validation



- Threat: Ineffective encoding/interaction idiom
- Immediate Validation: justification
  – carefully justify the design of the idiom with respect to known perceptual and cognitive principles
- Downstream Validation: qualitative image analysis, quality metrics, lab study
  – qualitative discussion of results, usage scenarios – show examples
  – quality metrics – quantitative measurement of result images (e.g., number of edge crossings)
  – lab study – controlled experiment in lab setting

Idiom Validation

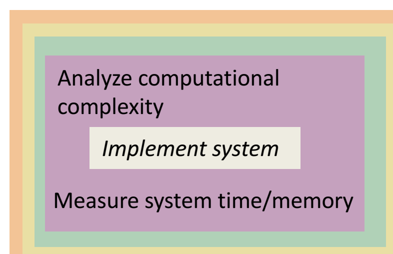


- Threat: Ineffective encoding/interaction idiom
- Immediate Validation: justification
  - ***carefully justify the design of the idiom with respect to known perceptual and cognitive principles***

Your project papers should have these

- Downstream Validation: qualitative image analysis, quality metrics, lab study
  - ***qualitative discussion of results, usage scenarios – show examples***
  - quality metrics – quantitative measurement of result images (e.g., number of edge crossings)
  - lab study – controlled experiment in lab setting

Algorithm Validation



- Threat: Slow algorithm
- Immediate Validation: analyze complexity
  - use standard algorithm analysis methods
  - based on number of items in the dataset or pixels in the display
- Downstream Validation: measure performance
  - measure wall-clock time and memory performance for the implemented algorithm
  - typical consideration is how dataset size affects algorithm speed

Example

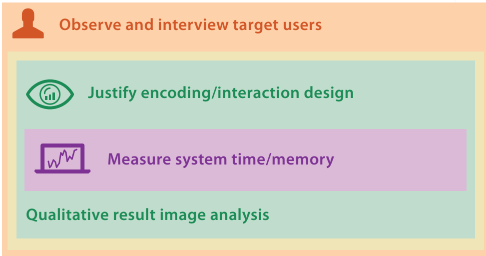


Fig 4.9, VAD, Munzner

IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, VOL. 12, NO. 5, SEPTEMBER/OCTOBER 2006

MatrixExplorer: a Dual-Representation System to Explore Social Networks

Nathalie Henry and Jean-Daniel Fekete

Abstract— MatrixExplorer is a network visualization system that uses two representations: node-link diagrams and matrices. Its design comes from a list of requirements formalized after several interviews and a participatory design session conducted with social science researchers. Although matrices are commonly used in social networks analysis, very few systems support the matrix-based representations to visualize and analyze networks.

MatrixExplorer provides several novel features to support the exploration of social networks with a matrix-based representation, in addition to the standard interactive filtering and clustering functions. It provides tools to reorder (layout) matrices, to annotate and compare findings across different layouts and find consensus among several clusterings. MatrixExplorer also supports Node-link diagram views which are familiar to most users and remain a convenient way to publish or communicate exploration results. Matrix and node-link representations are kept synchronized at all stages of the exploration process.

Index Terms— social networks visualization, node-link diagrams, matrix-based representations, exploratory process, matrix ordering, interactive clustering, consensus.

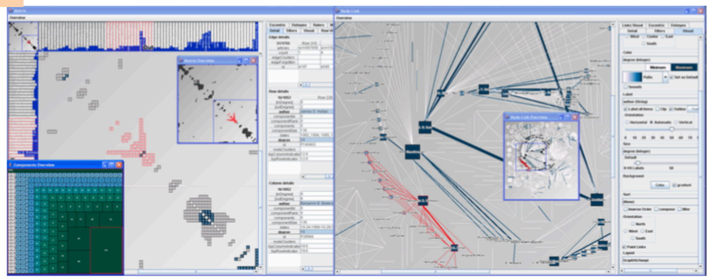


Fig. 1. MatrixExplorer showing two synchronized representations of the same network: matrix on the left and node-link on the right.

Domain

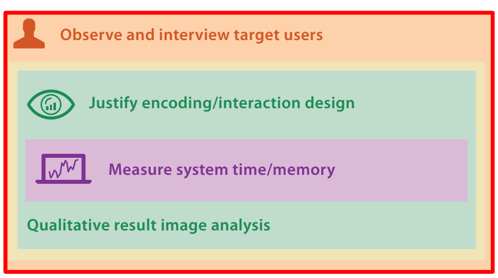


Fig 4.9, VAD, Munzner

3 Exploratory Analysis Requirements

We used participatory design techniques describes by Mackay [29] to understand the needs of social science researchers. After several interviews, we organized a participatory design session with professional social science researchers, selected for their frequent use of social network analysis tools. The participants included: a sociologist, a psychologist, a social network analysis specialist, two historians and five computer science researchers in the fields of HCI and Information Visualization. We focused on three specific questions:

1. How would you like to create a social network?
2. How would you like to edit a created social network?
3. How would you like to explore an unknown social network?

The session was organized in four stages. First, we presented participants the state-of-the-art tools in the domain of social network analysis and a broad range of novel HCI and InfoVis techniques for interacting with graphs and data. We explicitly avoided guiding them towards specific design techniques or tools. In the second stage, they split into small groups and generated ideas in a brainstorming session, which were then ranked. In the third stage, participants captured their ideas by creating paper prototypes (Figure 2) and then filming what it would be like to interact with them[2]. In the last stage, we reviewed the ideas altogether and gathered the common and important ones. Summarizing the working sessions, we ended up with a list of requirements for social networks exploratory analysis.

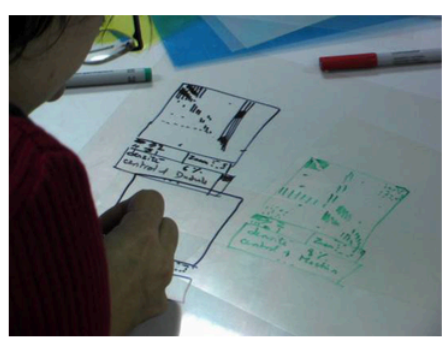


Fig. 2. Video Brainstorming showing a historian describing her ideas about using matrix-based representations to compare two networks.

Encoding/Interaction - Immediate

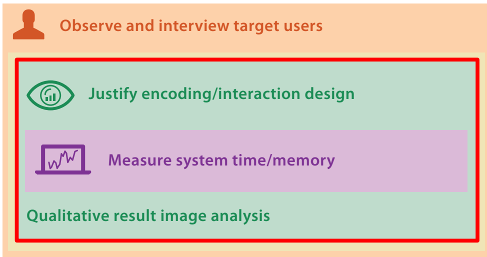


Fig 4.9, VAD, Munzner

justify interaction

4.1 Coupling node-link diagrams and matrices

MatrixExplorer is based on two representations: matrix-based and node-link diagrams meeting the first requirement of our users (*R1*). Node-link and matrix visualizations are synchronized in order to let the user work with both representations, switching smoothly from one to the other.

Multiple visualizations are synchronized by selection and filtering. Basically, if a user selects a set of actors in the matrix, this same set will be selected in all other visualizations (selection) and data filtered in one visualization will disappear from all others (filtering). Selection improves the transition from one representation to the other and constitutes the core of the coupling. Filtering preserves the coherence of the visualizations by presenting the same data, even if the attributes visualized are different.

In addition, visualizations can be synchronized by any visual attribute, simply by interactively setting the same attribute for the same visual variable. Thus, the user still has the possibility to not synchronize the visualizations in order to compare two attributes. With our system, users explore their networks using both representations, accomplishing tasks more easily with one representation or the other and visualizing the effect of a selection, or filtering, on all visualizations and their overviews.

Figure 3 shows a dual-representation of a co-authoring network and the correspondence of visual patterns in matrix and node-link representations. The process to obtain both representations follows: the user first automatically ordered the matrix, identified clusters (communities) and attributed colors to identify them. He then switched to a node-link diagram, displaying the community colors and laying the network out manually in order to better visualize how communities are linked and organized. Finally, moving back and forth between both representations, he identified the global structure of the network.

justify encoding

2.2 Matrix-Based Representations

Bertin in "Semiology of graphics" [5] introduced visual matrices to represent networks. Ghoniem et al. [18] showed that matrices outperform node-link diagrams for large graphs or dense graphs in several low-level reading tasks, except path finding. Bertin showed that matrices can be used to exhibit high-level structures by finding good permutations of their rows and columns. Thus, he qualified matrices as "reorderable". Reordering rows and columns of an adjacency matrix is similar to computing the layout for a node-link diagram: finding a layout that reveals some structure in the data. Related works can be divided into two categories: automatic and interactive systems.

Algorithm

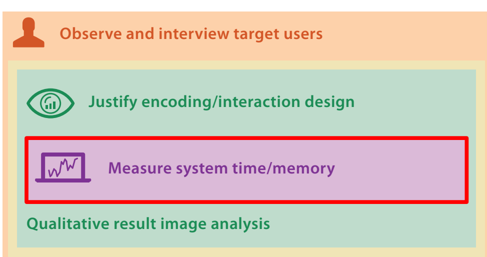


Fig 4.9, VAD, Munzner

To do this, we use the matrix of shortest paths (SP matrix) instead of the adjacency matrix. Our algorithm is:

Compute connected components
For each component
    Compute the SP matrix
Compute a matrix of distances between rows
        Apply the algorithm to find a linear order
        Compute a matrix of distances between columns
        Apply the algorithm to find a linear order
End for.

Connected components are independent blocks in the adjacency matrix so an order for their rows and columns is computed for each of them. Computing the SP matrix improves notably the order quality: it reduces the impact of noise (which is important in real datasets) and gives more information for low degree vertices (for which the rows and columns are very sparse). Computing the SP matrix is quadratic, as is the computation of the distance matrix for rows and the distance matrix for the columns. This has an important impact, since we want to use automatic ordering interactively. Therefore, we chose two fast ordering algorithms from the bioinformatics field. The first one is based on a hierarchical clustering, followed by a seriation (HCS) and is described in [14]; the second one is based on the traveling salesman problem (TSP) as presented in [9]. To solve TSP, we use a fast heuristic described in [21]. Matrices up to 1000 rows* 1000 columns can be ordered in seconds. Ordering larger matrices introduces a noticeable delay. So far, our user did not provide us with networks having a connected component larger than a thousand actors. However, we are investigating faster algorithms such as AMADO [7].

Encoding/Interaction - Downstream

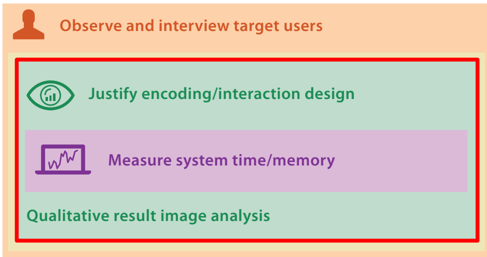


Fig 4.9, VAD, Munzner

Figure 6 presents a matrix reordered using TSP. The resulting matrix exhibits clearer blocks (diagonal with dense blocks); users can identify more clusters (*R10*) and articulation vertices between these clusters as dark color crosses here. A well-ordered matrix also helps identify outliers (*R11*) such as isolated relations, missing relation in a community, or actor with special connection patterns.

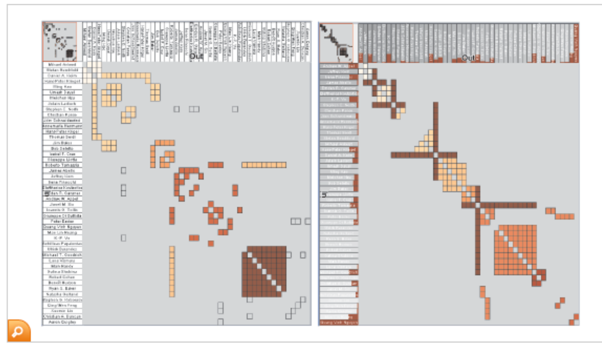


Fig. 6 Initial order (left) and TSP order (right). Colors represent clusters found by the user. Clusters are different in the two representations. Users found more clusters with TSP order. Headers red indicators (right) represents the distance between adjacent rows/columns.

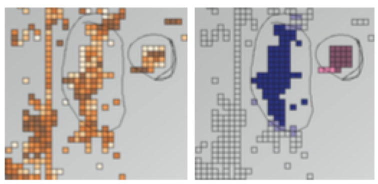


Fig. 10. Lasso selection on values visualization mode and resulting cluster visualization.

Evaluation Paper

Not all papers have (or even should have) validation for every level

Sizing the Horizon: The Effects of Chart Size and Layering on the Graphical Perception of Time Series Visualizations

Jeffrey Heer[1], Nicholas Kong[2], and Maneesh Agrawala[2]

[1] Computer Science Department
Stanford University
Stanford, CA 94305 USA
jheer@cs.stanford.edu

[2] Computer Science Division
University of California, Berkeley
Berkeley, CA 94720-1776 USA
{nkong, maneesh}@cs.berkeley.edu

ABSTRACT

We investigate techniques for visualizing time series data and evaluate their effect in value comparison tasks. We compare line charts with *horizon graphs* — a space-efficient time series visualization technique—across a range of chart sizes, measuring the speed and accuracy of subjects' estimates of value differences between charts. We identify transition points at which reducing the chart height results in significantly differing drops in estimation accuracy across the compared chart types, and we find optimal positions in the speed-accuracy tradeoff curve at which viewers performed quickly without attendant drops in accuracy. Based on these results, we propose approaches for increasing data density that optimize graphical perception.

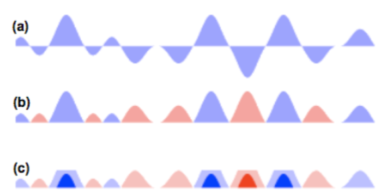
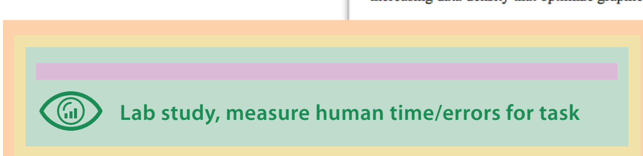


Figure 1. (a) Filled line chart. Area between data values on line and zero is filled in. **(b) "Mirrored" chart.** Negative values are flipped and colored red, cutting the chart height by half. **(c) 2-band horizon graph.** The chart is divided into bands and overlaid, again halving the height.



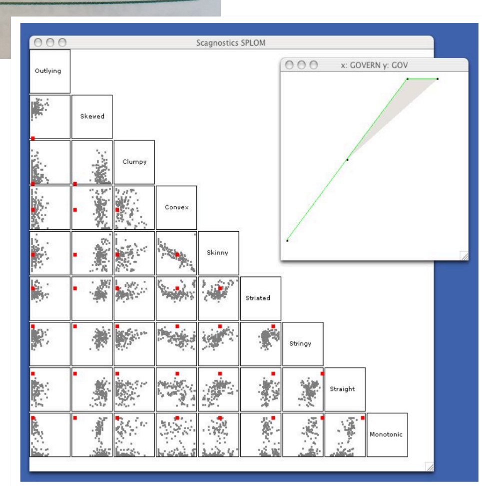
In both experiments, subjects completed discrimination and estimation tasks for points on time series graphs. Since the use case of horizon graphs is to compare data across several time series plots, we asked subjects to simultaneously view two separate graphs and compare a point on one graph to a point on the other, as shown in Figure 3. Subjects first reported which point represented the greater value and then estimated the absolute difference between the two. For each trial, we measured the *estimation error* as the absolute difference between a subject's estimation and the actual value difference between comparison points.

What-Why-How Table

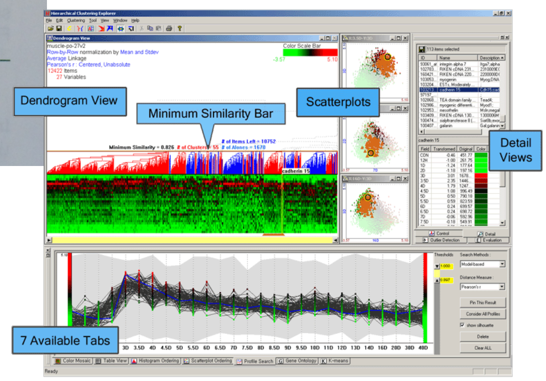
- There are some nice examples of how to frame this in Chapter 15.
- For each of the visualization tools, there's a table that describes things like "what: data", "what: derived", "why: tasks", "how: encode".
  - what – see Chapter 2
  - why – see Chapter 3
  - how: encode – see Chapters 7-10
  - how: manipulate – see Chapter 11
  - how: facet – see Chapter 12
  - how: reduce – see Chapters 13-14
- The "what" and "why" tasks should be driving how you developed the "how".
- Your system may not include all of the “how” items, but you need to address the ones that it does have.

System	**Scagnostics**
What: Data	Table.
What: Derived	Nine quantitative attributes per scatterplot (pairwise combination of original attributes).
Why: Tasks	Identify, compare, and summarize; distributions and correlation.
How: Encode	Scatterplot, scatterplot matrix.
How: Manipulate	Select.
How: Facet	Juxtaposed small-multiple views coordinated with linked highlighting, popup detail view.
Scale	Original attributes: dozens.

Wilkinson, Anand, Grossman. “Graph-Theoretic Scagnostics”, InfoVis 2005



System	Hierarchical Clustering Explorer (HCE)
What: Data	Multidimensional table: two categorical key attributes (genes, conditions); one quantitative value attribute (gene activity level in condition).
What: Derived	Hierarchical clustering of table rows and columns (for cluster heatmap); quantitative derived attributes for each attribute and pairwise attribute combination; quantitative derived attribute for each ranking criterion and original attribute combination.
Why: Tasks	Find correlation between attributes; find clusters, gaps, outliers, trends within items.
How: Encode	Cluster heatmap, scatterplots, histograms, boxplots. Rank-by-feature overviews: continuous diverging colormaps on area marks in reorderable 2D matrix or 1D list alignment.
How: Reduce	Dynamic filtering; dynamic aggregation.
How: Manipulate	Navigate with pan/scroll.
How: Facet	Multiform with linked highlighting and shared spatial position; overview–detail with selection in overview populating detail view.
Scale	Genes (key attribute): 20,000. Conditions (key attribute): 80. Gene activity in condition (quantitative value attribute): $20{,}000 \times 80 = 1{,}600{,}000$.



Seo, Shneiderman. “Interactively Exploring Hierarchical Clustering Results”, *IEEE Computer,* 2002