POLITECNICO DI MILANO

Unit 4

Data and Results Visualization

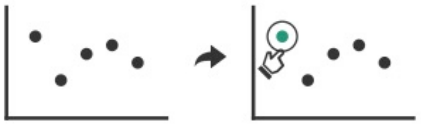
Daniele Loiacono

INTERACTIVE VISUALIZATION

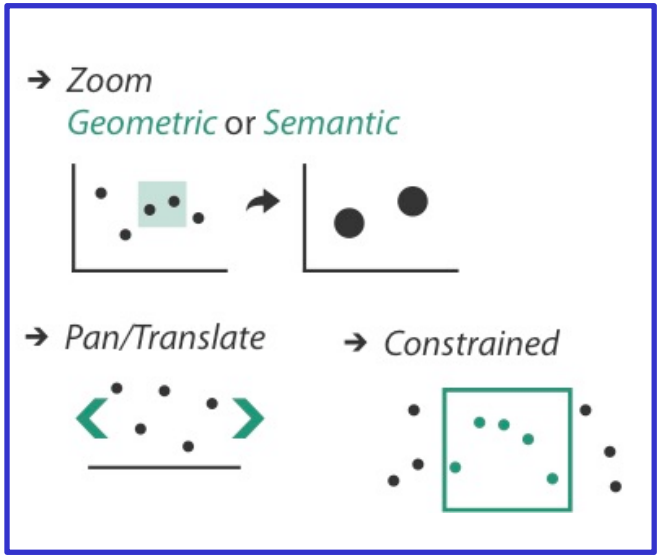
Overview

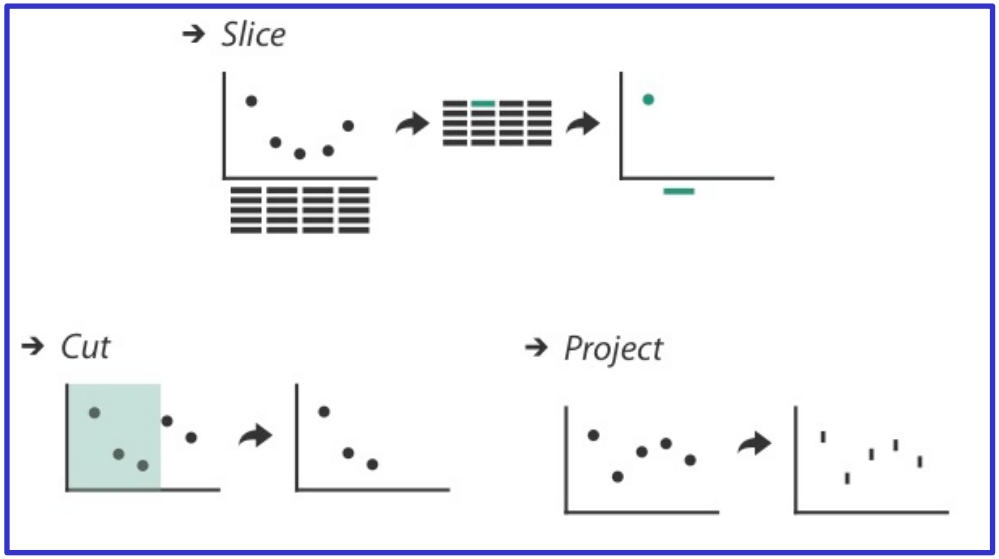
- Change

- Select



- Navigation
  - Through items
  - Through attributes



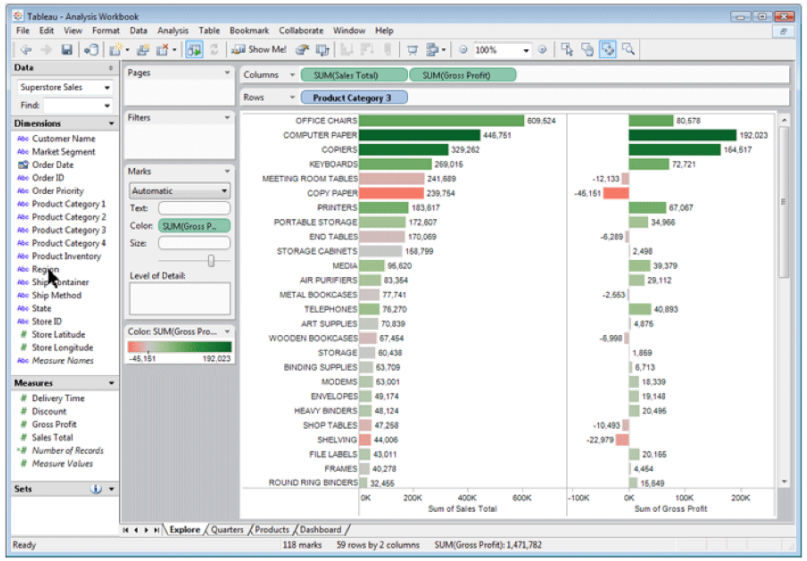


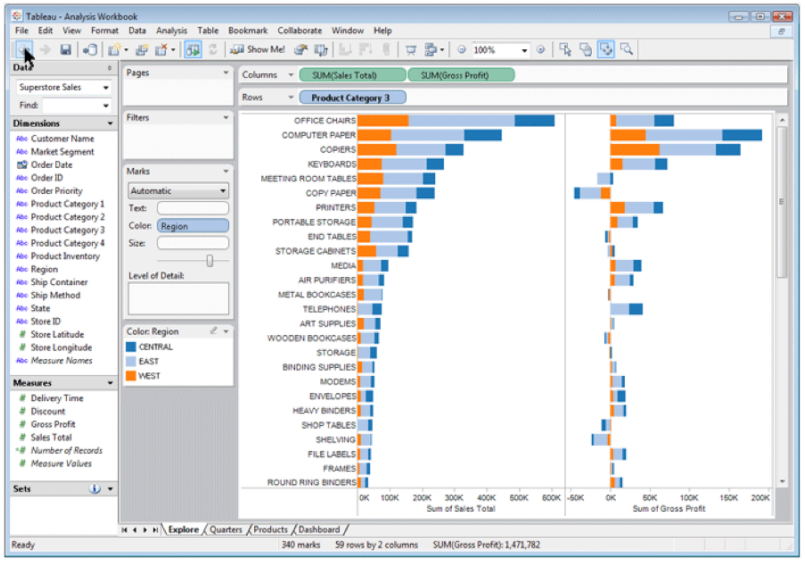
Change

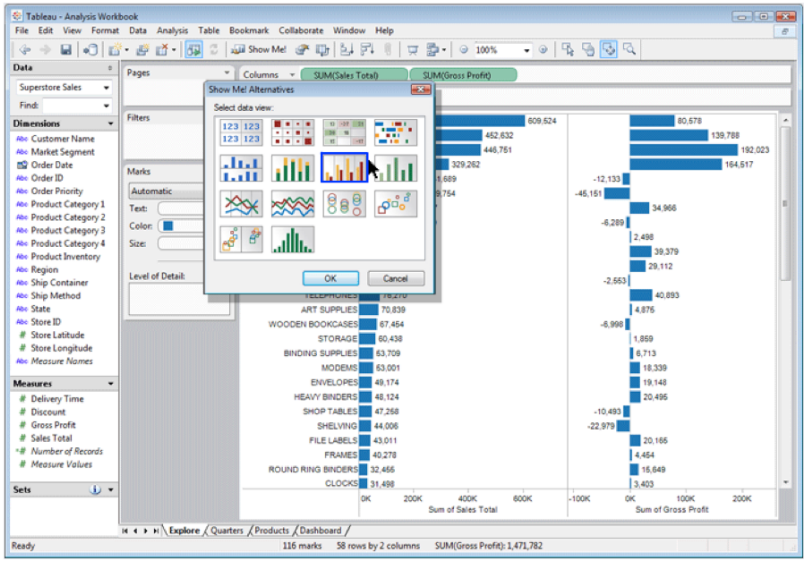
Change view

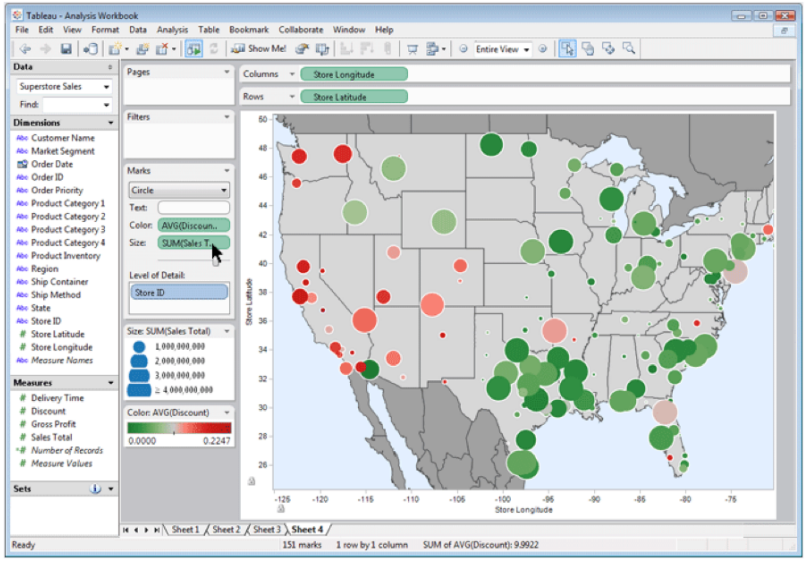
- Which changes?
  - encoding
  - parameters
  - arrangement
  - ordering
  - aggregation
- User driven or animated

Example: change encoding



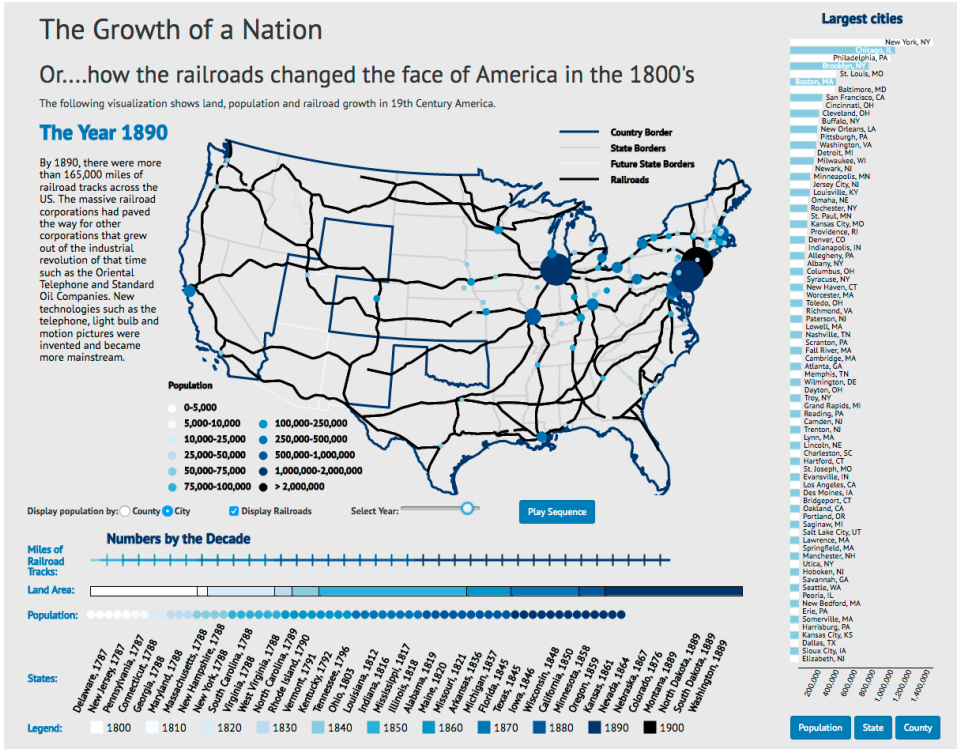






Change parameters

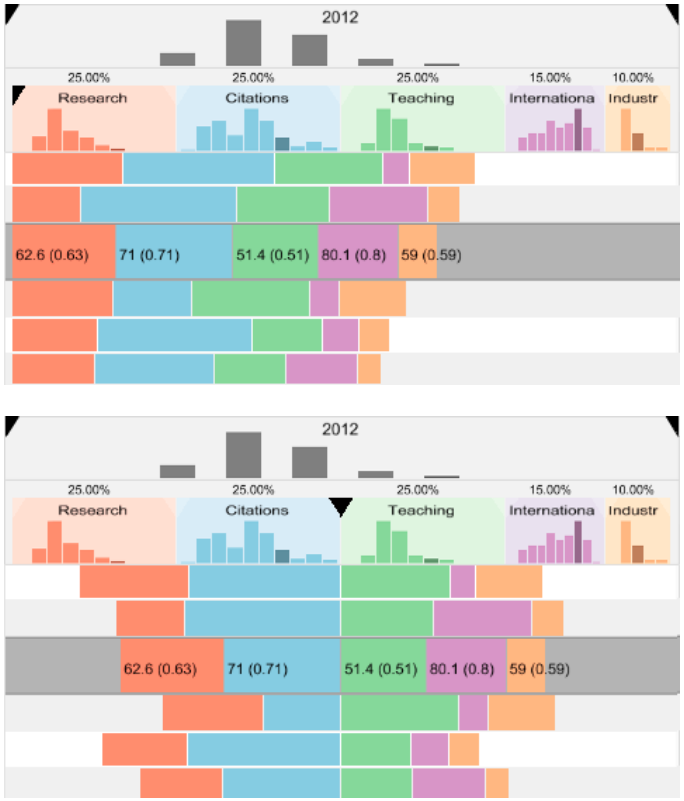
- Add sliders, buttons, checkboxes, dropdowns, etc
- Pros
  - user friendly
  - self-explicative (with labels)
- Cons
  - uses screen space



[Growth of a Nation]

Example: arrangement

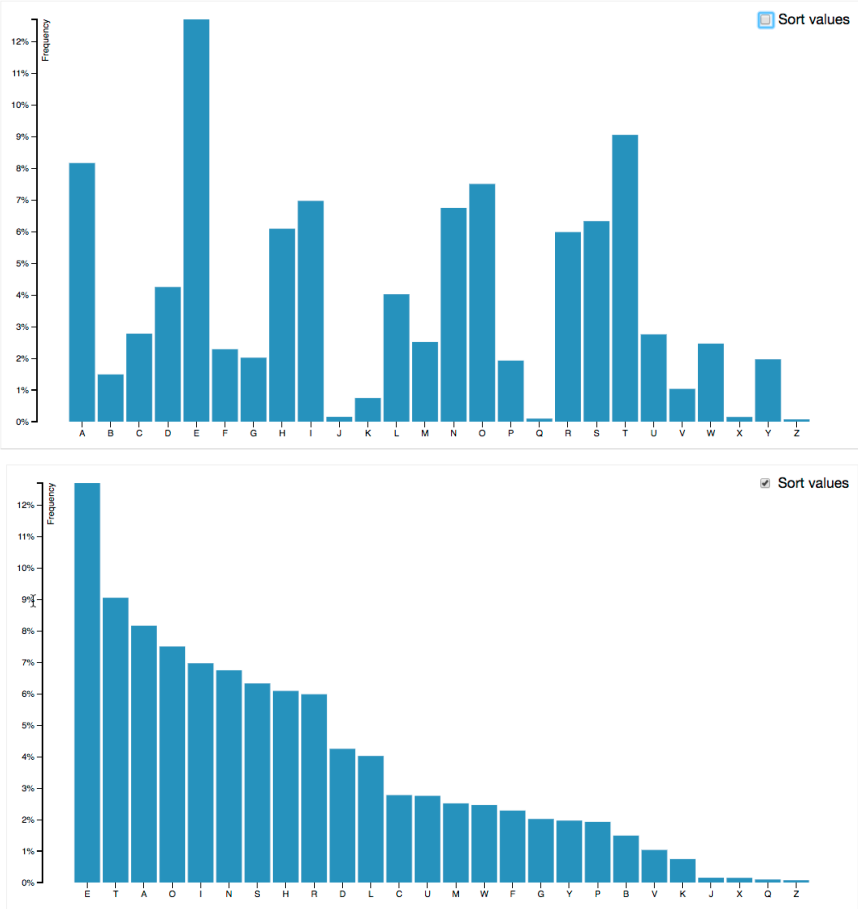
- Change alignment to change the focus of comparion:



[LineUp: Visual Analysis of Multi-Attribute Rankings.Gratzl, Lex, Gehlenborg, Pfister, and Streit, 2013]

Example: reordering

- Data drive reordering to find extremes or trends



[Sortable Bar Chart]

Animated Transitions in Statistical Data Graphics

Jeffrey Heer
George G. Robertson

Microsoft Research

Selection

Items selection

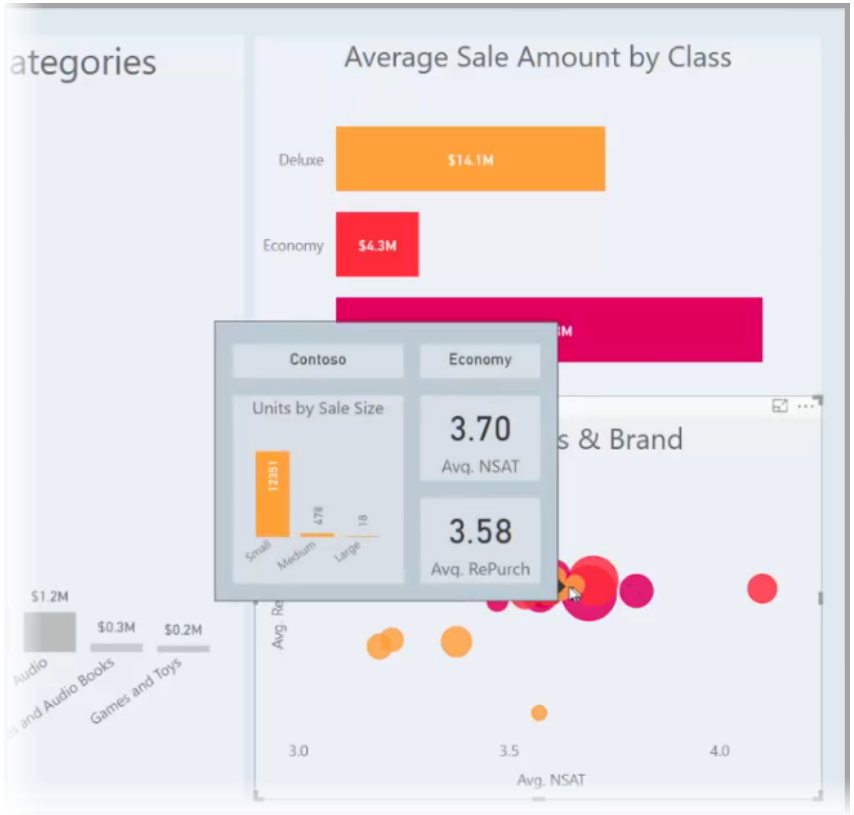
- How to select the items?
  - Click
  - Hover
- Click
  - Heavy
  - Allows multiple interactions (e.g., right-click)
- Hoover
  - Light
  - Not available on touch
  - Proximity
- Semantics
  - Incremental selection
  - Single/Multiple selection
  - Null selection is possible?

Highlighting

- Changing visual encoding of seleced items is an important feedback
- Design choices
  - change item color
    - clash wiht existing color coding?
  - add outline mark
  - change size
  - motion: usually avoid for single view
    - with multiple views, could justify to draw attention to other views

Tooltips

- Popup information for selection
  - hover or click
  - can provide useful additional detail on demand
- Does not support overview: if possible encode directly
- Assume users will not see it!



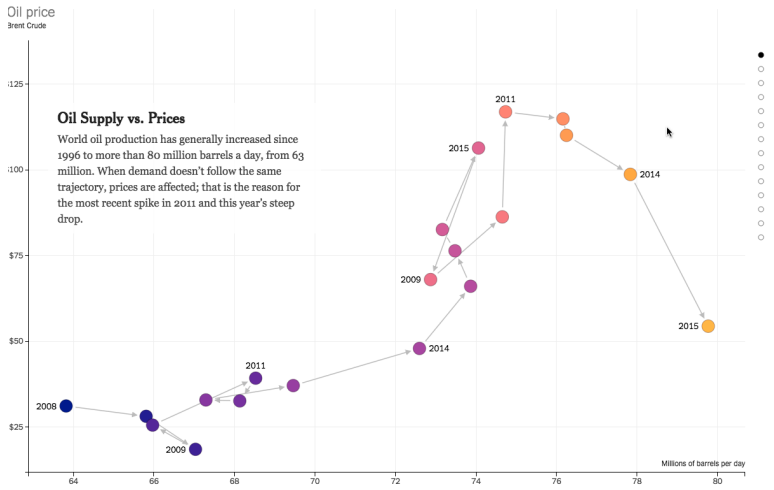
Navigation (through items)

Navigate: Changing viewpoint/visibility

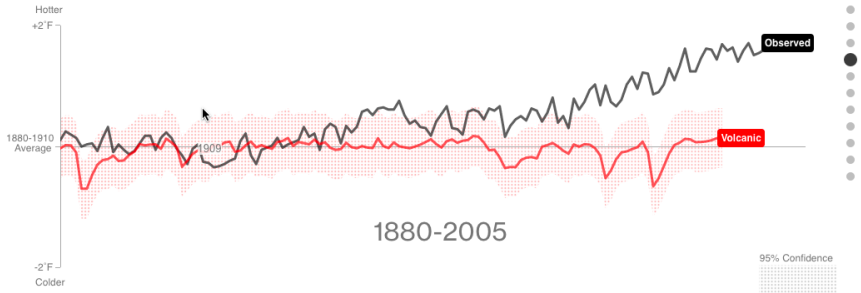
- Change viewpoint
  - changes which items are visible within view
- camera metaphor
  - pan/translate/scroll
    - move up/down/sideways
  - rotate/spin
    - typically in 3D
  - zoom in/out
    - enlarge/shrink world == move camera closer/further
    - geometric zoom: standard, like moving physical object

About panning down (scrolling)

- Navigate page by scrolling
- Pros:
  - familiar
  - linear
- Cons:
  - no direct access
  - unexpected behaviour
  - continuous control for discrete steps
- More advice here and here



link



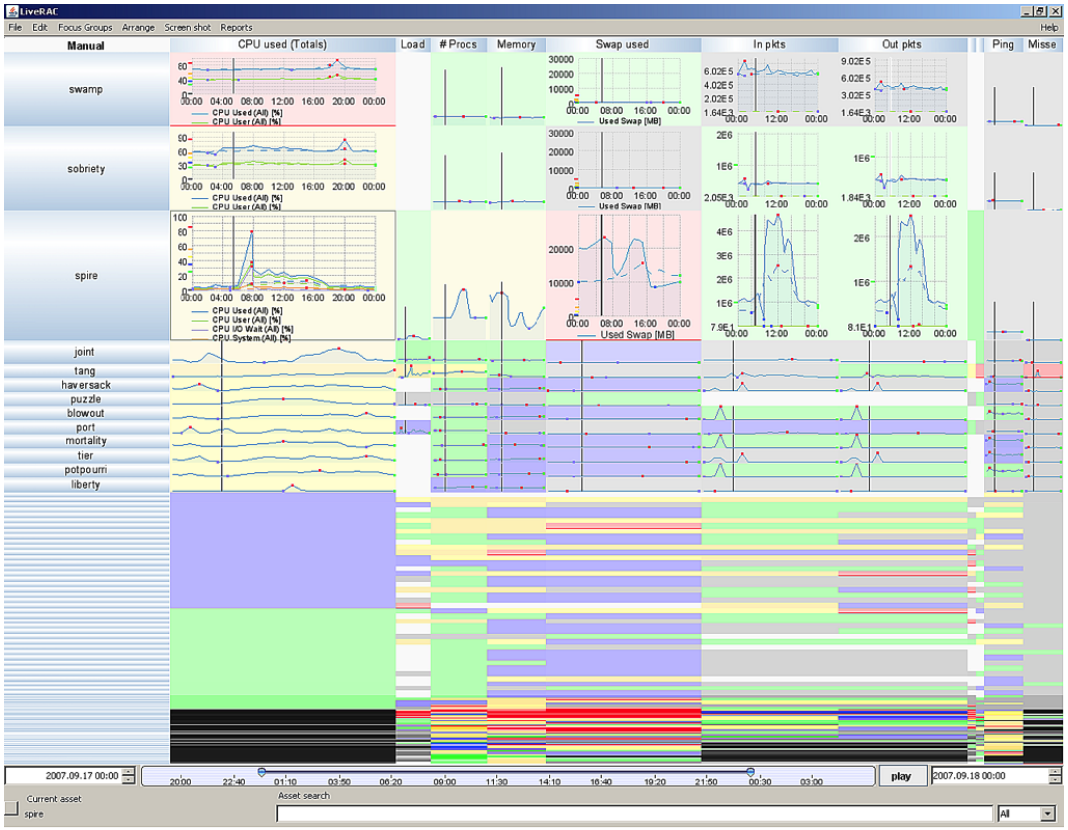
link

Navigate: Unconstrained vs constrained

- Unconstrained navigation
  - easy to implement for designer
  - hard to control for user
    - easy to overshoot/undershoot
- Constrained navigation
  - typically uses animated transitions
  - trajectory automatically computed based on selection
    - just click; selection ends up framed nicely in final viewport
- Examples
  - Zoom to Bounding Box
  - Zoomable Icicle

Semantic zooming

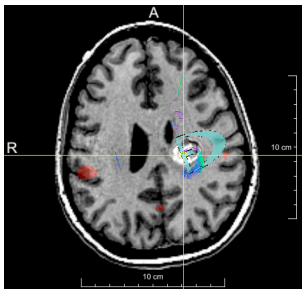
- Not just geometric zoom:
  - Resolution-aware layout adapts to available space
  - legible at multiple scales
  - Might involve encoding changes

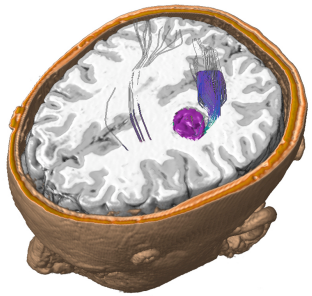


Navigation (through attributes)

Navigate: Reducing attributes

- Slice
  - show only items matching specific value for given attribute: slicing plane
  - axis aligned, or arbitrary alignment
- Cut
  - show only items on far slide of plane from camera
- Project
  - change mathematics of image creation
    - orthographic
    - perspective





Summary

Interaction benefits

- Major advantage of computer-based vs paper-based visualization
- Flexible, powerful, intuitive
  - Exploratory data analysis: change as you go during analysis process.
  - Fluid task switching: different visual encodings support different tasks.
- Animated transitions provide excellent support
  - Empirical evidence that animated transitions help people stay oriented.

Interaction limitations

- Interaction has a time cost
  - sometimes minor, sometimes significant
  - degenerates to human-powered search in worst case
- Remembering previous state imposes cognitive load (eyes beat memory!)
- Controls may take a lot of screen space
- Users may not interact as planned by designer
  - E.g., NYTimes logs show ~90% don’t interact beyond scrollytelling (Aisch, 2016)

DASHBOARDS

What and Why?

- Dashboard is a **single screen** information display that is used to **monitor** something.
  - It can be used wither real-time or once in awhile
  - It must help users to focus on most important data
- Why using dashboards?
  - They exploits visual channel
  - Single screen avoid memory overloading (eyes beat memory!)
- References

https://www.perceptualedge.com/articles/Whitepapers/Dashboard_Design.pdf

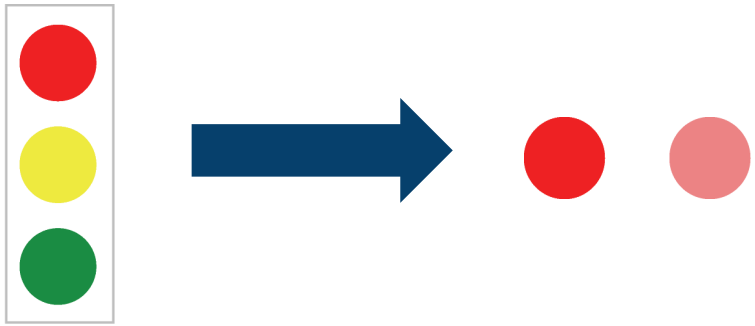
Pitfalls

Complexity

- Avoid too much complexity: dashboard should be as simple as possible.
- Real-time monitoring dashboard need to be even simpler than less frequently used dashboard.
  - Real-time monitoring requires quick responses
- Avoid showing more information than the amount users can perceive.
  - Keep in mind timing issues.

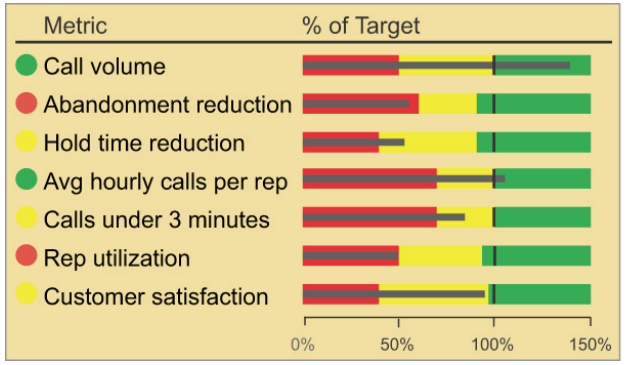
Avoid frequent and multi-level alerts

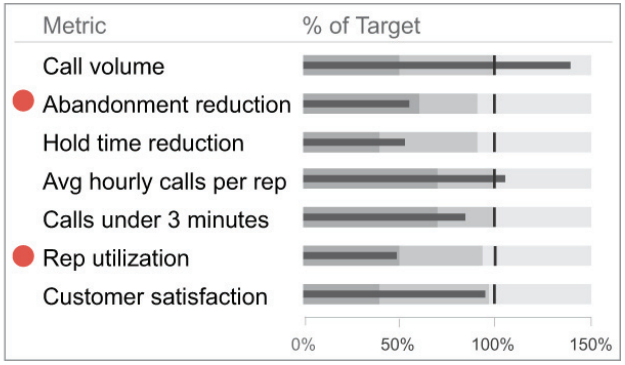
- Too frequent alerts will be soon ignored
  - Use alerts only for important situations
- Differentiated alerts are tricky to get right:
  - Distract or confuse the users
  - Not perceived properly (e.g., colorblindness)
- Try to use at max two-level alerts (preferably one)



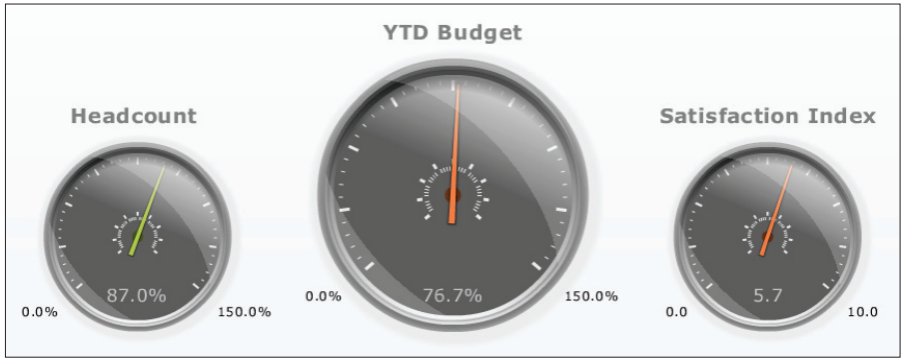
Overwhelming and distracting visuals

- Too many colors and too rich visuals might distract users



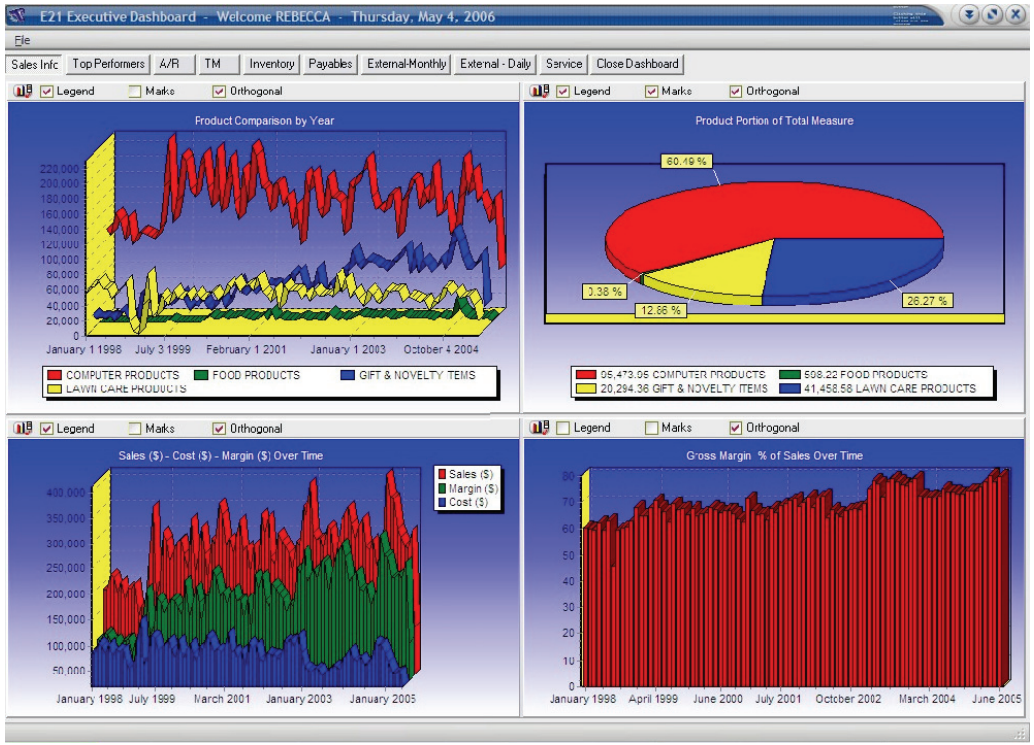


- Avoid distracting visuals



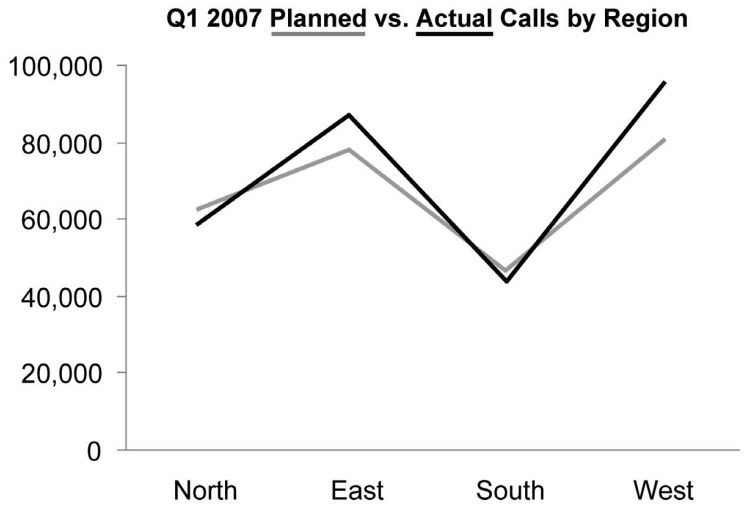
Inappropriate salience

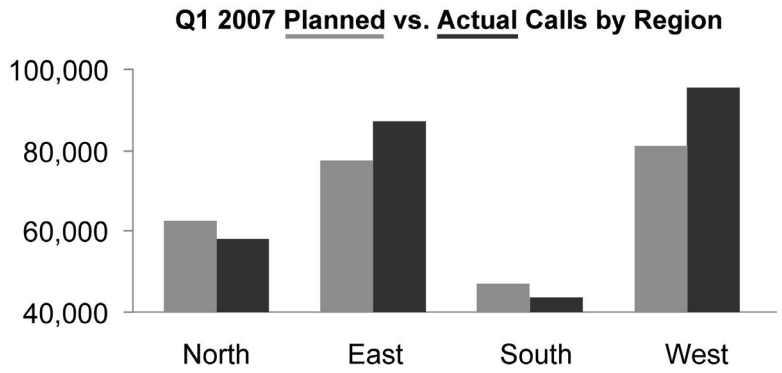
- All data in dashboard should be important, but it must be clear what data is most important.



Misleading visualization

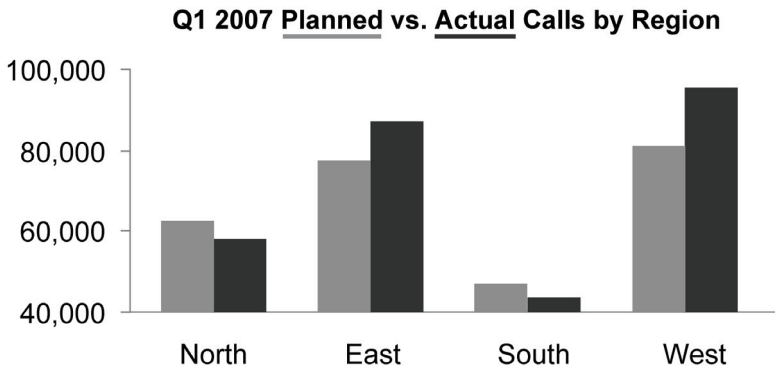
- Very harmful in a dashboard
- Examples

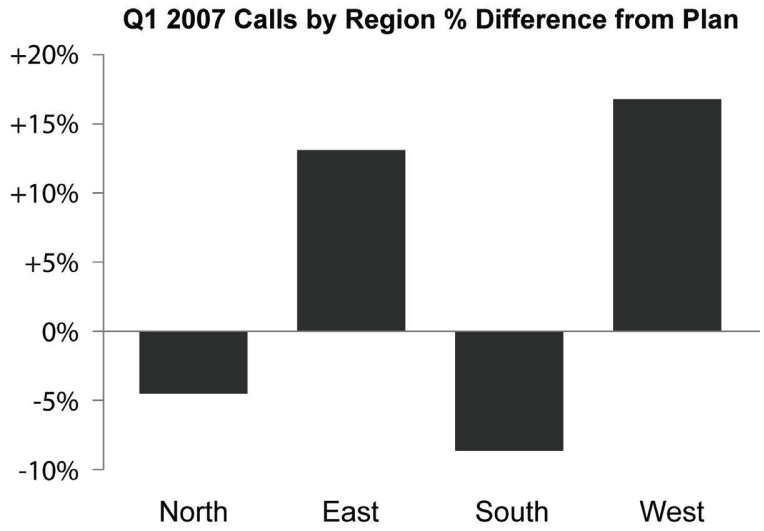




Indirect measures

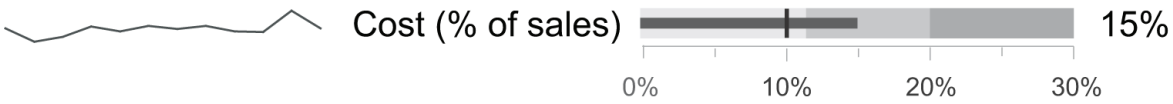
- When possible always visualize directly the information the users **need to act.**
- An example:





Lacking context

- Information without context is not very useful:



Design principles

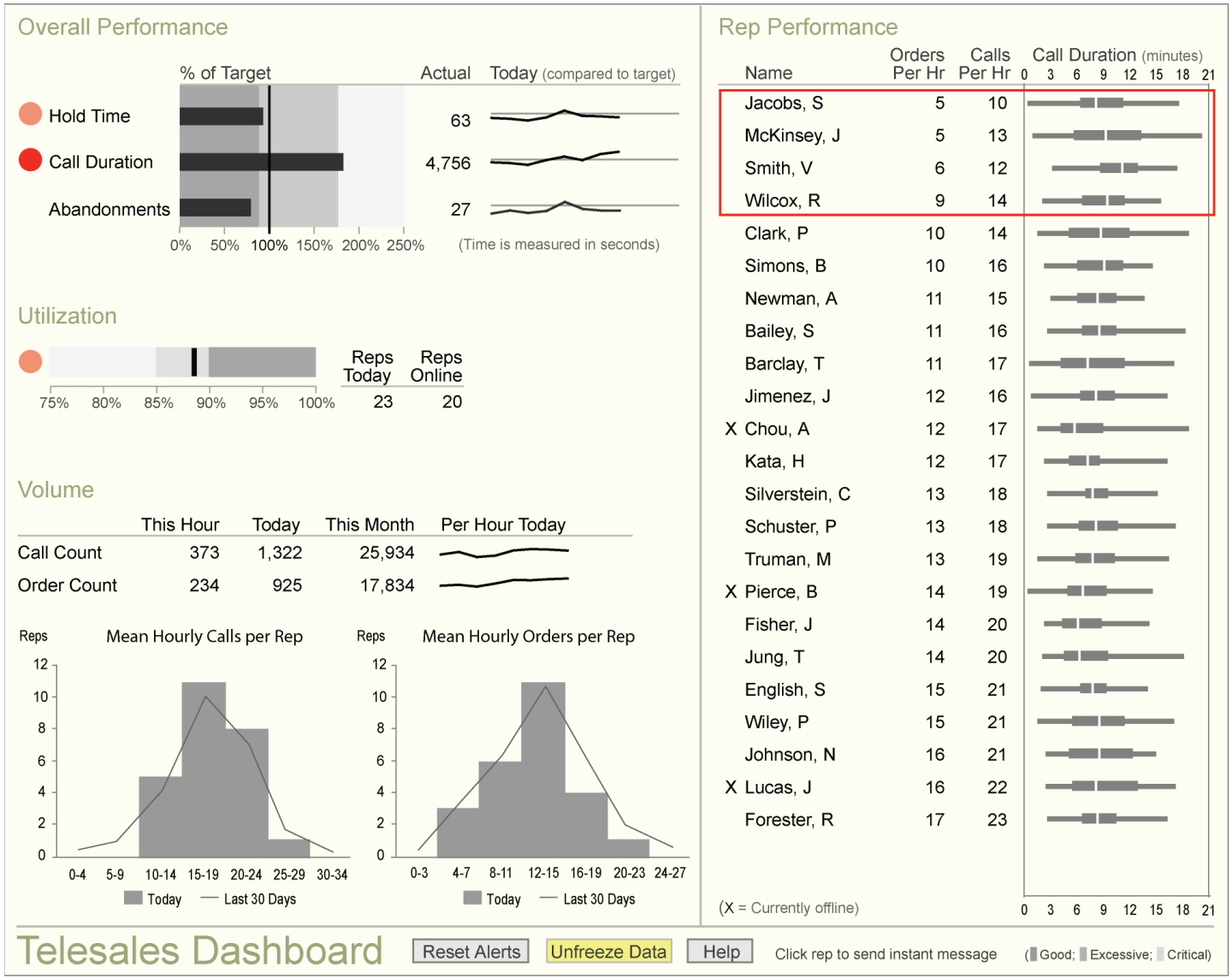
Flickering

- Flickering is a very powerful mean to attract the focus
- Use it only for urgent matter
  - If used too frequently it becomes annoying and less powerful

Design pro-active dashboard

- Encourage active-thinking about monitored data
  - Help users to be aware about what is happening and how things will evolve.
  - Help users to prevent an “alarm-situation”.
  - Make as easy as possible the actions required to deal with current issues (i.e., integration of dashboard in some control system).
- Design common dashboard (at least partially) for all the users (when having different roles) to give them a big picture.
- Match the mental model of the users of the dashboard about the underlying process monitored.

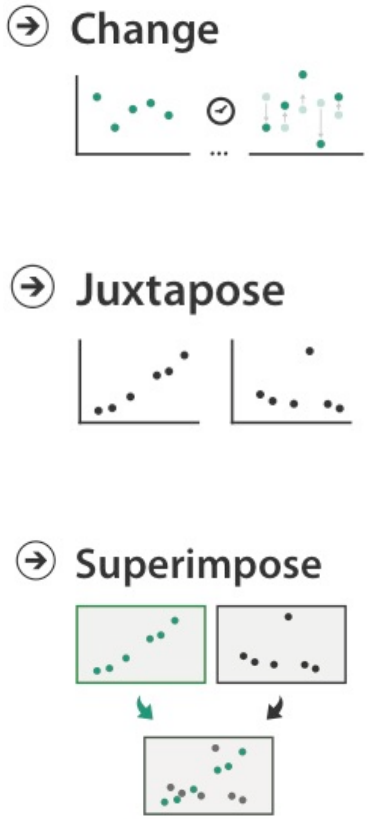
Example


Overall Performance
% of Target
Actual
Today (compared to target)
Hold Time
Call Duration
Abandonments
63
4,756
27
0% 50% 100% 150% 200% 250%
(Time is measured in seconds)
Utilization
Reps Today
Reps Online
75% 80% 85% 90% 95% 100%
23
20
Volume
This Hour
Today
This Month
Per Hour Today
Call Count 373 1,322 25,934
Order Count 234 925 17,834
Reps
Mean Hourly Calls per Rep
0-4 5-9 10-14 15-19 20-24 25-29 30-34
Today
Last 30 Days
Reps
Mean Hourly Orders per Rep
0-3 4-7 8-11 12-15 16-19 20-23 24-27
Today
Last 30 Days
Rep Performance
Name
Orders Per Hr
Calls Per Hr
Call Duration (minutes)
0 3 6 9 12 15 18 21
Jacobs, S 5 10
McKinsey, J 5 13
Smith, V 6 12
Wilcox, R 9 14
Clark, P 10 14
Simons, B 10 16
Newman, A 11 15
Bailey, S 11 16
Barclay, T 11 17
Jimenez, J 12 16
X Chou, A 12 17
Kata, H 12 17
Silverstein, C 13 18
Schuster, P 13 18
Truman, M 13 19
X Pierce, B 14 19
Fisher, J 14 20
Jung, T 14 20
English, S 15 21
Wiley, P 15 21
Johnson, N 16 21
X Lucas, J 16 22
Forester, R 17 23
(X = Currently offline)
0 3 6 9 12 15 18 21
Telesales Dashboard
Reset Alerts
Unfreeze Data
Help
Click rep to send instant message
(Good; Excessive; Critical)

DEALING WITH COMPLEXITY

How to deal with complexity?

- Working on data
  - Computing derived data
  - Aggregation
  - Filtering
- Facet across multiple views
  - Change views (interactive viz)
  - Juxtapose views
  - Superimpose view
- Not mutually exclusive



Filtering

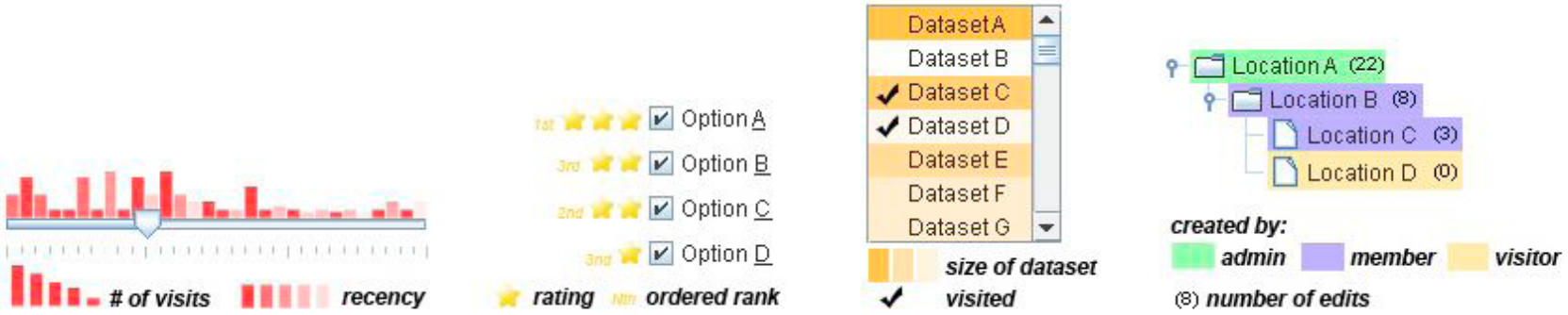
Overview of filtering

- What?
  - Items
  - Attributes
- Pros
  - Intuitive for users
  - Straightforward to compute
- Cons
  - Might easily lead to information loss

- Items filtering is mainly used in interactive visualization

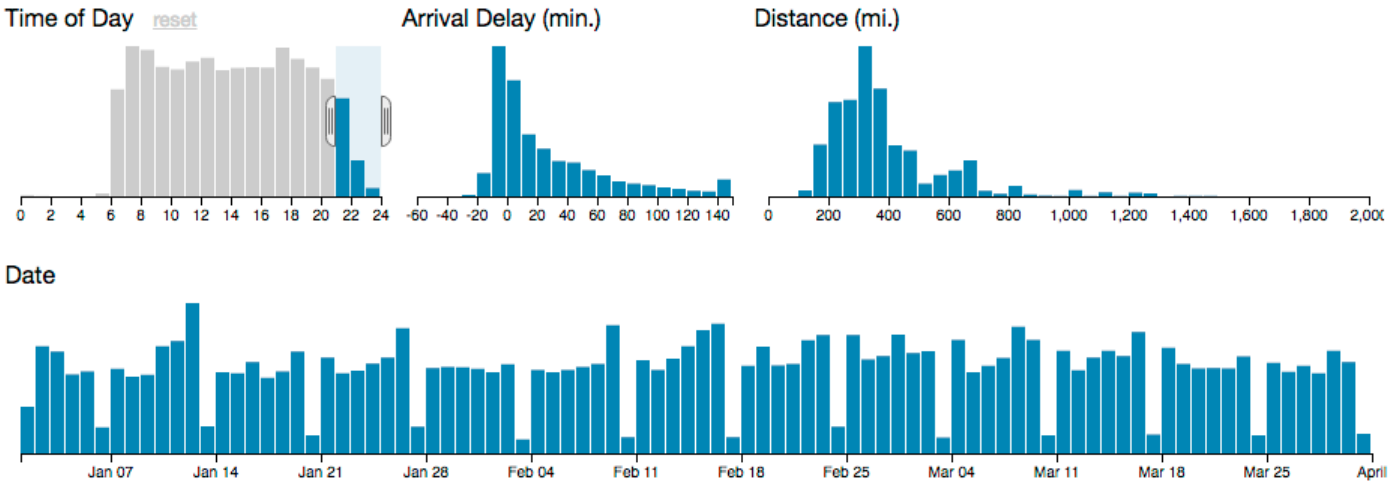
Item filtering: sciented widgets

- To suppor items filtering in interactive visualization, widget can be **augmented** to show **information scent**



Item filtering: cross filtering

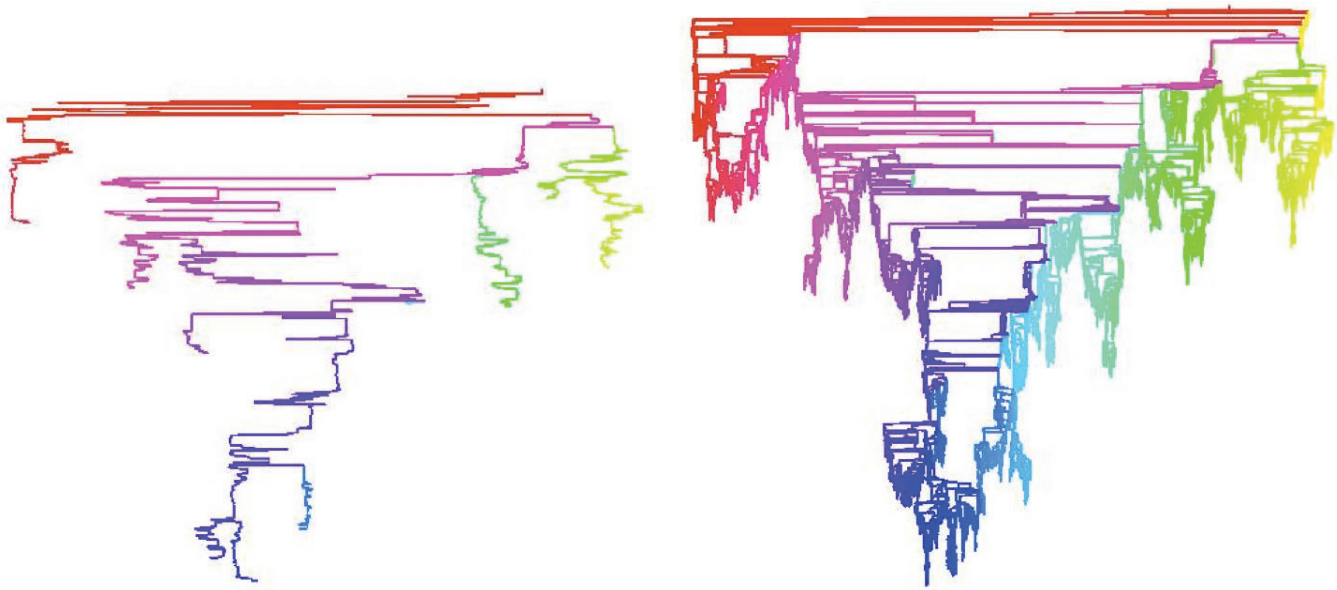
- Used in interactive visualization with linked multiple views
- Filtering happens is performed in one view and reflact on the others



[http://square.github.io/crossfilter/]

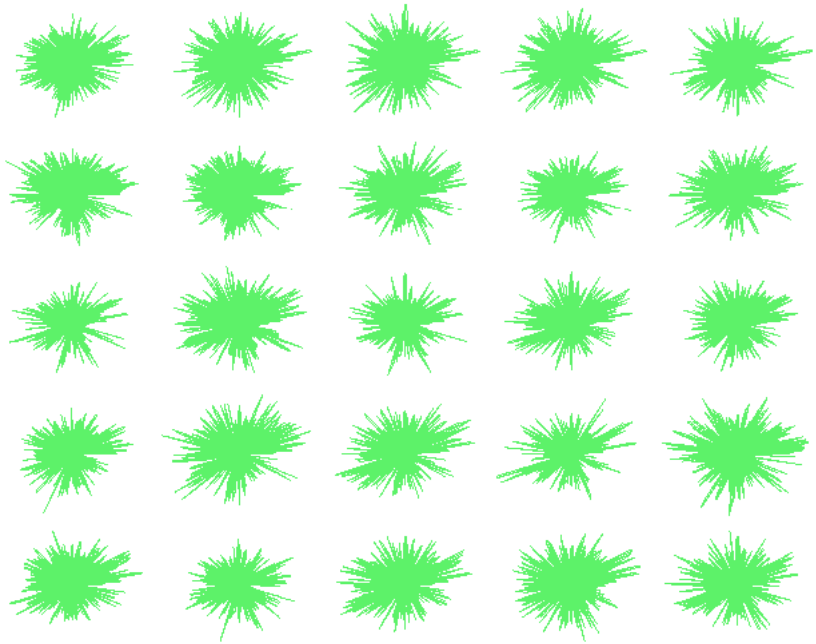
Items filtering: importance-based

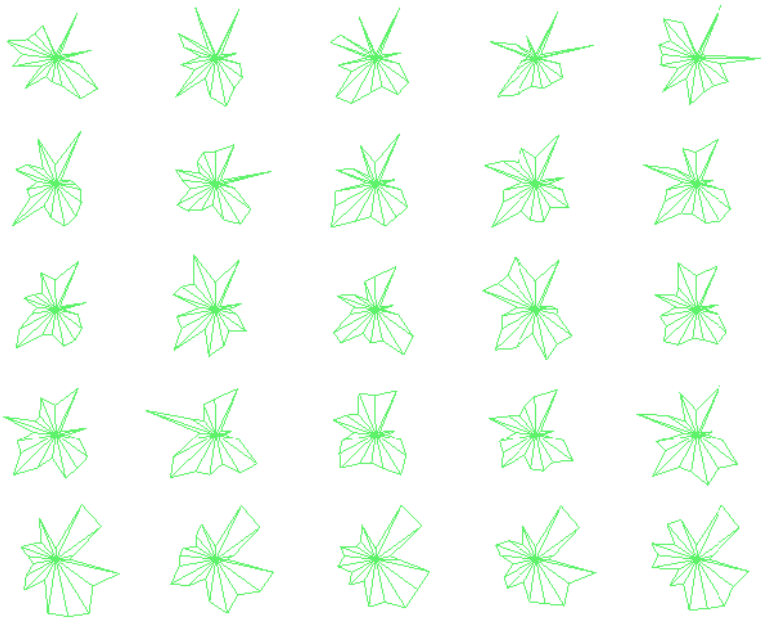
- Computing a derived quantitative attribute to measure the importance of each item.
  - Difficult to design good measures.
- Filter to remove items with low importance.
- Example (a tree filtered on the basis of centrality)



Attribute filtering: similarity-based

- Filter attributes based on both importance and similarities
- Mainly used on dense plot to make data readable
- Example:





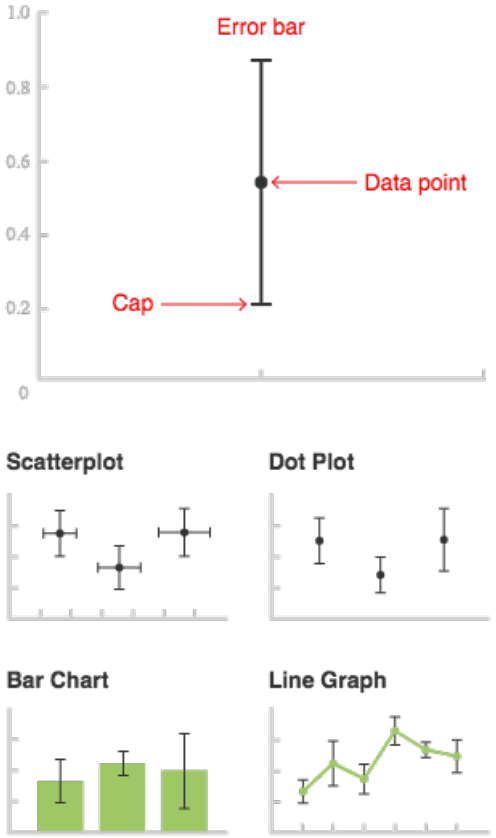
Item Aggregation

Item aggregation: derived data

- Mainly based on computing aggregate measures (e.g., mean, max, min, count, etc.) as derived data.
- Typically, several derived data is visually encoded together using sophisticated glyphs (e.g., boxplot).
- The goal is to avoid missing important details of data when using a more compact visualization.
- Notable examples:
  - Density plot
  - Continuous scatter plot
  - Error bars
  - Boxplot, Letter, and Violin plot
  - Histograms
  - Bullet chart

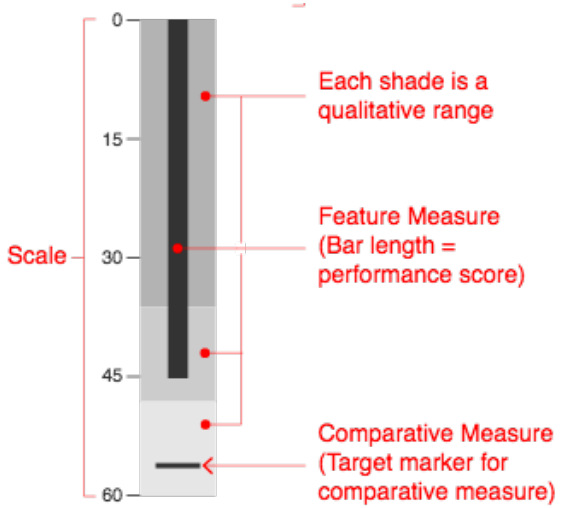
Error bar: anatomy

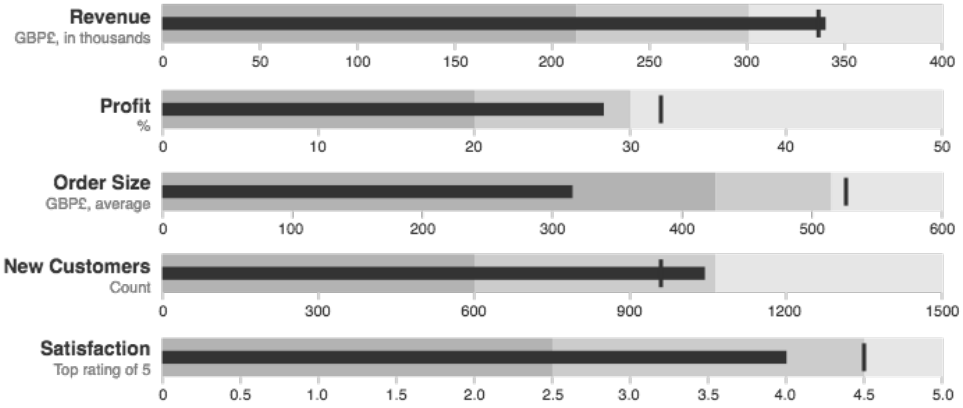
- What?
  - 1 quantitative attribute
- Why?
  - Identify range or distribution
- Remarks
  - Used as glyphs in other idioms



Bullet chart: anatomy

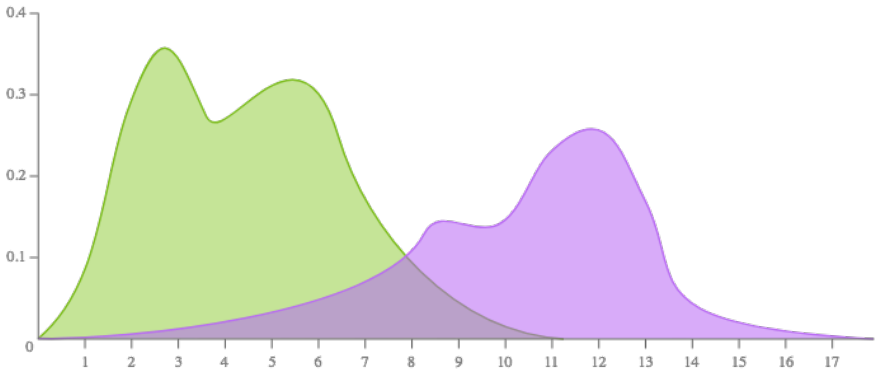
- What?
  - 1 categorical key
  - 1 quantitative attribute
  - 3 quantitative attribute (comparative references)
- Why?
  - Compare and lookup values
  - Identify outliers
- Remarks
  - Careful use of color (hue)





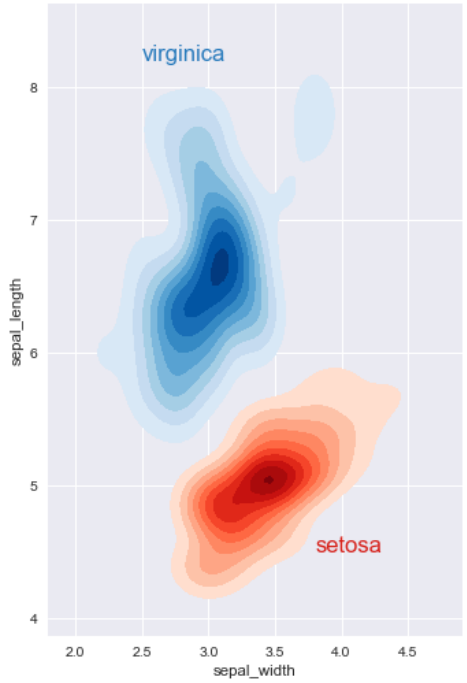
Density plot: anatomy

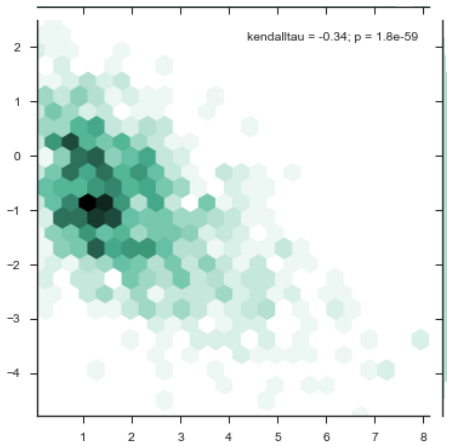
- What?
  - 1 quantitative attribute
  - 1 derived density
- Why?
  - Identify distribution and range
- Remarks
  - Color can encode a categorical key (or to compare more quantitative attributes)
  - Scale up to 2-3 keys/attributes but not limited by items



Continuous scatter plot: anatomy

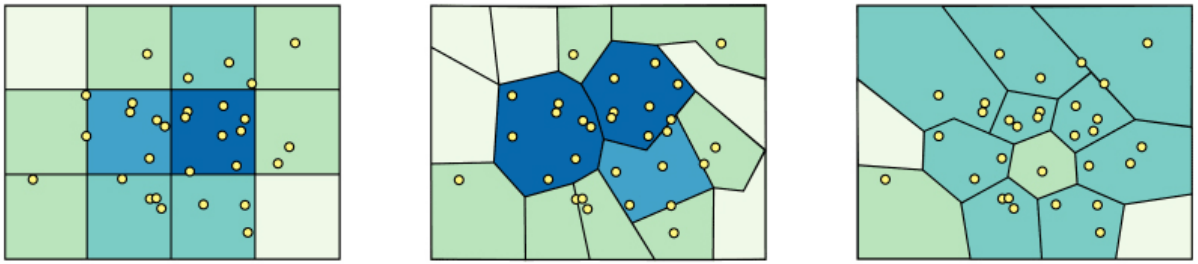
- What?
  - 2 quantitative attribute
  - Derived density function
- Why?
  - Correlation and distribution
  - Identify patterns/clusters
- Remarks
  - Used to overcome limitation of scatter plot
  - Color encode density (occasionally hue might encode additional categorical)
  - Bins or contours are possible alternatives



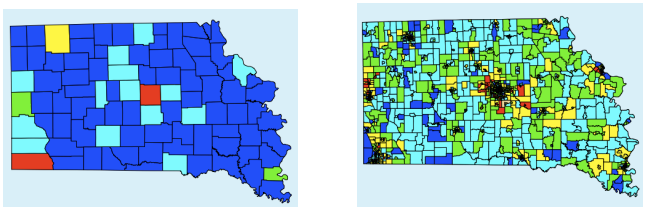


Item aggregation: spatial data

- Modifiable Areal Unit Problem
  - Region definition would affect significantly the outcome



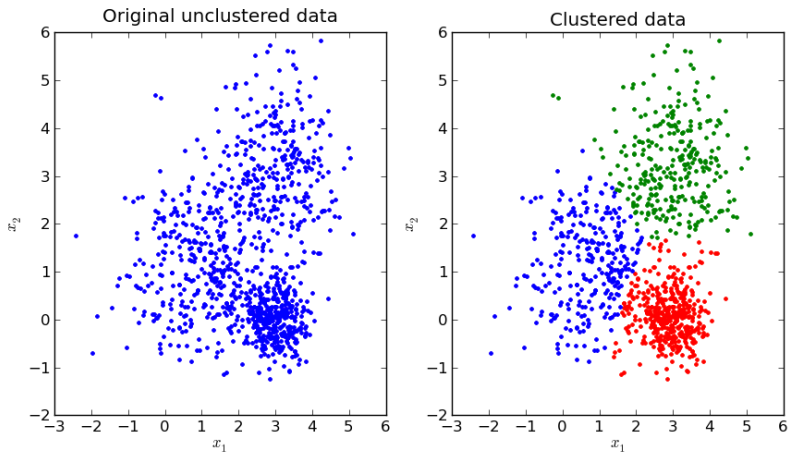
  - Scale effects

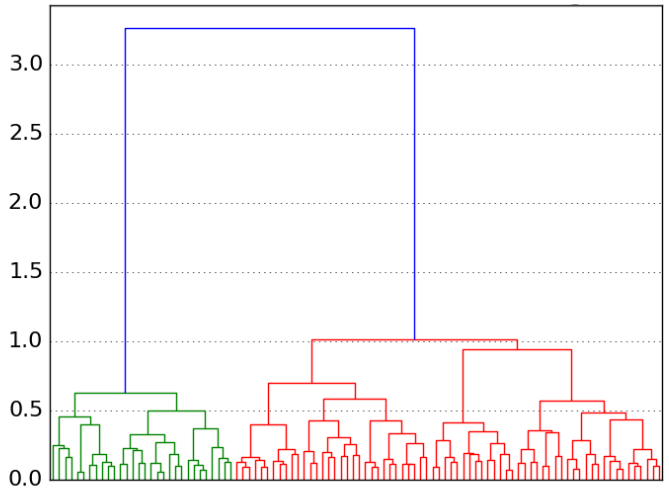


- Encoding of weighted averages (based on proximity) might help.

Item aggregation: clustering

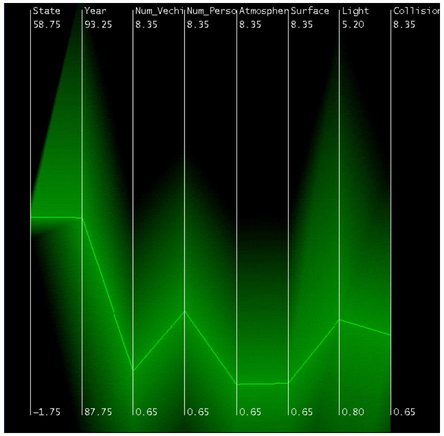
- Clustering consists of identifying groups of items that share similarities.
- Hierarchical clustering allows also to create hierarchies of cluster than can be also used for ordering data.

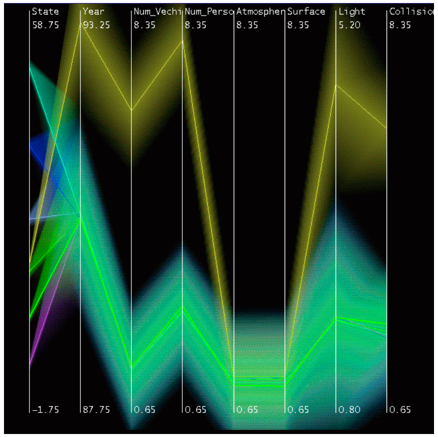


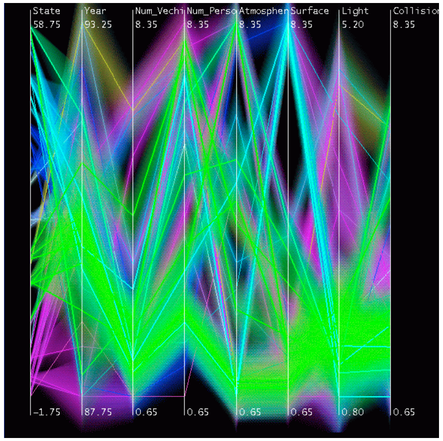


Hierarchical parallel coordinates: anatomy

- What?
  - N quantitative/ordered attributes
  - 5 derived quantitative
- Why?
  - Correlation
  - Identify outliers, range and patterns
- Remarks
  - Builds a hierarchy and compute, mean, max, min, count, a depth for each attribute in each cluster
  - Derived data is encoded with opacity and
  - Scale up to 100k items
  - Clusters can be filtered

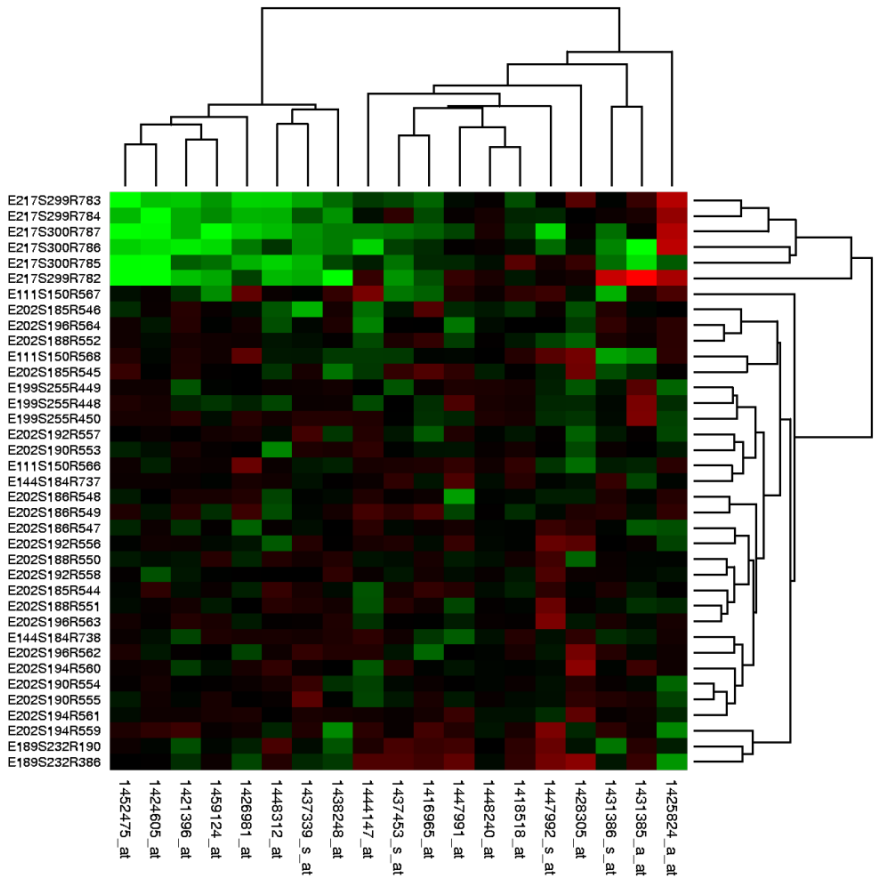






Clustered heatmap: anatomy

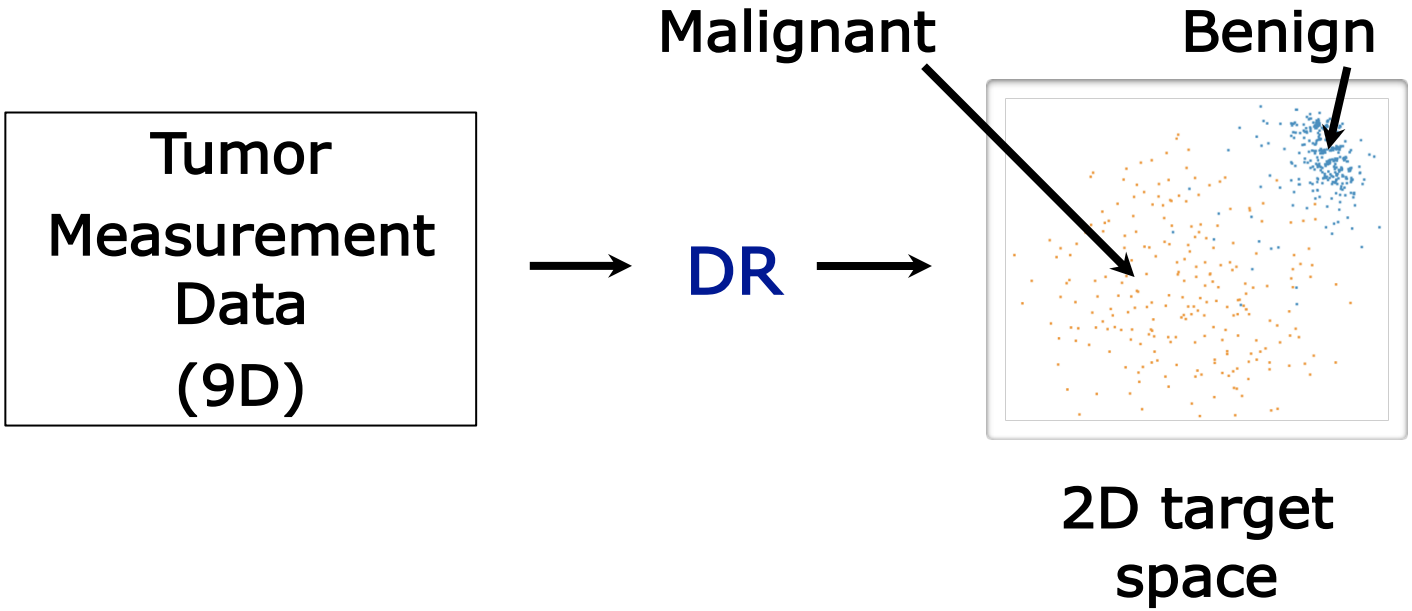
- What?
  - 2 categorical keys
  - 1 quantitative attribute
  - derived hierarchy
- Why?
  - Discover patterns, outliers
  - Correlation
- Remarks
  - Scale up to ~1M of items



Dimensionality Reduction
(Attribute Aggregation)

What is dimensionality reduction?

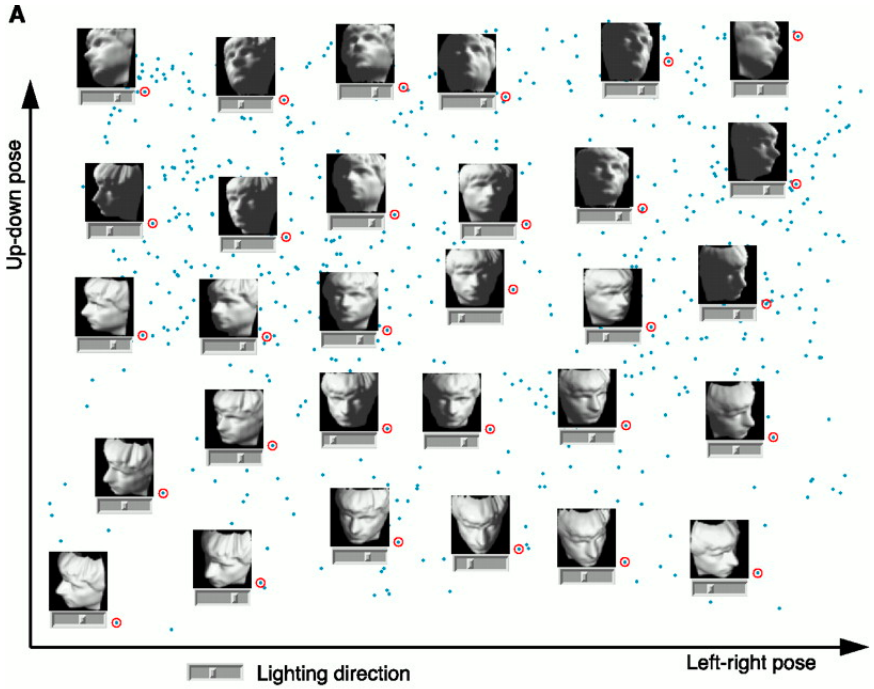
- It consists of mapping data from high-dimensional space to a low-dimensional target space.
- The goal is to capture most of variance with minimal error
- Used for practical reason when true dimensionality of dataset conjectured to be smaller than dimensionality of measurements

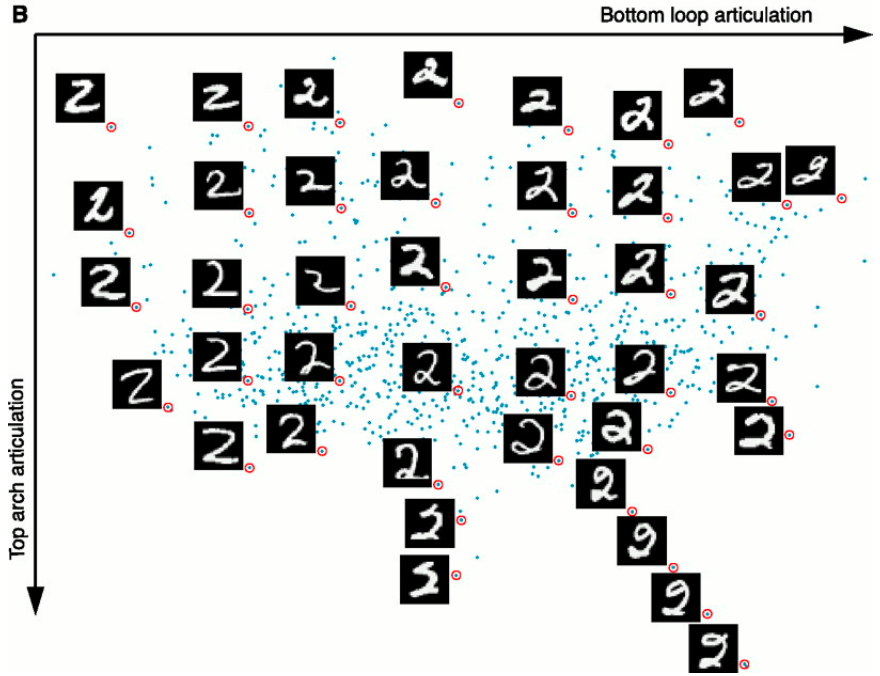


Dimensionality reduction & visualization

- Why applying DR?
  - Improve performance of downstream algorithms (avoid curse of dimensionality).
  - Allow visualization of data.
- Dimension-oriented tasks
  - naming synthesized dims, mapping synthesized dims to original dims
- Cluster-oriented tasks
  - verifying clusters, naming clusters, matching clusters and classes

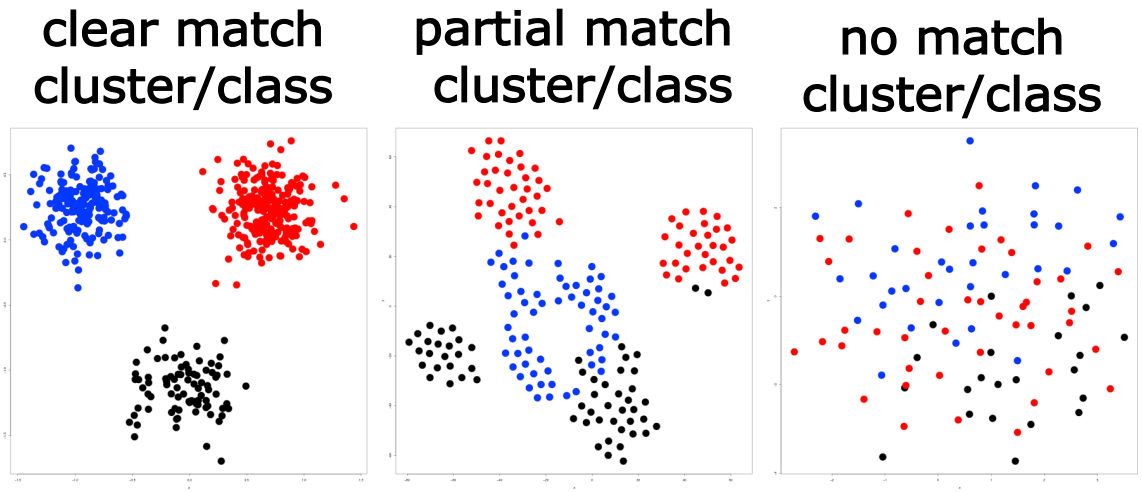
Dimension-oriented tasks: example





[A global geometric framework for nonlinear dimensionality reduction. Tenenbaum, de Silva, and Langford. Science, 290(5500):2319–2323, 2000.]

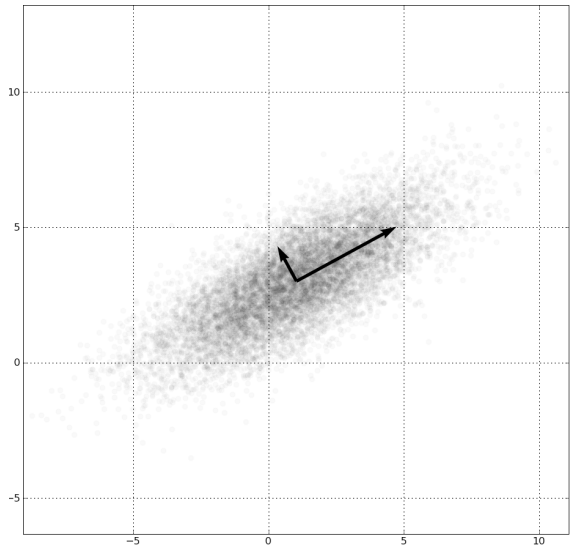
Cluster-oriented tasks



[Visualizing Dimensionally-Reduced Data: Interviews with Analysts and a Characterization of Task Sequences. Brehmer, Sedlmair, Ingram, and Munzner. Proc. BELIV 2014.]

How? Linear DR

- principal components analysis (PCA)
  - finding axes: first with most variance, second with next most, …
  - describe location of each point as linear combination of weights for each axis
  - mapping synthesized dims to original dims



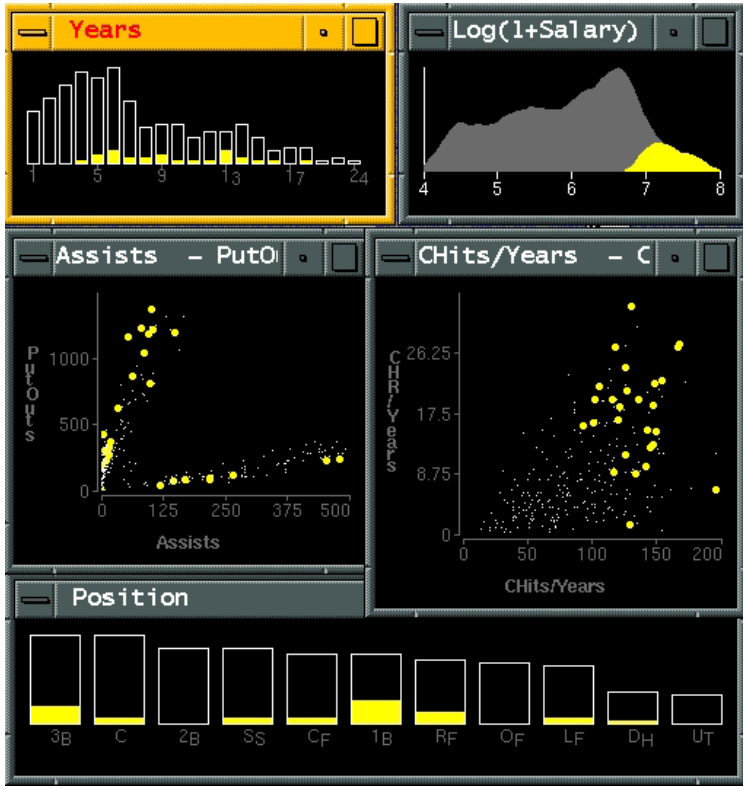
How? Nonlinear DR

- More powerful…
- … lose all ties to original dims/attribs:
  - new dimensions often cannot be easily related to originals
- Approaches (from both visualization, ML, optimization)
  - t-SNE
  - MDS (multidimensional scaling)
  - charting
  - isomap
  - LLE
  - …

Faceting: Juxtapose

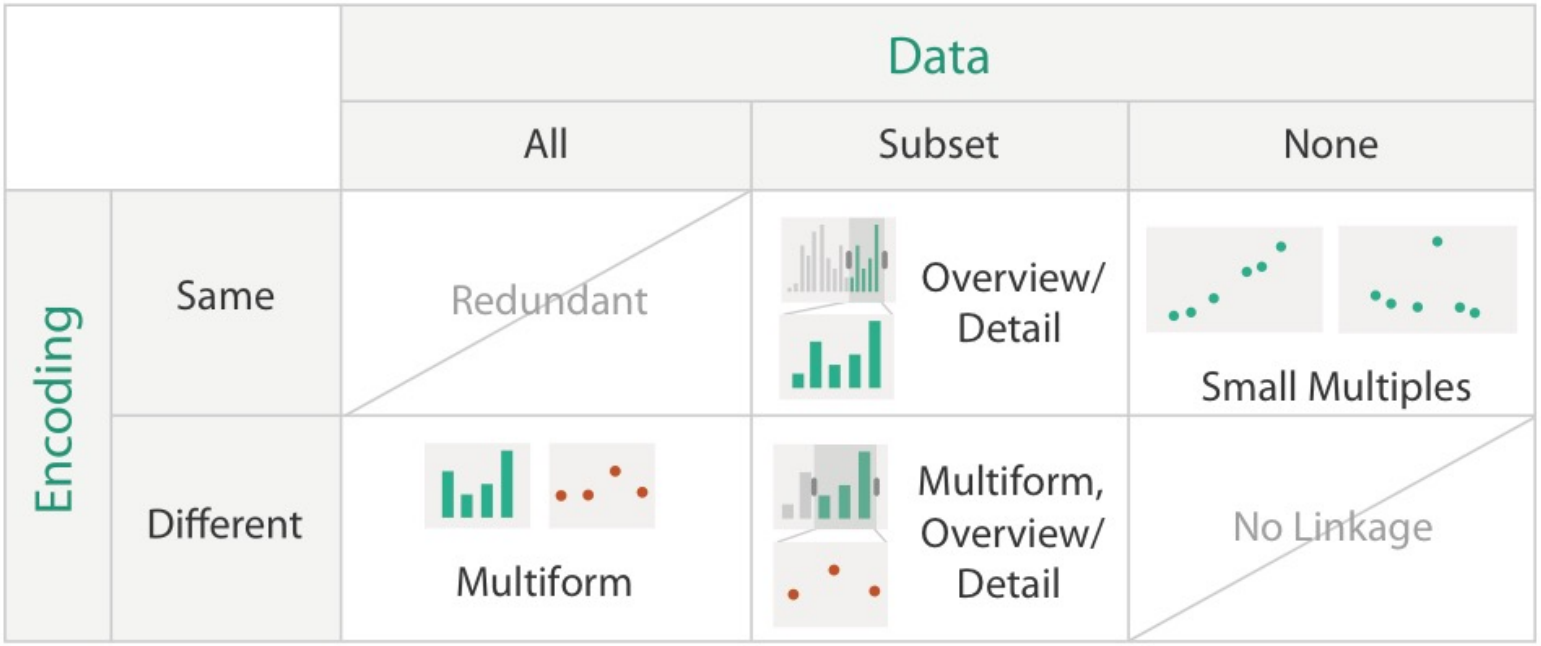
Juxtapose views

- Trade space for memory
  - lower cognitive load to move eyes between 2 views than remembering previous state with single changing view
  - display area, 2 views side by side each have only half the area of one view
- Coordinated views
  - Shared encoding
  - Shared (subset of) data
  - Shared interaction (e.g., highlighting)

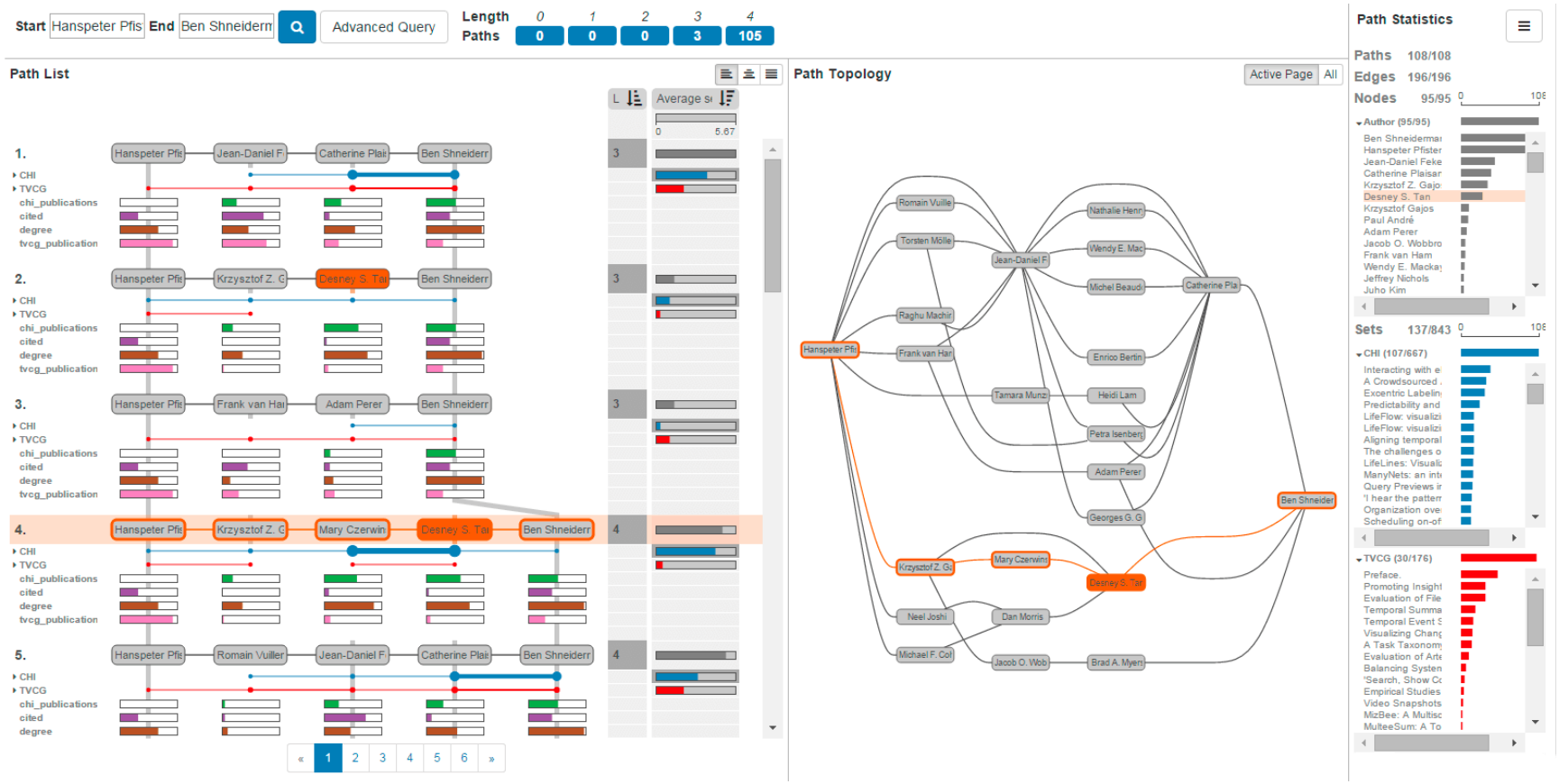


Design of juxtapose views

- What is shared?

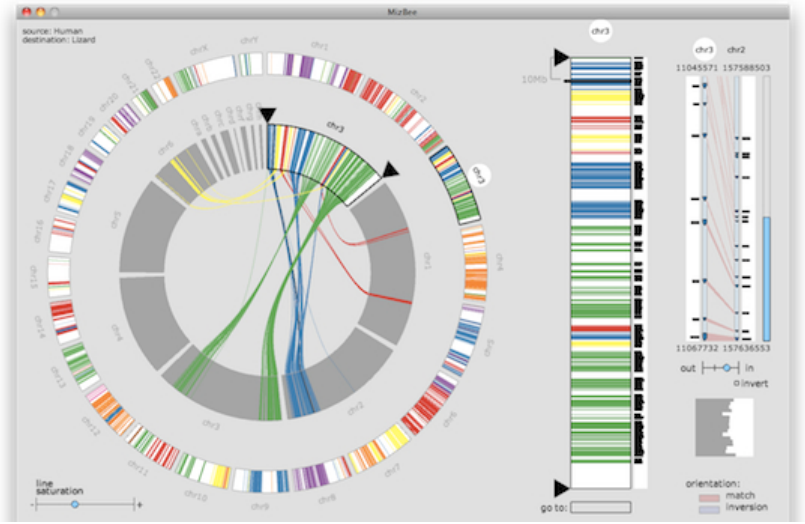


Example: multiform views



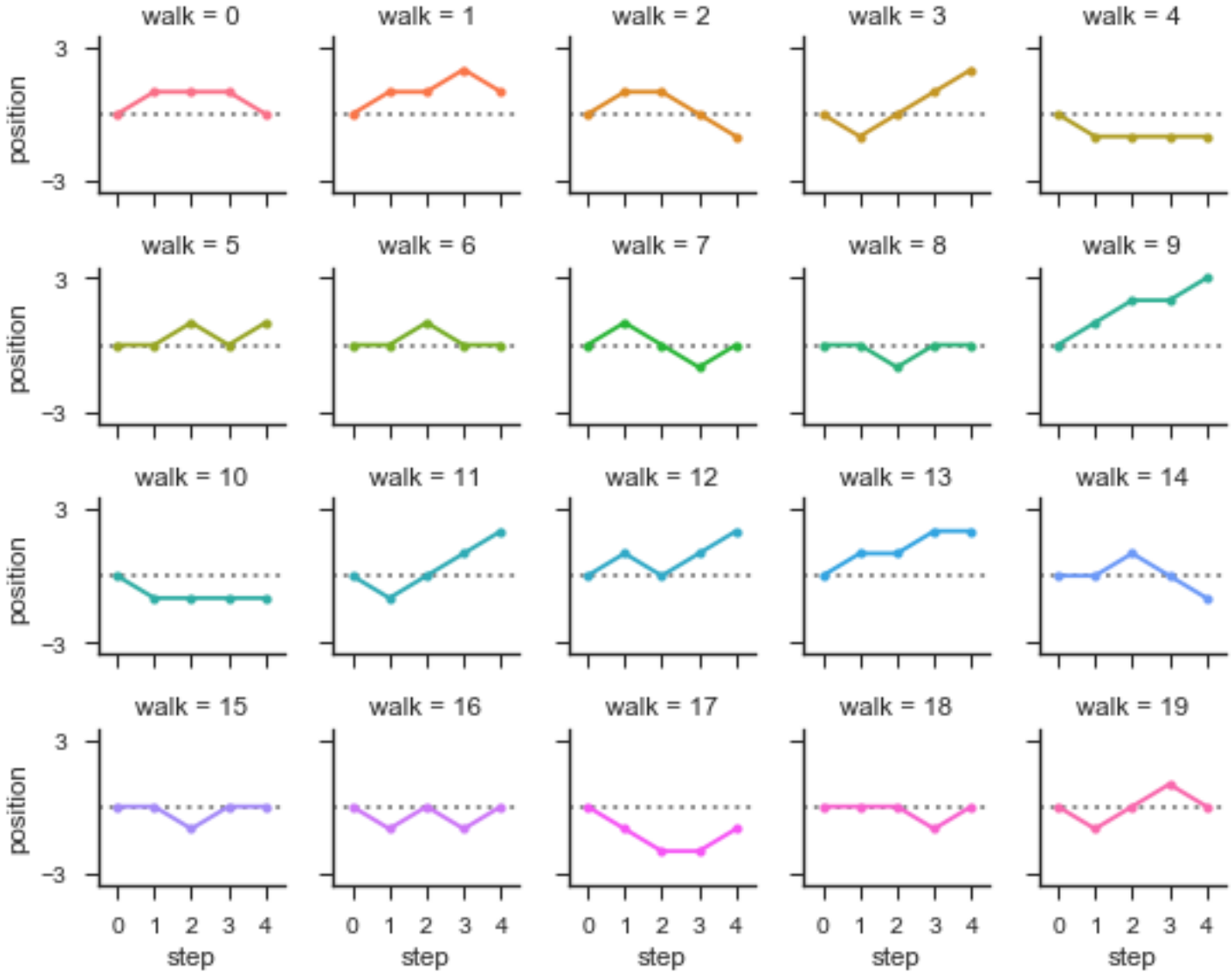
https://www.youtube.com/watch?v=aZF7AC8aNXo

Examples: overview-detail



https://www.youtube.com/watch?v=86p7brwuz2g

Example: small multiples

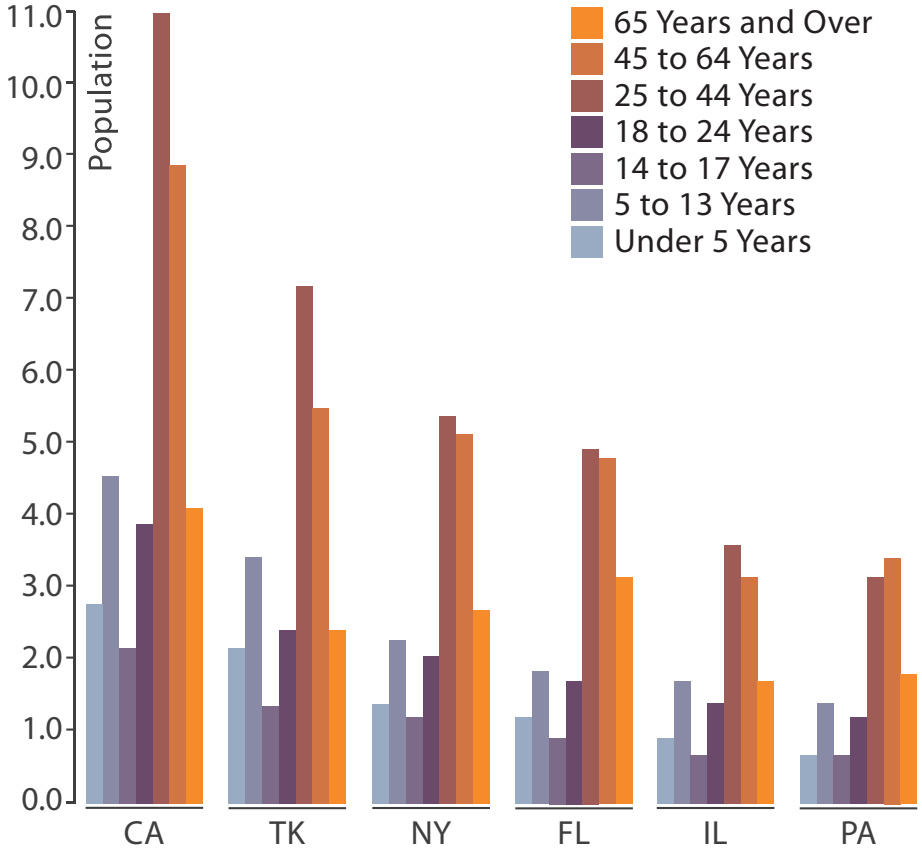


Design of small multiples

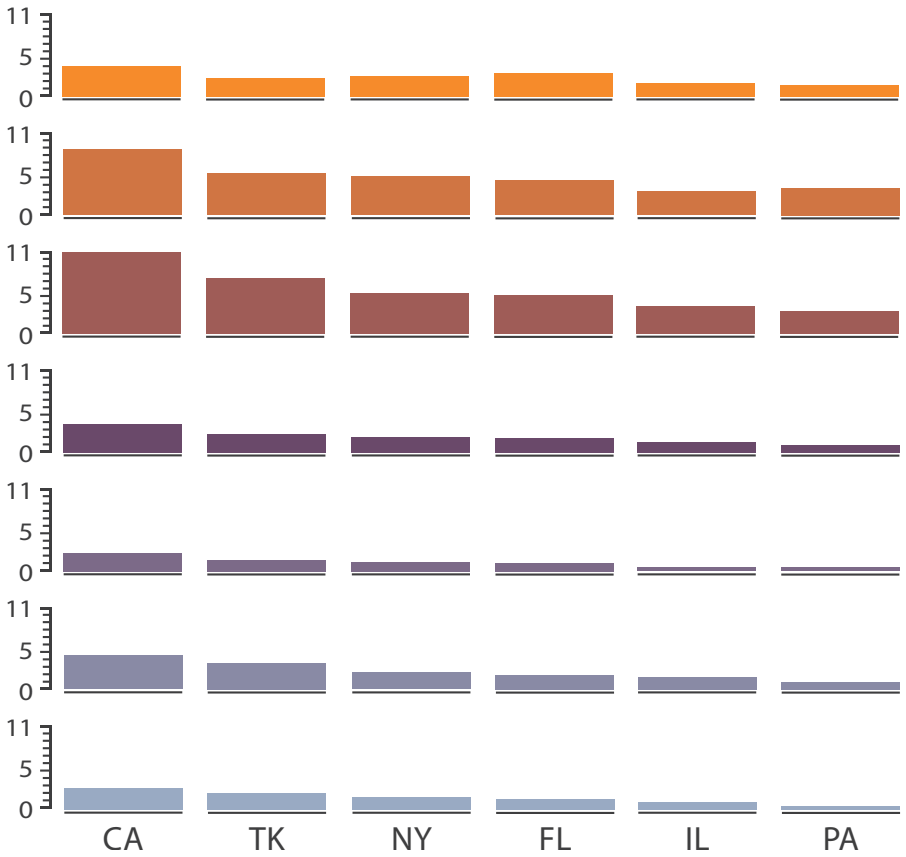
- How to partition data between views
  - Encodes association between items using spatial proximity
  - Split data into regions by attributes
  - Orde r of splits largely affects patterns visibility
- Alignment
  - List
  - Matrix
  - Recursive
- Views vs Glyphs
  - View is a big/detailed region that contains data visually encoded
  - Glyph is a small/iconic object composed of multiple marks
  - Distinction is not always clear

Example: Small multiple designs

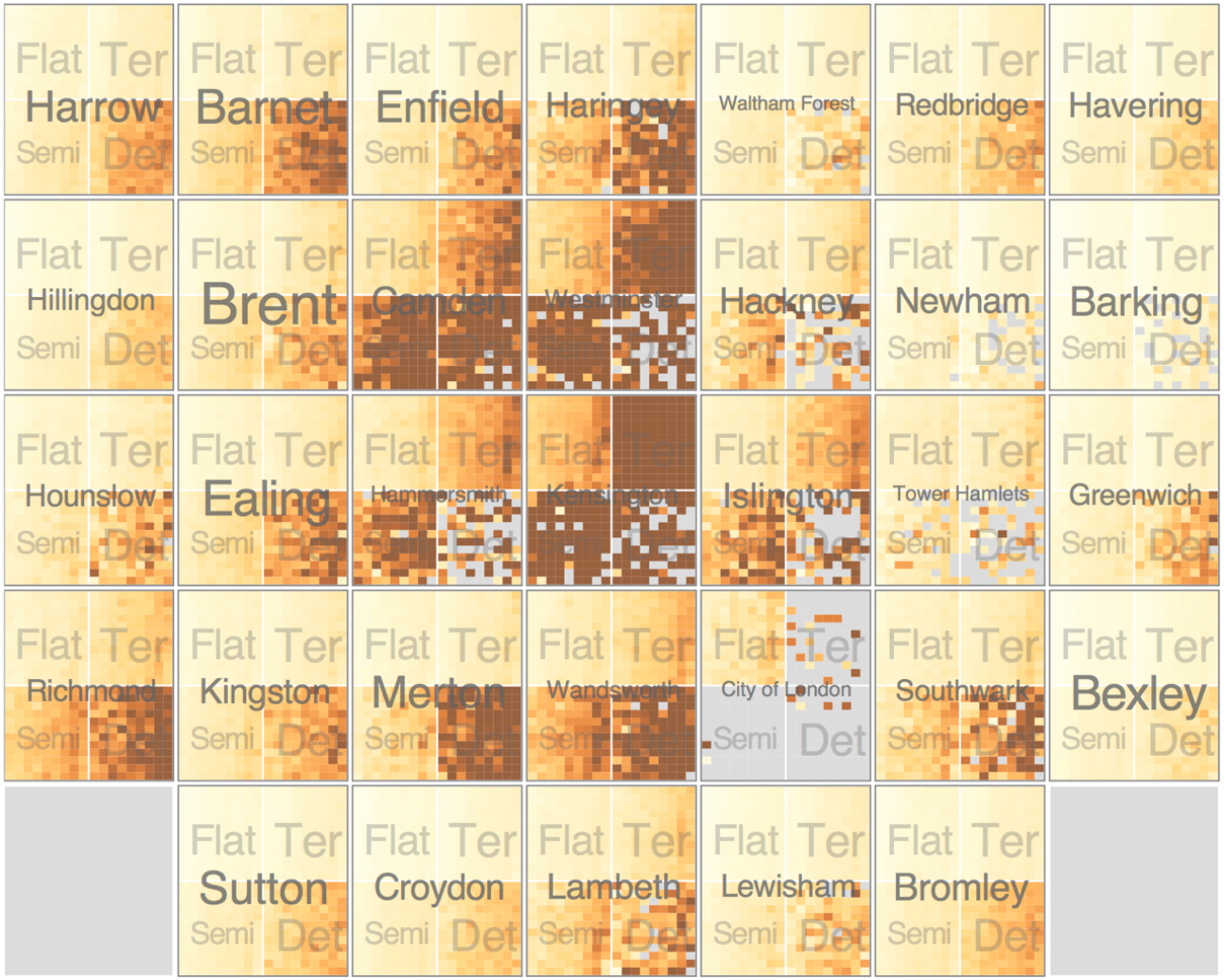
- List of complex glyph
- State comp. easier than ages



- Matrix of bars
- Age comp. easier than states



Example: small multiples recursive



Example: small multiples recursive

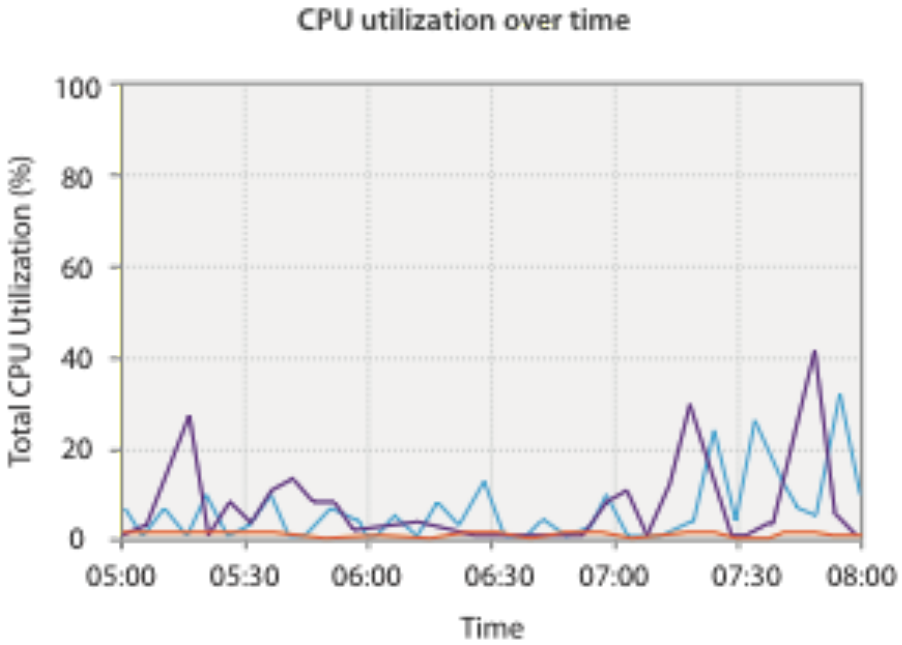
Faceting: Superimpose

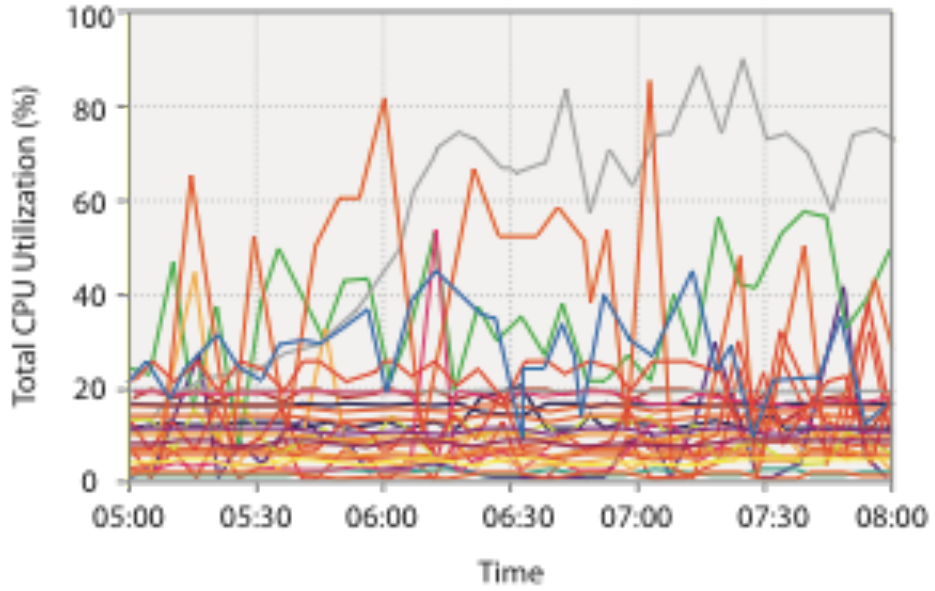
Superimpose layers

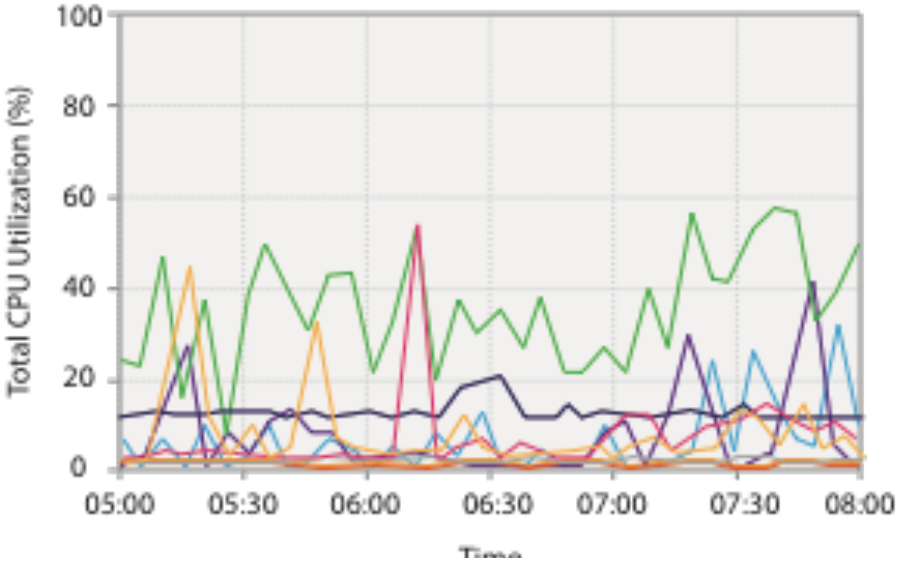
- A layer is a set of objects spread out over region
  - Each set is visually distinguishable group
  - It might extend through the whole view
- Design choices
  - how many layers?
    - two layers achieveable, three with careful design
  - how to distinguish?
    - encode with different, nonoverlapping channels
  - small static set, or dynamic selecting from many possibility?

Example: superimposing lines

- Scale up to few lines (a dozen if well separated)

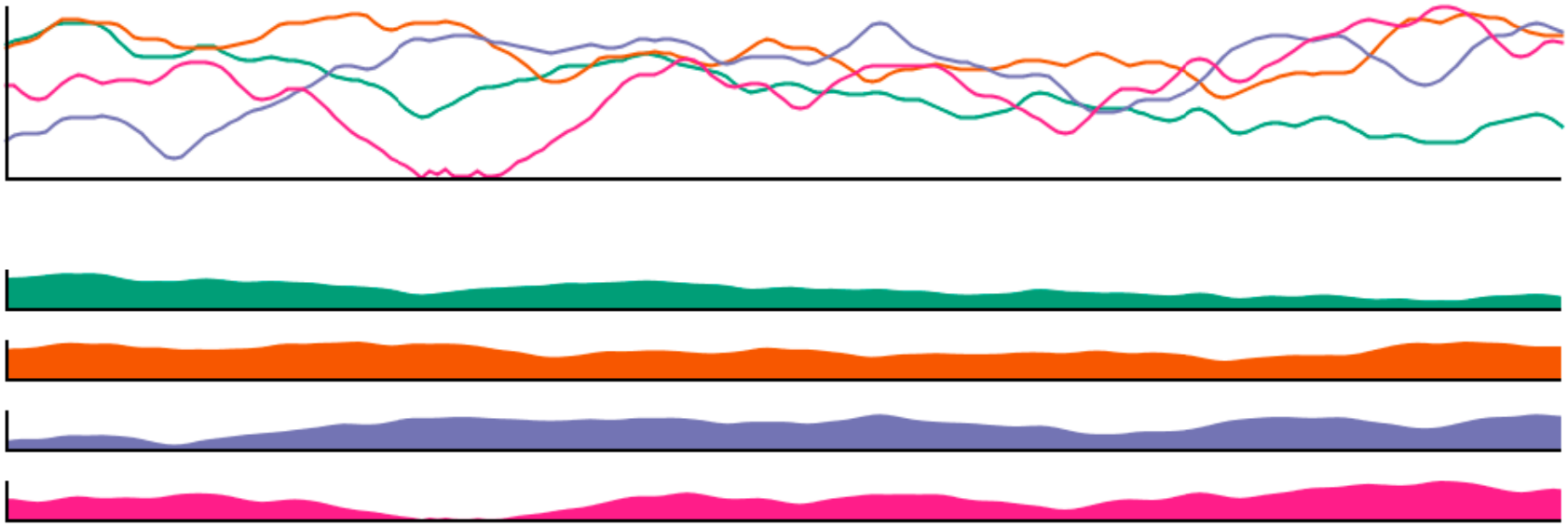




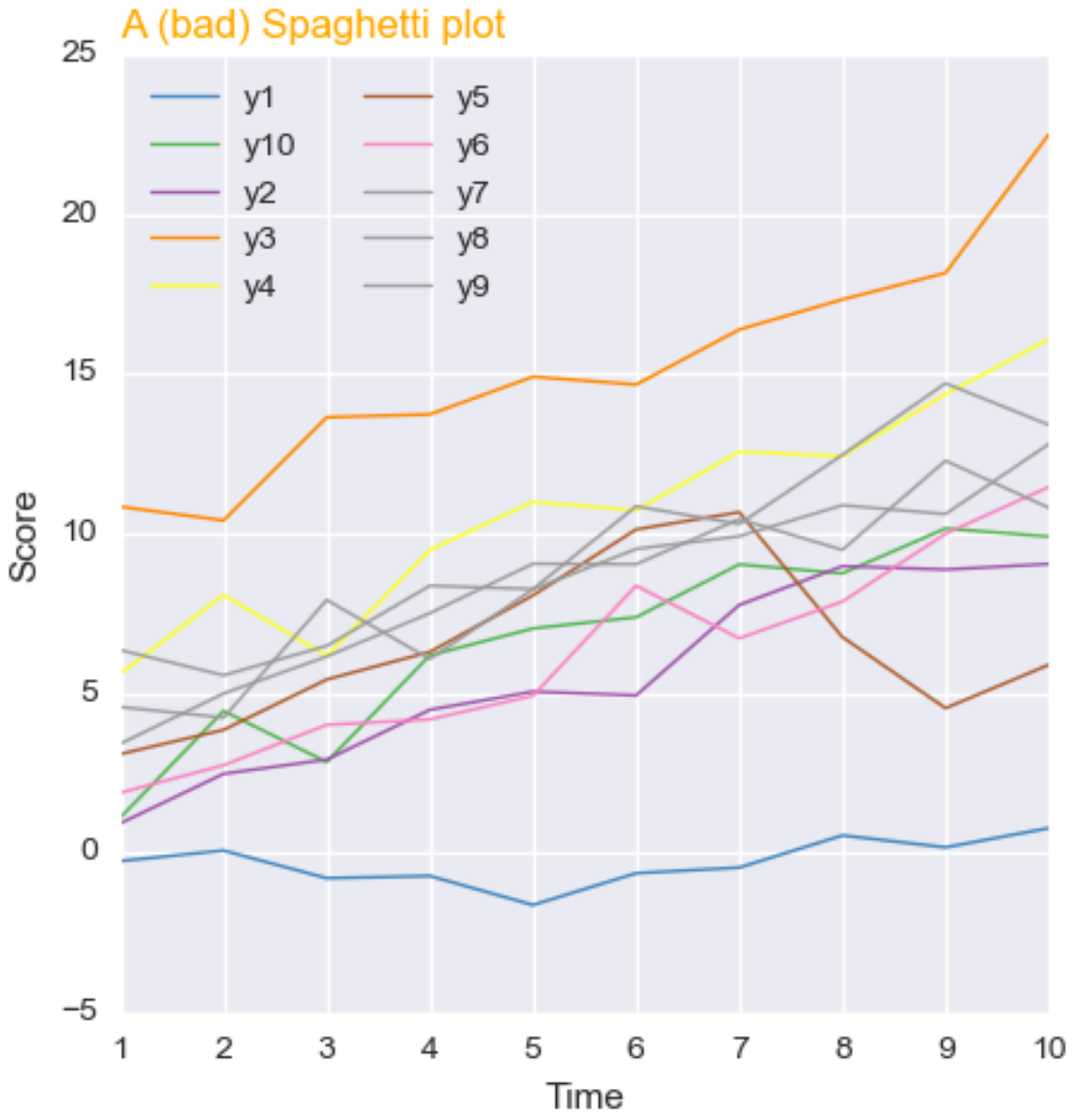


Superimpose vs Juxtapose

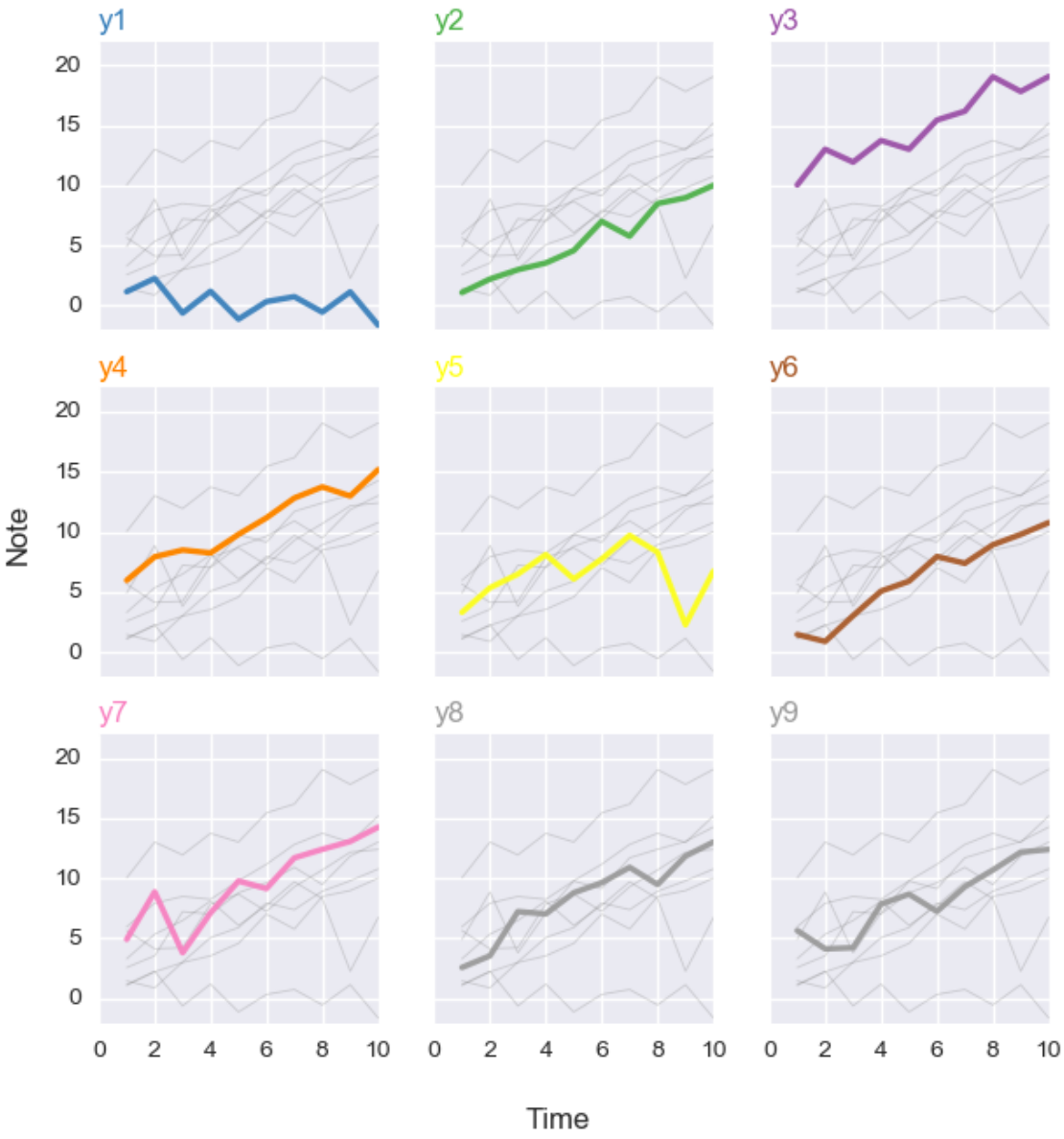
- Empirical studies show that superimposing is better for local tasks as find maximum at given position.
- Juxtaposing supports better global tasks as pattern identification.
- Example



Combining together!



Combining together!



VALIDATION

How to get visualization “right”?

What are the steps to follow?

And how do you validate these steps?

Get it right?

1. Solving the **right problem**:
   Not a problem your users don't have.

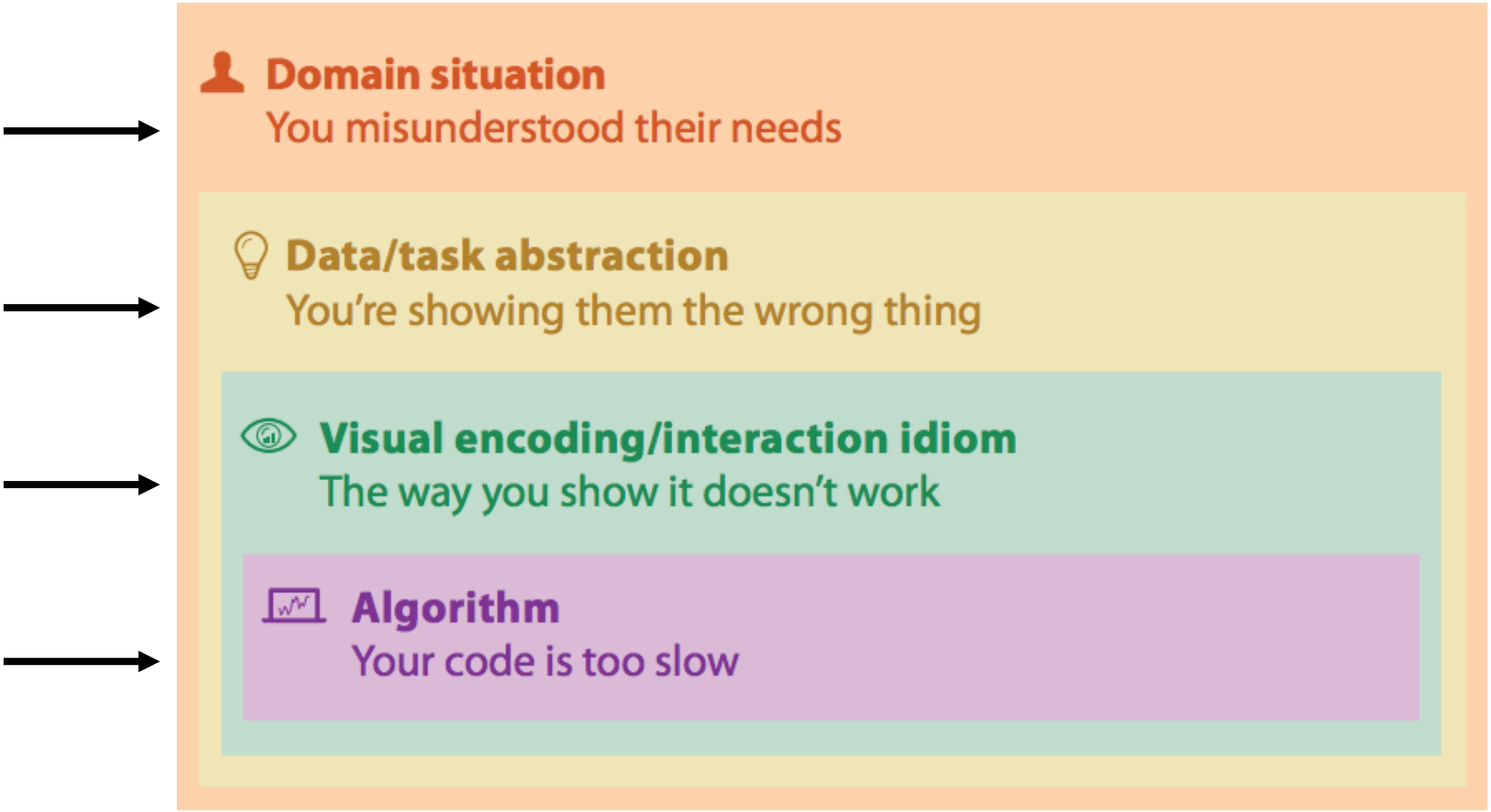
2. Solving the **problem right**:
   your solution is *effective*.

Effective?

- There are many ways to define effectiveness but ultimately what counts is that your *users* or *readers* are able to extract information out of data **accurately**, with **reasonable effort**, and **high confidence.**

- The ultimate test is **knowledge gain.** If your users do not learn anything new you are in trouble. Knowledge gain must be your yardstick.

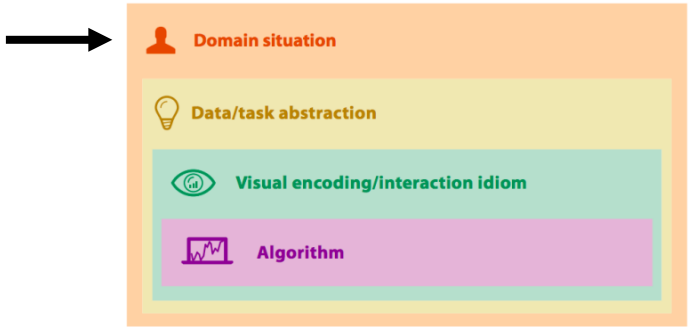
Threats


Domain situation
You misunderstood their needs
Data/task abstraction
You're showing them the wrong thing
Visual encoding/interaction idiom
The way you show it doesn't work
Algorithm
Your code is too slow

Domain

- Understanding who are your target users and what problems they need to solve with (their) data.

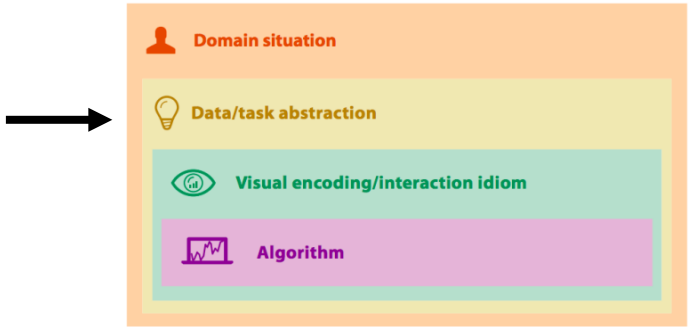
- **Outcome of this step**: a series of questions or actions your users want to carry out with this data.



Abstraction

- After *domain situation* you need to understand **what** needs to be visualized for **what** tasks.

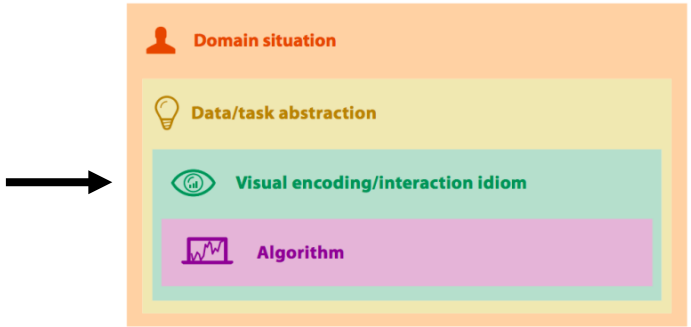
- If you fail here it means you are not showing the right things and/or supporting the right operations.



Visual Encoding and Interaction

- You need to design effective visual representations and interactions for the data and tasks identified.

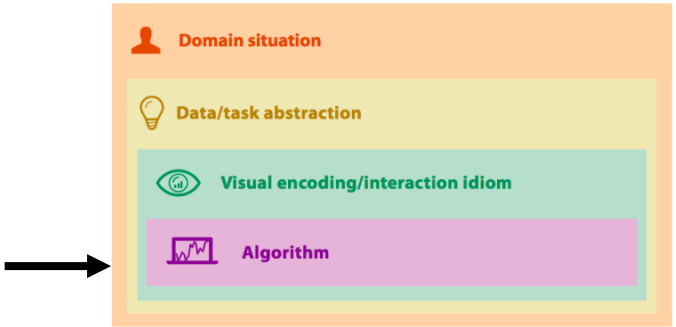
- This is where you need to know how to get rid of poor matches and ideate good designs.



Algorithm

- This is the lower lever where the algorithm and the logic that creates the visualization itself is designed and implemented.

- Here you want to make sure your algorithm is fast, accurate and effective.



Practical suggestions

Strategies

- **Gather** all this information from your target users (your assigned mentors and domain experts) through interviews.
- **Generate** as an output a description of the problem, the data, and a set of questions they have on the data.
- **Engage** keep asking questions, double-check to see if you got it right, read material they use/produce, observe what they do, learn the language.
- **Explore** the data don't wait to have a visualization out, give a look to your data with existing tools.
- **Draw/prototype** don't talk about visualization, show it / draw sketches, make small prototypes, be ready to trash it all!

Common issues

- Problem not well-defined or misunderstood: Talk to your users and observe them doing their work / learn their language and domain.
- End-users not well-defined or not available: End-users are people who are actually using your system / ask to have access to them.
- Data not available: Stop! No way you can work without real data!
- Visualization not needed: Ask yourself: why visualization? Are there better ways to solve this without visualization? Can I solve this algorithmically?

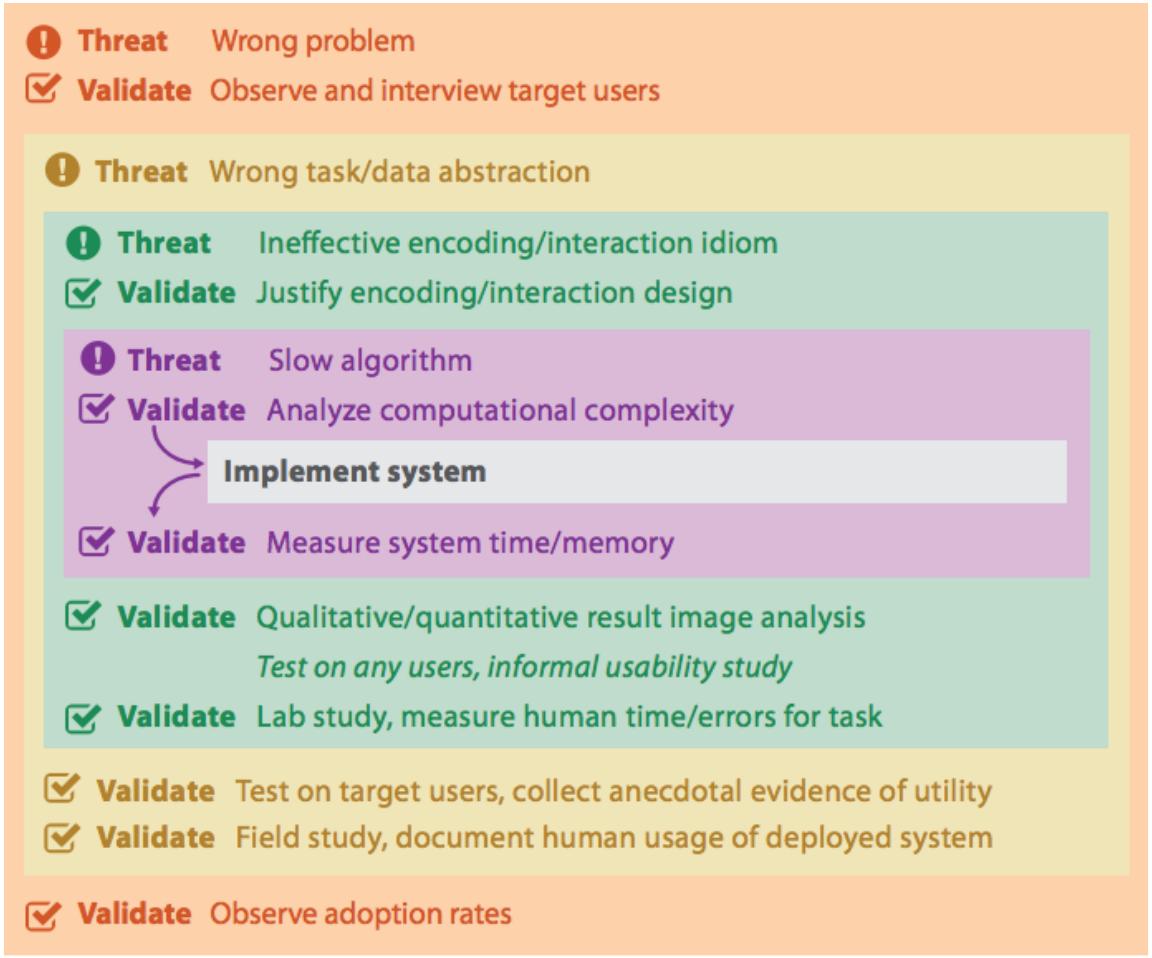
A final advice…

- Most encodings are suboptimal and you keep learning how to create better ones by failing. The more you fail the more your encoding improves.

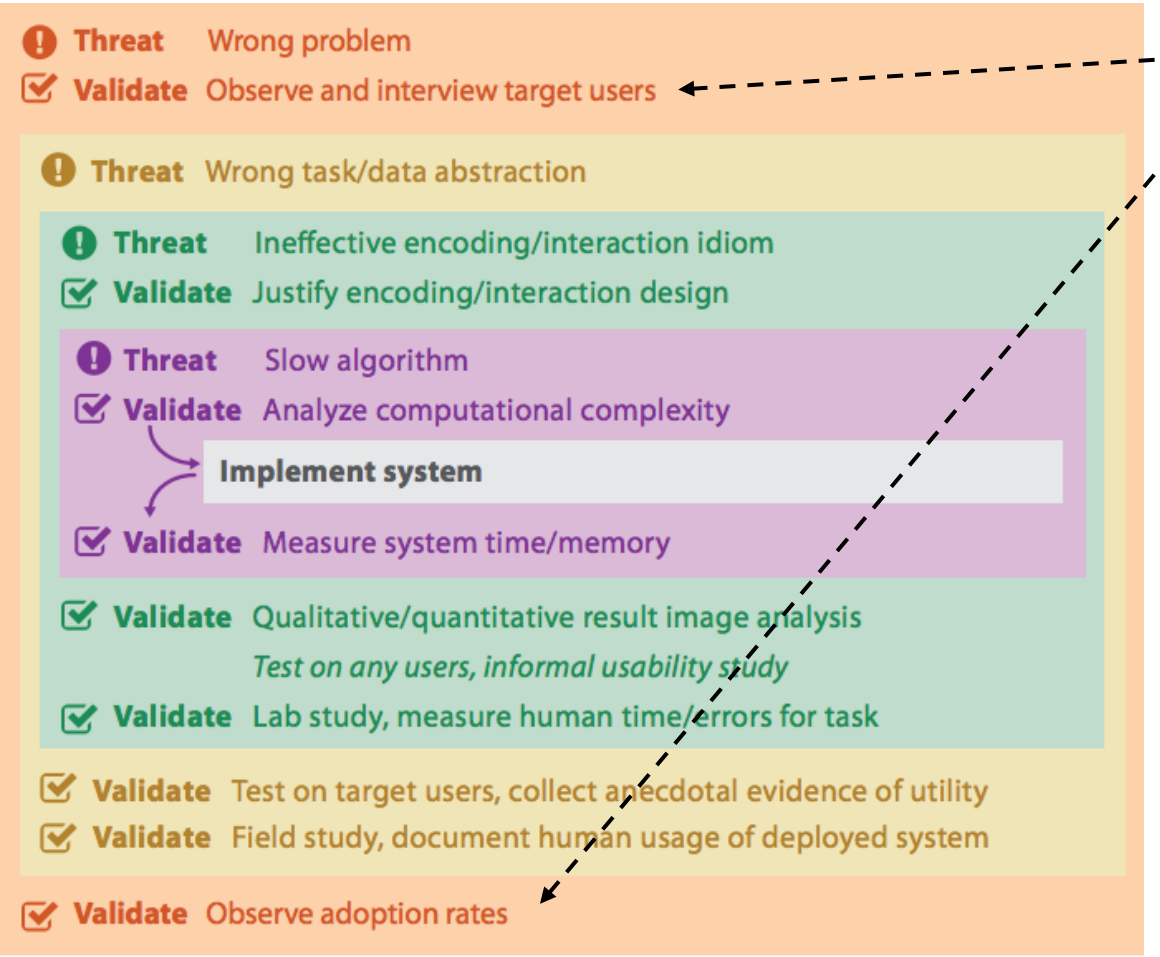
- Do not stop iterating. Don’t be satisfied with the first one that comes into your mind! Generate/compare alternatives.

Validation strategies

Validation approaches

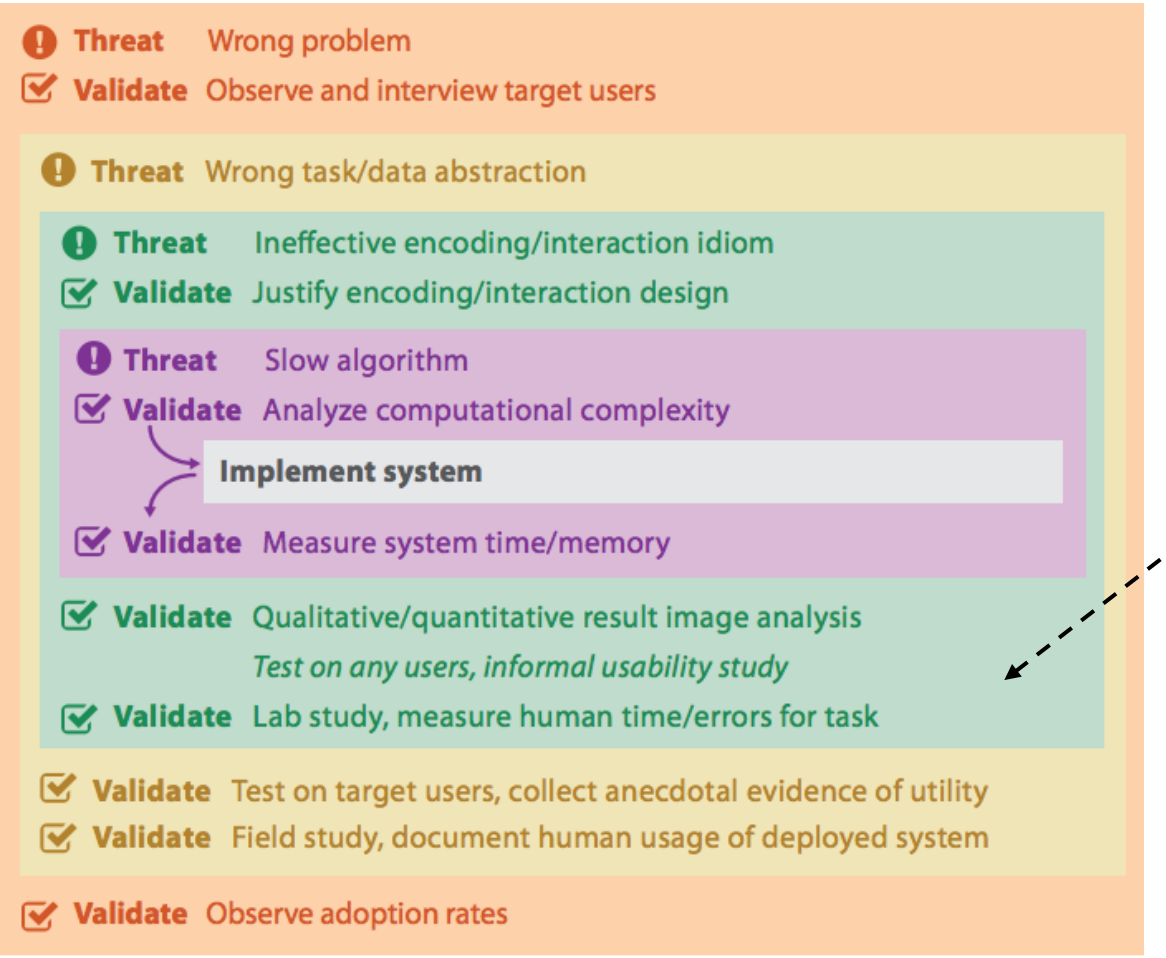


Validation approaches



Validating
Domain Situation

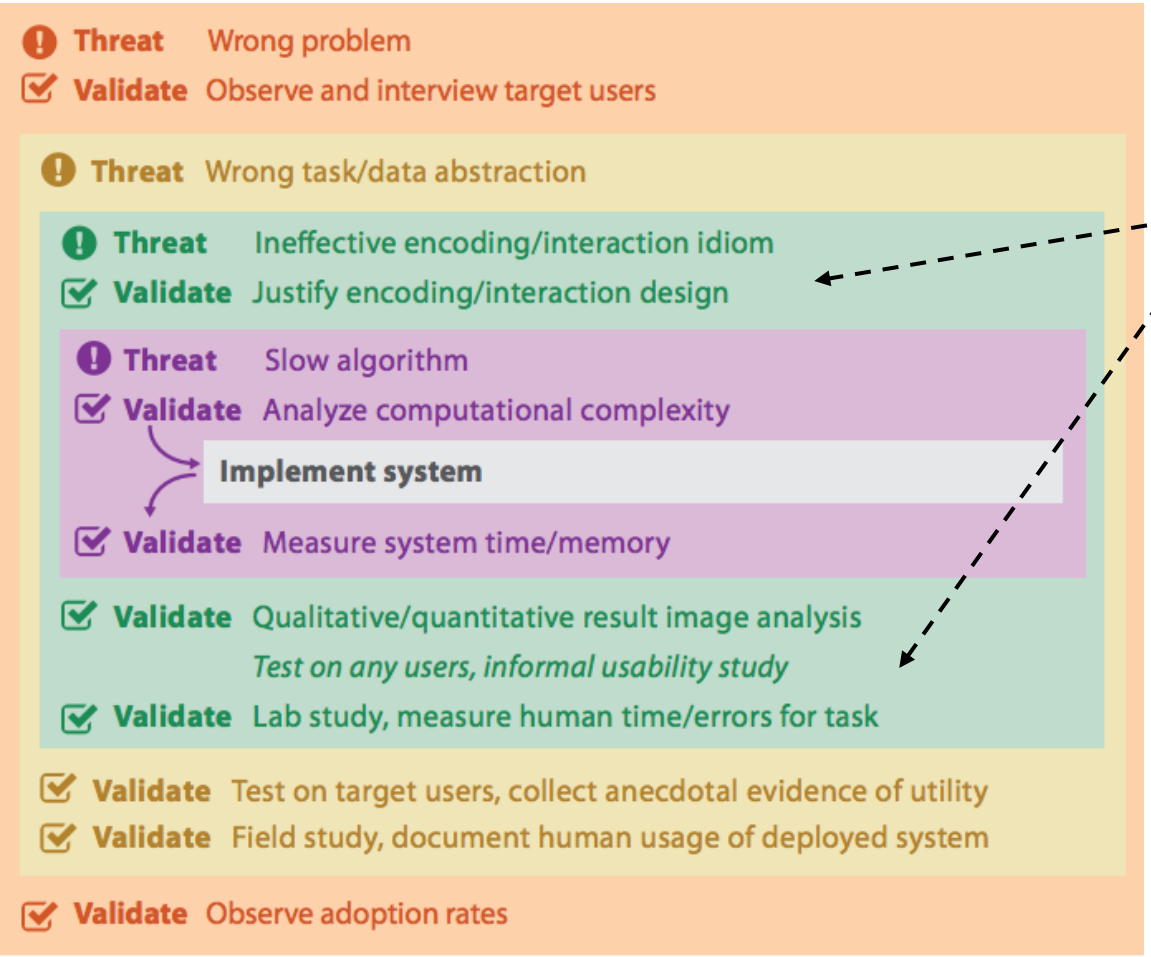
Validation approaches



Validating Task and Data Abstraction

It's only downstream but you can (must!) use prototypes to go through several iterations.

Validation Approaches

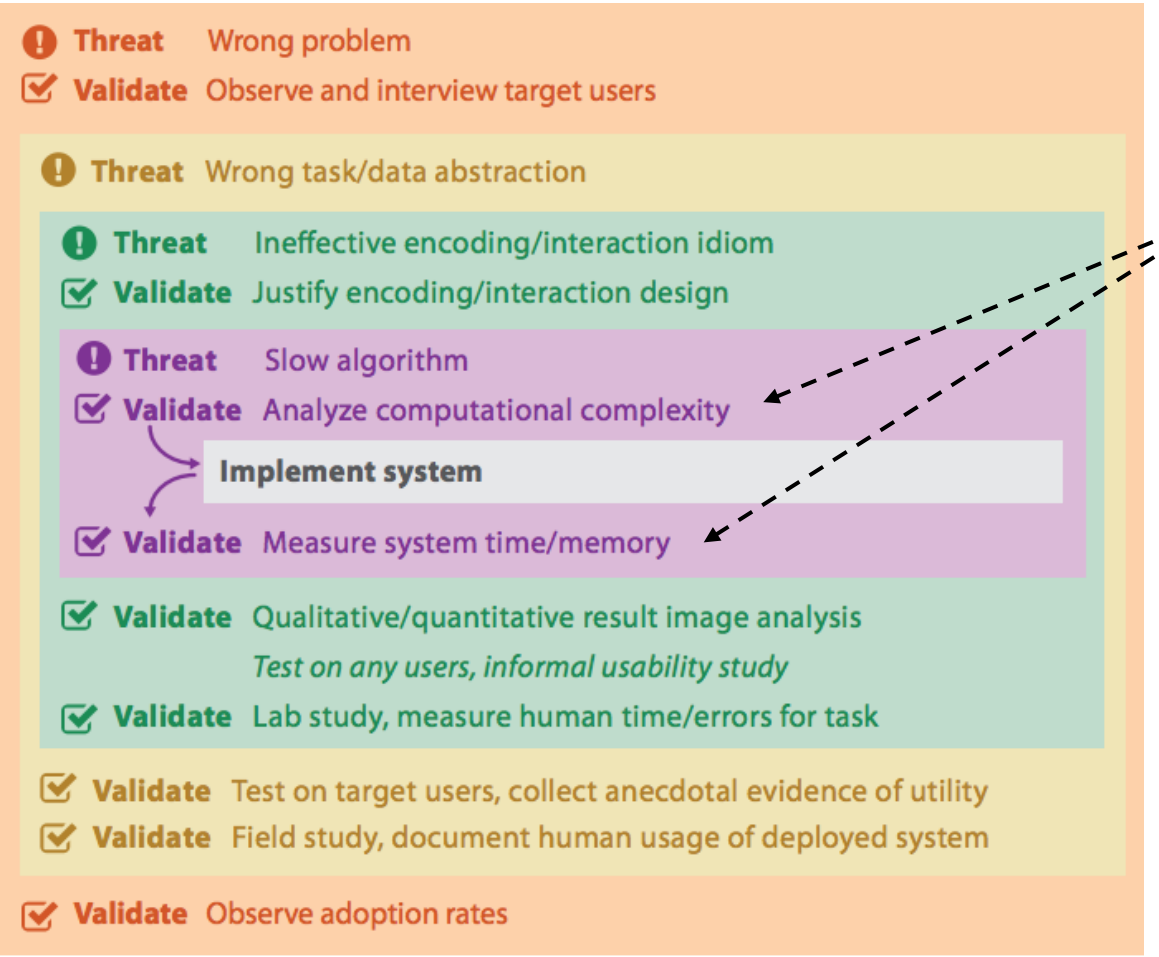


Validating Idioms

We give a lot of weight to *justification of encoding/interaction design*.

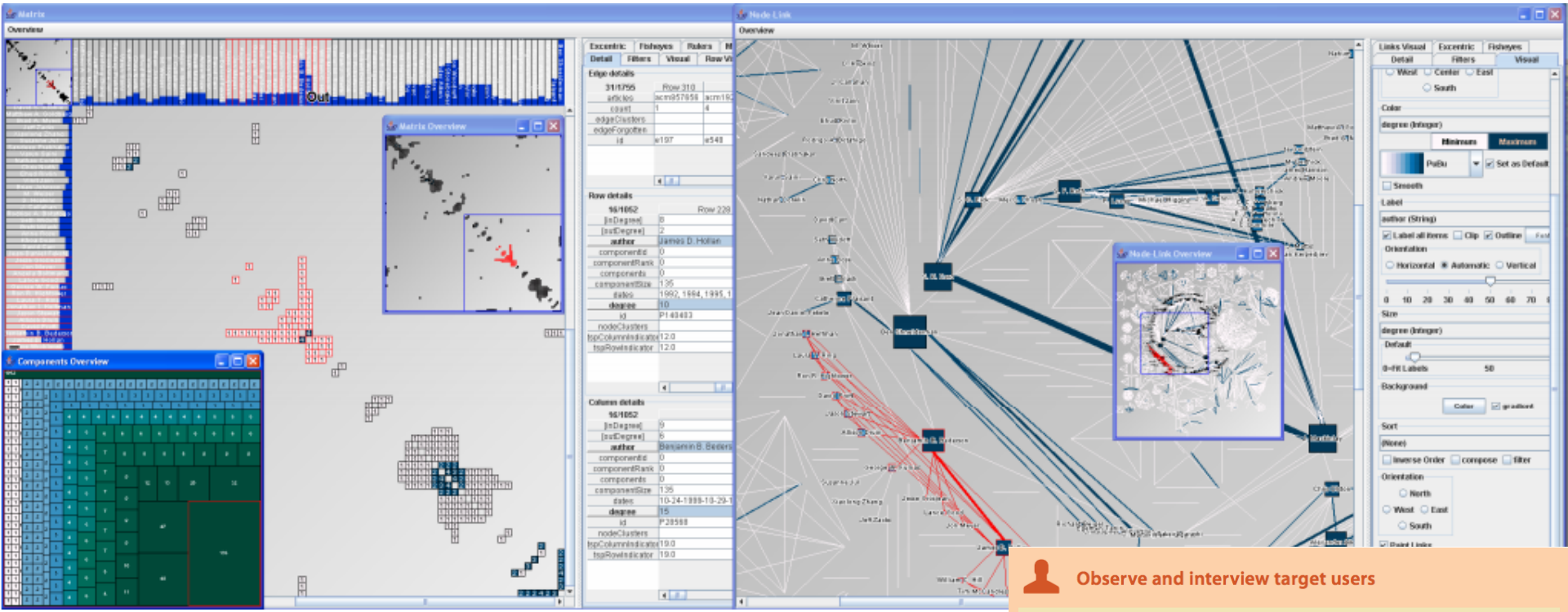
It is also important to show convincing images that show the encoding is effective

Validation approaches



Validating Algorithms

Examples



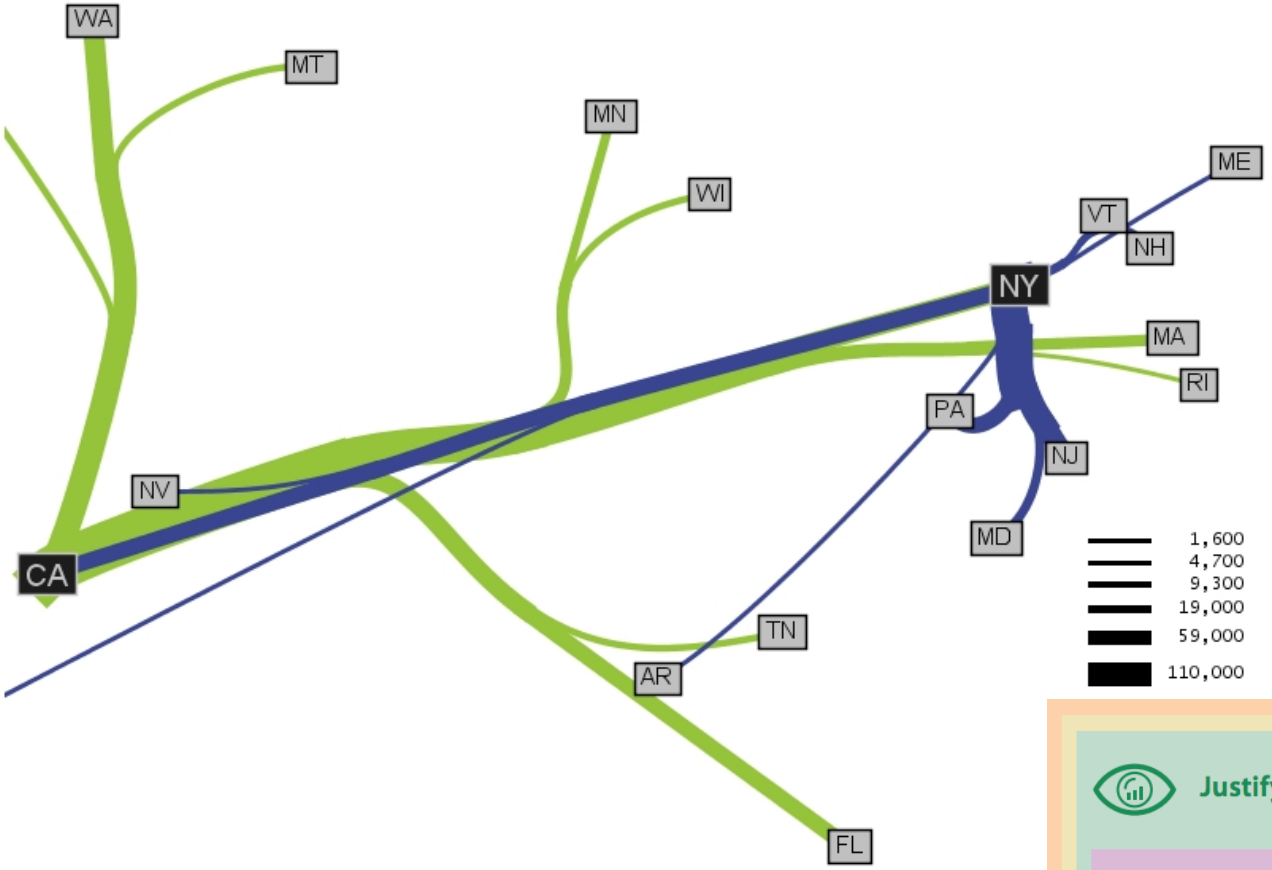
Observe and interview target users

Justify encoding/interaction design

Measure system time/memory

Qualitative result image analysis

Henry, Nathalie, and Jean-Daniel Fekete. "MatrixExplorer: a dual-representation system to explore social networks." Visualization and Computer Graphics, IEEE Transactions on 12.5 (2006): 677-684.

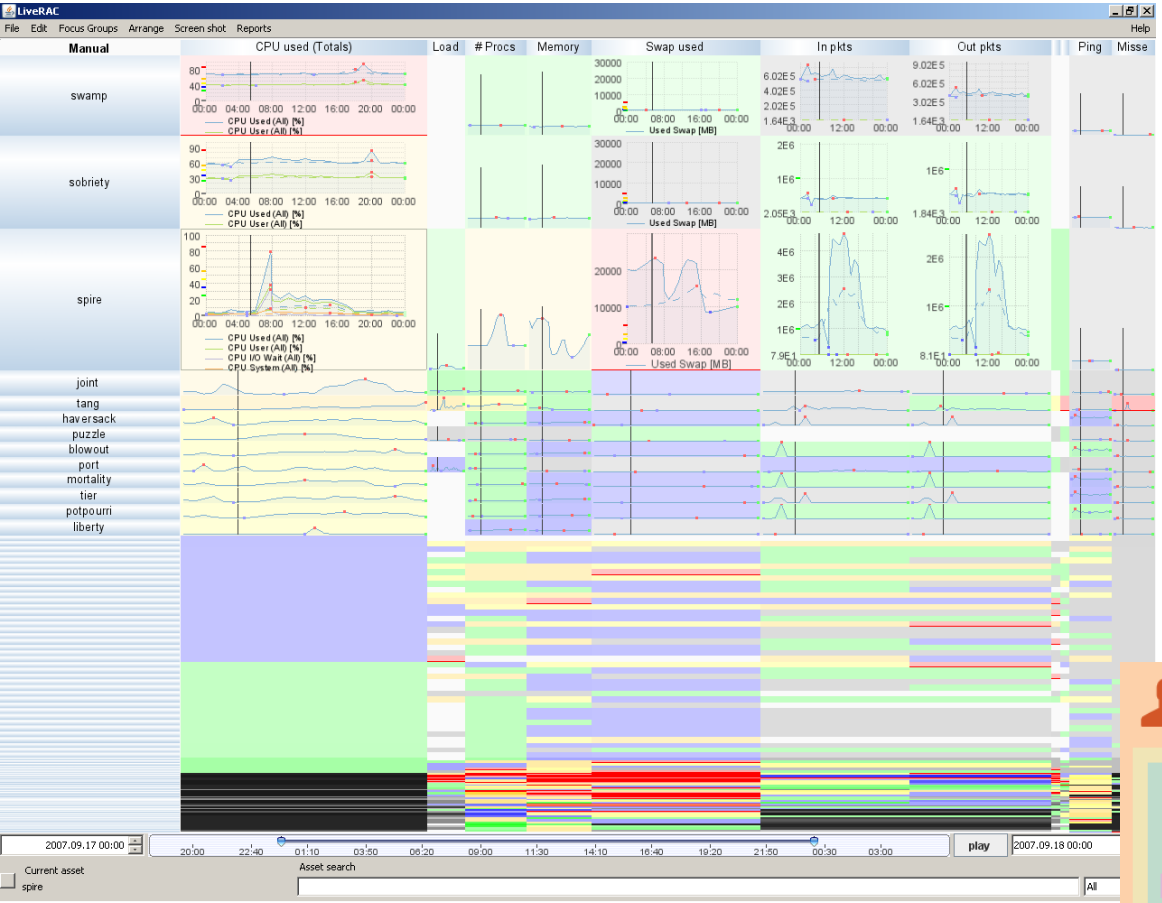


Phan, Doantam, et al. "Flow Map Layout." IEEE Symposium on Information Visualization, 2005. INFOVIS 2005. IEEE, 2005.

Justify encoding/interaction design

Computation complexity analysis
Measure system time/memory

Qualitative result image analysis



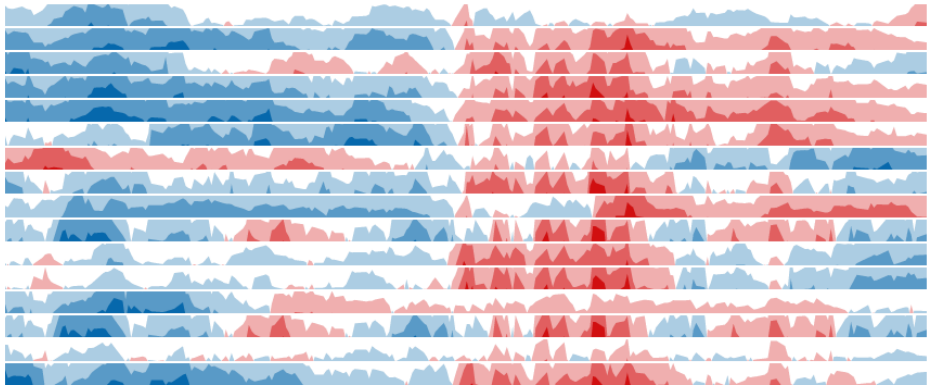
McLachlan, Peter, et al. "LiveRAC: Interactive Visual Exploration of System Management Time-Series Data." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2008.

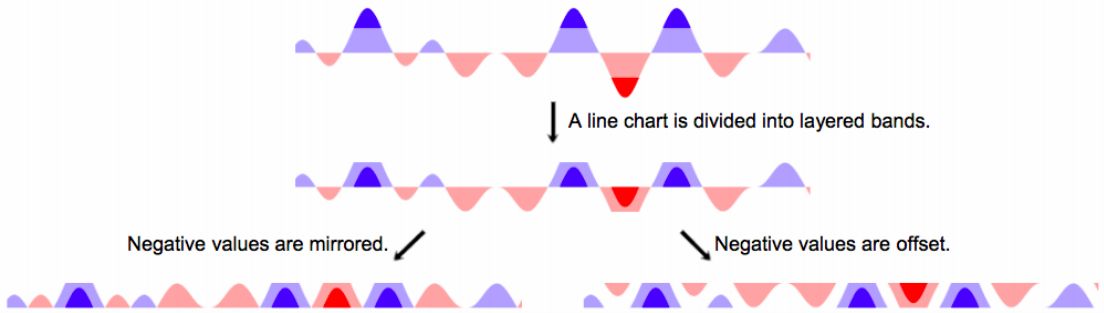
Observe and interview target users

Justify encoding/interaction design

Qualitative result image analysis

Field study, document usage of deployed system





How does mirrored vs. offset compare?

How does the number of bands affect performance?

Heer, Jeffrey, Nicholas Kong, and Maneesh Agrawala. "Sizing the horizon: the effects of chart size and layering on the graphical perception of time series visualizations." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2009.

Lab study, measure human time/errors for task