goto; conference

Explorations in Interactive Visual Analytics

Supporting Analysis and Visualization At Scale

Doug Talbott
Bedarra Research Labs
doug.talbott@bedarra.com

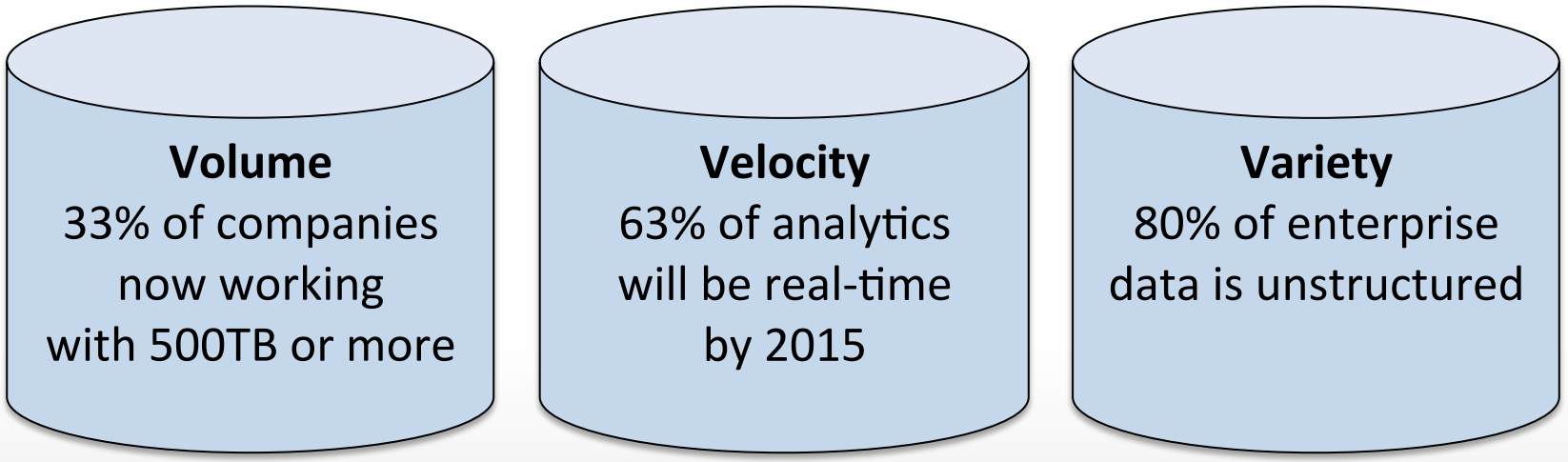
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE

What we're going to talk about

- What is big data?
- Who is the big data user?
- What are we trying to accomplish in this space?
- Challenges we've encountered
- Things we've learned

What is big data?

Datasets that are too large or complex to be managed using traditional methods and tools…



Various sources including Gartner. IDC. Forrester

Who is the big data user?

- **Domain experts in science, business, engineering, arts**
  - Also known as "data scientist"
  - Critical thinking skills
  - Statistical and model-based programming skills
  - 150-190K shortage in US alone

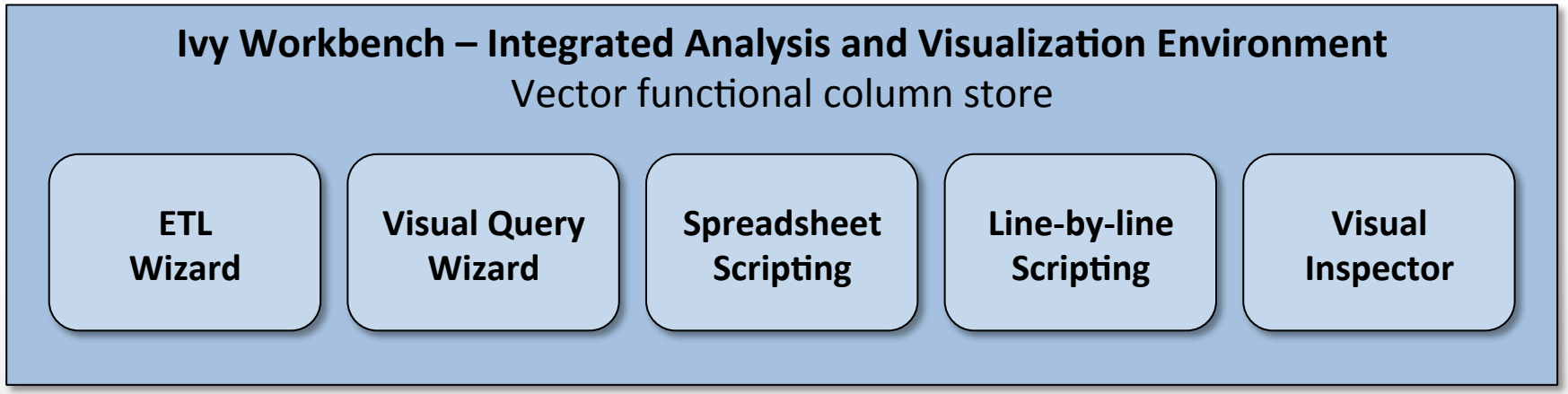
- **Range of languages**
  - 91% using R, SAS, Python, SQL*
  - 9% using Java, Hadoop (Pig, Hive, etc), SPSS, MatLab, Scala, C/C++*
  - Becoming familiar with clouds, clusters, grids, hadoop, etc

- **Range of applications**
  - Sentiment analysis, fraud detection, customer churn analysis, network monitoring, risk modelling, social network analysis, etc

*2014 KD Nuggets poll respondents

What are we trying to accomplish?

1. Believe there is value in looking at big data
2. Trying to deliver a 'small data' user experience at 'big data' scale in terms of response times, interactivity and ease-of-use…



What are the challenges to delivering that kind of user experience at scale?

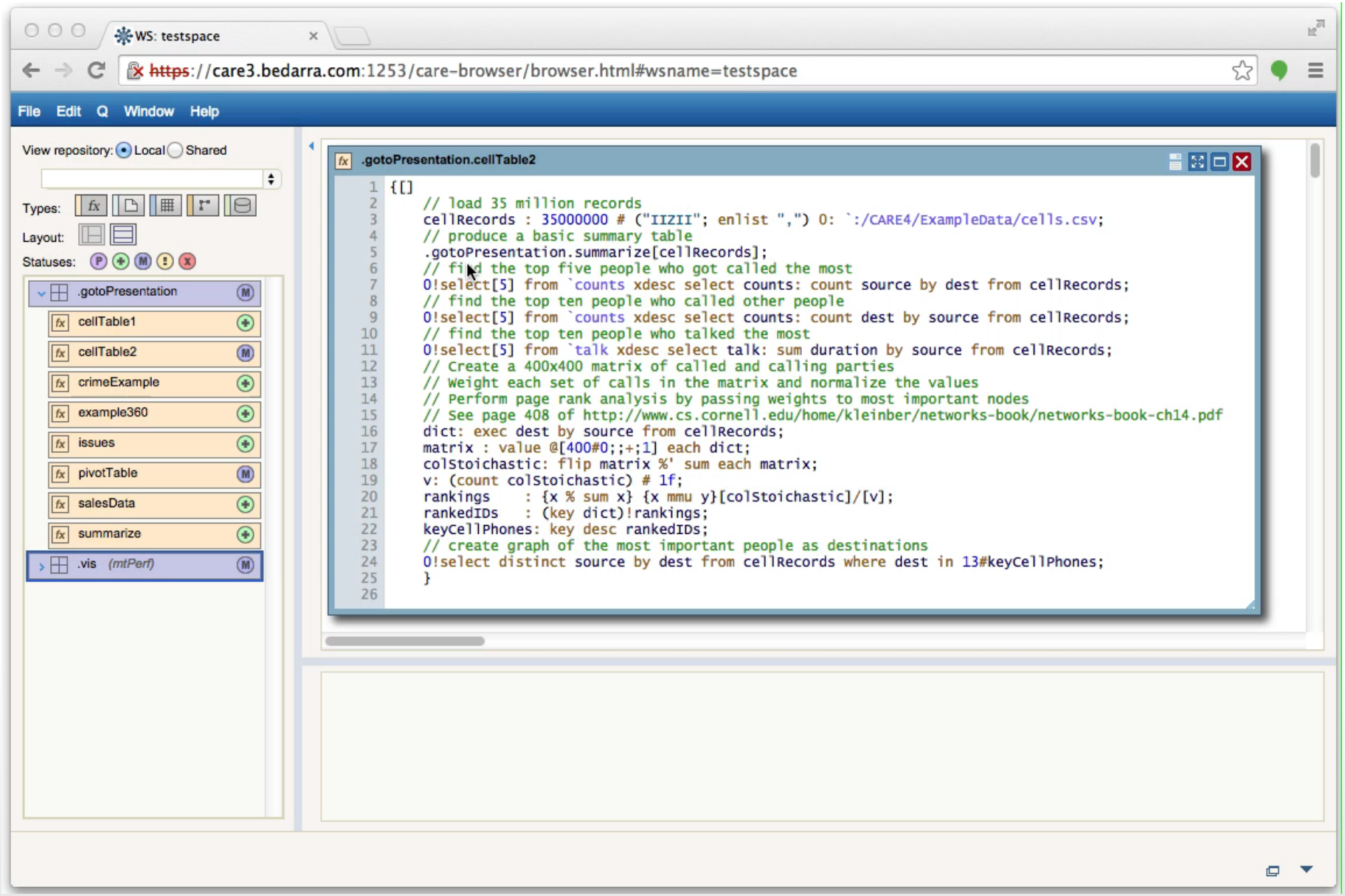
Potential problems	Potential solutions
Short attention spans	Increase processing and rendering speeds Reduce latency
Loss of context	Increase processing and rendering speeds Reduce latency
Limited short term memory	Keep UI controls, data, scripts visible Increase processing and rendering speeds Reduce latency
Increased cognitive load	Design UI controls to handle complexity Provide flexible approaches to analysis
Increased perceptual load	Leverage pre-attentive processing skills Encourage pattern matching Provide flexible approaches to visualization

A typical analysis workflow example

- **VAST Challenge Case Study**
  - Analyse cell phone records to identify a social network of criminals
  - Cell phone records contain source, dest, duration, tower, timestamp
  - Phones are numbered 0 to 400
  - Original set was tiny so we 'poofed' it up to 35 million records
  - Suspicious behaviors to look for…
    - Destinations receiving lots of calls from different sources
    - Destinations suddenly disappear, destinations suddenly appear and receive lots of calls from the same set of different sources

- **Technology**
  - Ivy workbench, but other solutions may do similar things (e.g. R Studio, Rapid Miner, Datawatch, Weka, Tableau, Pentahoe, Palantir, SpotFire)
  - Internal server using 4 cores

Case Study Demonstration



So what did we see in the example?

- **Real-time 'tumbling of data' to explore and discover relationships**
  - Very fast loading of records
  - Real-time analysis and visualization of 35 million records
  - Summary, page rank, and query scripting 'on-the-fly'
  - Different plot types
  - Different column configurations
  - Different levels of granularity (bin sizing)
  - Different levels of detail (zooming)

- **Minimal human factors impediments**
  - No attention span problems (fast processing)
  - No loss of context (fast processing)
  - No memory problems (visible controls, scripts, data)
  - Reduced cognitive load (flexible, scalable control interfaces)
  - Reduced perceptual load (binning to leverage pre-attentive processing)

What are the key learnings so far?

- **Supporting analysis at scale**
  - Support integration of analysis and visualization tasks
  - Support an interative, non-linear task flow at scale
- **Supporting visualization at scale**
  - Leverage server-side computing and rendering
  - Leverage techniques that encourage pattern matching
  - Allow users to deviate from accepted visual design rules
- **Support interaction at scale**
  - Manage the data-to-ui ratio
  - Provide interaction control to support data complexity
  - Make scripting a primary task

Support integration of analysis and visualization tasks

Need to tightly couple analysis and visualization

- **Schneiderman's Data Visualization Mantra**
  - "Overview first, zoom & filter, details on demand"
- **Visualization tasks (blue) merged with analysis tasks (orange)**
- **Realistically you want to do these tasks in almost any order**

Analysis and Visualization Tasks	Example
Extract/load	Only take records from Denmark out of the dataset
Transform or lookup/load	Change gender from male, M, MALE, Male to zero
Script	Write a query, write a cluster analysis routine
Visualize (high-level, details)	View data as scatter chart
Filter	All records where sex equals female
Sort, categorize, relate, organize, mine	Rank children by height in grade 2
Brush, highlight, telescope	Rollover to see record details
Drill, zoom, pan	Rubberband select a region and zoom in
Annotate	Add a comment to a visualization
Collaborate	Sharing the visualizations, filters and computations

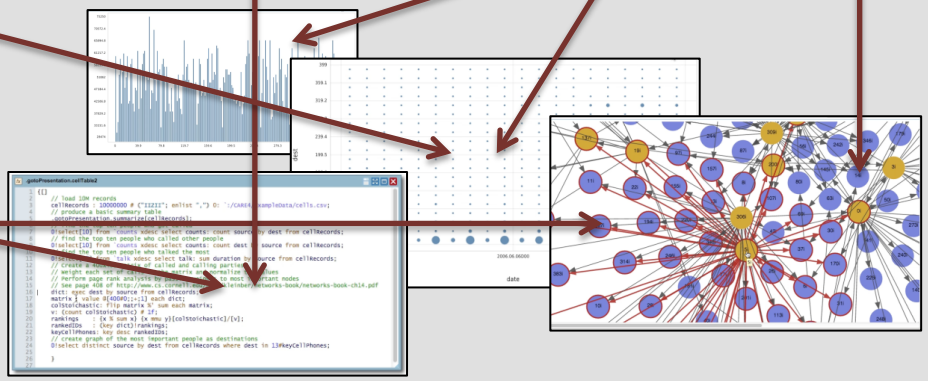
Support an iterative, non-linear taskflow at scale

Users need to be able to 'tumble' their data in order to explore and find relationships

Rankings
Parts-to-whole
Across time
Deviations
Distributions
Correlations

Organization Techniques				
Location	**Alpha**	**Time**	**Category**	**Heirarchy**
Maps	Tables, Pivot, Text	Timelines	Pie, Bar, Line, Scatter, Heatmap, Histogram	Decision Tree, Treemap, Graphs

Analysis Techniques	
Summarization	Keyword analysis
Clustering	K-means
Association Rules	Shopping cart analysis
Anomaly Detection	Outliers, link analysis
Classification	Groupings
Regression	Statistical models



Richard Saul Wurman - Information Anxiety The LATCH Model (orange)
Summarization Techniques for Visualization of Large Multidimensional Datasets Technical Report S. M. Kocherlakota, C. G. Healey (blue)

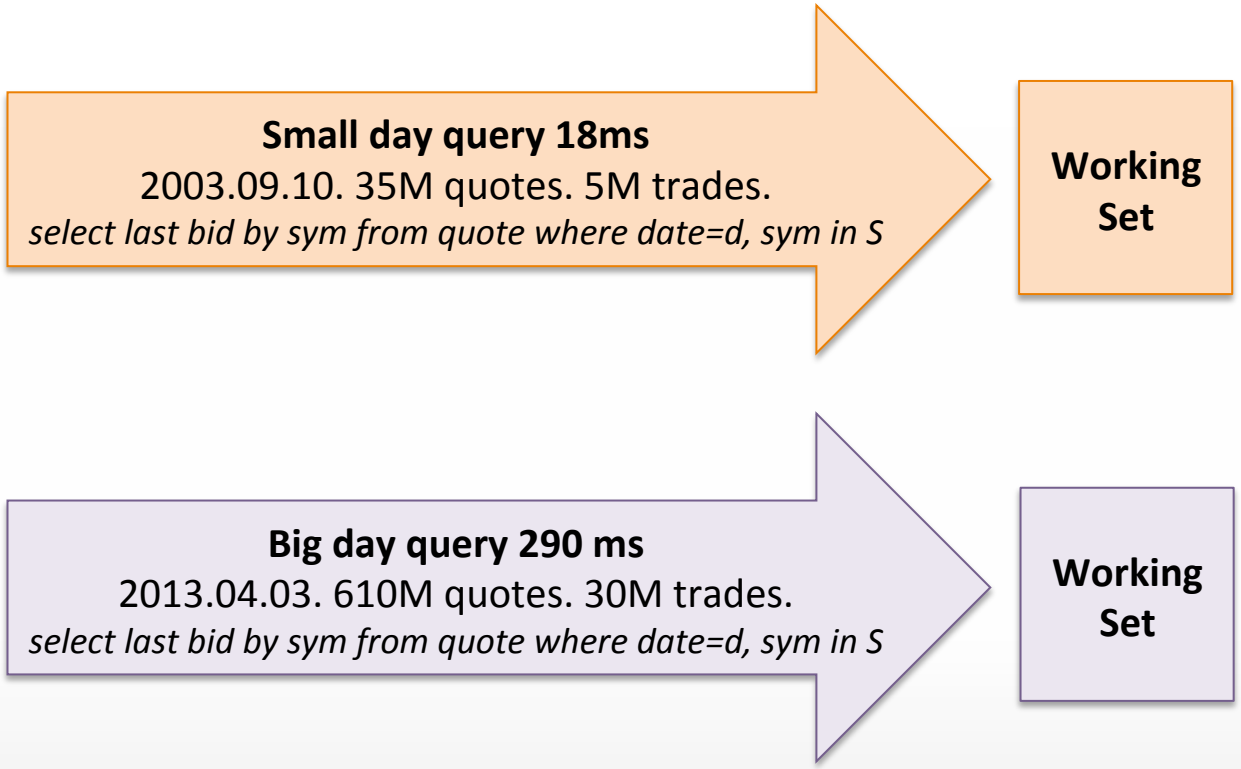
Leverage server-side
processsing and rendering
to improve interactivity at scale

Server-side processing lets you query and compute quickly…

100TB
NYSE TAQ
5000 days
1.1 trillion quotes
65 billion trades

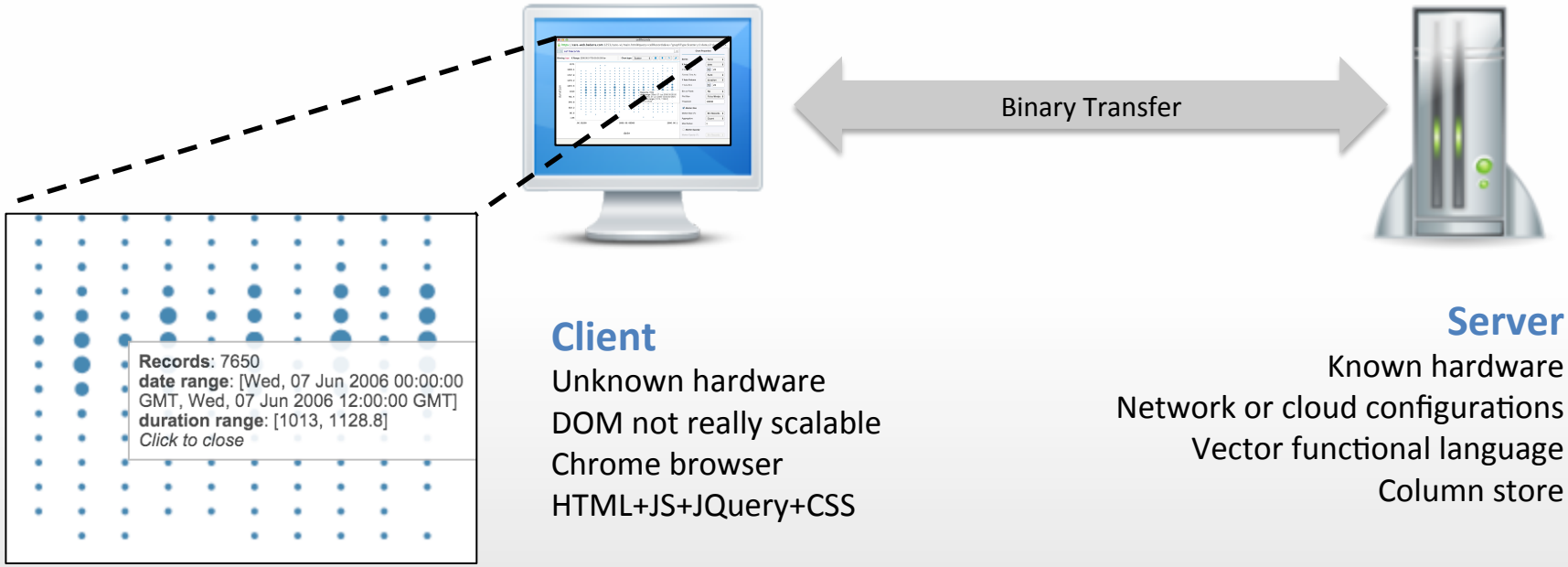
16 cores
256GB RAM
Date partitioned
Sym indexed

Column store
Vector fx queries
In Memory
Parallel across cores



Server-side rendering lets you visualize faster

- Tried rendering on client first
- Client is for interacting
- Server is for processing and rendering
- Images are shipped from server to client
- Data brushing is handled by shipping clicks to server



Client

Unknown hardware
DOM not really scalable
Chrome browser
HTML+JS+JQuery+CSS

Server

Known hardware
Network or cloud configurations
Vector functional language
Column store

A few numbers…

- **Rendering 1 million points**
  - Client: JS, D3, SVG, Canvas objects ~22 seconds (Apple)
  - Server: PNG shipped to client ~ 4 seconds (Vanilla Amazon cloud)

- **Rendering 10 million points**
  - Client: JS, D3, SVG, Canvas -- Browser Screen of Death (Apple)
  - Server: PNG shipped to client ~28 seconds (Vanilla Amazon cloud)

- **Rendering points using binning**
  - Rendering 1 million ~ 1 second (Vanilla Amazon cloud)
  - Rendering 10 million ~ 4 seconds (Vanilla Amazon cloud)
  - Rendering 100 million ~22 seconds (Internal 4 cores)
  - Rendering 500 million ~116 seconds (Internal 4 cores)

Apple 2.8 GHz Intel Core 16 Mb RAM
Vanilla Amazon cloud (Ottawa – Virginia)

Leverage techniques that encourage pattern finding and matching

Pattern matching techniques already exist and can be applied to big data

- **Huge range of techniques already exist**
  - Aggregation (e.g. count, sum, average, min, max, etc)*
  - Summarization (e.g. keyword extraction)*
  - Clustering (e.g. k-means, graph/link analysis)*
  - Anomaly detection (e.g. outlier, change, deviation, graph/link analysis)*
  - Association rules (e.g. shopping basket)

- **Many of these techniques can:**
  - leverage our ability to pre-attentively process data
  - be performed at scale 'on-the-fly'
  - be applied to different visualization types (e.g. bar, line, box plots)
  - replace or be augmented by visualization (e.g. page rank analysis)

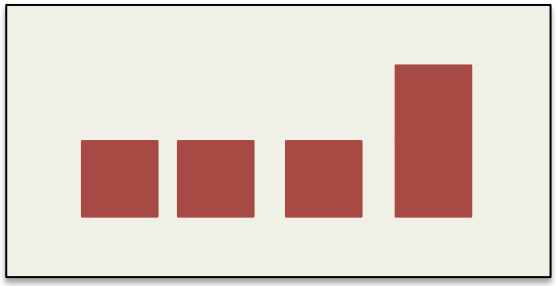
Summarization Techniques for Visualization of
Large Multidimensional Datasets Technical Report TR-2005-35
S. M. Kocherlakota, C. G. Healey

Pattern matching leverages our ability to pre-attentively process information

- **Huge body of work on pre-attentive processing (Bertin, Mackinlay)**
- **Without paying attention, people notice certain visual attributes**

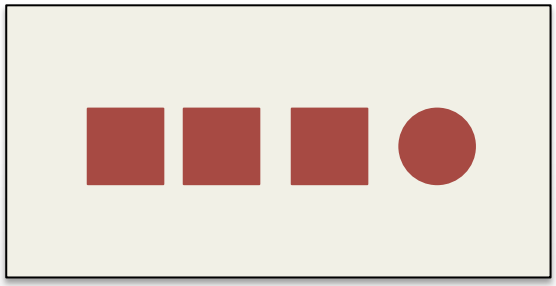
Quantitative Data

1. Position
2. Length
3. Angle (slope)
4. 2D Area/ 3D Volume (size)
5. Density (opacity)
6. Color Saturation
7. Color Hue
8. Texture (not applicable)
9. Connection (not applicable)
10. Containment (not applicable)
11. Shape (not applicable)



Length

Last item is twice as big as the others



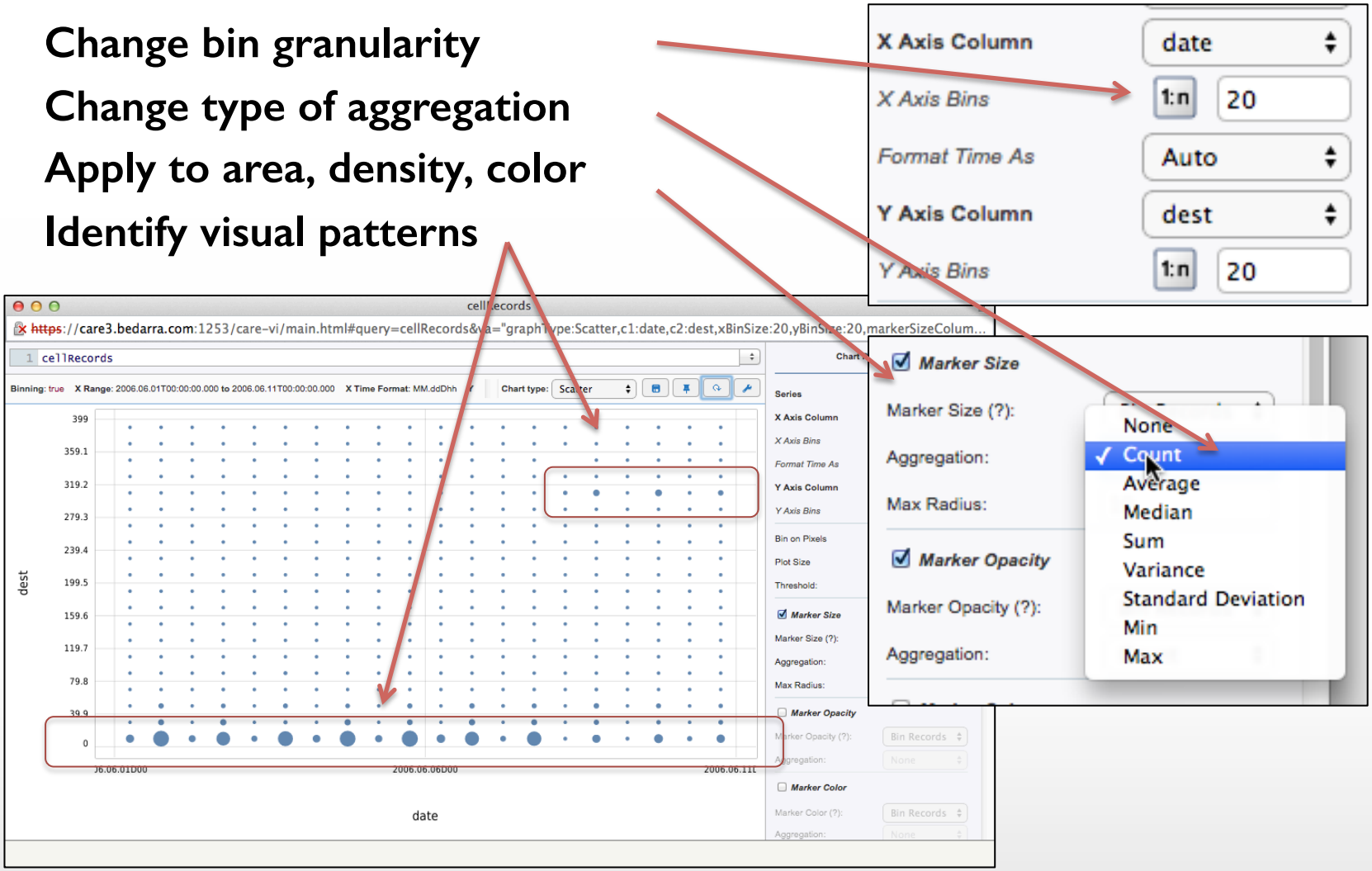
Shape

Last one is different but cannot tell how it is different

Automating the design of graphical representations of relational information J.D. Mackinlay 1986
High-speed visual estimation using preattentive processing, C.G. Healey, K.S. Booth, J.T. Enns 1996
Seminology of graphics, J. Bertin, 1967

Aggregation allows you to identify visual patterns

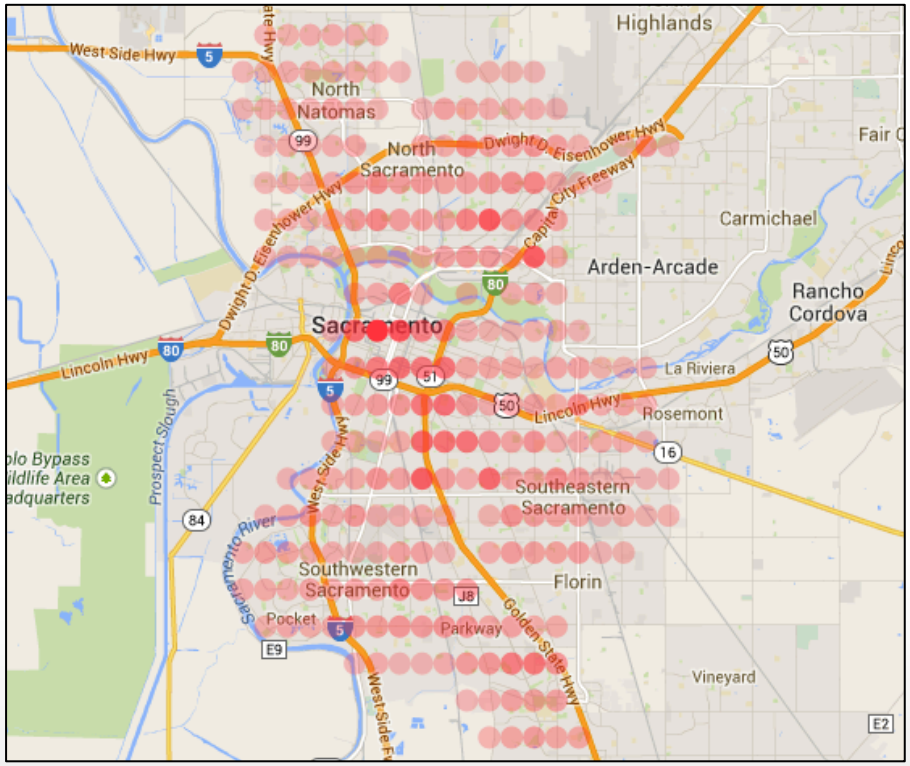
- Change bin granularity
- Change type of aggregation
- Apply to area, density, color
- Identify visual patterns



Applied to geomap

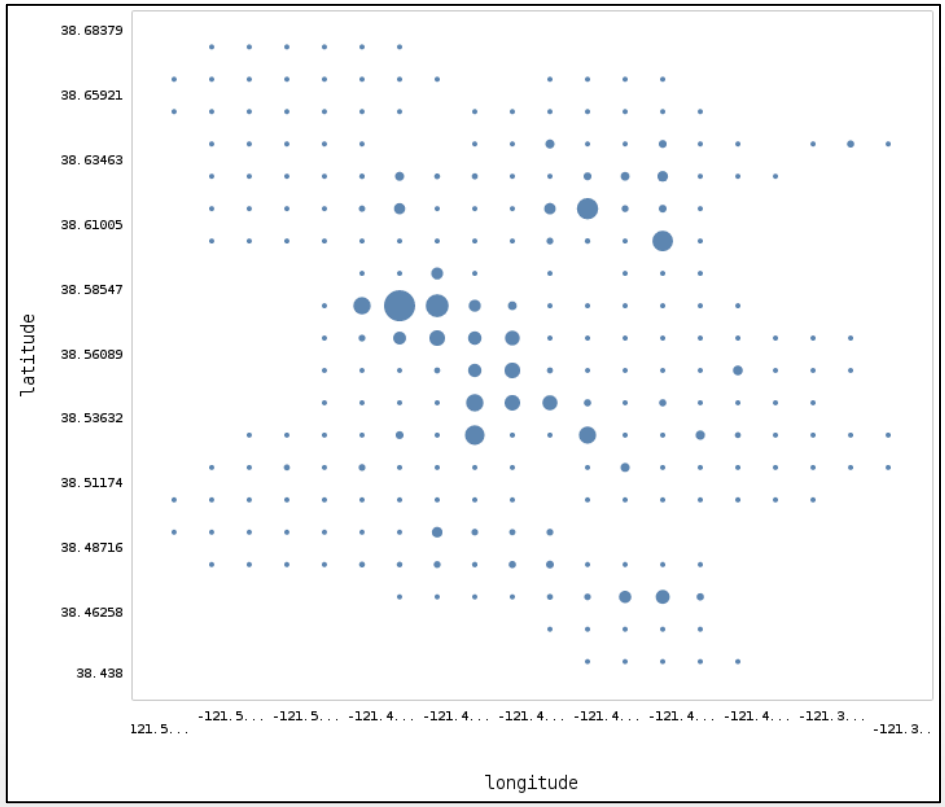
Where you don't want to live in Sacramento

- **10M crimes (20 x 20 bins rendered ~3 seconds)**
- **Bin count applied to opacity attribute**



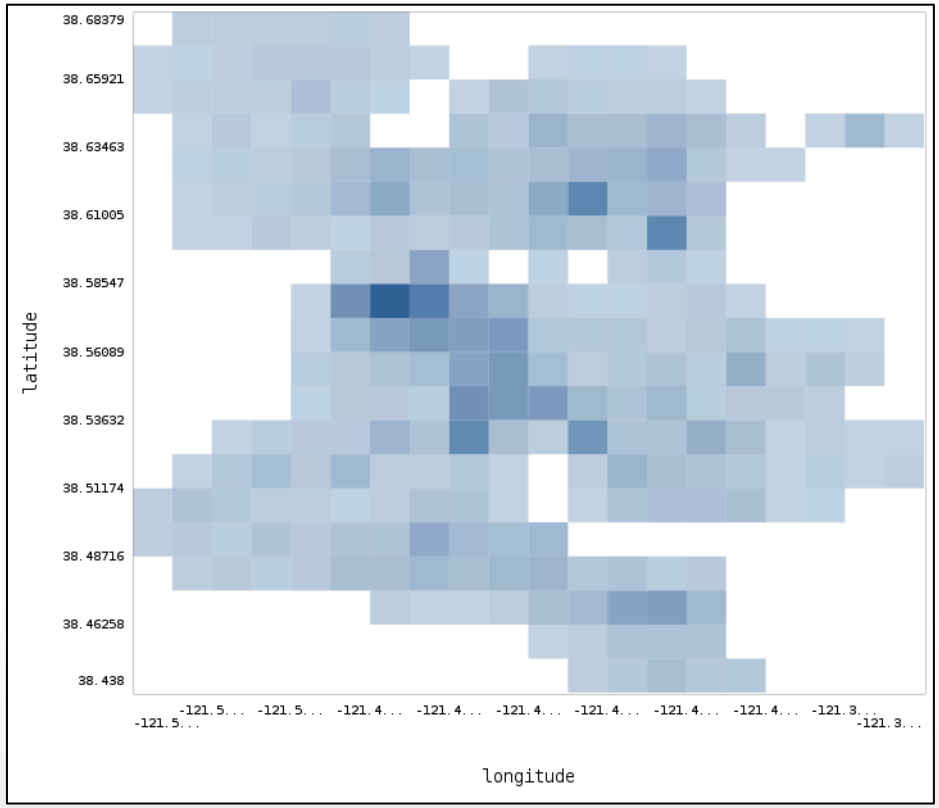
Applied to scatter

- 10M crimes (20x20 bins) renders in ~2 seconds
- Bin count applied to size attribute



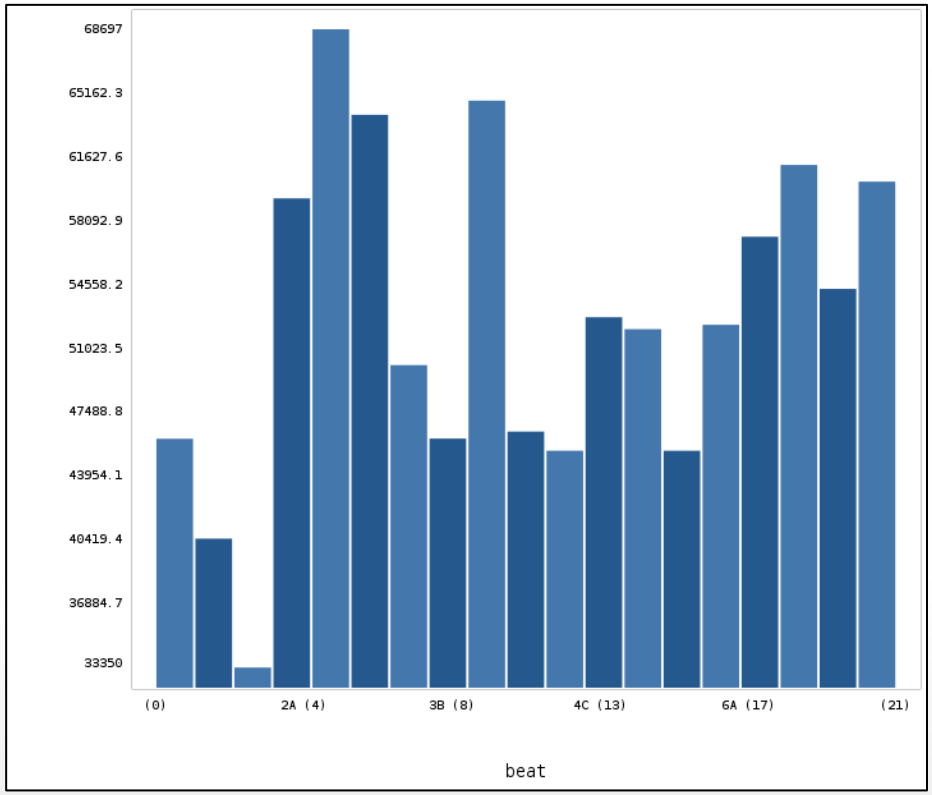
Applied to heatmap

- 10M crimes (20 x 20 bins rendered in ~2 seconds)
- Bin count applied to opacity attribute



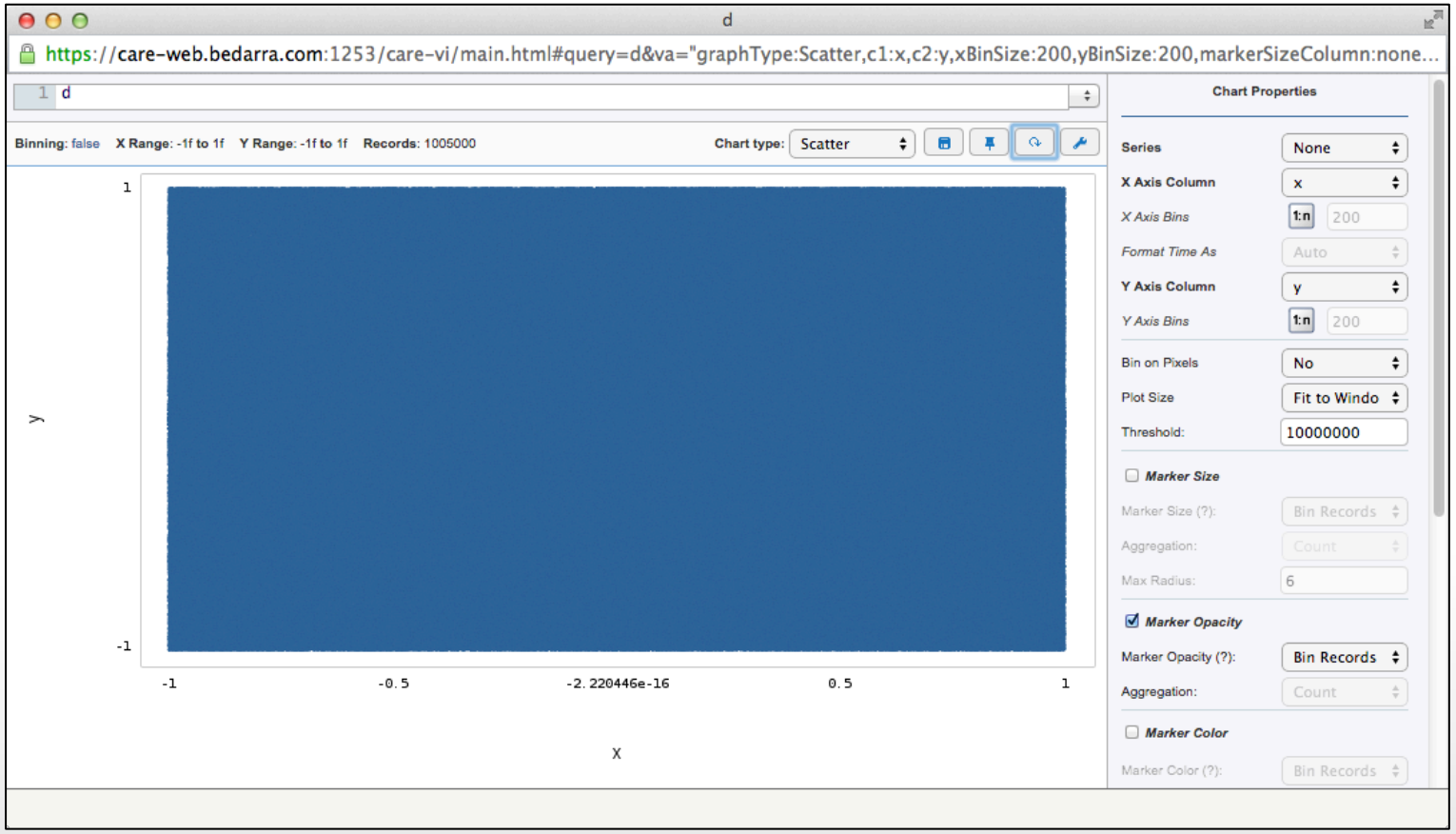
Applied to histogram

- 10M crimes by police zone (19 bins rendered in ~2 seconds)
- Bin count applied to length attribute



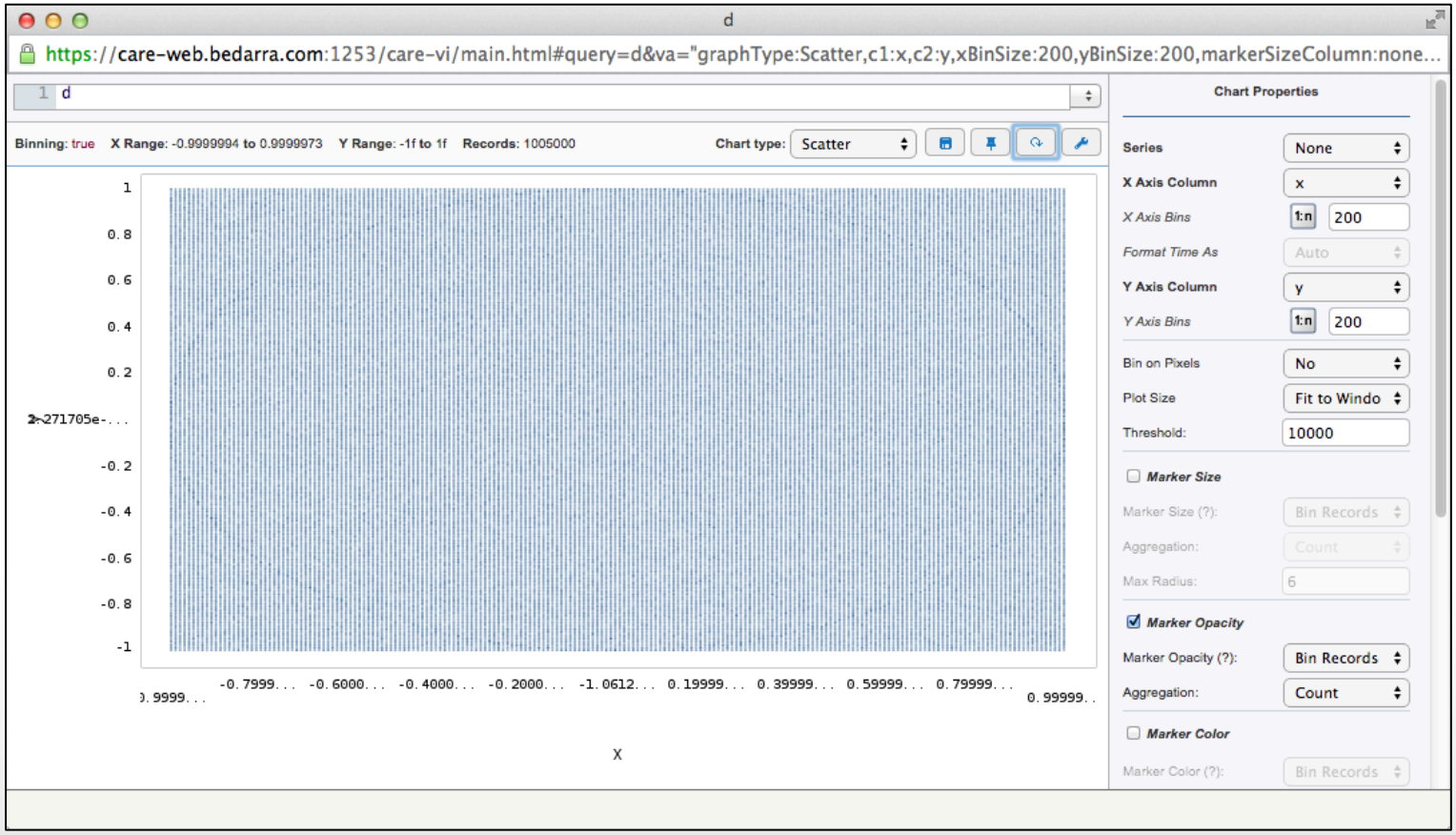
Plotting ‘all the data’ can actually hid the data

- 1,005,000 points between zero and one using opacity attribute
- Plotting all the points took ~5 seconds and produced no pattern



Plotting by bins actually reveals the pattern and it's faster

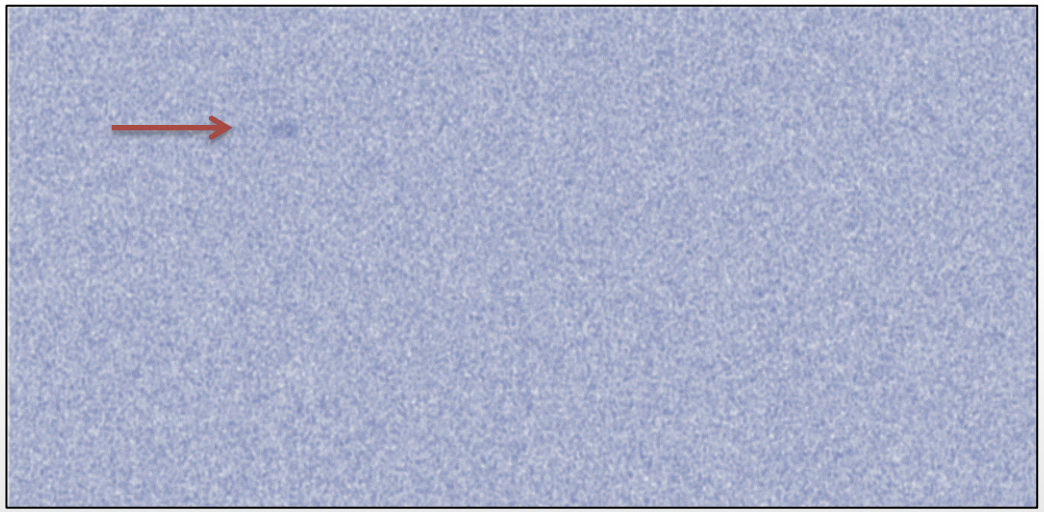
- Same graph plotted with 200x200 bins in ~2 seconds
- Now you can see the pattern



Patterns might even let you find a needle in a haystack

- **Distributed patterns appear to hold at 2000:1,000,000**

- **Concentrated patterns appear to hold at 250:1,000,000**

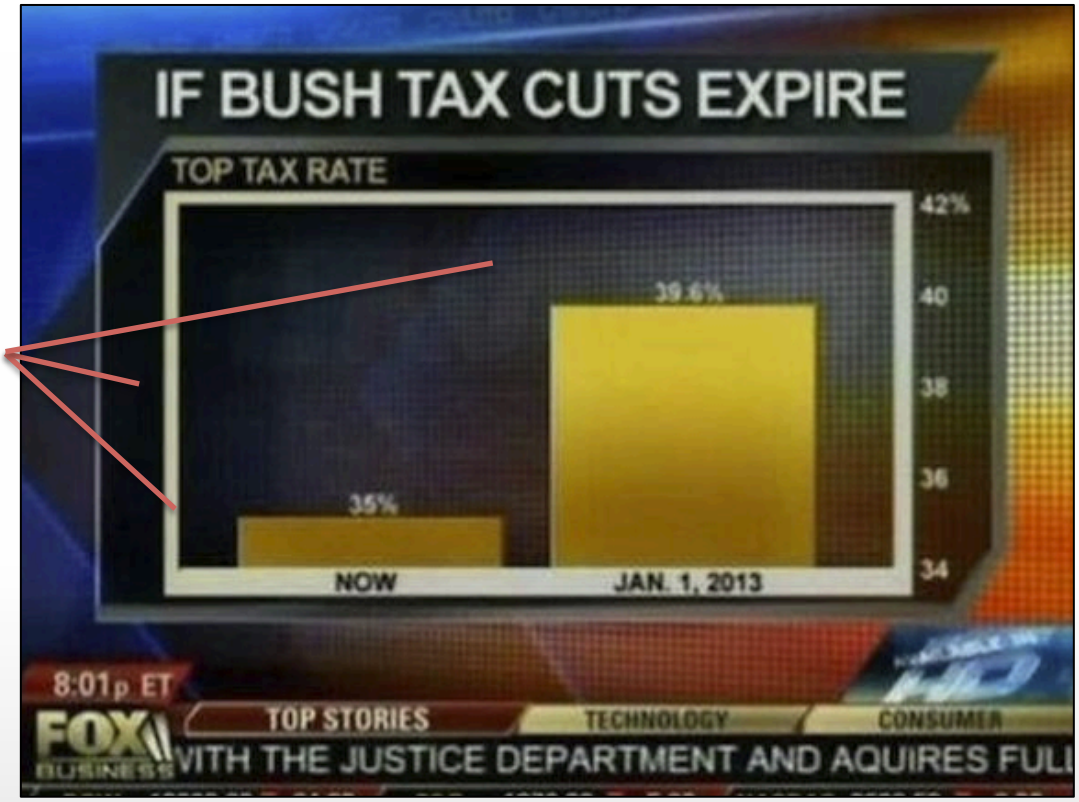


Allow users to ‘bend’
the rules of good visual design

‘Analysing’ rules may need to be different than ‘viewing’ rules

- **Tufte developed a great set of design rules for *viewing graphs***
- **This terrible graph violates several of his rules**

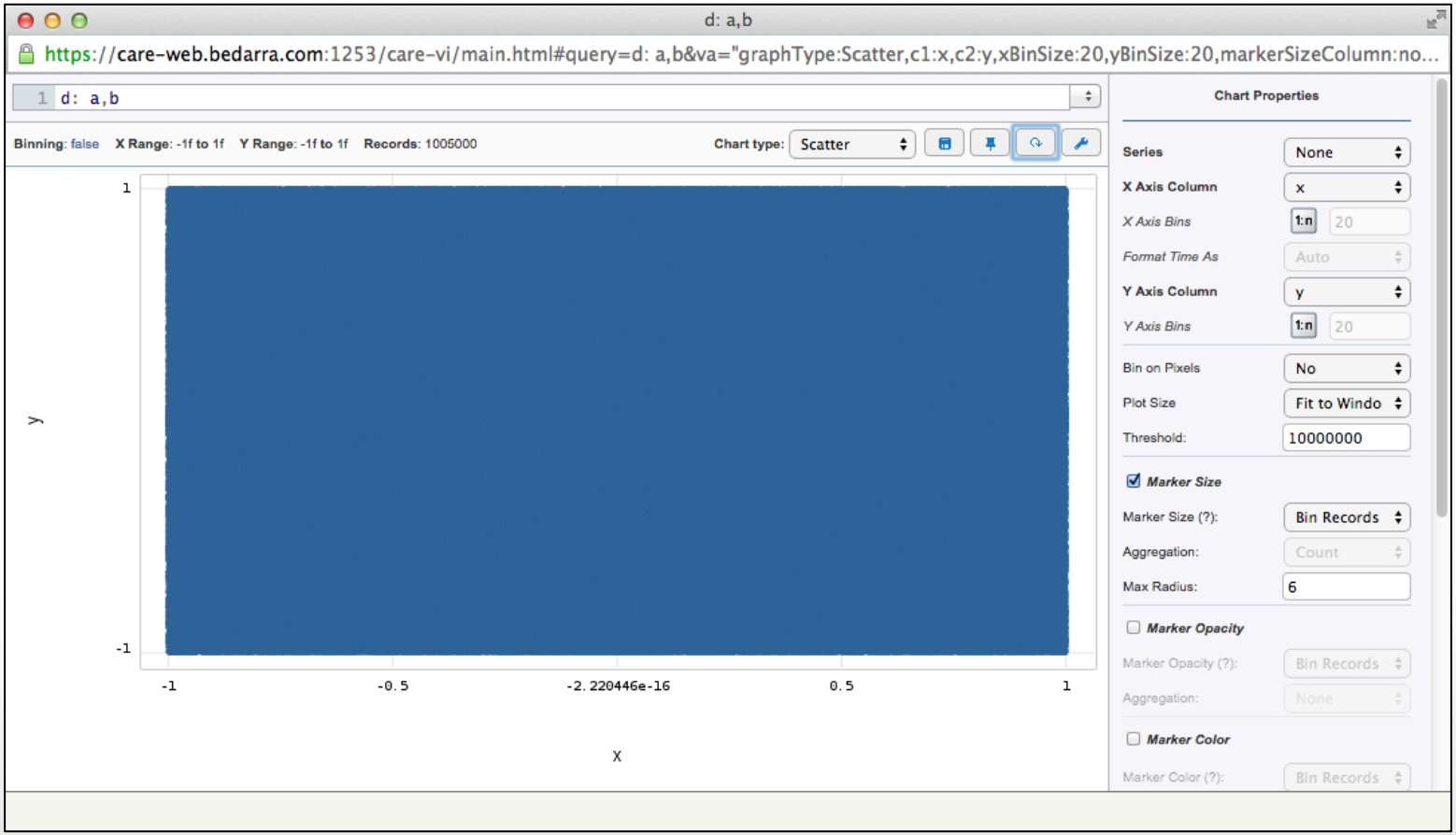
“Erase non-data ink”



“Tell the truth with your data”
and
“Show your data”

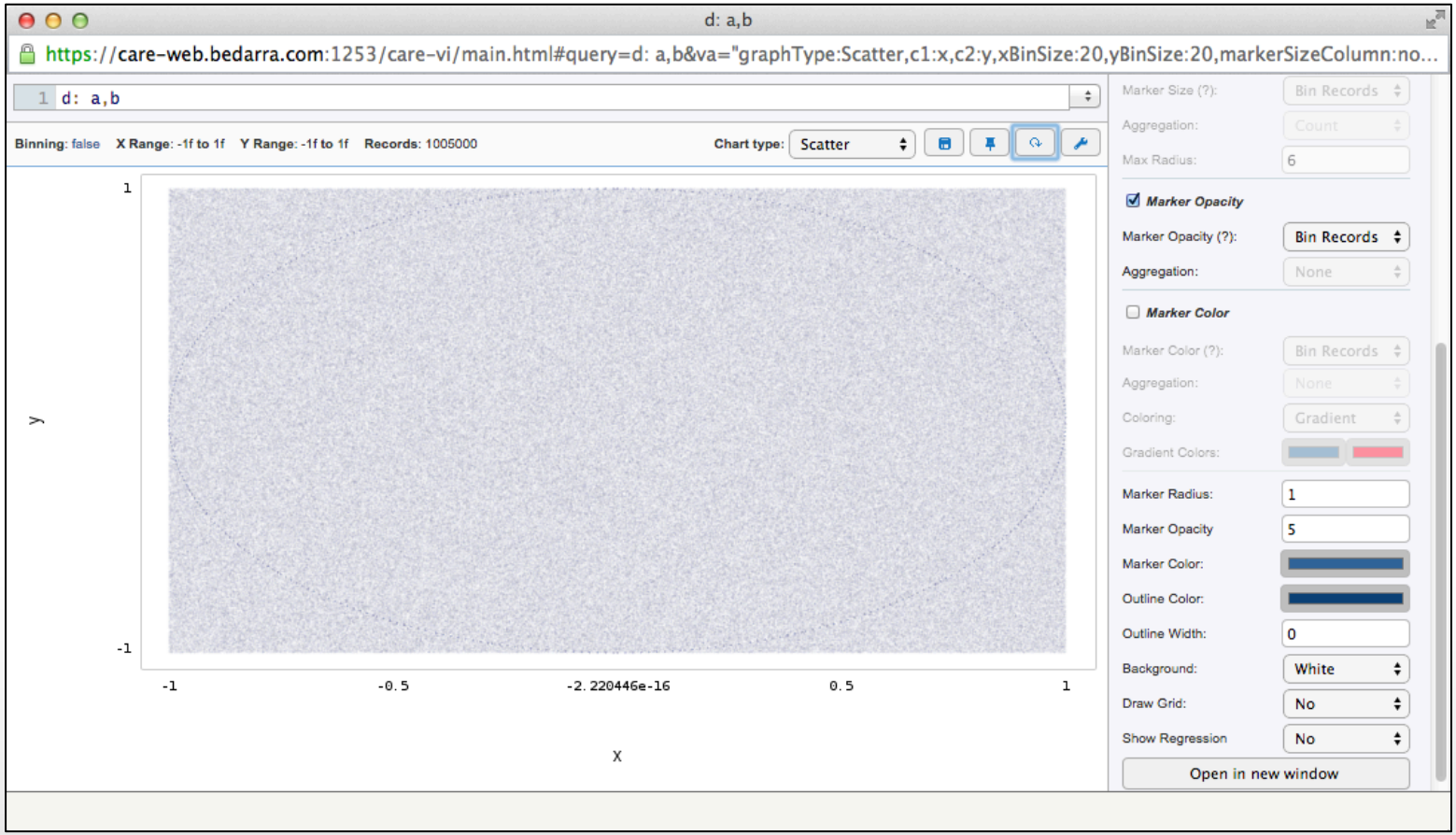
However… at scale sometimes "showing the data" can hide the data

- One million points in ~5 seconds
- No pattern



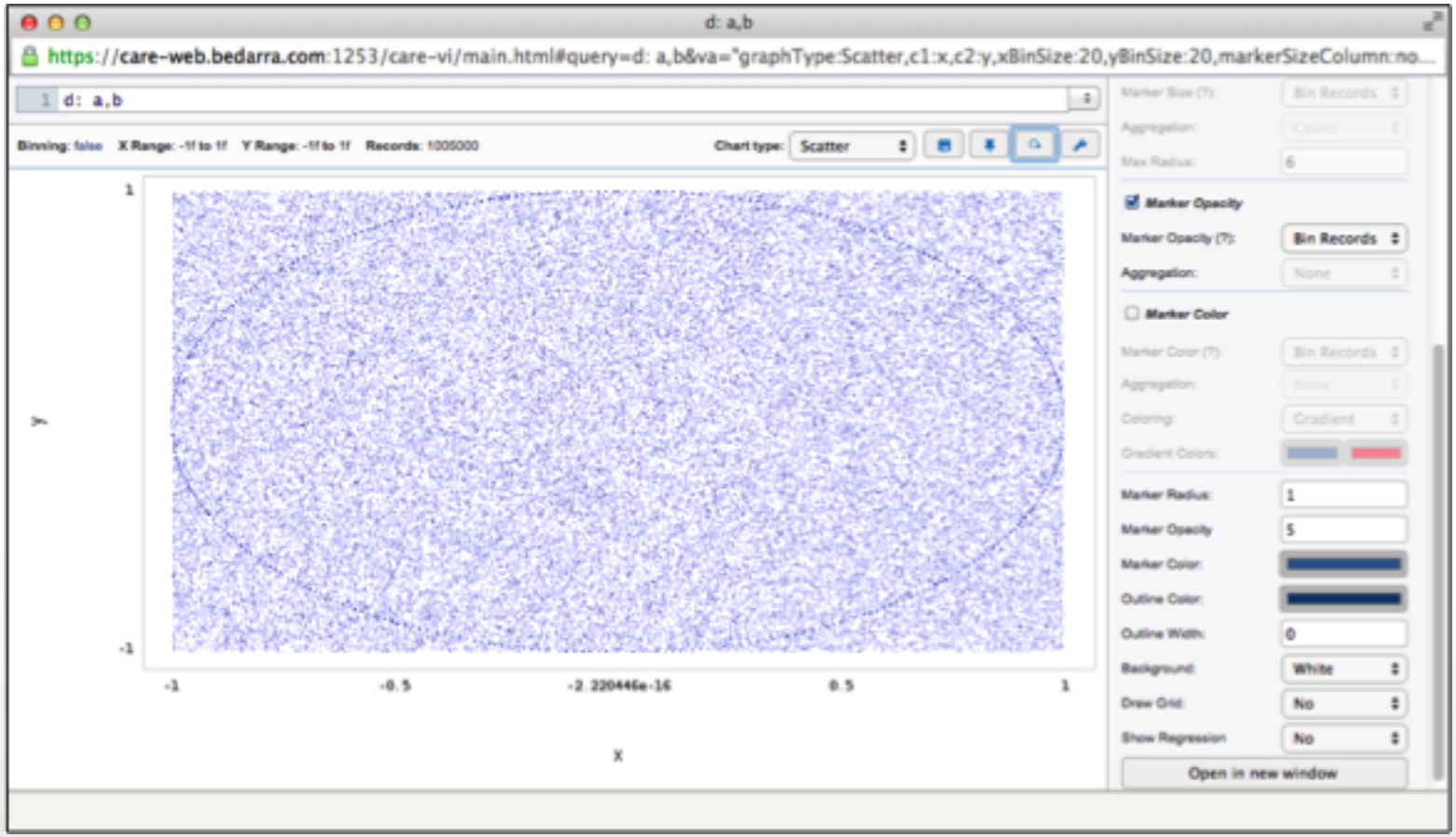
Not ‘showing all the data” can reveal a pattern

- One million points in ~5 seconds
- Rendering at 5% opacity yields pattern



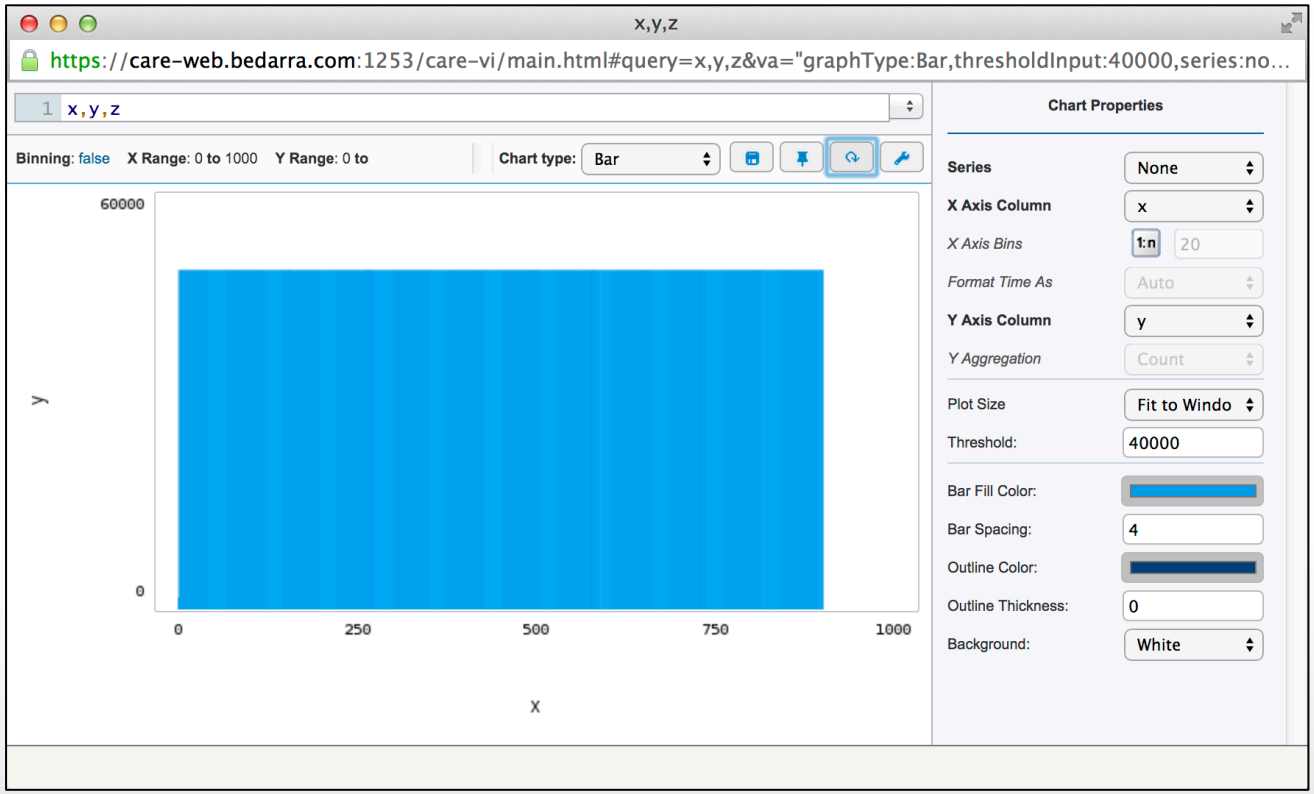
Not showing all the data, and it's ugly, but it works!

- Manipulating the RGB levels removes points and enhances pattern
- Can be done completely in client using JS image libraries



Sometimes 'telling the truth with your data' provides little value

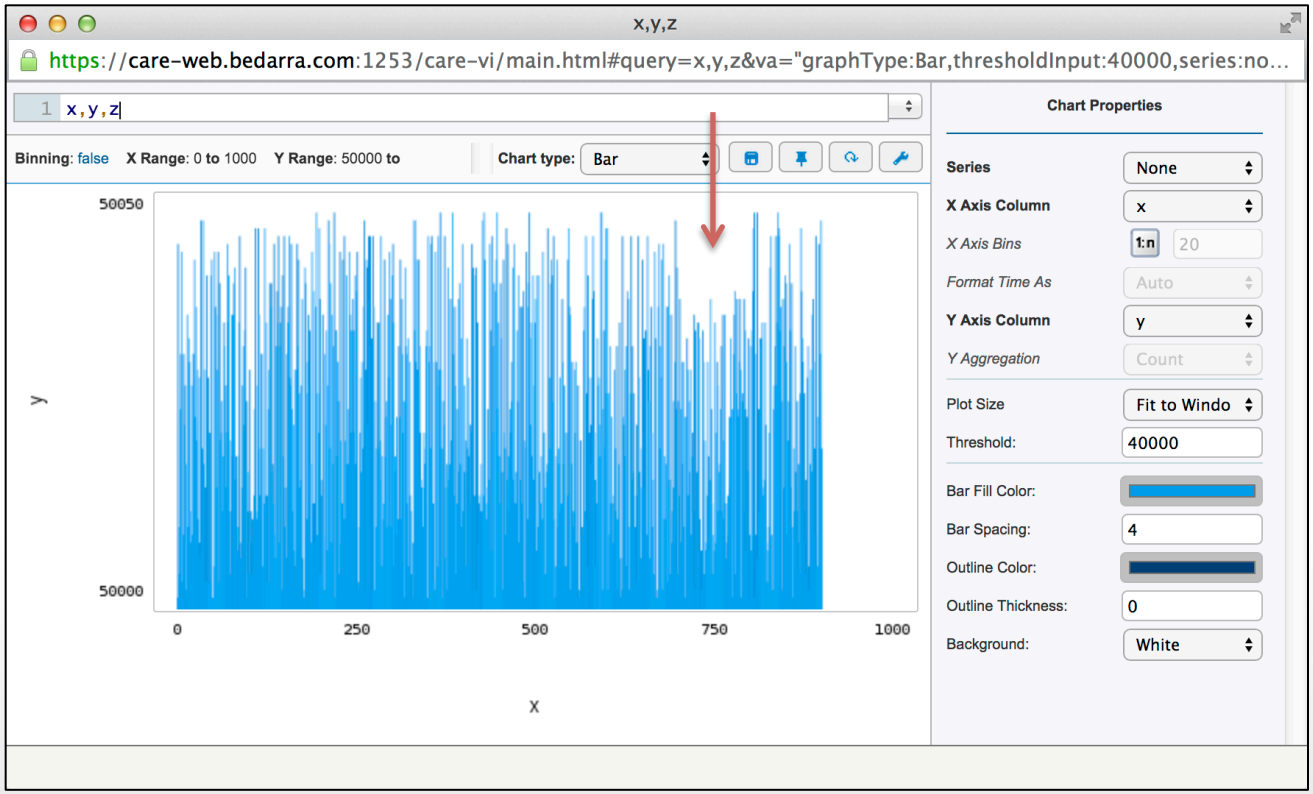
- The Rule: Plot Y axis from zero so data is not exaggerated
- 1000 Y bars between 50,000 and 50,050*
- When plotted from zero, this plot is kind of useless



* One Y value is plotted at 0,0 to trick our renderer

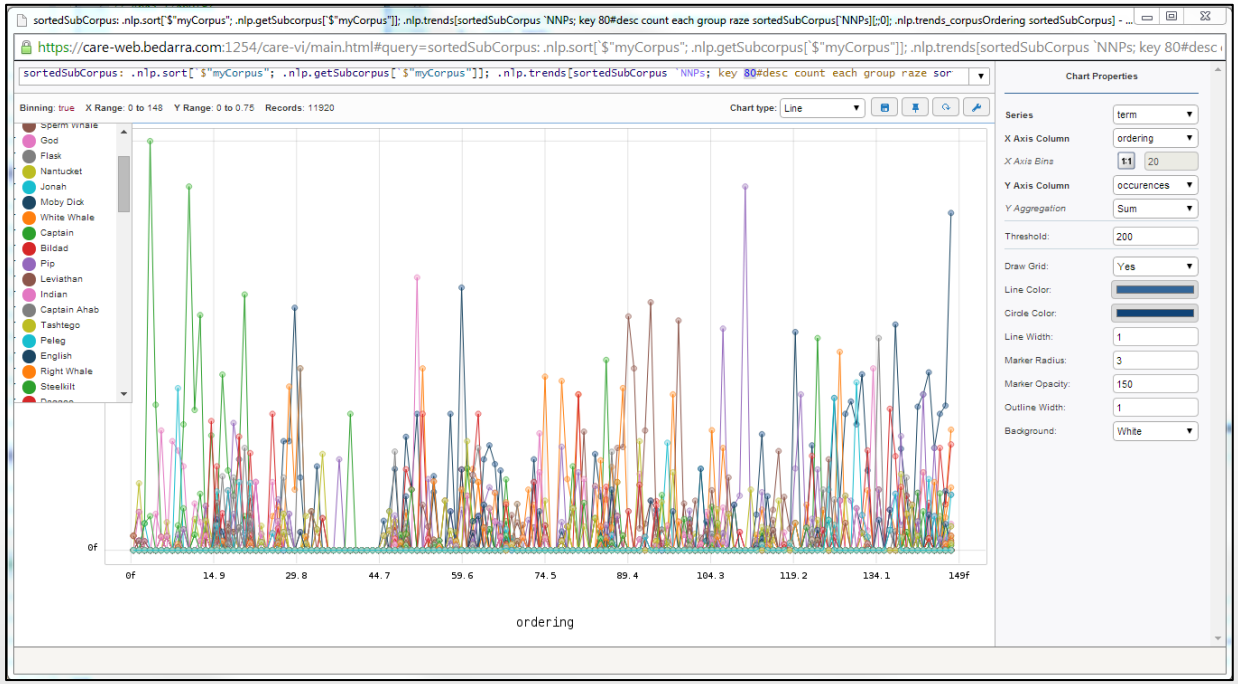
Plotting from min and max values at least shows range

- Same points plotted from min (50,000) and max (50,050) range
- When plotted by range, we can see a bit of a gap
- Let the analyst decide what is important



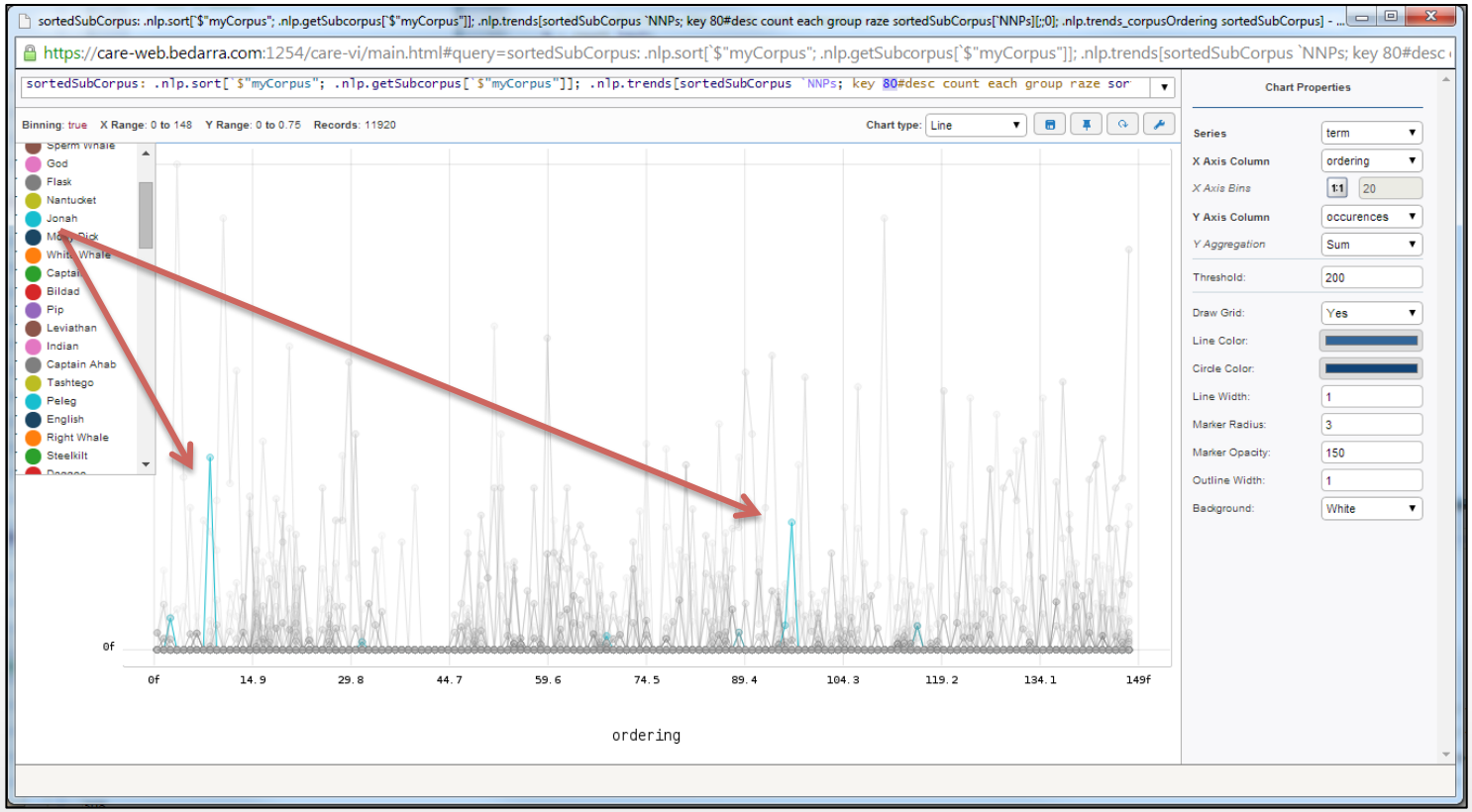
Sometimes ‘too much data’ can actually show you things

- **The rule: Don’t use more than 7 +/- 2 legend items**
- **However, this example contains 80 legend items**
  - Occurences of top 80 keywords in Moby Dick (Y) by chapter (X)
  - Easily see chapters are largely oriented on single keywords
  - Easily see a few chapters contain no keywords



Too much data can always be managed

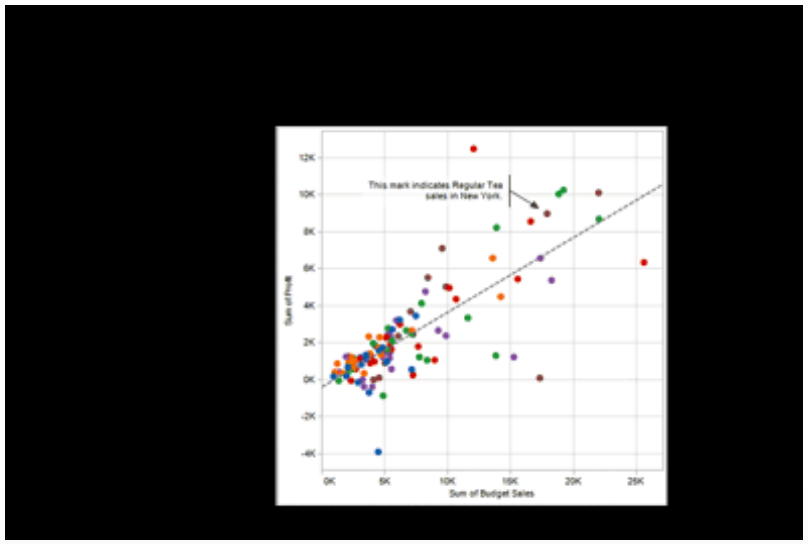
- Filtering out noise is handled interactively (e.g. rollovers)
- Example of highlighting Jonah's importance across chapters



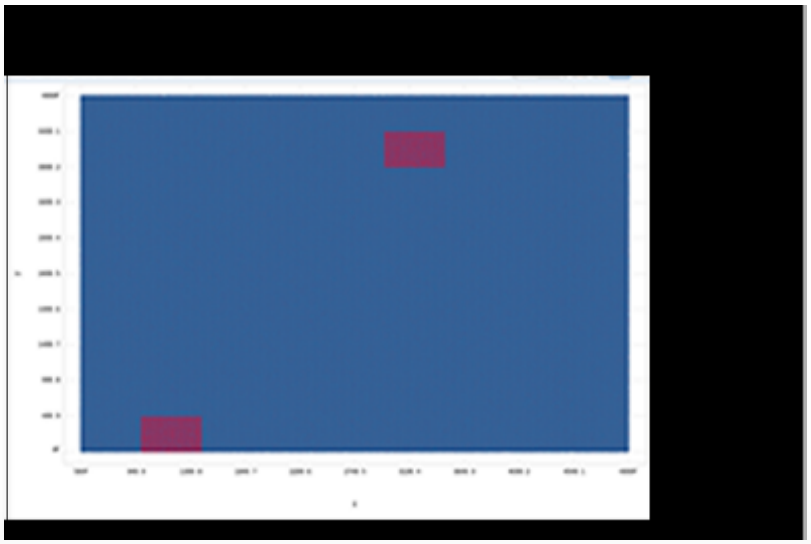
Maximize the plot-to-ui ratio

Striking the balance between ui control area and plot area

- Big data plots often require lots of pixels for pattern identification
- But small data UIs allocate quite a bit of the screen to ui controls
- Need to strike better balance between 'ui controls' and 'plot'



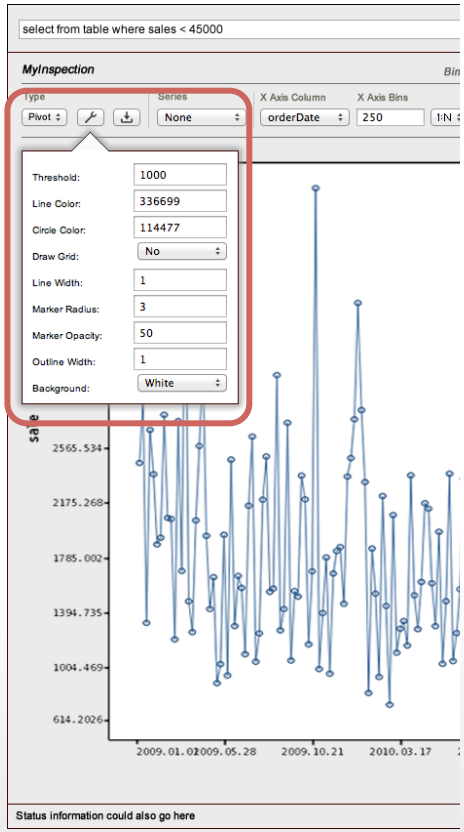
Most of this interface allocated to ui controls



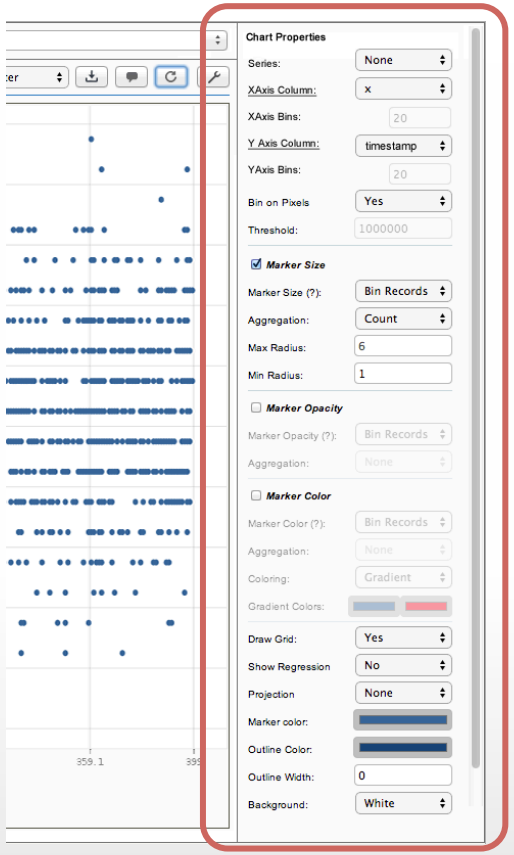
Most of this interface allocated to plot

Controls are shown but scroll so we can maximize plot size

- Analysts use controls constantly
- Tradeoff between control accessibility and plot size



Early iteration: Hidden controls

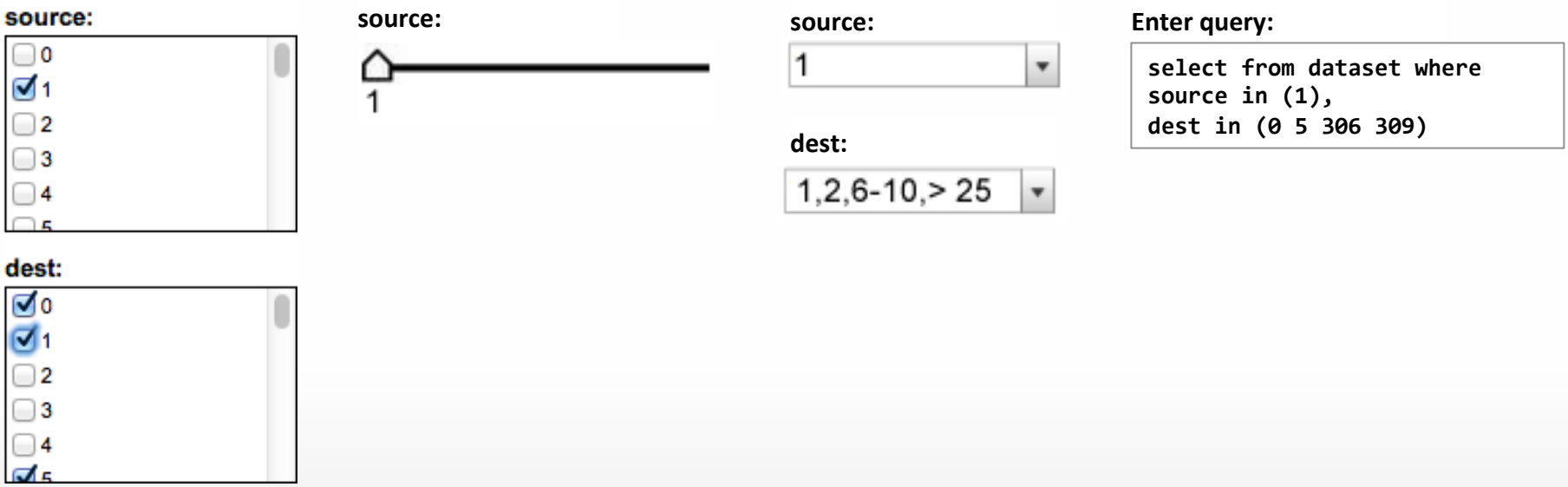


Current interation: Scrolls controls

Provide interaction control support data complexity

Rethinking standard UI controls at scale

- Standard controls work for few columns, small ranges and iterations
- Tedious with more columns, broader ranges, and lots of iterations
- Tradeoff between ease of learn and easy to use



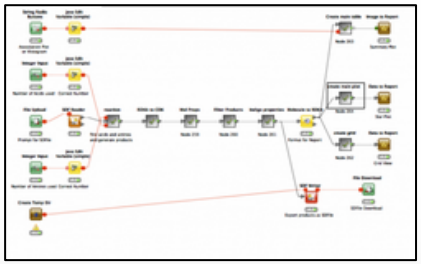
Easier to learn
Harder to use with lots of columns, ranges, iterations

Harder to learn
Easier to use with lots of columns, ranges, iterations

Rethinking standard UI interfaces at scale

- **Same issue with general query interfaces**
  - Node & link, tree, spreadsheet, text interface styles
- **Opted for spreadsheet style as an 80% solution**
  - Harder to learn but easier to express complex queries
  - Handles the majority of queries

Node and link style



Spreadsheet style

column	beat	grid	district	district
where	=`3C	=1115		
or	=`2A			
or			=1	
or			=2	
or			>3	<4

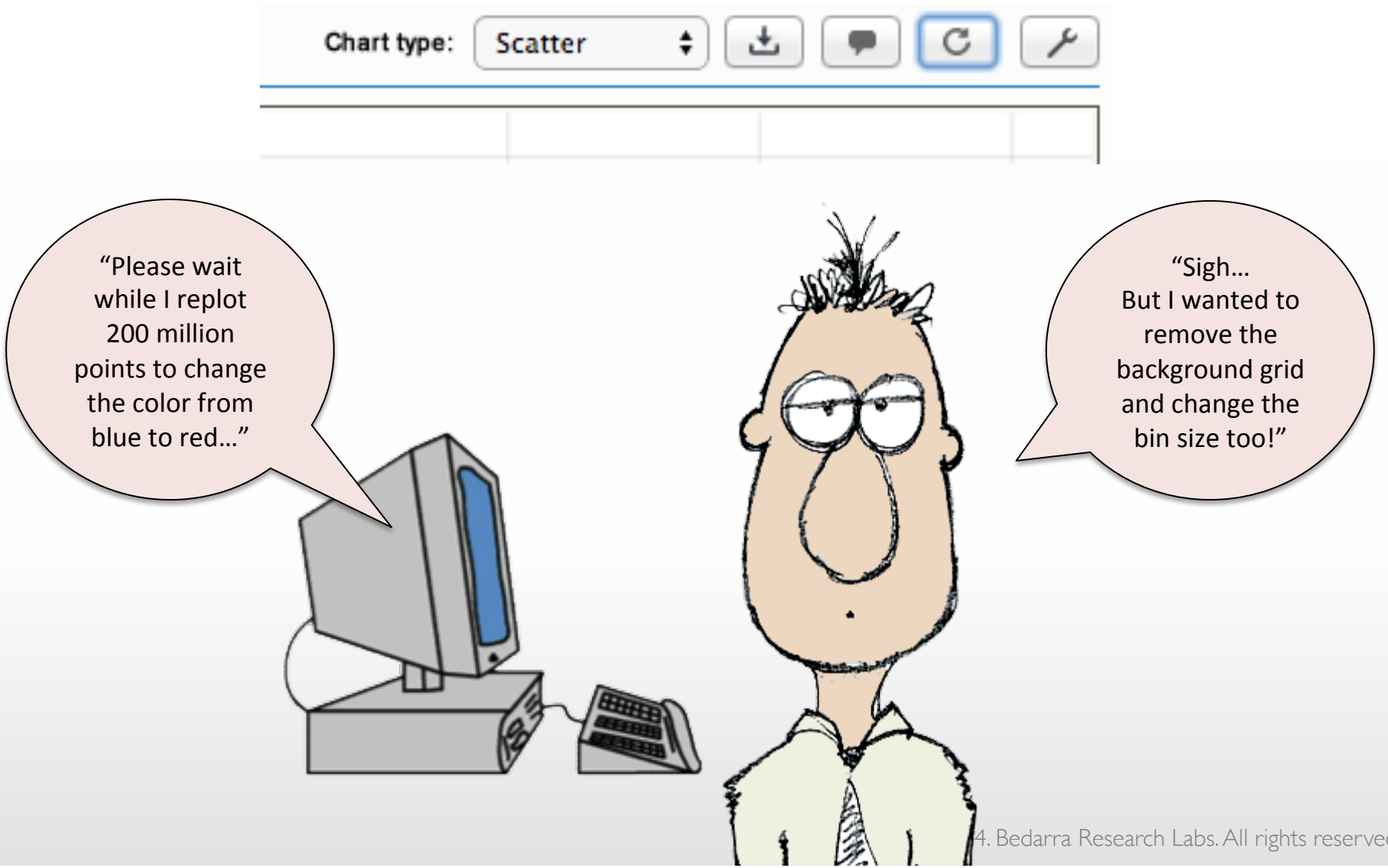
Text style

```
select from crime where ((beat
=`3C) and (grid =1115)) or ((beat
=`2A)) or ((district =1)) or
((district =2)) or ((district >3) and
(district <4))
```

Easier to learn
Less expressive and flexible
Real estate issues

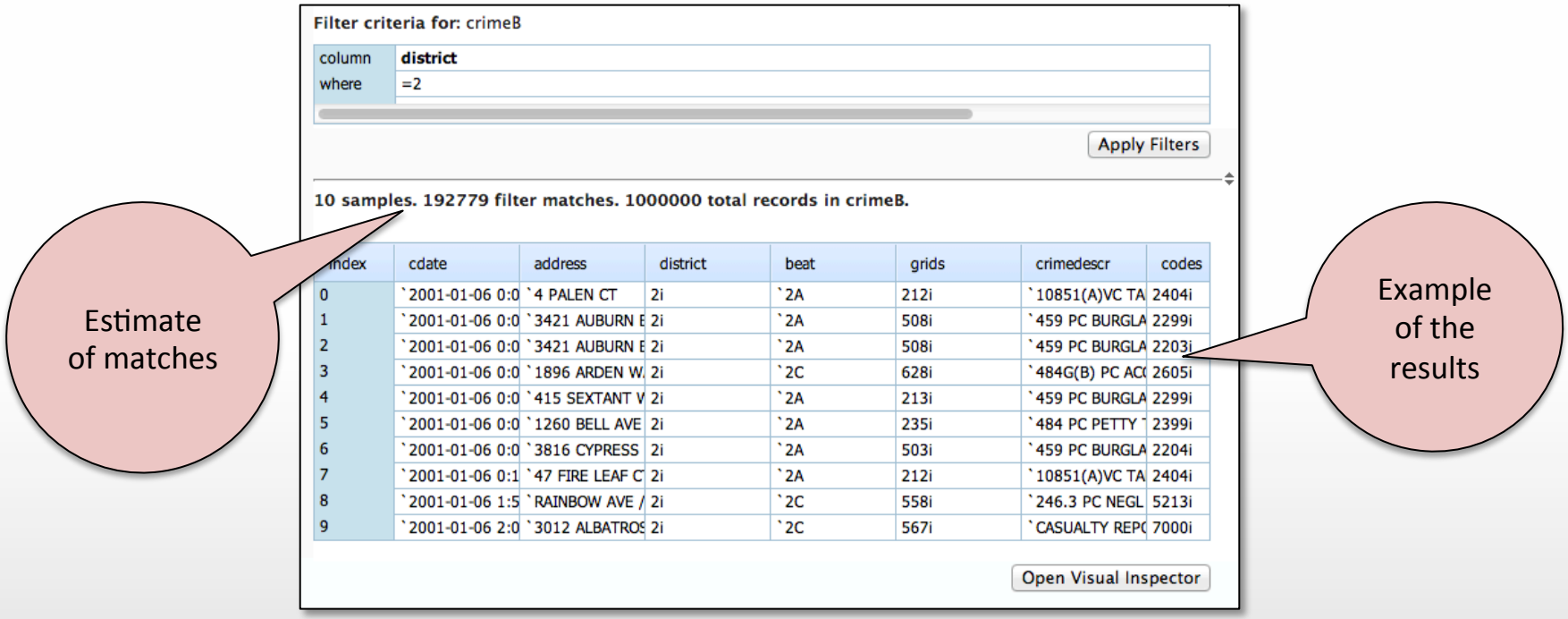
Harder to learn
More expressive and flexible
Syntax issues

Small data interfaces tend to auto update…
Big data interfaces need to manually update



Small data interfaces tend to auto execute…
Big data interfaces need to manually execute

- Small data UIs execute operations automatically
- Big data UIs should execute operations manually
- Big data UIs should provide ‘examples’ and ‘estimates of results’ first
- Same principles can be applied to any large operation

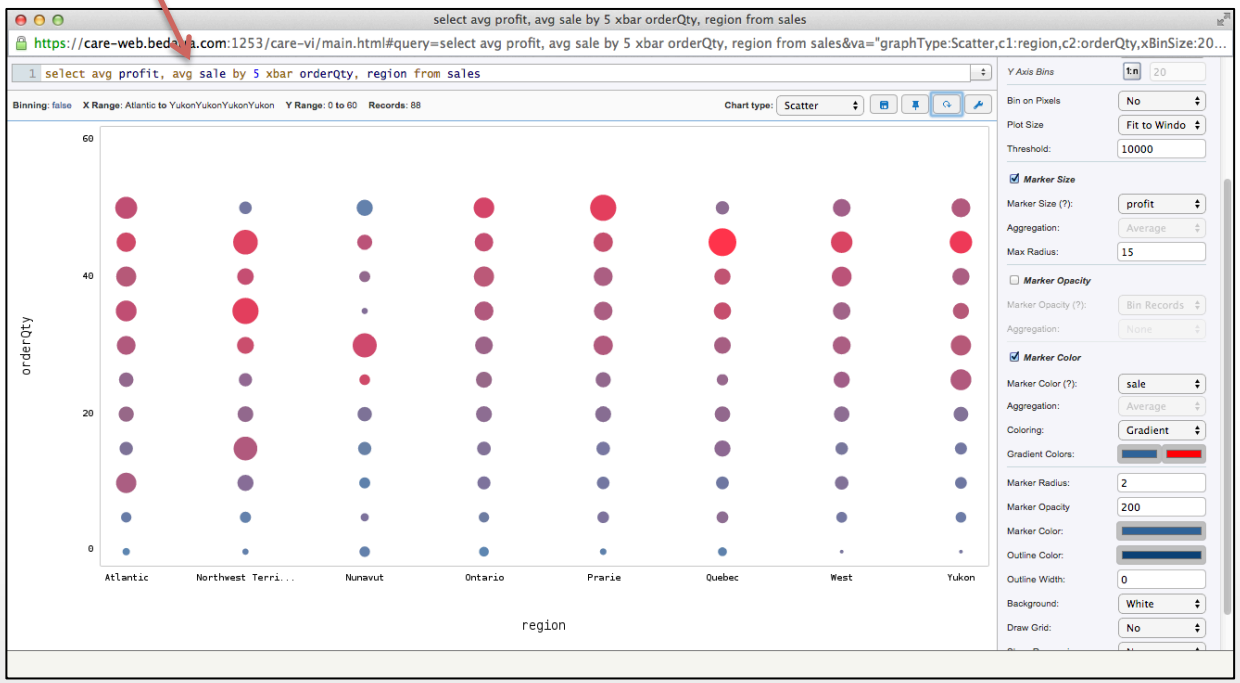


Make scripting a primary task

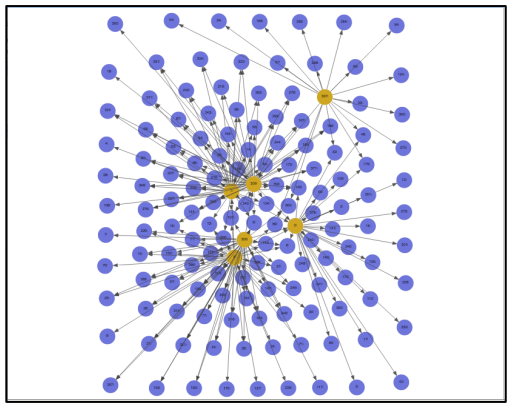
A little scripting can go a long way…

- Via interface: 500 million 20x20 bins ~116 seconds
- Via scripting: 500 million 20x20 bins ~27 seconds

```
1 select avg profit, avg sale by 5 xbar orderQty, region from sales
```



Scripting can make the difference between the right and wrong answer



1 Visual analysis allowed us to identify key players

```
// Read a small dataset and replicate it up to 1 million records
cellRecords : 1000000 # ("IIZII"; enlist ",") 0: `:/CARE3/ExampleData/vast/cellRecords.csv;

// Create a 400x400 matrix of called and calling parties
// Weight each set of calls in the matrix and normalize the values
// Perform page rank analysis by passing weights to most important nodes
// See page 408 of http://www.cs.cornell.edu/home/kleinber/networks-book/networks-book-ch14.pdf
dict: 1 _ exec dest by source from cellRecords;
matrix : value @[400#0;;+;1] each dict;
colStoichastic: flip matrix %' sum each matrix;
v: (count colStoichastic) # 1f;
rankings    : {x % sum x} {x mmu y}[colStoichastic]/[v];
rankedIDs   : (key dict)!rankings;
keyCellPhones: key desc rankedIDs;

// call graph of the most important people as destinations
select distinct source, dest from cellRecords where dest in 15#keyCellPhones;
```

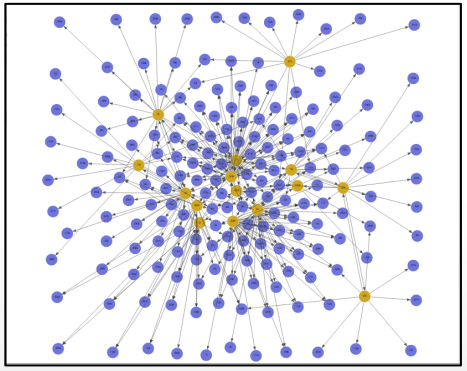
```
5 200 1 0 2 3 20 61 309 137 19 306 15 92 360
```

3 Page rank analysis revealed additional key players

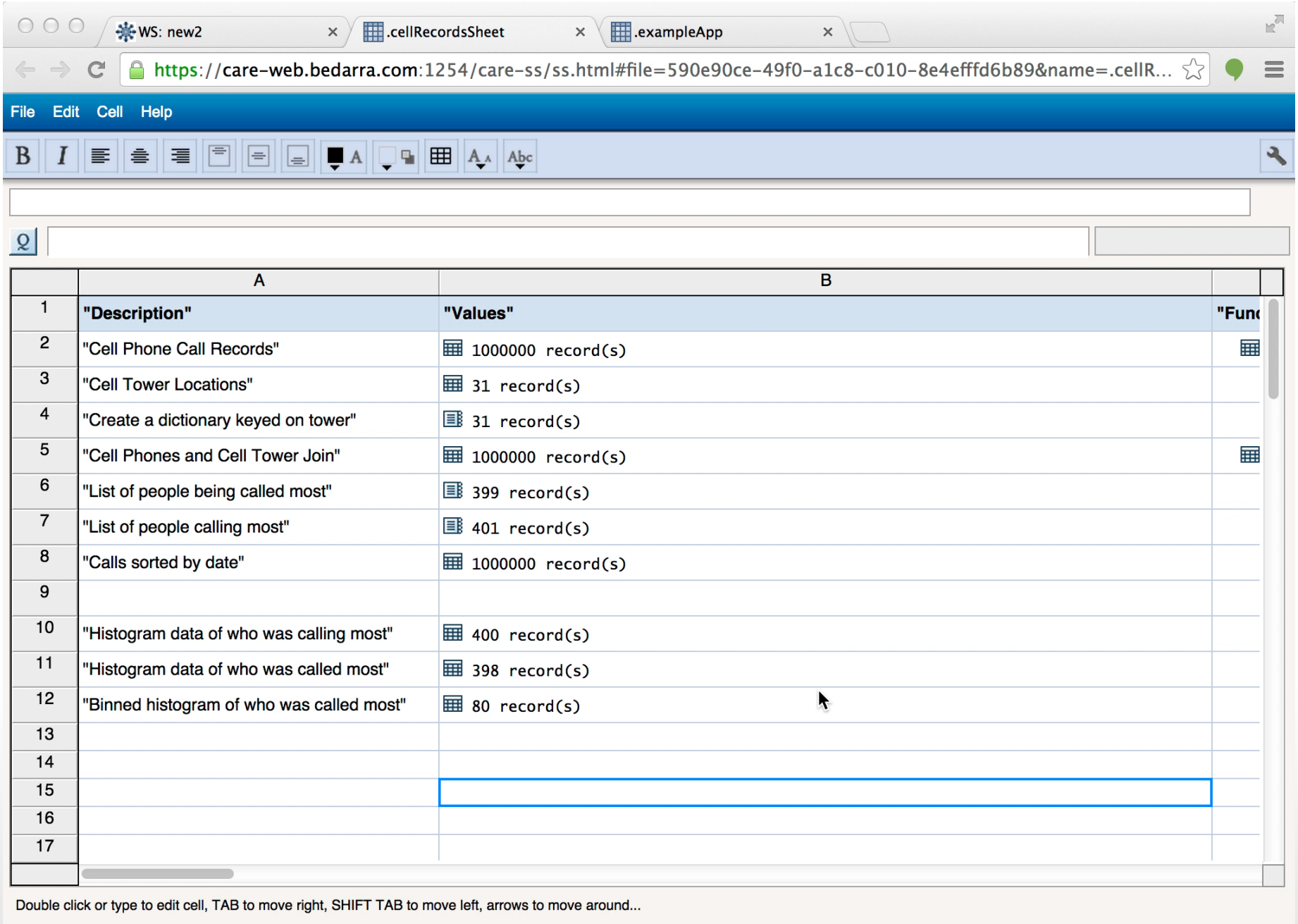
```
// Load/Generate the data
cellRecords : 1000000 # ("IIZII"; enlist ",") 0: `:/CARE3/ExampleData/vast/ce

// Make a dictionary where the keys are the IDs of people,
// and the values are the IDs of the phones for each call they make
// With the 1 _ to get rid of null records
dict: 1 _ exec dest by source from cellRecords;

// Make a matrix where each row shows the number
matrix : value @[400#0;;+;1] each dict;

// Make the matrix undirected (though still weig
clusters: .nlp.findClusters matrix + flip matrix

// Sort by cluster size
clusters idesc count each clusters;
```

```
7 76 82 171 209 224 316 343
63 72 109 232 297 334 375
5 99 163 250 288 306
200 228 262 281 286 383
50 100 184 210 323 393
13 157 167 219 322
41 107 122 240 314
49 147 274 339 367
57 64 189 315 352
133 141 191 324 346
102 181 293 350 388
212 284 299 317 364
2 130 226 397
8 77 245 289
39 128 340 355
59 104 110 356
120 178 214 363
149 201 249 321
124 186 263 277
```

2 Markov cluster analysis revealed nothing

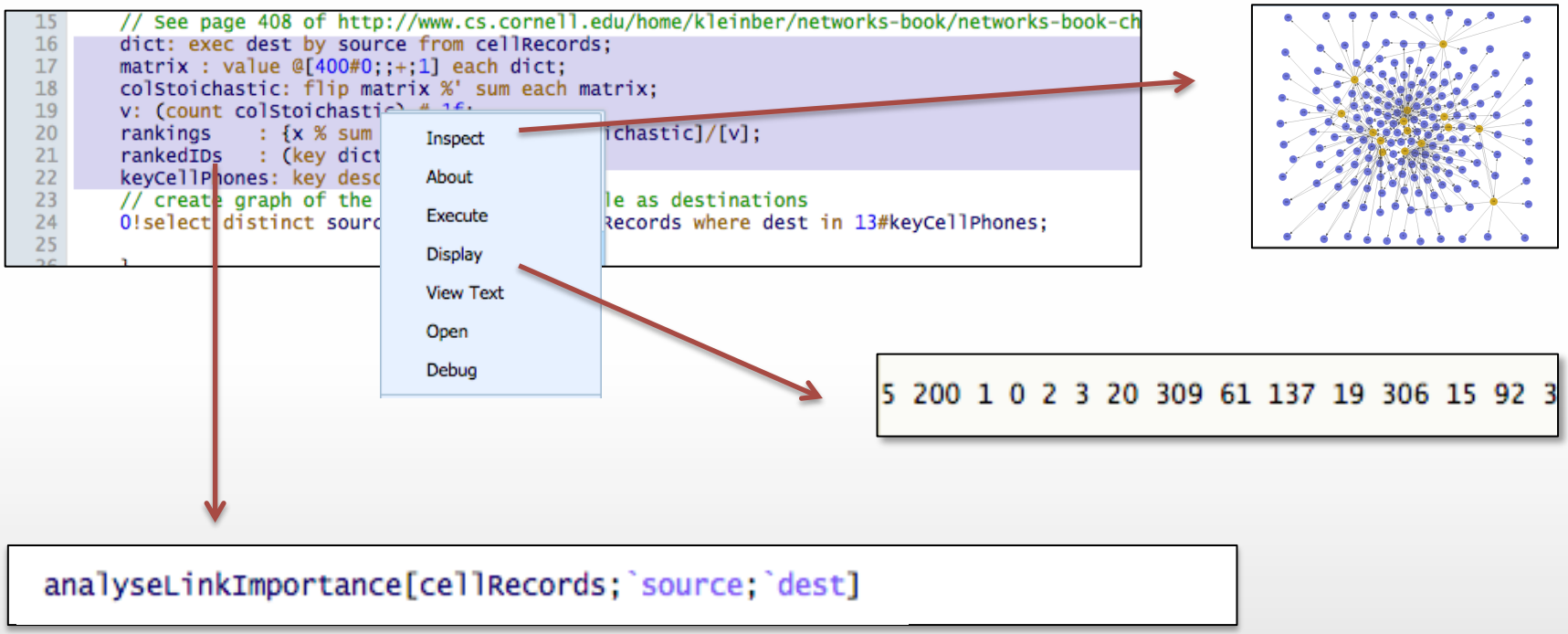


4 Page rank analysis augmented by directed graph visualization



Scripting should support these behaviors

- Allow users to script ‘on-the-fly’
- Provide a method of receiving continuous feedback loop
- Allow users to move freely between scripting and visualizing
- Allow users to create their own languages (e.g. DSLs)



Summary

- **To provide a 'small data' experience at 'big data' scale in terms of response times, interactivity and ease-of-use:**
  - Support integration of analysis and visualization tasks
  - Support an interative, non-linear task flow at scale
  - Leverage server-side computing and rendering
  - Leverage techniques that encourage pattern matching
  - Allow users to 'bend' the rules of good visual design
  - Manage the data-to-ui ratio
  - Ensure interaction controls support data complexity
  - Make scripting a primary task

Thank you!

Doug Talbott
Bedarra Research Labs
doug.talbott@bedarra.com
www.bedarra.com