CSE 442 - Data Visualization

Scalable Visualization

Jeffrey Heer  University of Washington
Session Outline

The Varieties of “Big Data”
Scalable Plotting Techniques
Scalable Interaction
Why Latency Matters
Sampling Methods
The Varieties of “Big Data”
Tall Data

Lots of records
Large DBs have petabytes or more
(but median DB still fits in RAM!)

How to manage?
Parallel data processing
Reduction: Filter, aggregate
Sample or approximate

Not just about systems. Consider perceptual / cognitive scalability.
Tall Data

Wide data

Lots of variables (100s-1000s…)
Select relevant subset
Dimensionality reduction
Statistical methods can suggest and order related variables
Requires human judgment
<table>
<thead>
<tr>
<th>Tall Data</th>
<th>Wide data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diverse data

- Tall Data: Data with a high number of rows compared to columns.
- Wide data: Data with a high number of columns compared to rows.
- Diverse data: Data that includes various types of information, such as graphs, tables, and text.
How can we visualize and interact with billion+ record databases in real-time?
Two Challenges:

1. Effective **visual encoding**
2. Real-time **interaction**
Perceptual and interactive scalability should be limited by the chosen resolution of the visualized data, not the number of records.
Scalable Plotting Techniques
Data

Sampling

Modeling
Data

Sampling

Binning

Modeling
How to **Visualize** a Billion+ Records

Decouple the visual complexity from the raw data through aggregation.
Bin > Aggregate (> Smooth) > Plot

1. Bin  Divide data domain into discrete “buckets”

*Categories*: Already discrete (but watch out for high cardinality)

*Numbers*: Choose bin intervals (uniform, quantile, ...)

*Time*: Choose time unit: Hour, Day, Month, etc.

*Geo*: Bin x, y coordinates *after* cartographic projection
Bin > Aggregate (> Smooth) > Plot

1. **Bin**  Divide data domain into discrete “buckets”
   
   Categories: Already discrete (but watch out for high cardinality)
   
   **Numbers**: Choose bin intervals (uniform, quantile, ...)
   
   **Time**: Choose time unit: Hour, Day, Month, etc.
   
   **Geo**: Bin x, y coordinates after cartographic projection

2. **Aggregate**  Count, Sum, Average, Min, Max, ...
Bin > Aggregate (> Smooth) > Plot

1. **Bin**  Divide data domain into discrete “buckets”

   *Categories*: Already discrete (but watch out for high cardinality)

   *Numbers*: Choose bin intervals (uniform, quantile, ...)

   *Time*: Choose time unit: Hour, Day, Month, etc.

   *Geo*: Bin x, y coordinates *after* cartographic projection

2. **Aggregate**  Count, Sum, Average, Min, Max, ...

3. **Smooth**  Optional: smooth aggregates [Wickham ’13]
Bin > Aggregate (＞ Smooth) > Plot

1. **Bin**  Divide data domain into discrete “buckets”  
*Categories*: Already discrete (but watch out for high cardinality)  
*Numbers*: Choose bin intervals (uniform, quantile, ...)  
*Time*: Choose time unit: Hour, Day, Month, etc.  
*Geo*: Bin x, y coordinates *after* cartographic projection

2. **Aggregate**  Count, Sum, Average, Min, Max, ...

3. **Smooth**  Optional: smooth aggregates [Wickham ’13]

4. **Plot**  Visualize the aggregate values
# Binned Plots by Data Type

<table>
<thead>
<tr>
<th></th>
<th>Numeric</th>
<th>Ordinal</th>
<th>Temporal</th>
<th>Geographic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1D</strong></td>
<td>Histogram</td>
<td>Bar Chart</td>
<td>Line Graph / Area Chart</td>
<td>Choropleth Map</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2D</strong></td>
<td>Binned Scatter Plot</td>
<td>Heatmap</td>
<td>Temporal Heatmap</td>
<td>Geographic Heatmap</td>
</tr>
</tbody>
</table>

- Numeric data is often best represented with a Histogram.
- Ordinal data can be shown with a Bar Chart.
- Temporal data is effectively visualized using a Line Graph or Area Chart.
- Geographic data is often depicted with a Choropleth Map.
- Binned data in a 2D format can be shown with a Binned Scatter Plot in 1D, a Heatmap in 2D, or a Temporal Heatmap in a temporal context.
- Geographic data in a 2D format is typically represented with a Geographic Heatmap.
Examples
Sampling
Google Fusion Tables
Binned Aggregation (imMens)
[Liu, Jiang, Heer '13]

Sampling
Google Fusion Tables
Binned Aggregation

[Liu, Jiang, Heer '13]
Example: Binned Scatter Plots

Scatterplot Matrix Techniques for Large N
[Carr et al. '87]
Example: Basketball Shot Chart

NBA Shooting 2011-12
[Goldsberry]
Time Series
Time Series: 1M samples, 1 sample/second
Time Series: 1M samples, 1 sample/second

↑ Value

670 pixels
Time Series: 1M samples, 1 sample/second

670 pixels
~1500 points / pixel!
How might we improve scalability?

Time Series: 1M samples, 1 sample/second

~1500 points / pixel!
Time-Series Aggregation [Jugel’14]

Insight: the resolution is bound by the number of pixels.
Time-Series Aggregation [Jugel’14]

Insight: the resolution is bound by the number of pixels.

1. Compute average value per pixel (1 point/pixel)
   ...this may miss extreme (min, max) values
Time-Series Aggregation [Jugel’14]

Insight: the resolution is bound by the number of pixels.

1. Compute average value per pixel (1 point/pixel)
   ...this may miss extreme (min, max) values

2. Plot min/max values per pixel (2 points/pixel)
   ...this does better, but still misrepresents
Time-Series Aggregation [Jugel’14]

Insight: the resolution is bound by the number of pixels.

1. Compute average value per pixel (1 point/pixel)
   …this may miss extreme (min, max) values

2. Plot min/max values per pixel (2 points/pixel)
   …this does better, but still misrepresents

3. **M4**: min/max values & timestamps (4 points/pixel)
   …this provides provable fidelity to the full data!
Data Reduction in the Database

SELECT t,v FROM Q JOIN
(SELECT round($w*(t-$t1)/($t2-$t1)) as k,
  min(v) as v_min, max(v) as v_max,
  min(t) as t_min, max(t) as t_max
FROM Q GROUP BY k) as QA
ON k = round($w*(t-$t1)/($t2-$t1))
AND (v = v_min OR v = v_max OR
  t = t_min OR t = t_max)

Q: query that returns a time series (t,v)
$w$: chart width in pixels
$t1$, $t2$: global min/max timestamps
Time Series: 1M samples, 1 sample/second

↑ Value

Sat 03  Mon 05  Wed 07  Fri 09  Dec 11  Time →
M4: 1M samples -> 2,653 plotted points
But what about multiple time-series?

Perceptual scalability breaks down...
The non-normalized heatmap suffers from artifacts, seen as vertical stripes. Binned charts convey high points across the top, a collective dip in stocks during the crash of 2008, and two distinct bands of $25 and $15 stocks.
Density Line Chart

[Moritz & Fisher]

Time Series

Repeat for each series

Non-Normalized
Density Line Chart

Time Series

Repeat for each series

Non-Normalized

[Moritz & Fisher]
**Density Line Chart**

[Moritz & Fisher]

**Time Series**

**Repeat for each series**

**Non-Normalized**

Approx. Arc-Length Normalized
Density Line Chart

[Moritz & Fisher]

Time Series

Repeat for each series

Non-Normalized

Approx. Arc-Length Normalized

Aggregate

Color
The density of the second group appears to increase to the right!
Without normalization, the steep lines are over-represented.
Design Subtleties
Hexagonal or Rectangular Bins?

Hex bins better estimate density for 2D plots, but the *improvement is marginal* [Scott 92]. Rectangles support *reuse* and *visual queries*. 
Color Scale: Discontinuity after Zero

**Standard Color Ramp**  
Counts near zero are white.

**Add Discontinuity after Zero**  
Counts near zero remain visible.
Color Ramps / Scale Transforms

Linear interpolation in RGBA is not perceptually linear.

Perceptual color spaces approximate perceptual linearity.
Questions?
Scalable Interaction
Flight Delays
250k Records
Gaia Star Catalog · 1.8B Records
Gaia Star Catalog · 1.8B Records
Interactive Scalability Strategies

1. Query Database
2. Client-Side Indexing / Data Cubes
3. Prefetching
4. Approximation
Interactive Scalability Strategies

1. Query Database  Offload to a scalable backend…
Tableau, for example, issues aggregation queries. Analytical databases are designed for fast, parallel execution. But round-trip queries to the DB may still be too slow…

2. Client-Side Indexing / Data Cubes

3. Prefetching

4. Approximation
Interactive Scalability Strategies

1. **Query Database** …or alternative data frame implementation

   *Python*: Vaex, Polars, Modin, cuDF

   *R*: dbplyr

   *All*: DuckDB

2. **Client-Side Indexing / Data Cubes**

3. **Prefetching**

4. **Approximation**
Interactive Scalability Strategies

1. Query Database

2. Client-Side Indexing / Data Cubes  Query data summaries
   Build sorted indices or data cubes to quickly re-calculate aggregations as needed on the client.

3. Prefetching

4. Approximation
Interactive Scalability Strategies

1. Query Database
2. Client-Side Indexing / Data Cubes
3. **Prefetching**  Request data *before* it is needed
   Reduce latency by speculatively querying for data before it is needed. Requires prediction models to guess what is needed.
4. Approximation
Interactive Scalability Strategies

1. Query Database
2. Client-Side Indexing / Data Cubes
3. Prefetching
4. Approximation  Give fast, approximate answers

Reduce latency by computing aggregates on a sample, ideally with approximation bounds characterizing the error.
Interactive Scalability Strategies

1. Query Database
2. Client-Side Indexing / Data Cubes
3. Prefetching
4. Approximation

These strategies are not mutually exclusive!
Systems can apply them in tandem.
Client-Side Indexes
Flight Delays
250k Records
Flight Delays
250k Records

Data Cube
Flight Delays
250k Records

Sum across rows to re-aggregate
Flight Delays
250k Records
Flight Delays
250k Records
Flight Delays
250k Records

Sum across rows to re-aggregate
Flight Delays
250k Records

Sum across columns to re-aggregate
5-D Data Cube

Month, Day, Hour, X, Y

~2.3B bins
5-D Data Cube

Month, Day, Hour, X, Y

~2.3B bins
Multivariate Data Tiles

1. Send data, not pixels
2. Embed multi-dim data
Full 5-D Cube
For any pair of 1D or 2D binned plots, the maximum number of dimensions needed to support brushing & linking is four.
Full 5-D Cube

Σ
Σ
Σ
Σ

3-D cubes

3-D data tiles

13 3-D Data Tiles
2.3B bins

Full 5-D Cube

~2.3B bins

Σ

17.6M bins

3-D Data Tiles

~17.6M bins (in 352KB!)

3-D Cubes

13 3-D Data Tiles
In-Memory Data Cube

Number of Data Points

Average Frames per Second

5 dimensions x 50 bins/dim x 25 plots

~50fps querying of visual summaries of 1B data points.
Limitations and Questions

But where do the multivariate data tiles come from?
They must be computed, either ahead of time or on-the-fly. Up to the 100M point range, an analytic database can do this on the fly. In the 1B point range, pre-computation avoids delays.

We can also *prefetch*: we can start computing new data tiles as soon as the pointer enters a chart, before a selection is made.

**Does super-low-latency interaction really matter?**
Is it worth it to go to all of this trouble? (Short answer: yes!)
High latency leads to reduced analytic output [Liu & Heer, InfoVis 2014]
Sampling Methods
Common Sampling Methods

**First-N**: Useful for transformation, but not inference.

**Random**: Good default, but may miss features of interest. Possible in one pass via reservoir sampling, or faster if stored in randomized order.

**Stratified**: Sample within groups, ensure coverage and balance across those categories.
Binned Aggregation
[Liu, Jiang, Heer '13]

Sampling
Google Fusion Tables
Online Aggregation [Hellerstein, Haas, Wang ’97]

Provide dynamic, progressive results as queries run: see results over growing samples. Visualize current results with confidence intervals to convey uncertainty of estimate. Challenge: difficult to ensure truly random sampling.
What if data is too large to query in a reasonable time?
Trust, but Verify: Optimistic Vis

[Moritz, Fisher, Ding & Wang ’17]

Strategies: Query Database, Approximation
Latencies reduce engagement and lead to fewer observations.

Approximation: Trade Accuracy for Speed

Approximate query processing (AQP)
Uncertainty estimation in statistics
Uncertainty visualization
Probabilistic programming
Approximate hardware
Pick your poison:
1. Trust the approximation, or
2. Wait for everything to complete.
Optimistic Visualization

Trust but Verify

This glass is half full
What if we think of the issues with approximation as user experience problems?
Optimistic Visualization

1. Analysts uses initial estimates.
2. Precise queries run in the background.

Analysts can use approximations and also trust them.
Optimistic Visualization
Visualize Uncertainty

Heatmap

X-Axis
- Field: DepDelay
- Binning: 64
- Sort by key: ✓

Y-Axis
- Field: ArrDelay
- Binning: 40
- Sort by key: ✓

Value
- Function: Count

Persistent Filters
@.AND(Carrier$ND([a, d])&DepDelay>30)

Expected Error
Relative

Approximate Values

Expected errors: 2.3%
Show a History of Previous Charts
Help Analysts Confirm Results
Evaluation

Case studies with teams at Microsoft who brought in their own data.

Approximation works
“seeing something right away at first glimpse is really great”

Need for guarantees
“[with a competitor] I was willing to wait 70-80 seconds. It wasn’t ideally interactive, but it meant I was looking at all the data.”

Optimism works
“I was thinking what to do next— and I saw that it had loaded, so I went back and checked it

. . . [the passive update is] very nice for not interrupting your workflow.”
In Conclusion...
Two Challenges:

1. Effective **visual encoding**
2. Real-time **interaction**
Perceptual and interactive scalability should be limited by the chosen resolution of the visualized data, not the number of records.
1. **Bin**  Divide data domain into discrete “buckets”

2. **Aggregate**  Count, Sum, Average, Min, Max, ...

3. **Smooth**  *Optional*: smooth aggregates [Wickham ‘13]

4. **Plot**  Visualize the aggregate values
Interactive Scalability Strategies

1. Query Database
2. Client-Side Indexing / Data Cubes
3. Prefetching
4. Approximation

These strategies are not mutually exclusive! Systems can apply them in tandem.