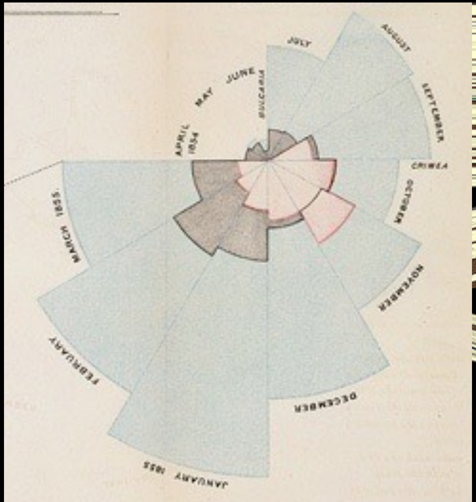
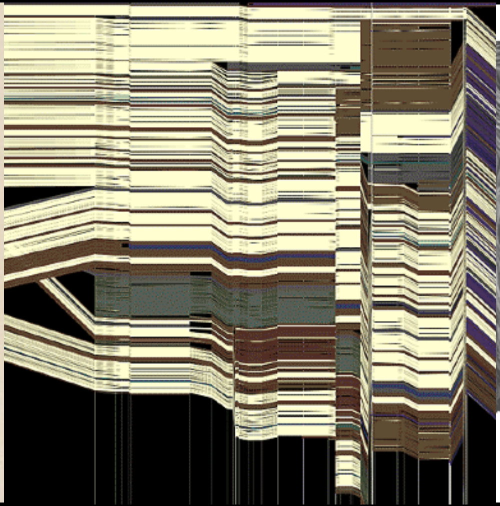
**CSE 442** - Data Visualization

# Scalable Visualization




Jeffrey Heer University of Washington

# Session Outline

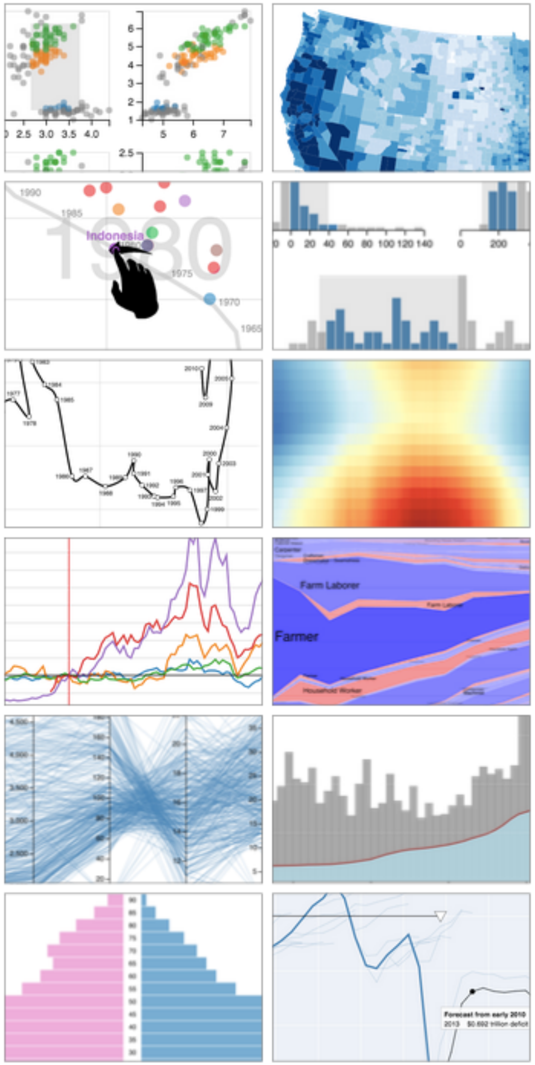
The Varieties of “Big Data”

Scalable Plotting Techniques

Scalable Interaction

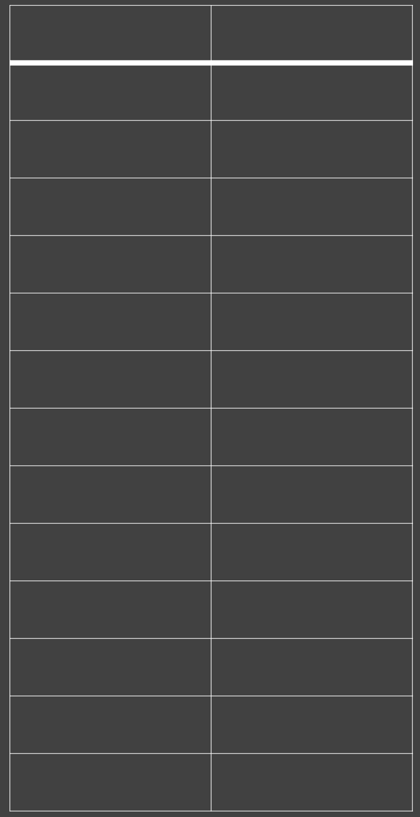
Why Latency Matters

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?

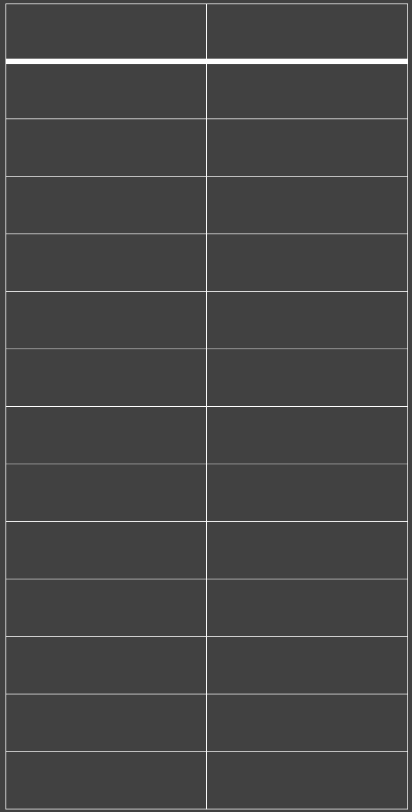
  Parallel data processing

  Reduction: Filter, aggregate

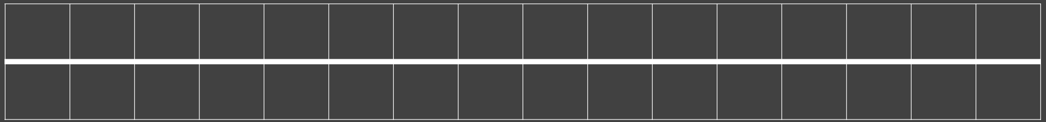
  Sample or approximate

Not just about systems. Consider perceptual / cognitive scalability.

## **Tall** Data



## **Wide** data

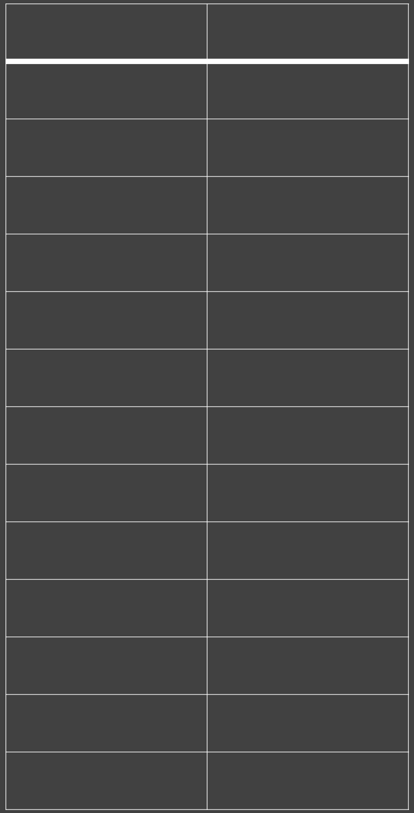


Lots of variables (100s-1000s…)

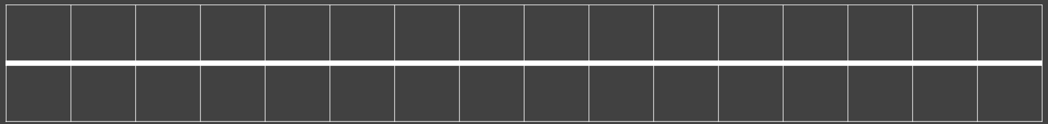
- Select relevant subset
- Dimensionality reduction
- Statistical methods can suggest and order related variables

Requires human judgment

**Tall** Data

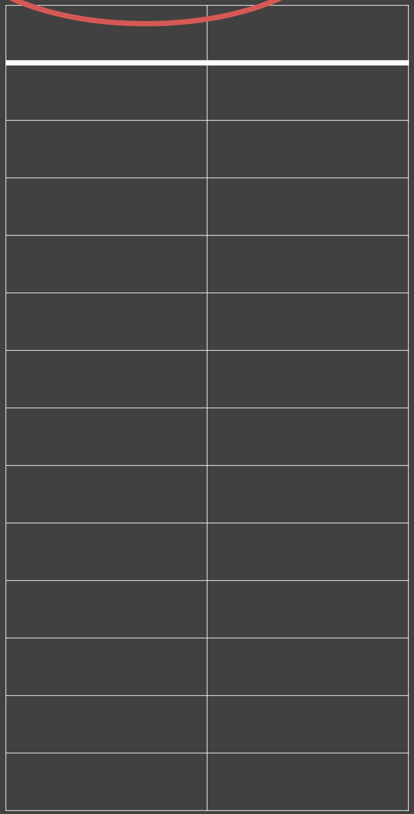


**Wide** data

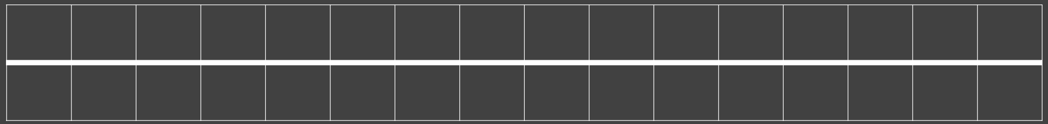


**Diverse** data

## **Tall** Data



## **Wide** data



## **Diverse** data

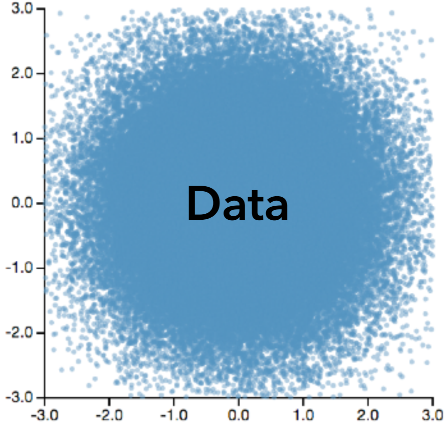
How can we visualize and interact with **billion+ record** databases in real-time?

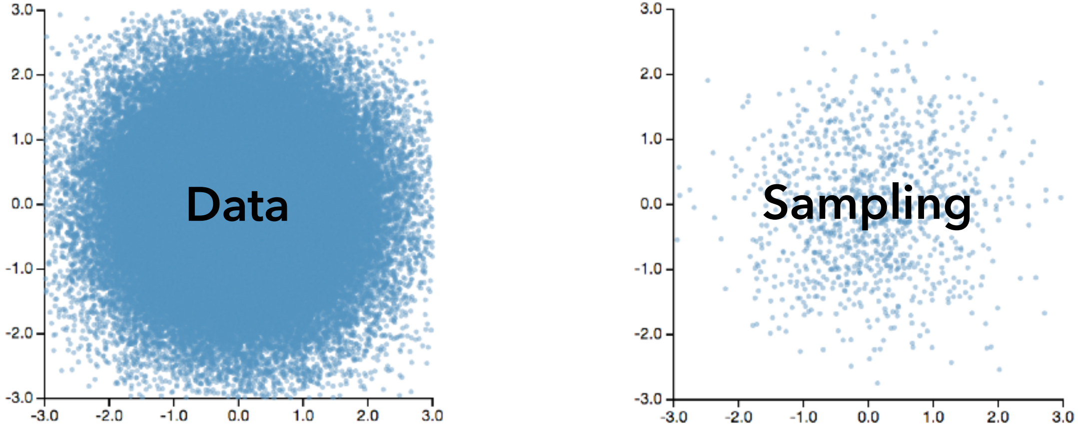
Two Challenges:

1. Effective **visual encoding**
2. Real-time **interaction**

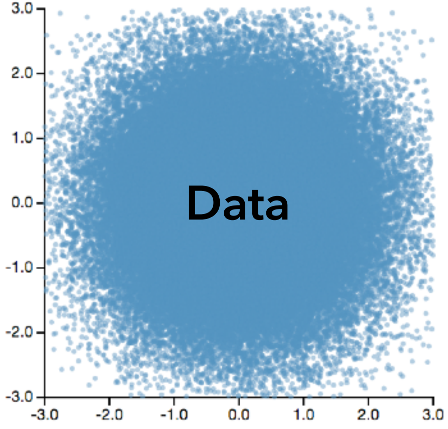
**Perceptual and interactive scalability** should be limited by the **chosen resolution** of the visualized data, not the number of records.

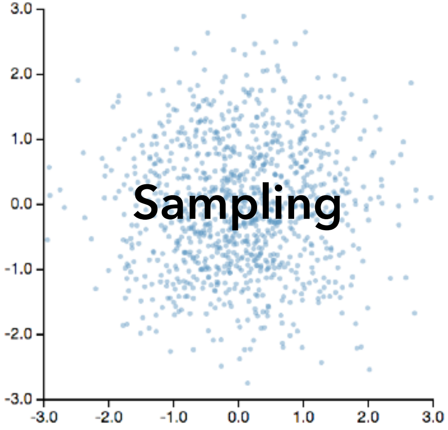
# Scalable Plotting Techniques

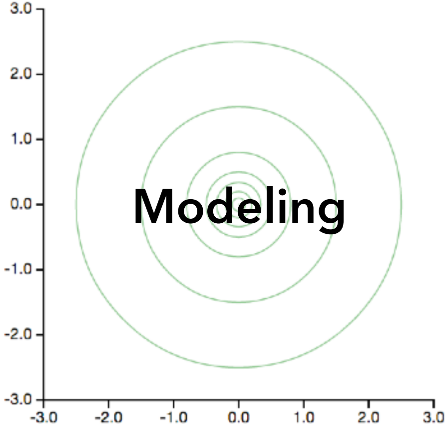

Data
3.0
2.0
1.0
0.0
-1.0
-2.0
-3.0
-3.0 -2.0 -1.0 0.0 1.0 2.0 3.0

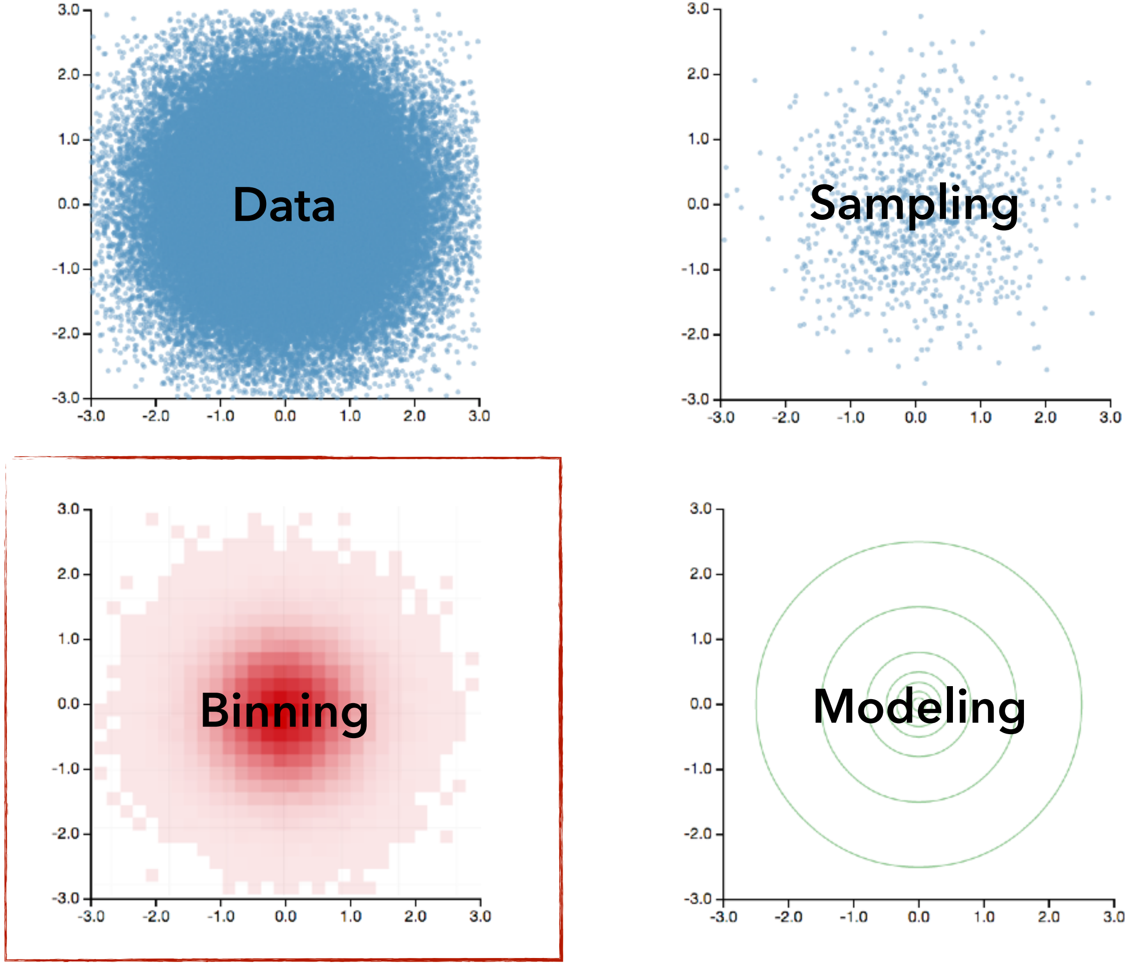


Data
Sampling
3.0
2.0
1.0
0.0
-1.0
-2.0
-3.0
-3.0
-2.0
-1.0
0.0
1.0
2.0
3.0

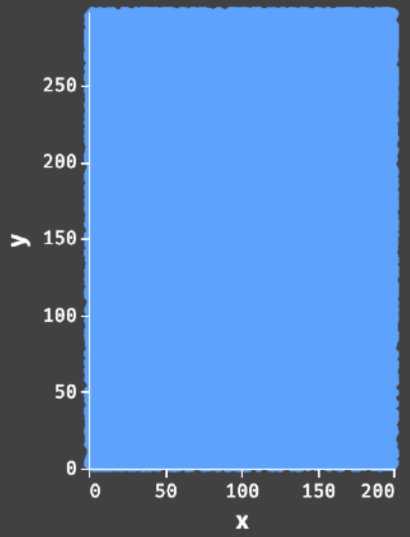

Data
3.0
2.0
1.0
0.0
-1.0
-2.0
-3.0
-3.0 -2.0 -1.0 0.0 1.0 2.0 3.0


Sampling
3.0
2.0
1.0
0.0
-1.0
-2.0
-3.0
-3.0 -2.0 -1.0 0.0 1.0 2.0 3.0

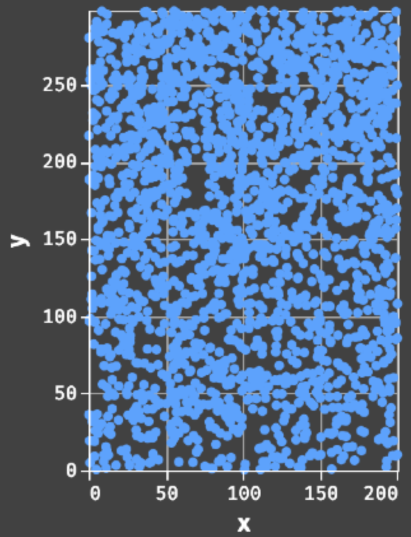

Modeling
3.0
2.0
1.0
0.0
-1.0
-2.0
-3.0
-3.0 -2.0 -1.0 0.0 1.0 2.0 3.0


Data
Sampling
Binning
Modeling

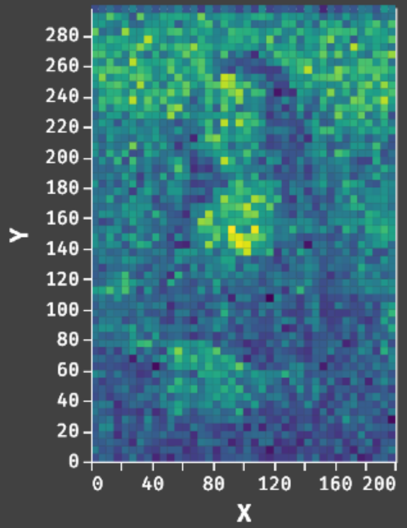
# How to **Visualize** a Billion+ Records




Data




Sampling




Binned Aggregation

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.

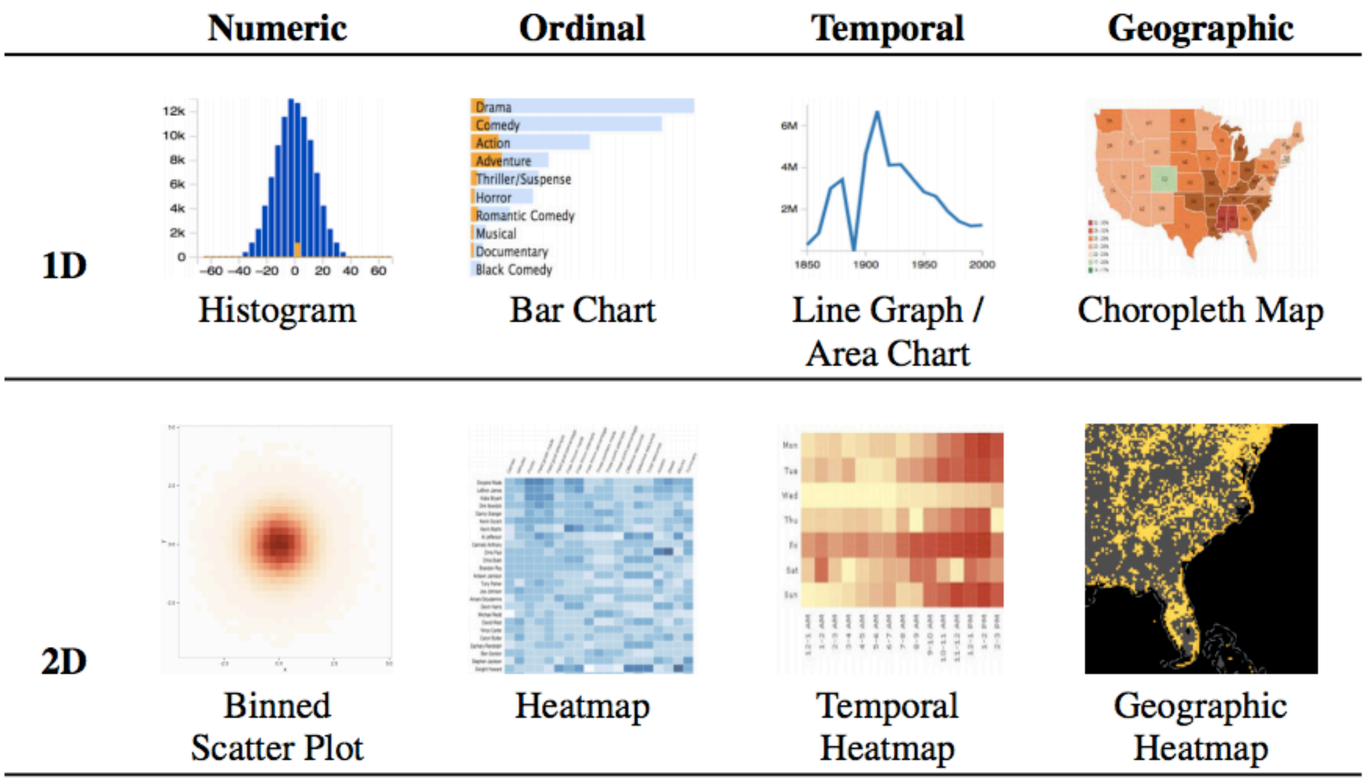
*Geo*: Bin x, y coordinates *after* cartographic projection

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

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

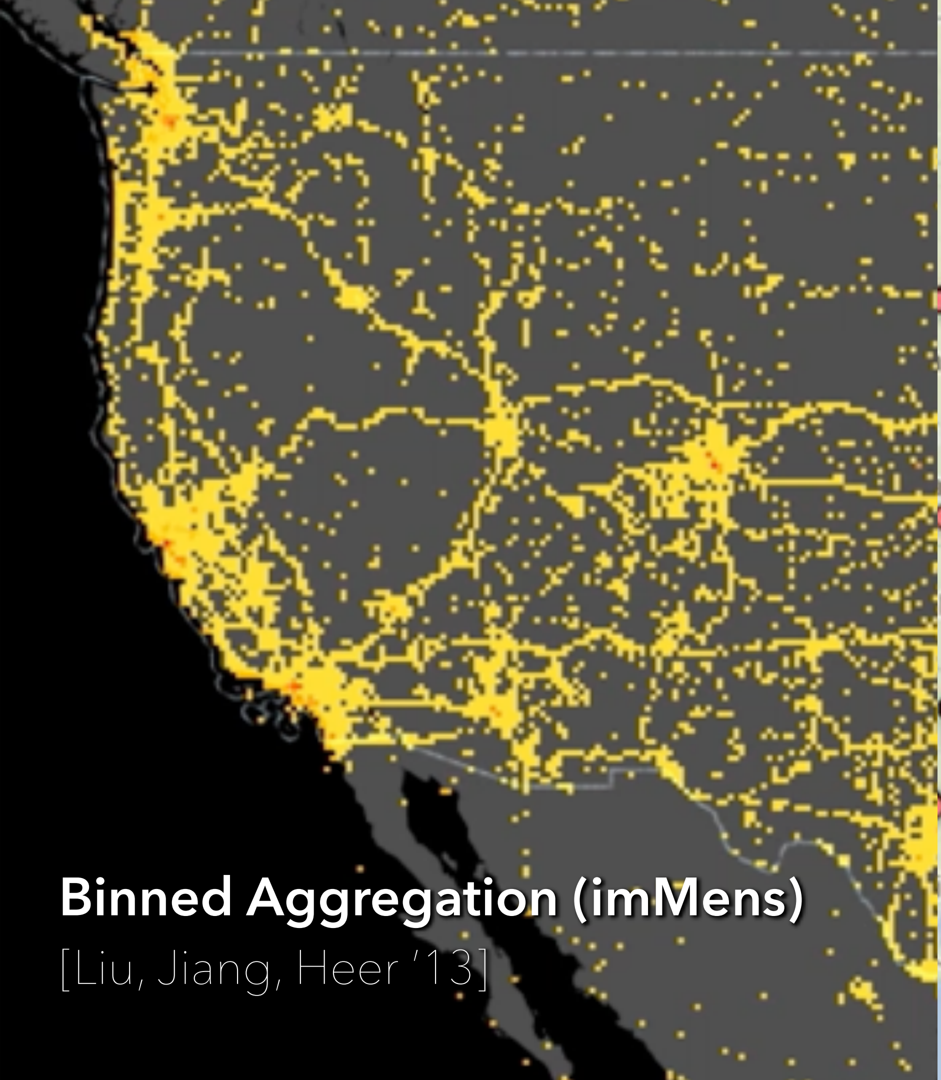
**4. Plot** Visualize the aggregate values

# Binned Plots by Data Type



# Examples


Sampling
Google Fusion Tables
Montana
Minnesota
Montreal
Maine
South Dakota
Toronto
Idaho
New York
Illinois
Nevada
Utah
Missouri
West Virginia
Kentucky
Virginia
Oklahoma
New Mexico
Mississippi
South Carolina
District of Columbia
Alabama
Georgia
Texas
Dallas
Phoenix
San Diego
Houston
San Antonio
Gulf of California
Monterrey
Mexico
Havana
Delaware
Maryland

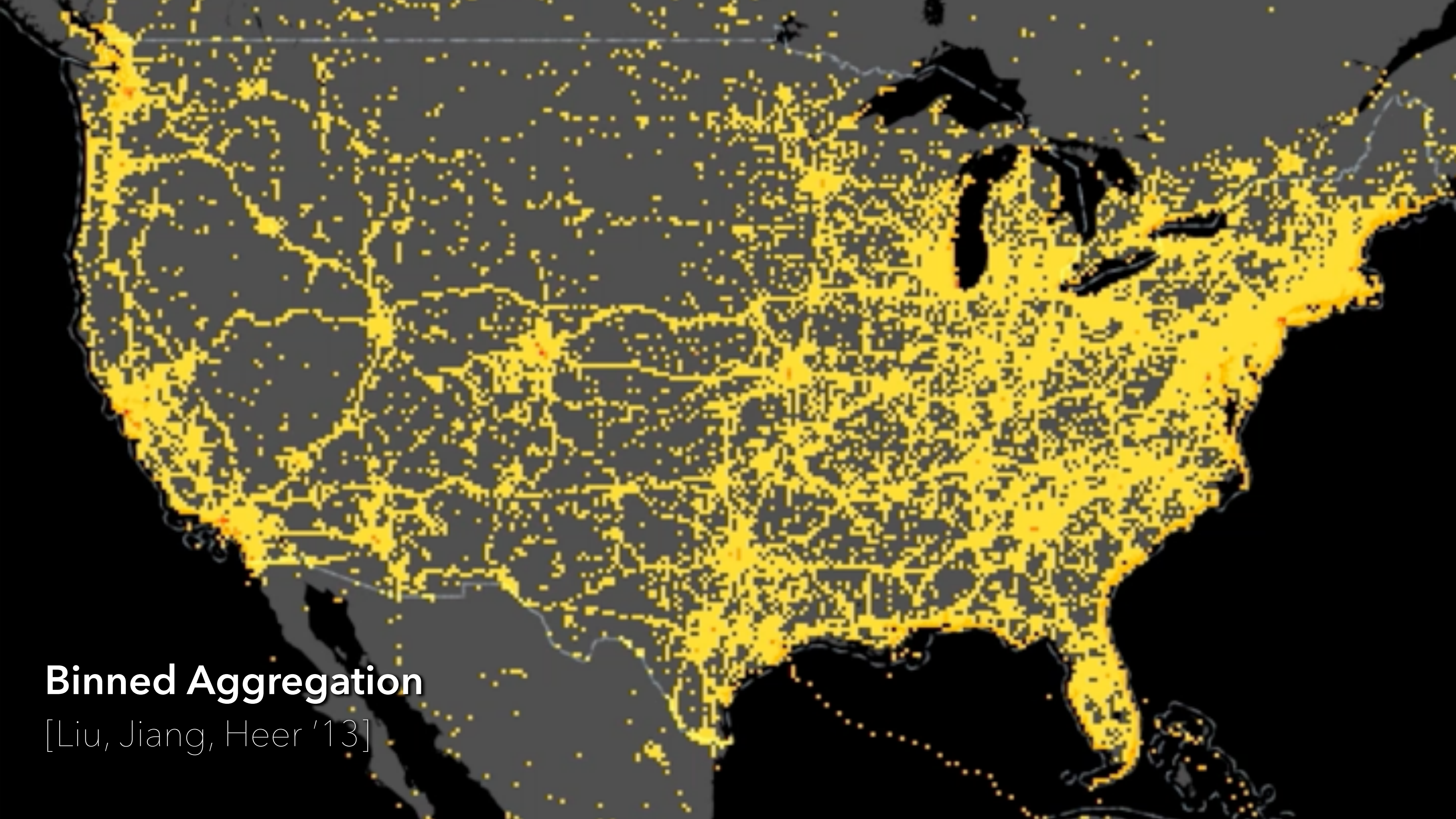


**Binned Aggregation (imMens)**

[Liu, Jiang, Heer '13]


**Sampling**

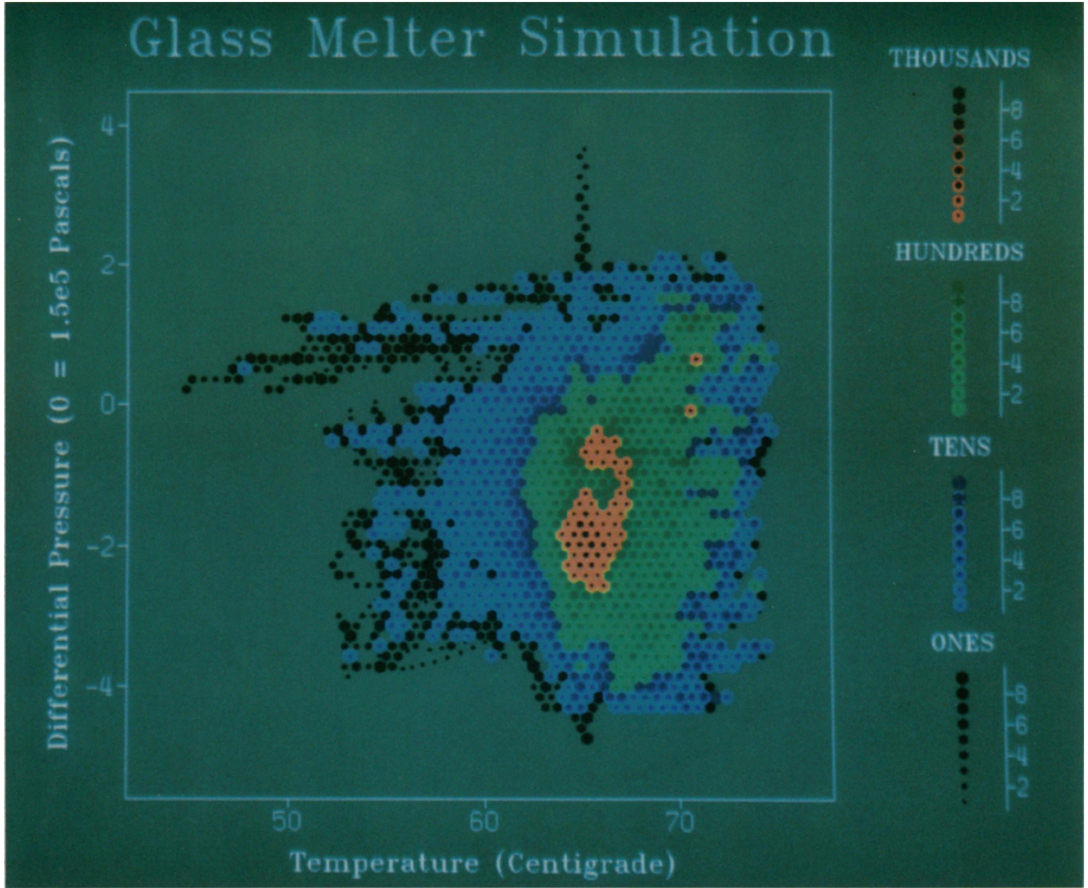
**Google Fusion Tables**



**Binned Aggregation**

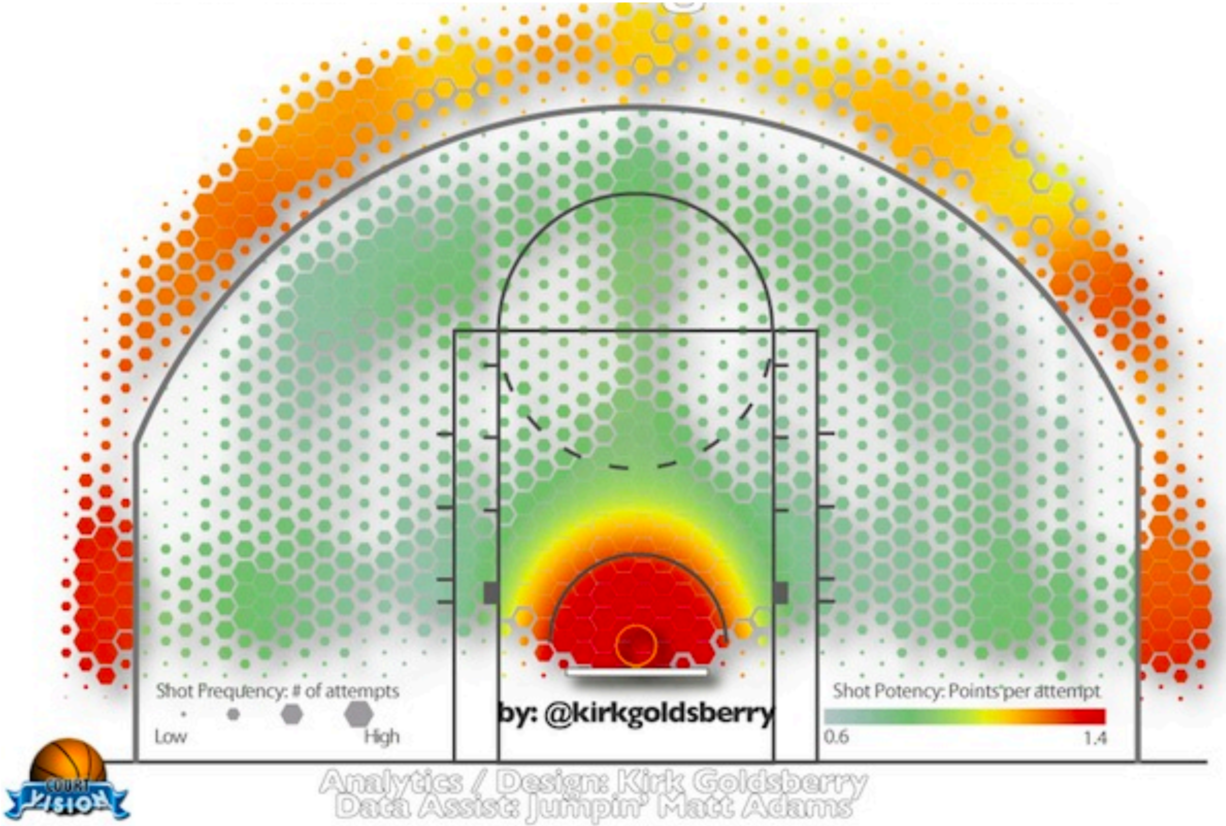
[Liu, Jiang, Heer '13]

# Example: Binned Scatter Plots




Scatterplot Matrix Techniques for Large N [Carr et al. '87]

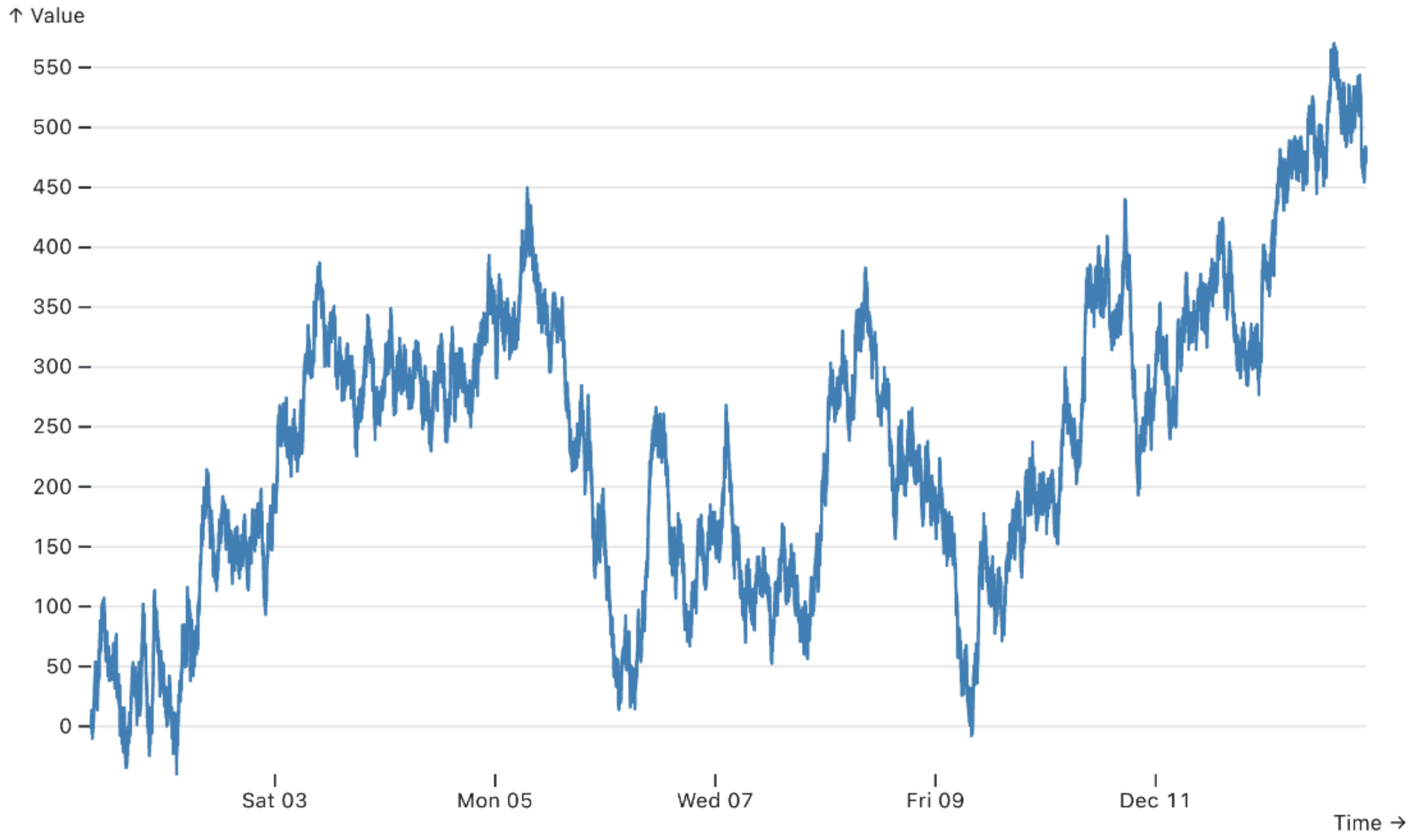
# Example: Basketball Shot Chart



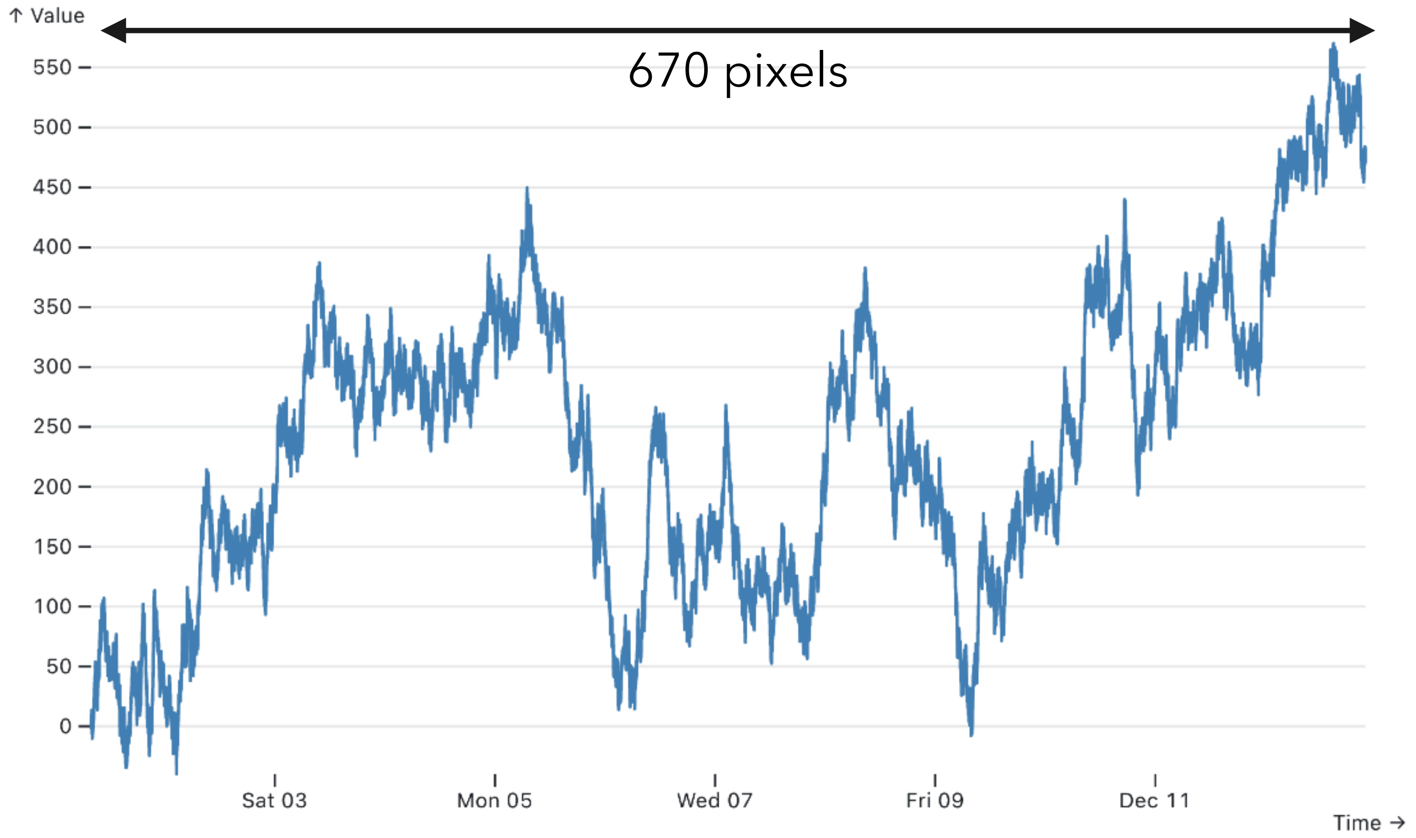

NBA Shooting 2011-12
[Goldsberry]

# Time Series

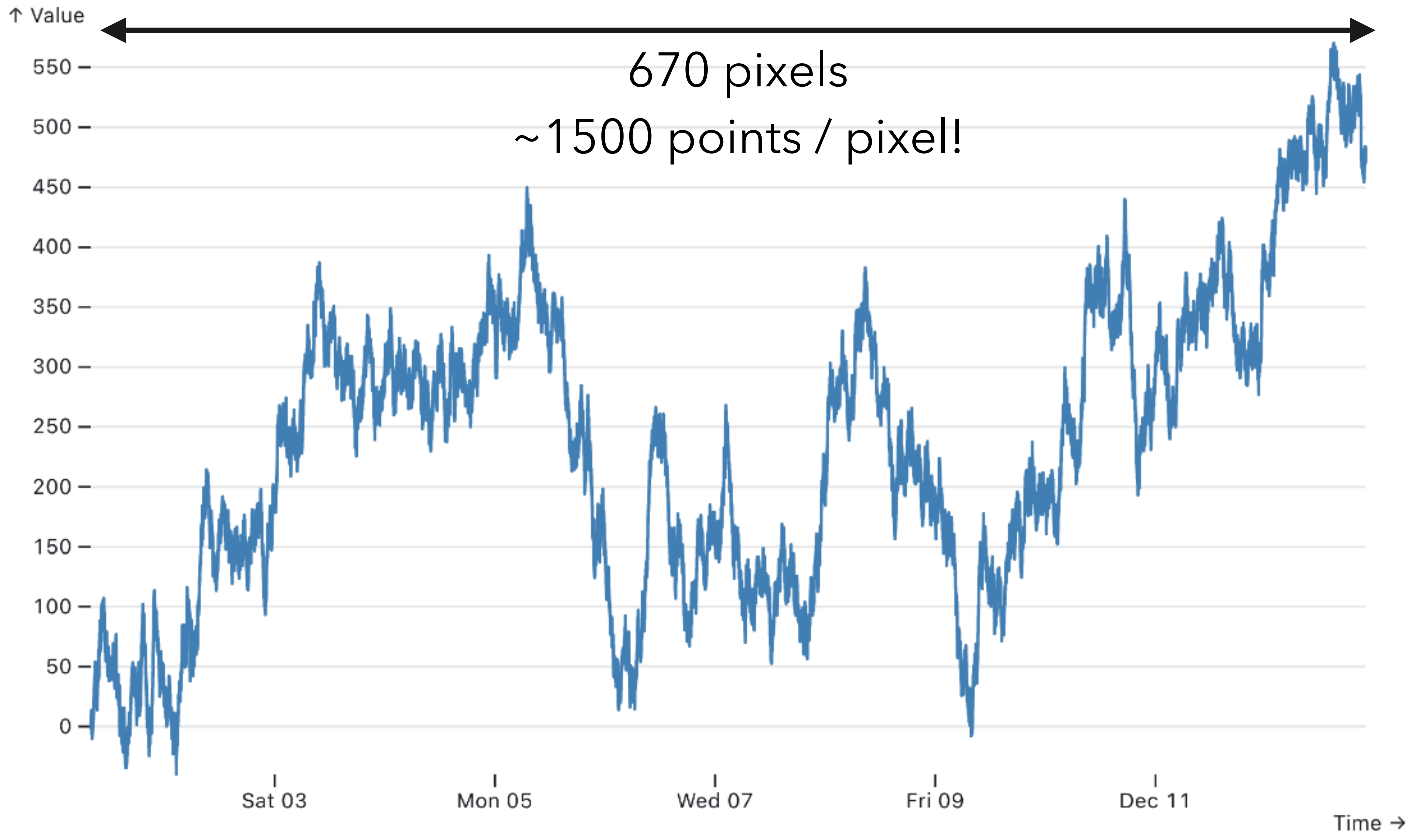
# Time Series: 1M samples, 1 sample/second



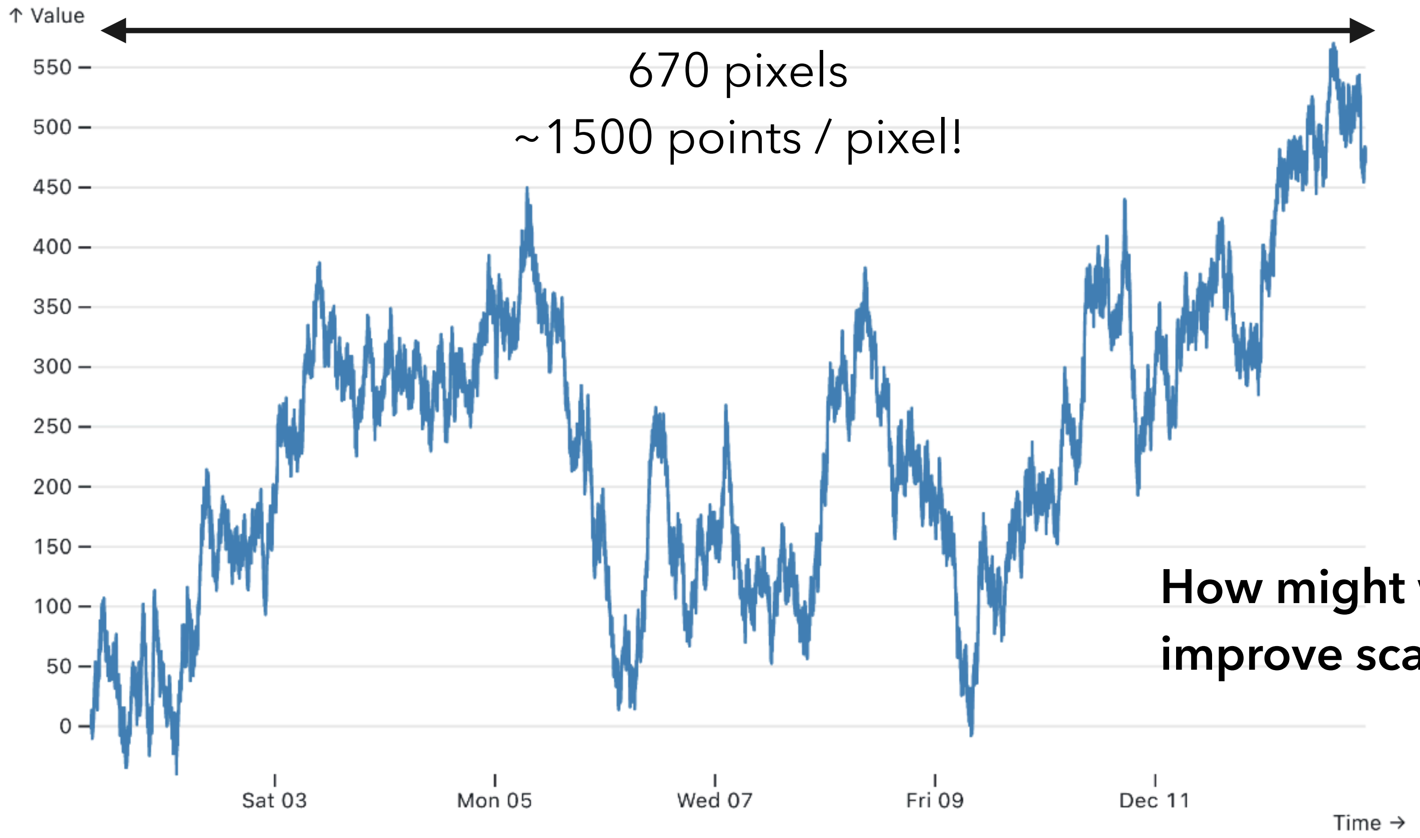
# Time Series: 1M samples, 1 sample/second



# Time Series: 1M samples, 1 sample/second

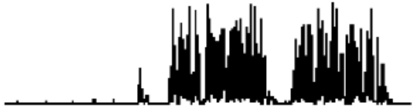


# Time Series: 1M samples, 1 sample/second



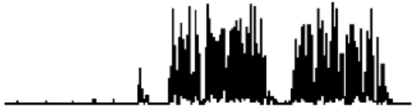

How might we improve scalability?

# Time-Series Aggregation [Jugel'14]



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

# Time-Series Aggregation [Jugel'14]

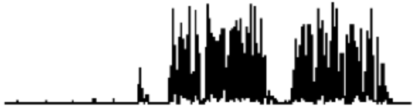


**Insight: the resolution is bound by the number of pixels.**

1. Compute average value per pixel (1 point/pixel)

…this may miss extreme (min, max) values

# Time-Series Aggregation [Jugel'14]



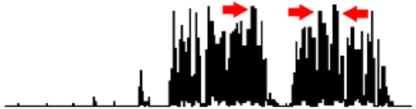
**Insight: the resolution is bound by the number of pixels.**

1. Compute average value per pixel (1 point/pixel)

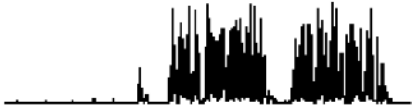
   …this may miss extreme (min, max) values

2. Plot min/max values per pixel (2 points/pixel)

   …this does better, but still misrepresents



# Time-Series Aggregation [Jugel'14]



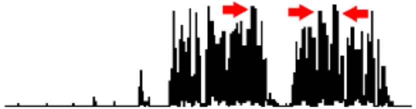
**Insight: the resolution is bound by the number of pixels.**

1. Compute average value per pixel (1 point/pixel)

   ...this may miss extreme (min, max) values

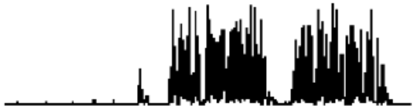
2. Plot min/max values per pixel (2 points/pixel)

   ...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

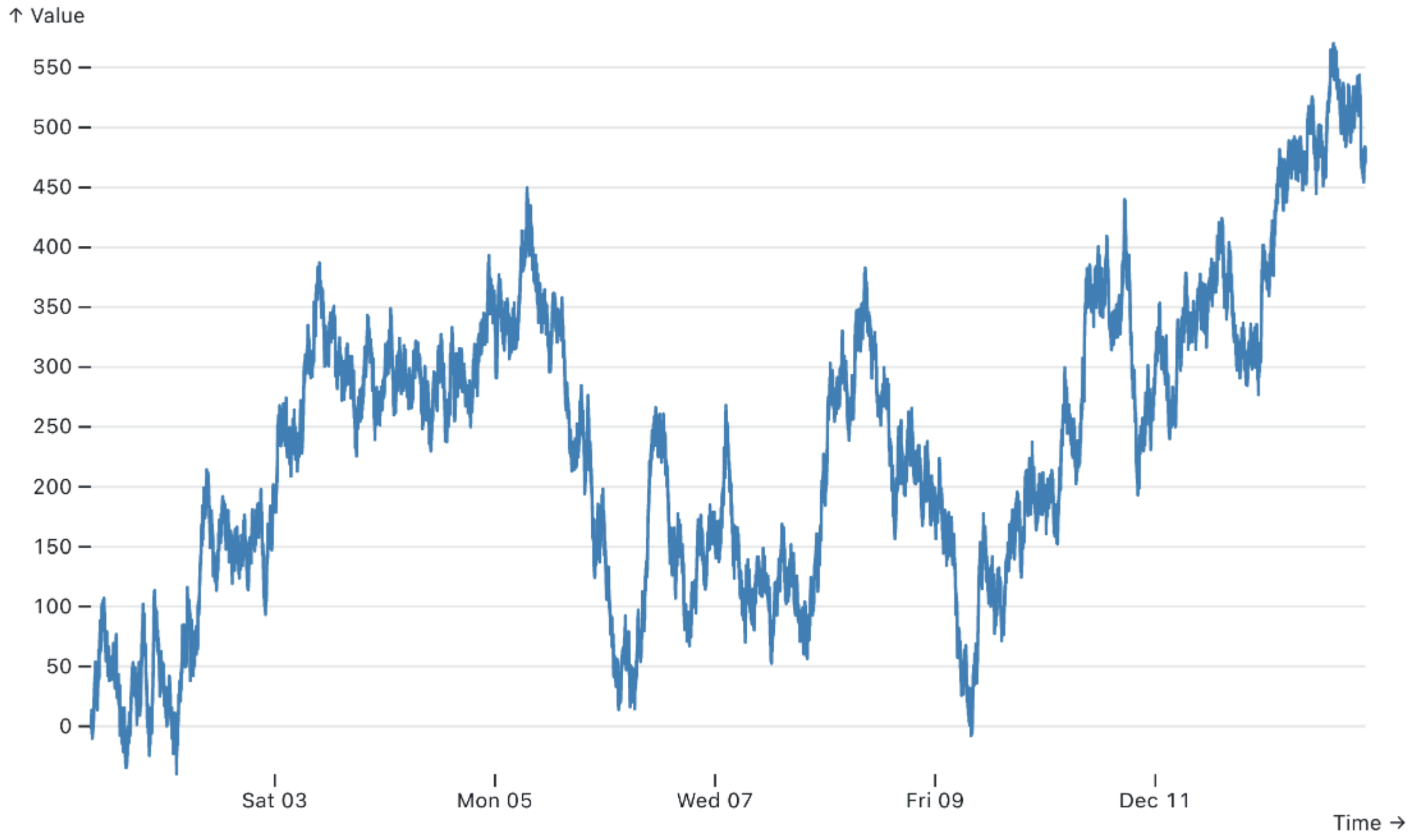
```
SELECT t,v FROM Q JOIN
(SELECT round($w*(t-$t1)/($t2-$t1)) as k,   --define key
        min(v) as v_min, max(v) as v_max,   --get min,max
        min(t) as t_min, max(t) as t_max    --get 1st,last
        FROM Q GROUP BY k) as QA            --group by k
ON k = round($w*(t-$t1)/($t2-$t1))          --join on k
       AND (v = v_min OR v = v_max OR       --&(min|max|
            t = t_min OR t = t_max)         -- 1st|last)
```

`Q`: query that returns a time series `(t,v)`

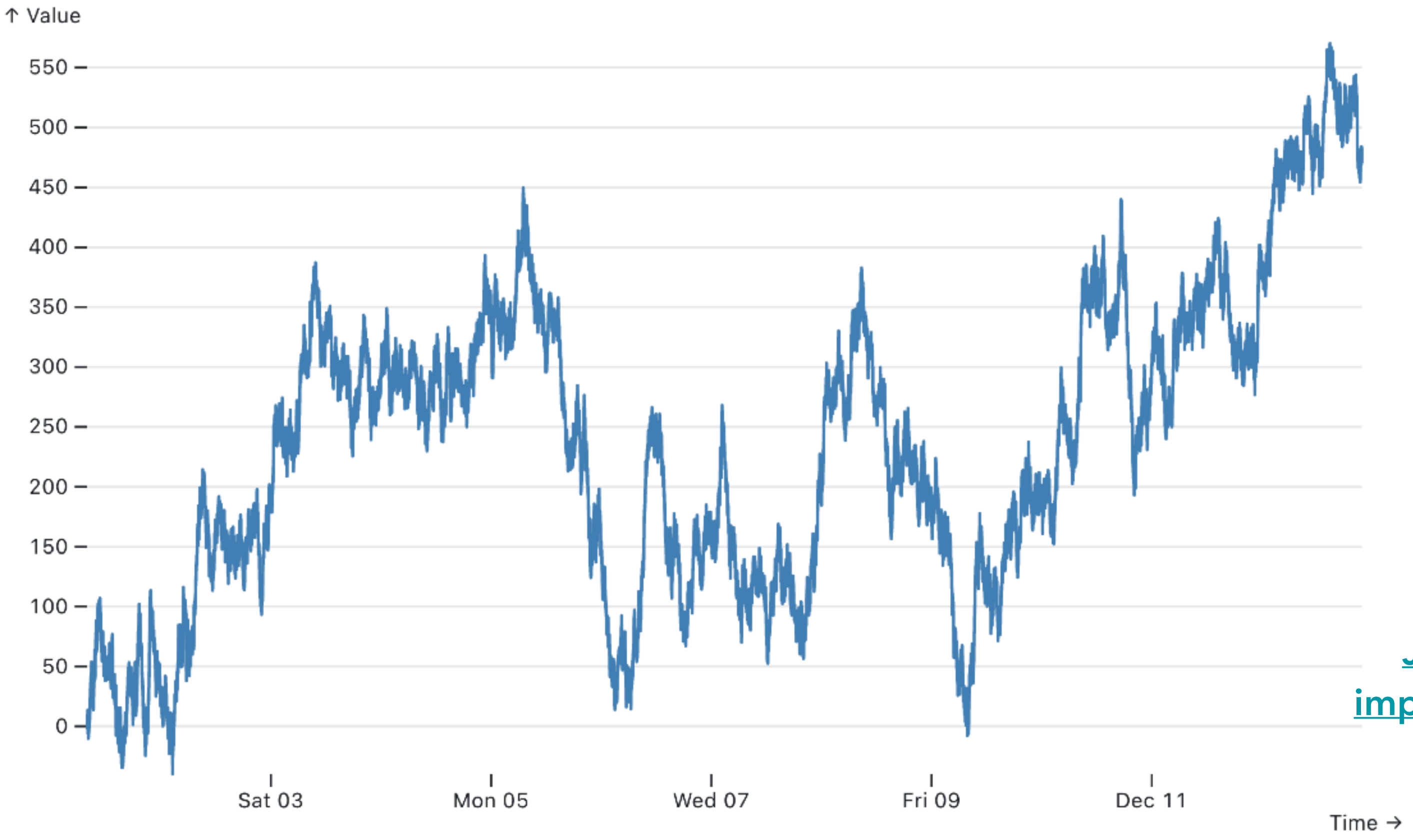
`$w`: chart width in pixels

`$t1, $t2`: global min/max timestamps

# Time Series: 1M samples, 1 sample/second

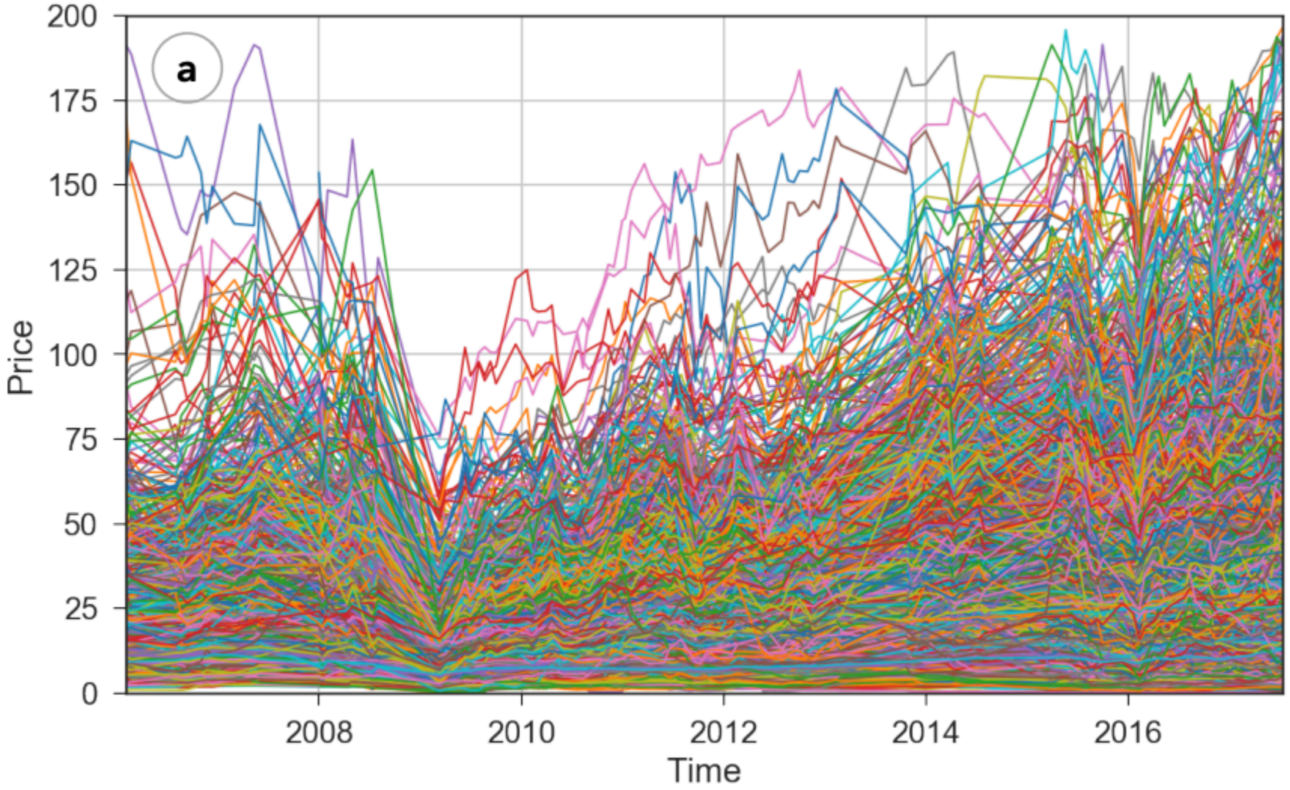


# M4: 1M samples -> 2,653 plotted points



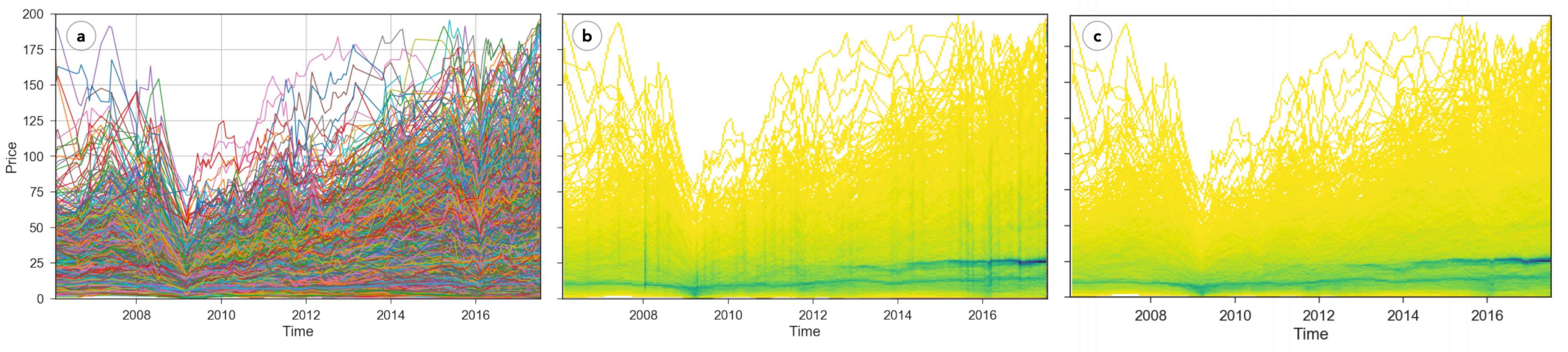

[JavaScript implementation]

# But what about multiple time-series?




Perceptual scalability breaks down…

# Density Line Chart [Moritz & Fisher]




Line Chart

Non-Normalized Heatmap

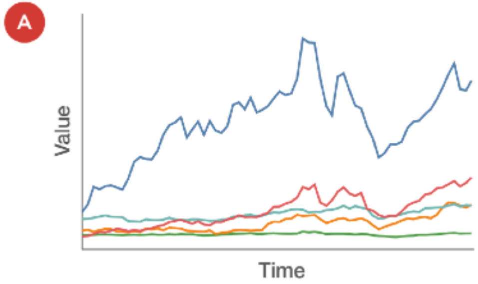
Normalized "DenseLines"

The non-normalized heatmap suffers from artifacts, seen as vertical stripes.

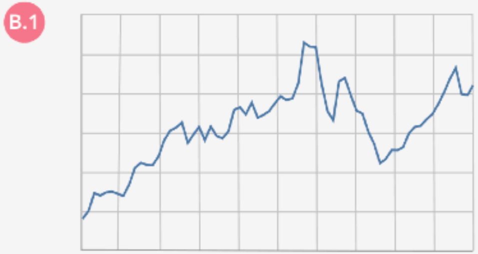
Binned charts convey high points across the top, a collective dip in stocks during the crash of 2008, and two distinct bands of $25 and $15 stocks.

# Density Line Chart [Moritz & Fisher]

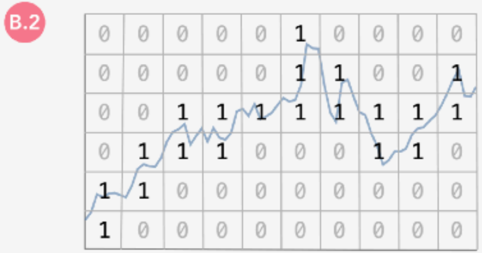
## Time Series



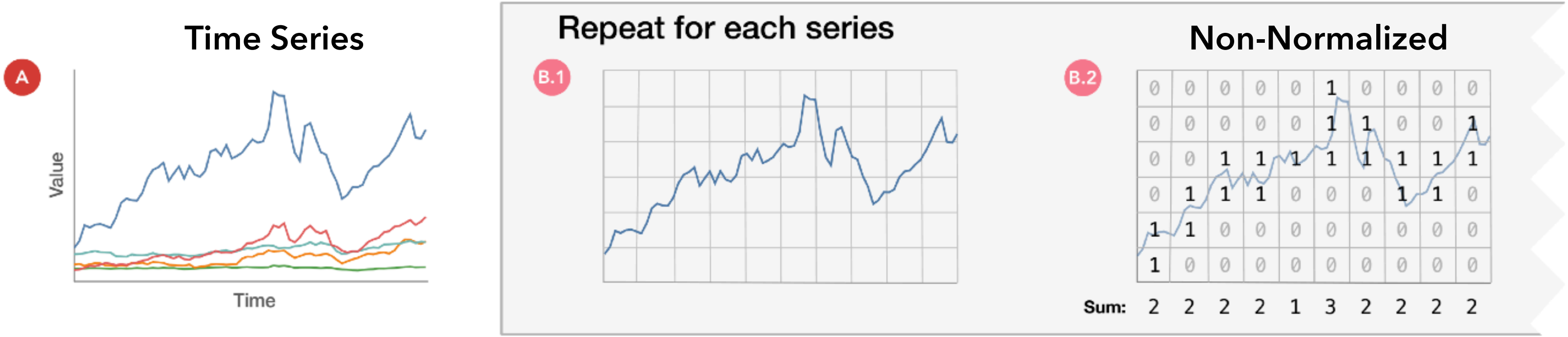

## Repeat for each series




## Non-Normalized



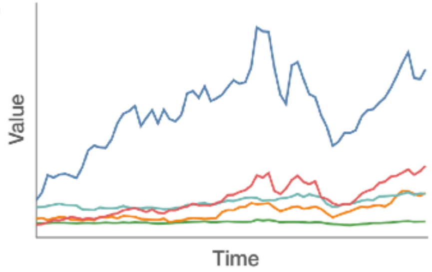
# Density Line Chart [Moritz & Fisher]



# Density Line Chart [Moritz & Fisher]

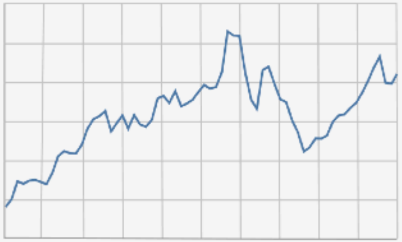
## Time Series

A



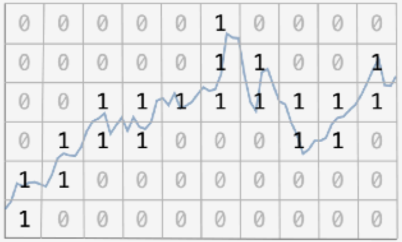

## Repeat for each series

B.1



## Non-Normalized

B.2



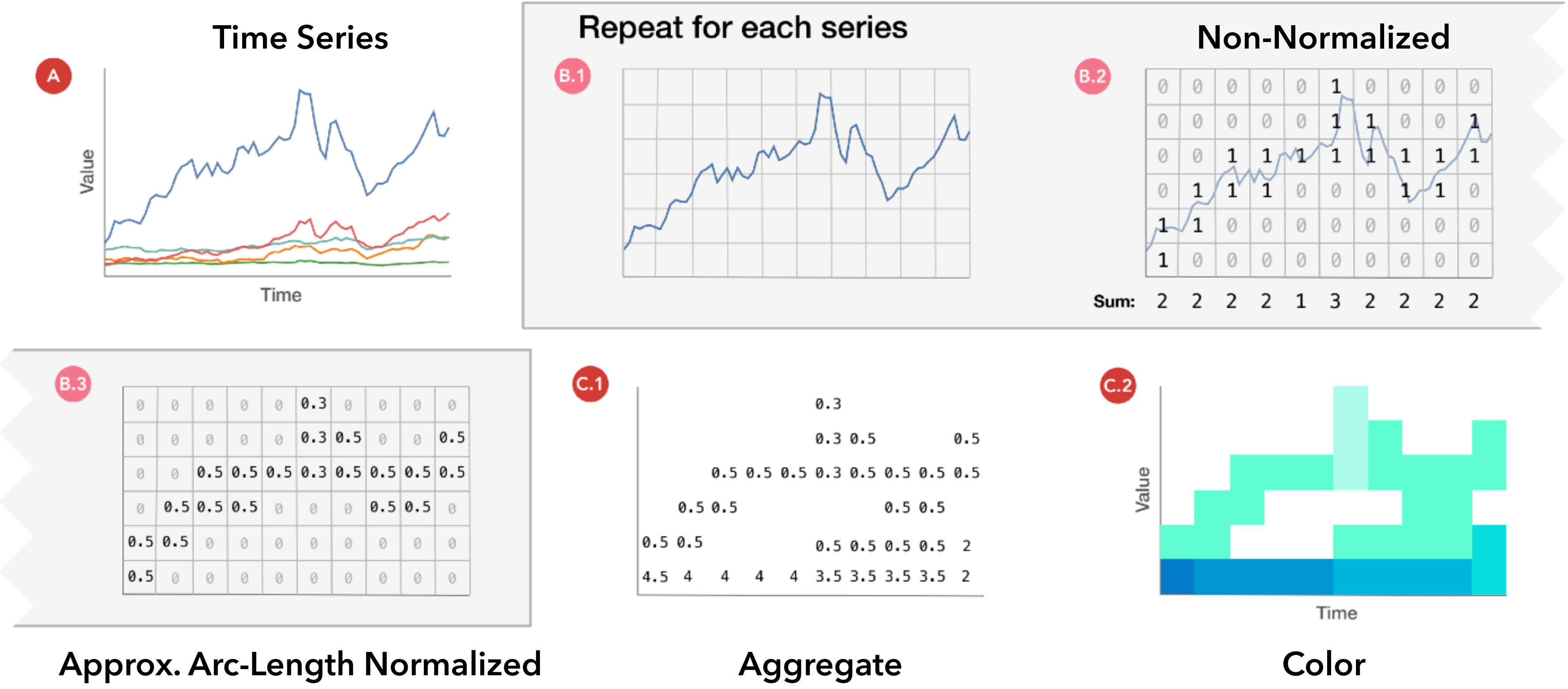
| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Sum:** | 2 | 2 | 2 | 2 | 1 | 3 | 2 | 2 | 2 | 2 |

B.3

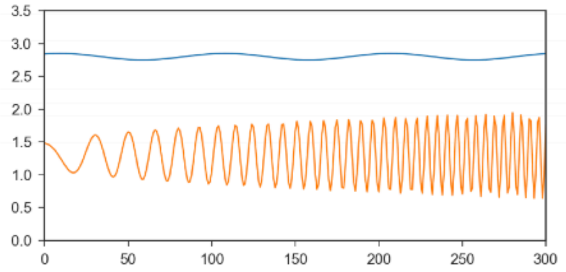
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0.3 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0.3 | 0.5 | 0 | 0 | 0.5 |
| 0 | 0 | 0.5 | 0.5 | 0.5 | 0.3 | 0.5 | 0.5 | 0.5 | 0.5 |
| 0 | 0.5 | 0.5 | 0.5 | 0 | 0 | 0 | 0.5 | 0.5 | 0 |
| 0.5 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## Approx. Arc-Length Normalized

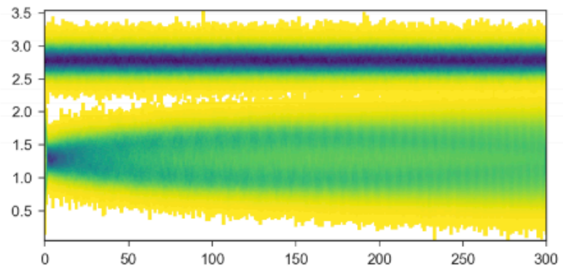
# Density Line Chart [Moritz & Fisher]



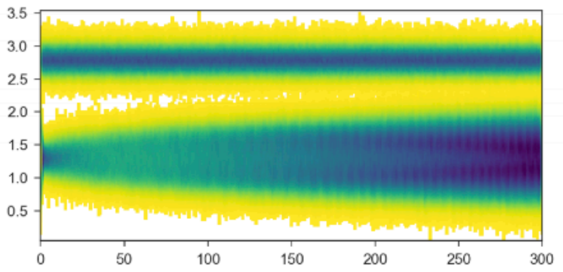
# Density Line Chart [Moritz & Fisher]




Example Time Series




10k Series, Normalized



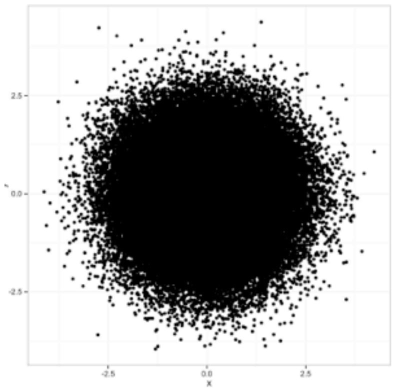

10k Series, Non-Normalized

The density of the second group appears to increase to the right!

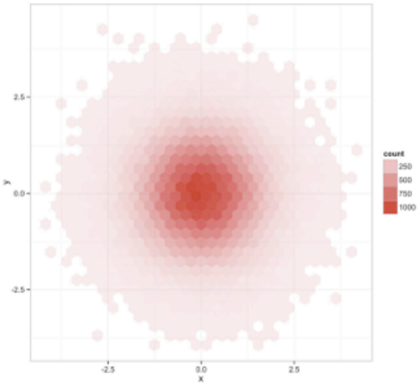
Without normalization, the steep lines are over-represented.

# Design Subtleties

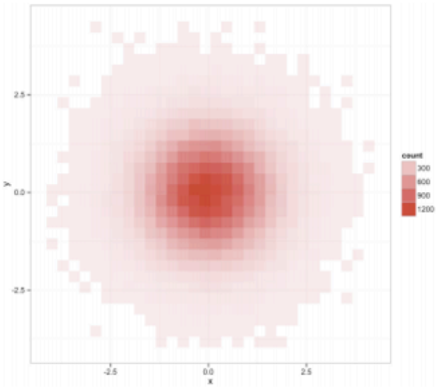
# Hexagonal or Rectangular Bins?




100,000 Data Points




Hexagonal Bins

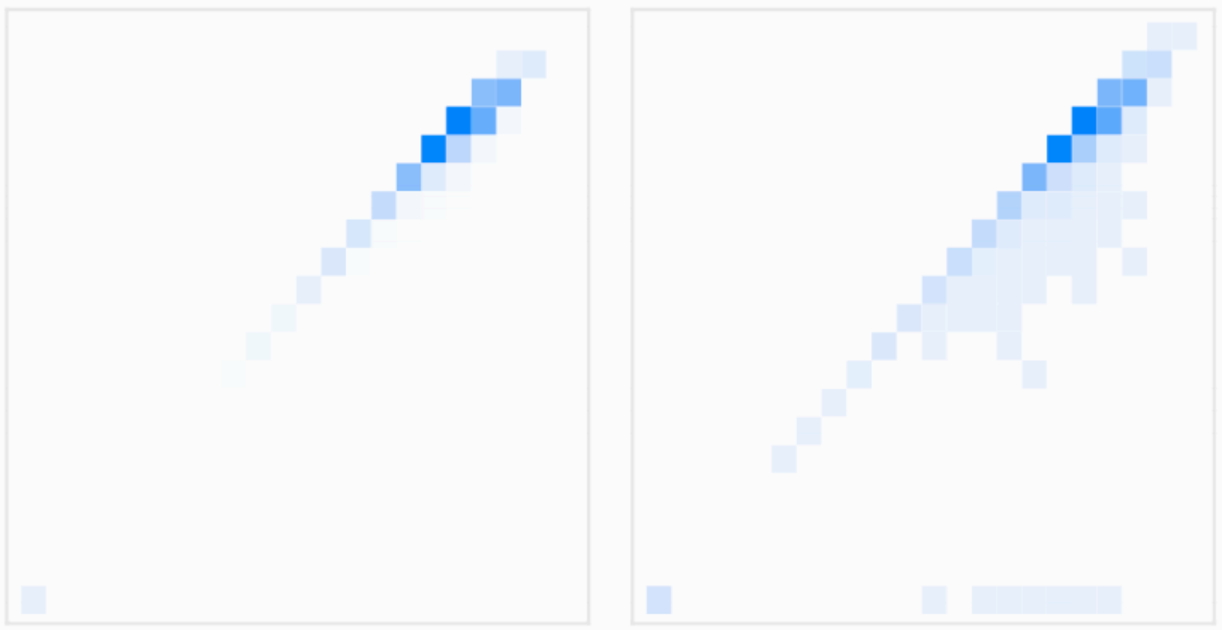



Rectangular Bins

Hex bins better estimate density for 2D plots, but the *improvement is marginal* [Scott 92].

Rectangles support *reuse* and *visual queries*.

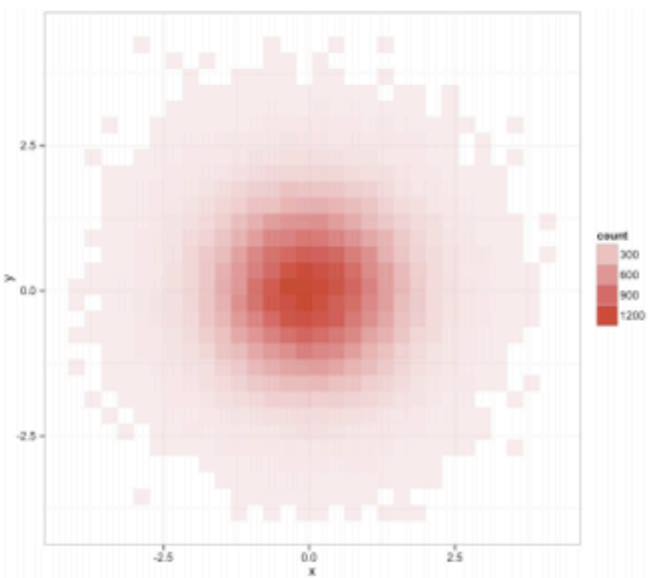
# Color Scale: Discontinuity after Zero



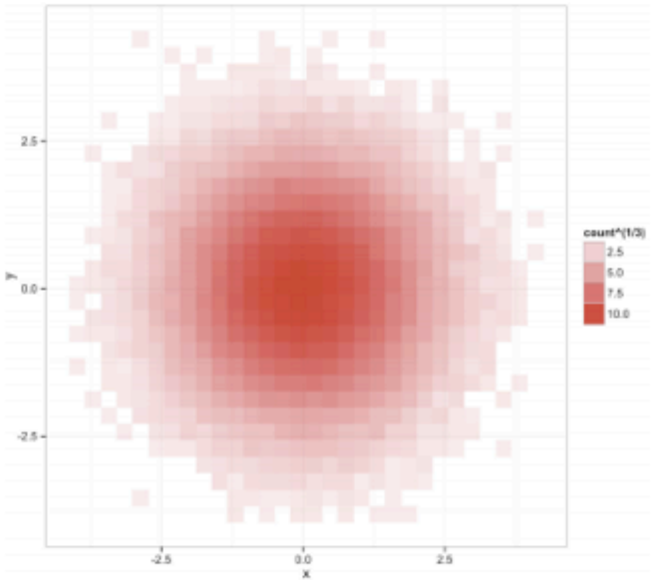
**Standard Color Ramp**
Counts near zero are white.

**Add Discontinuity after Zero**
Counts near zero remain visible.

# Color Ramps / Scale Transforms




**Linear interpolation in RGBA**
is not perceptually linear.



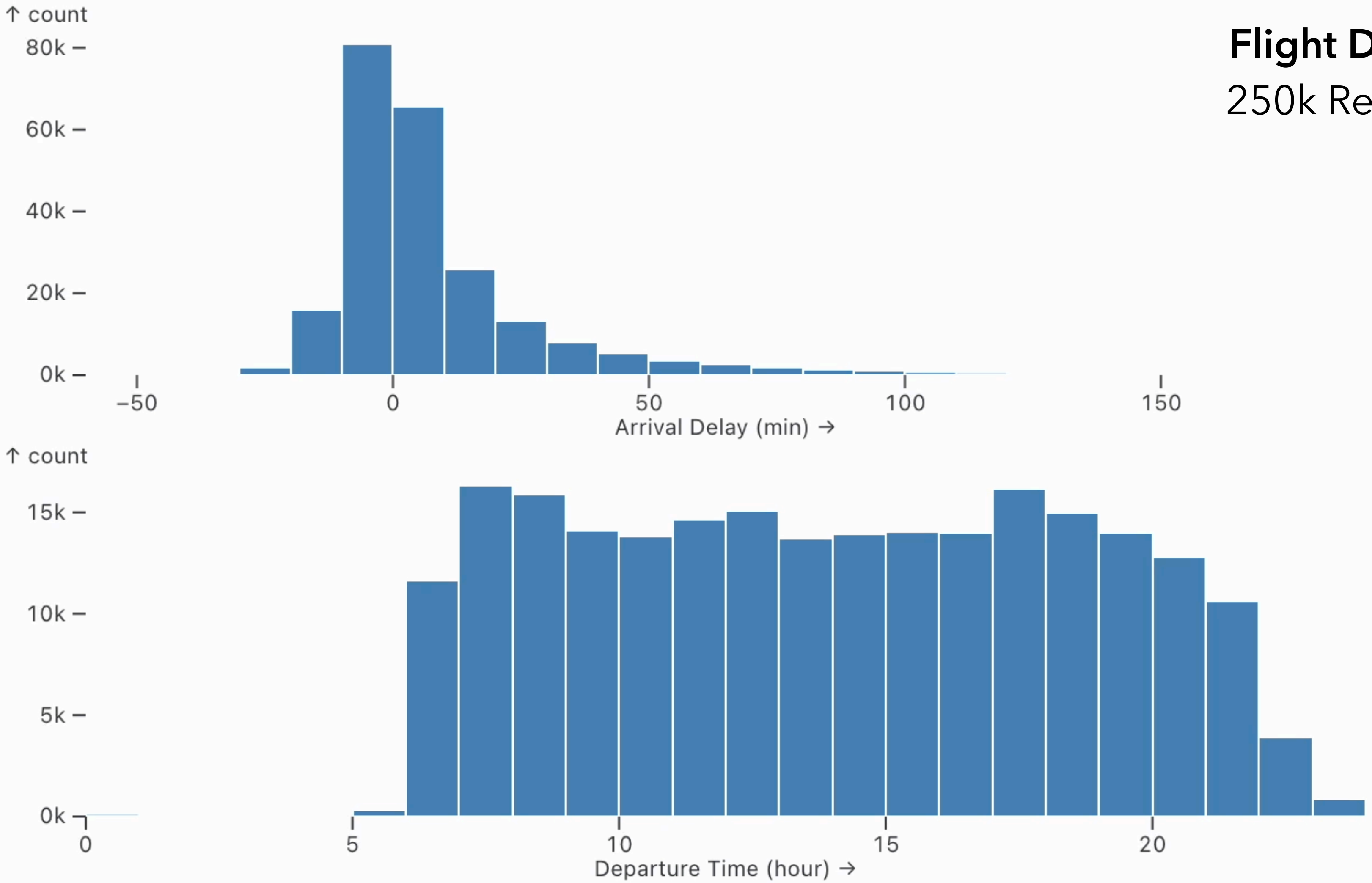

**Perceptual color spaces**
approximate perceptual linearity.

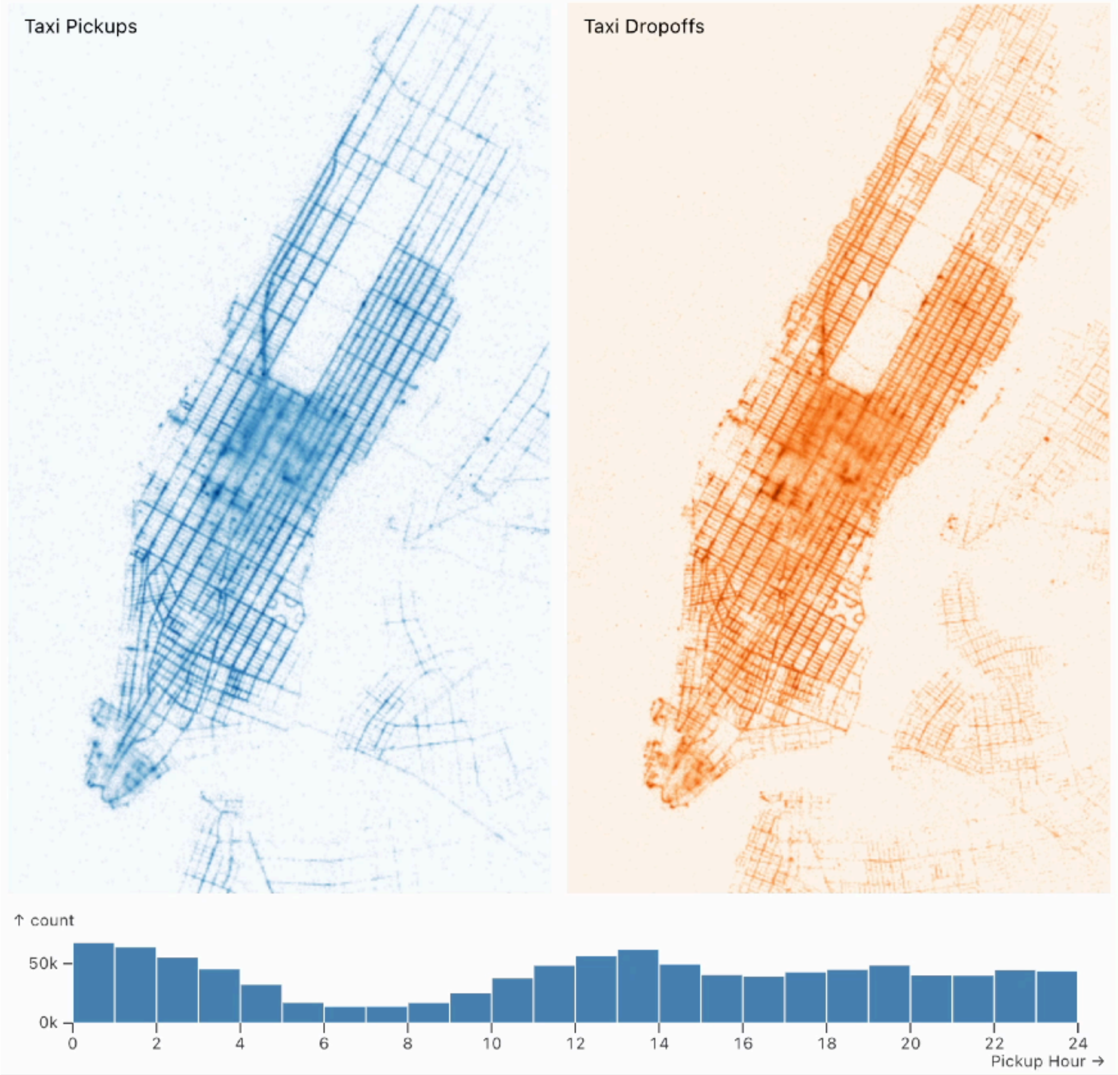
# Questions?

# Scalable Interaction

**Flight Delays**

250k Records

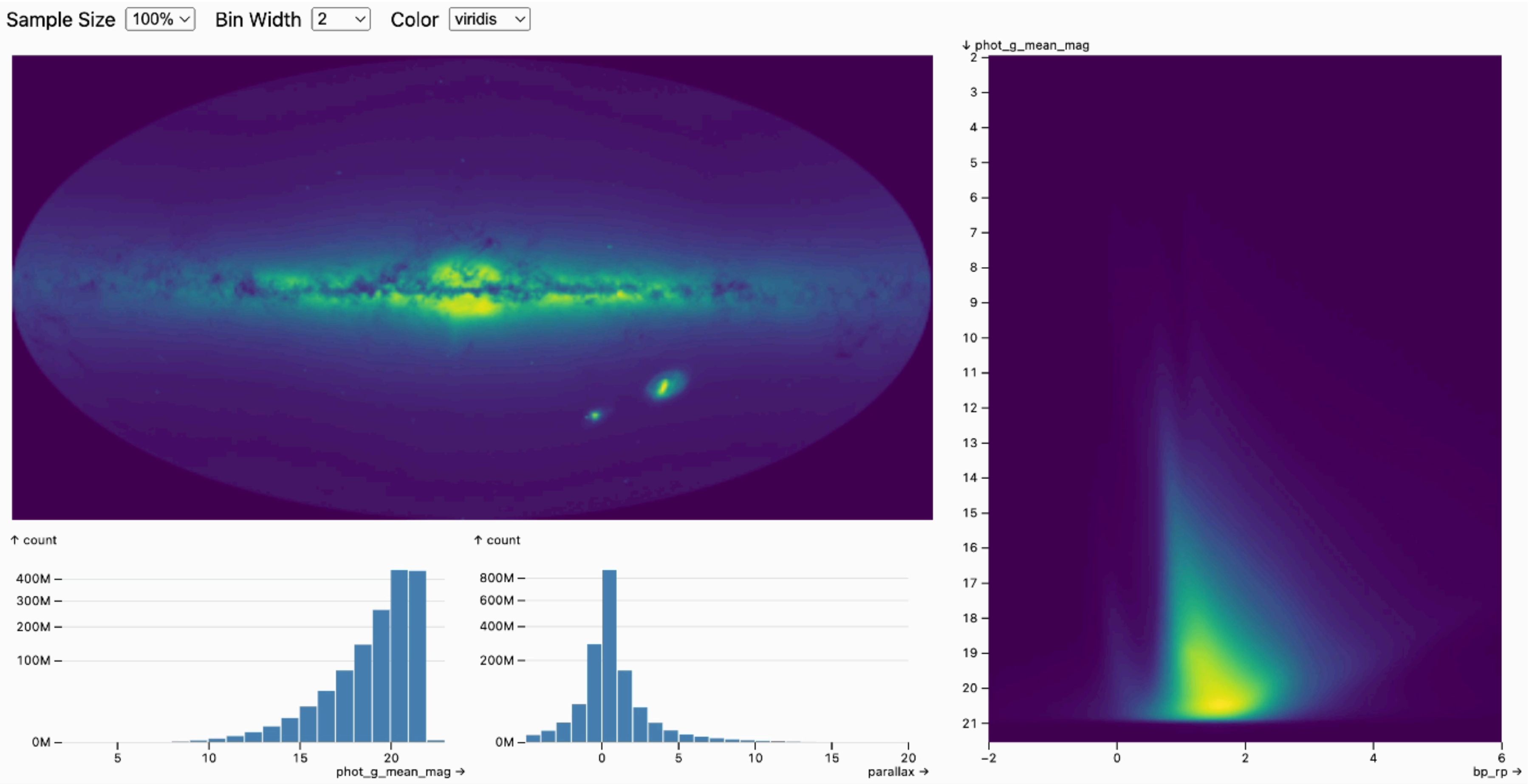




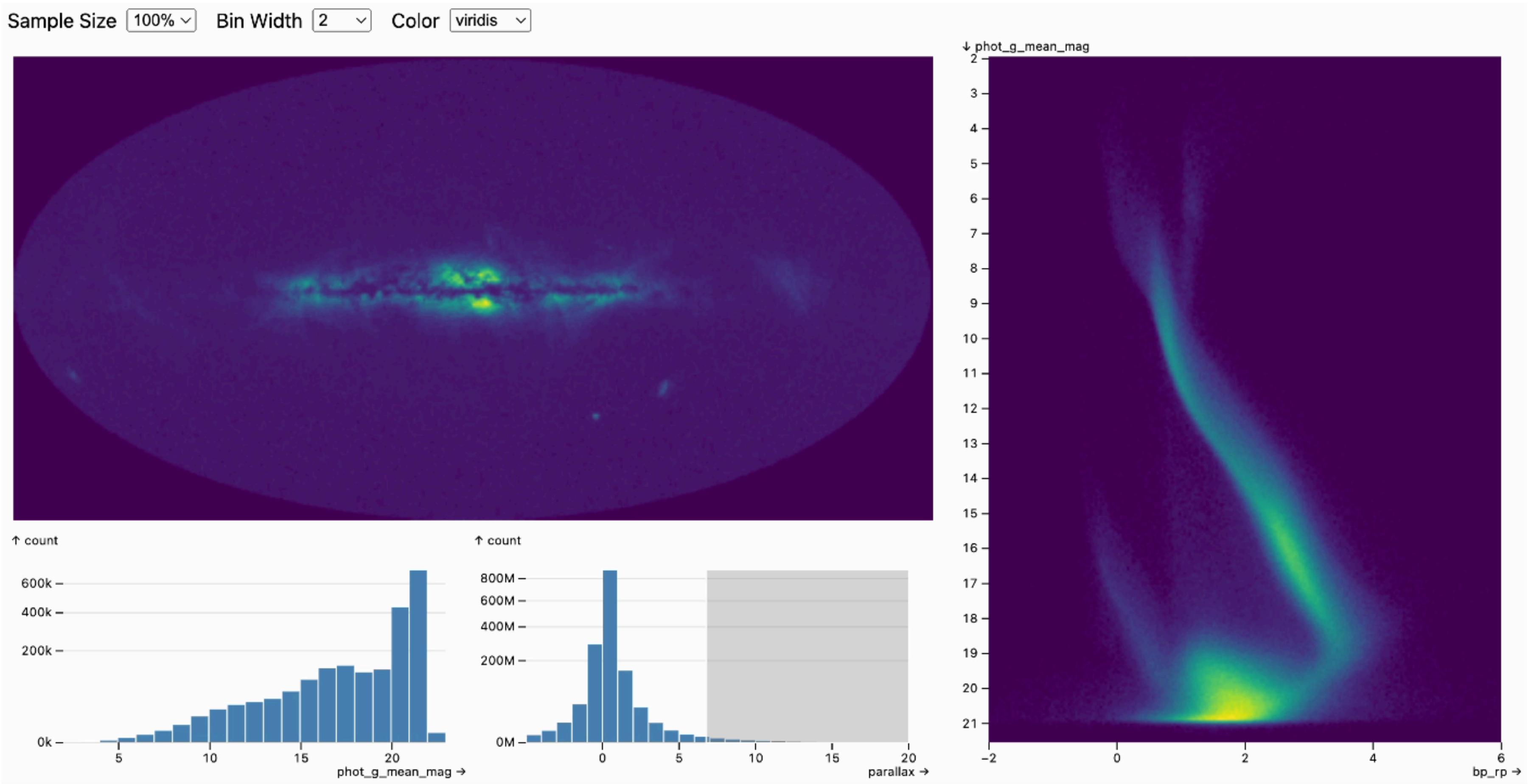

**NY Taxi Rides**

1M Records

Jan 1-3, 2010




**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.

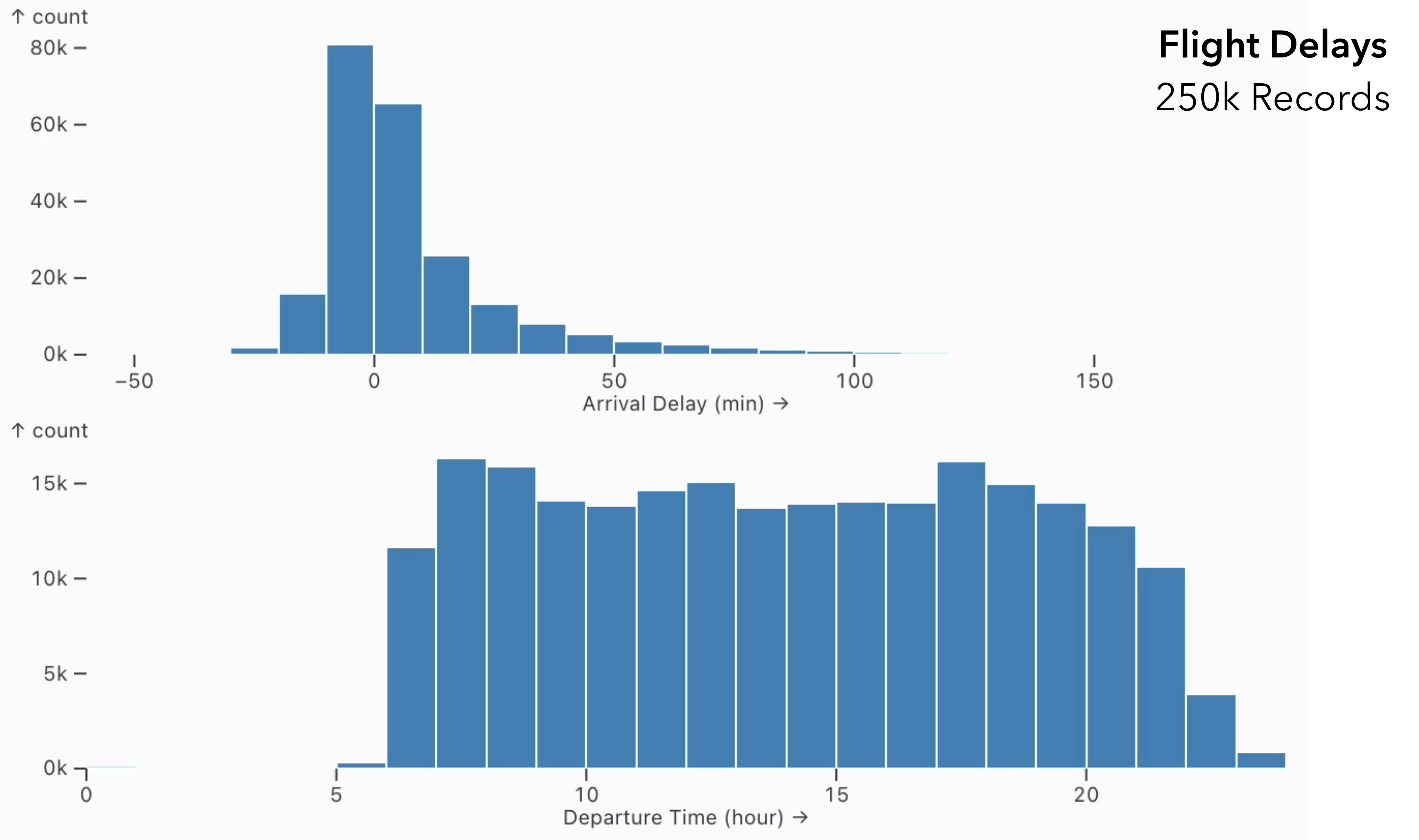
# Interactive Scalability Strategies

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

These strategies are **not** mutually exclusive!

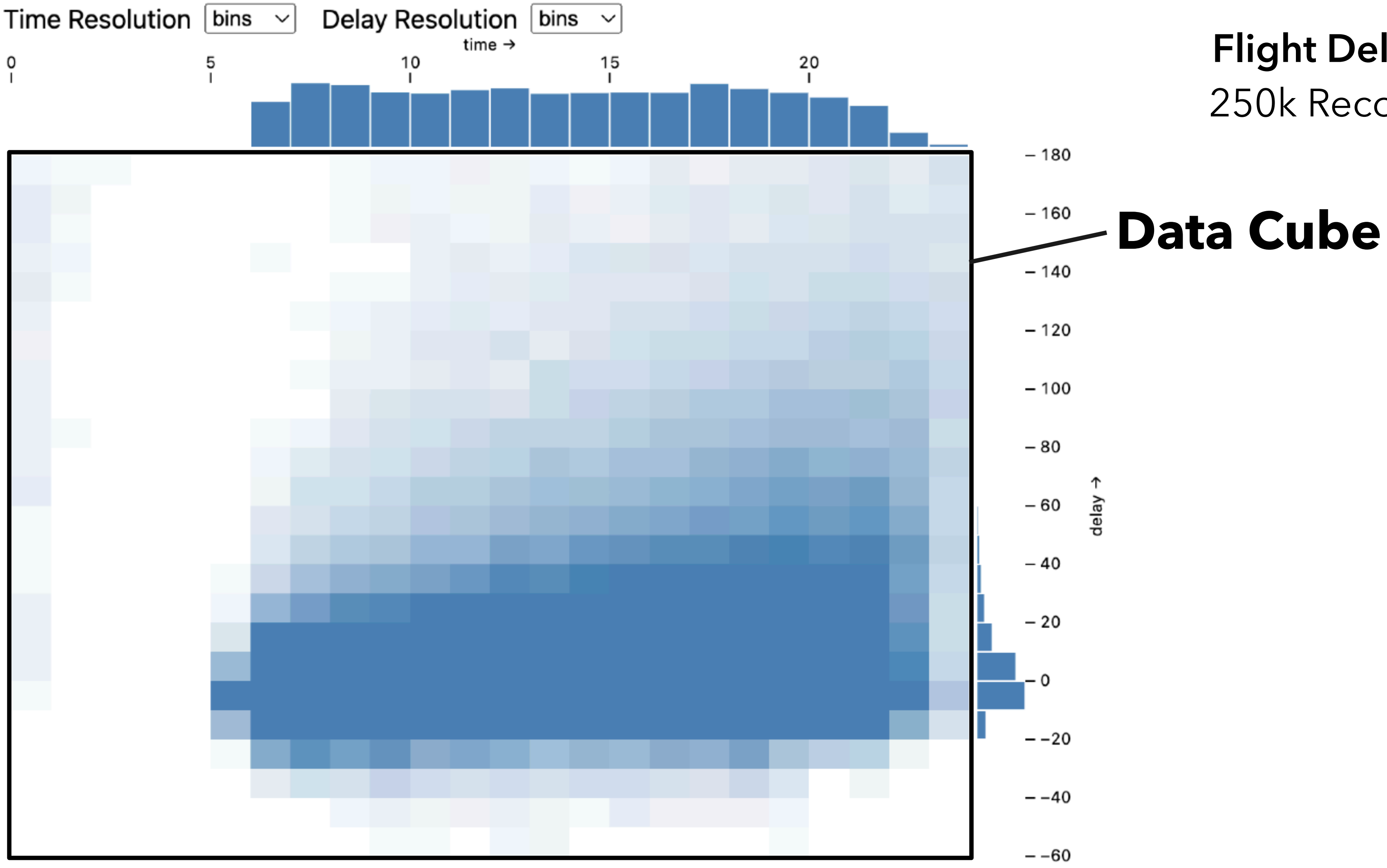
Systems can apply them in tandem.

# Client-Side Indexes


Flight Delays
250k Records
↑ count
80k
60k
40k
20k
0k
−50
0
50
100
150
Arrival Delay (min) →
↑ count
15k
10k
5k
0k
0
5
10
15
20
Departure Time (hour) →

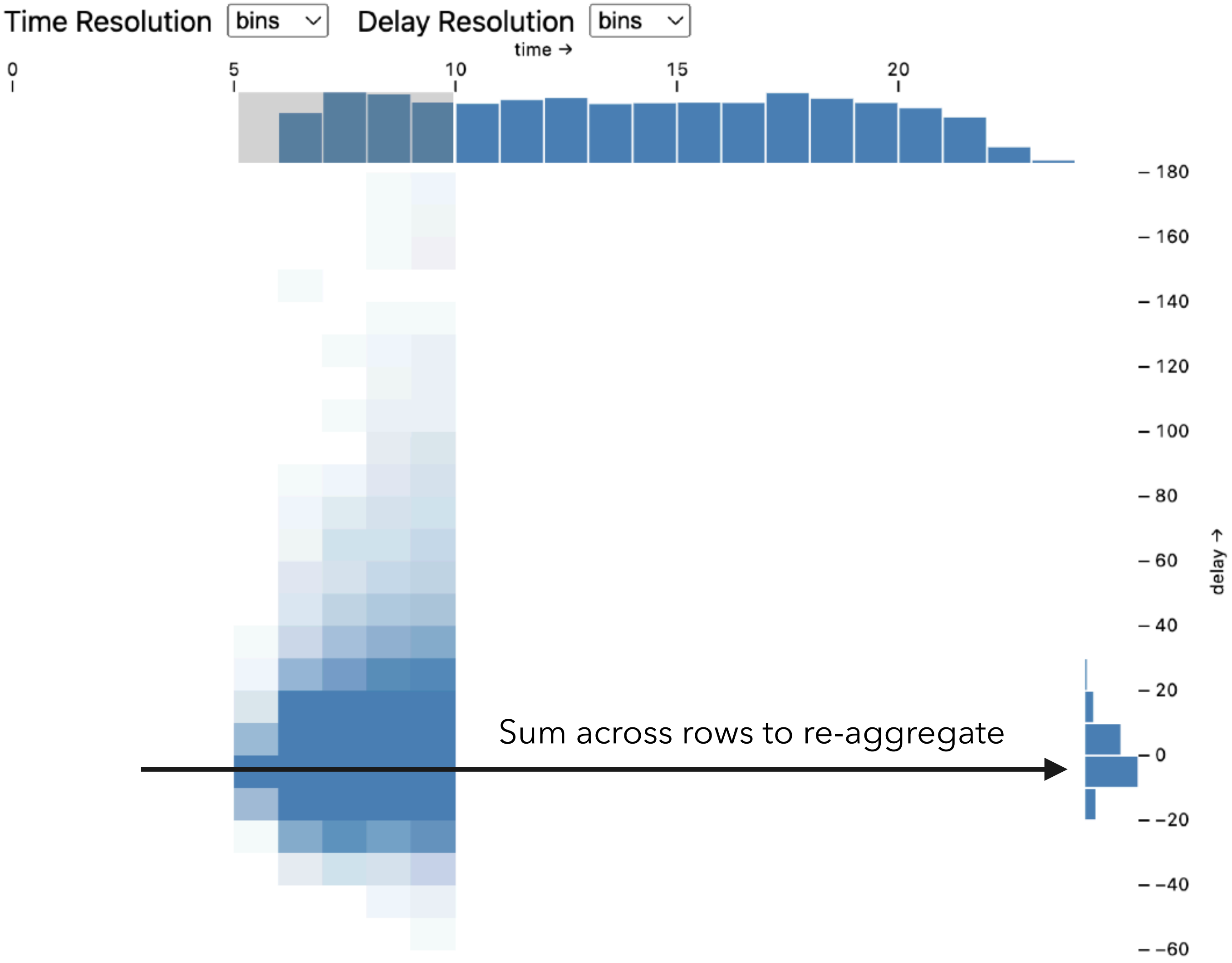
# Flight Delays

250k Records



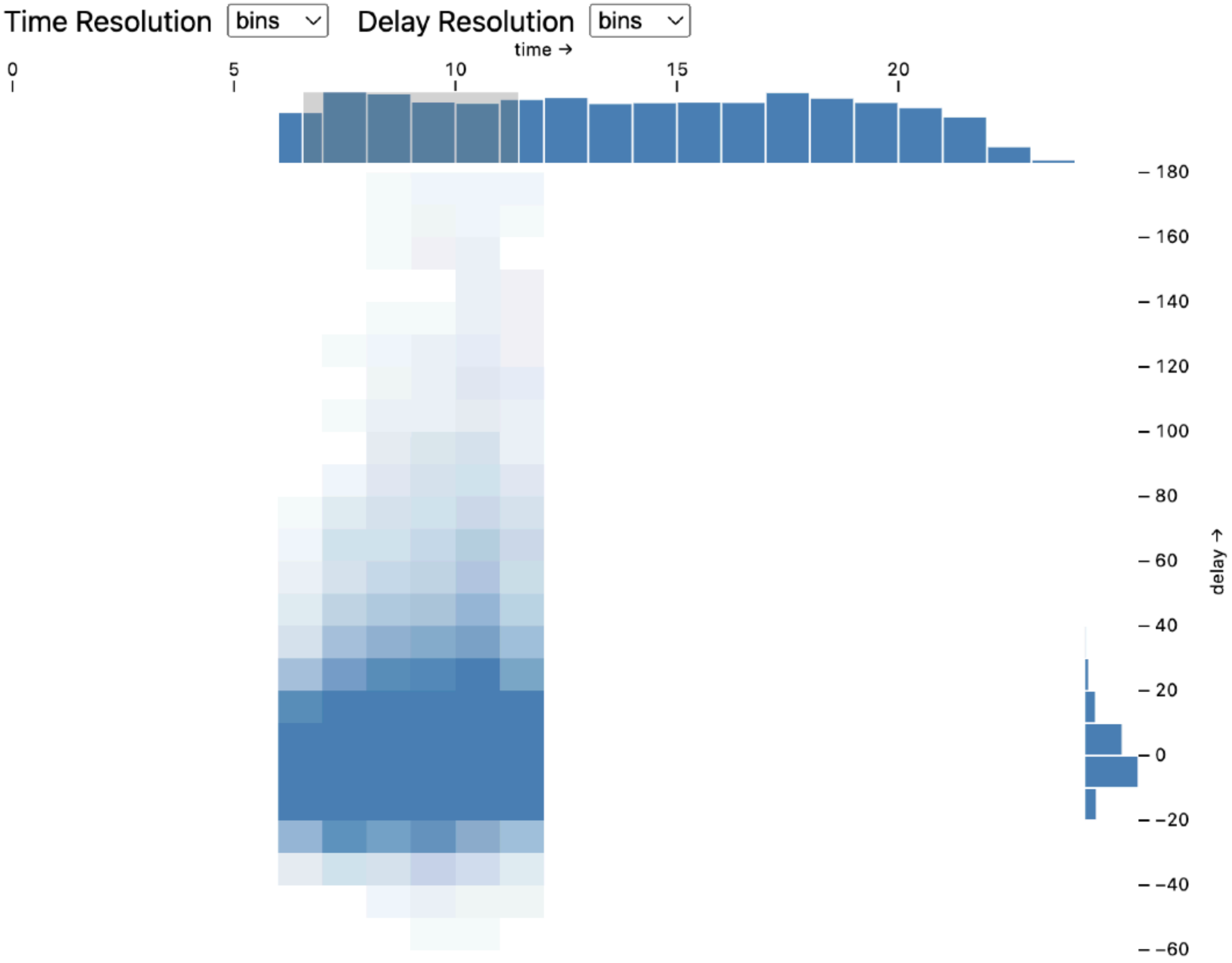
# Flight Delays

250k Records



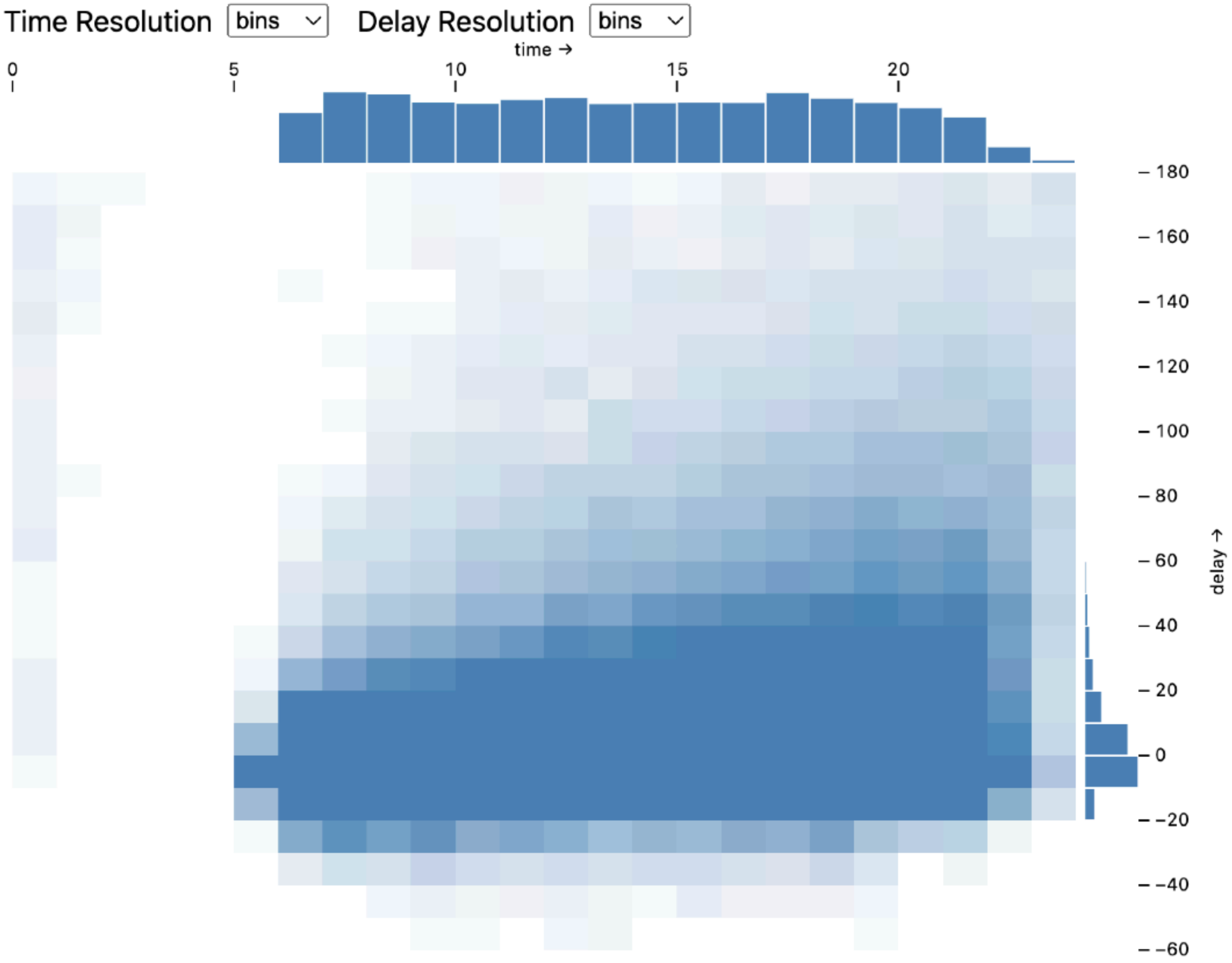
# Flight Delays

250k Records



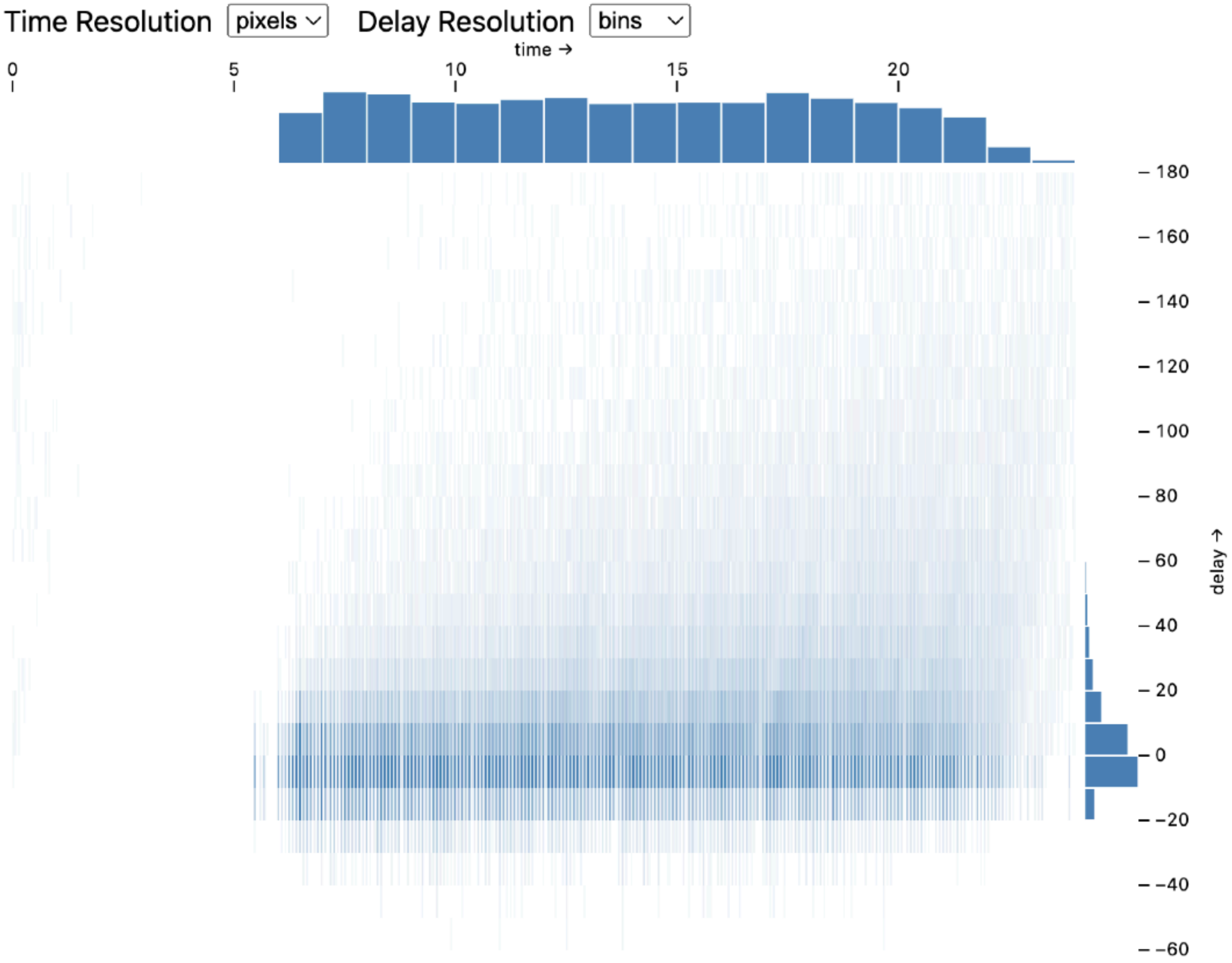
# Flight Delays

250k Records



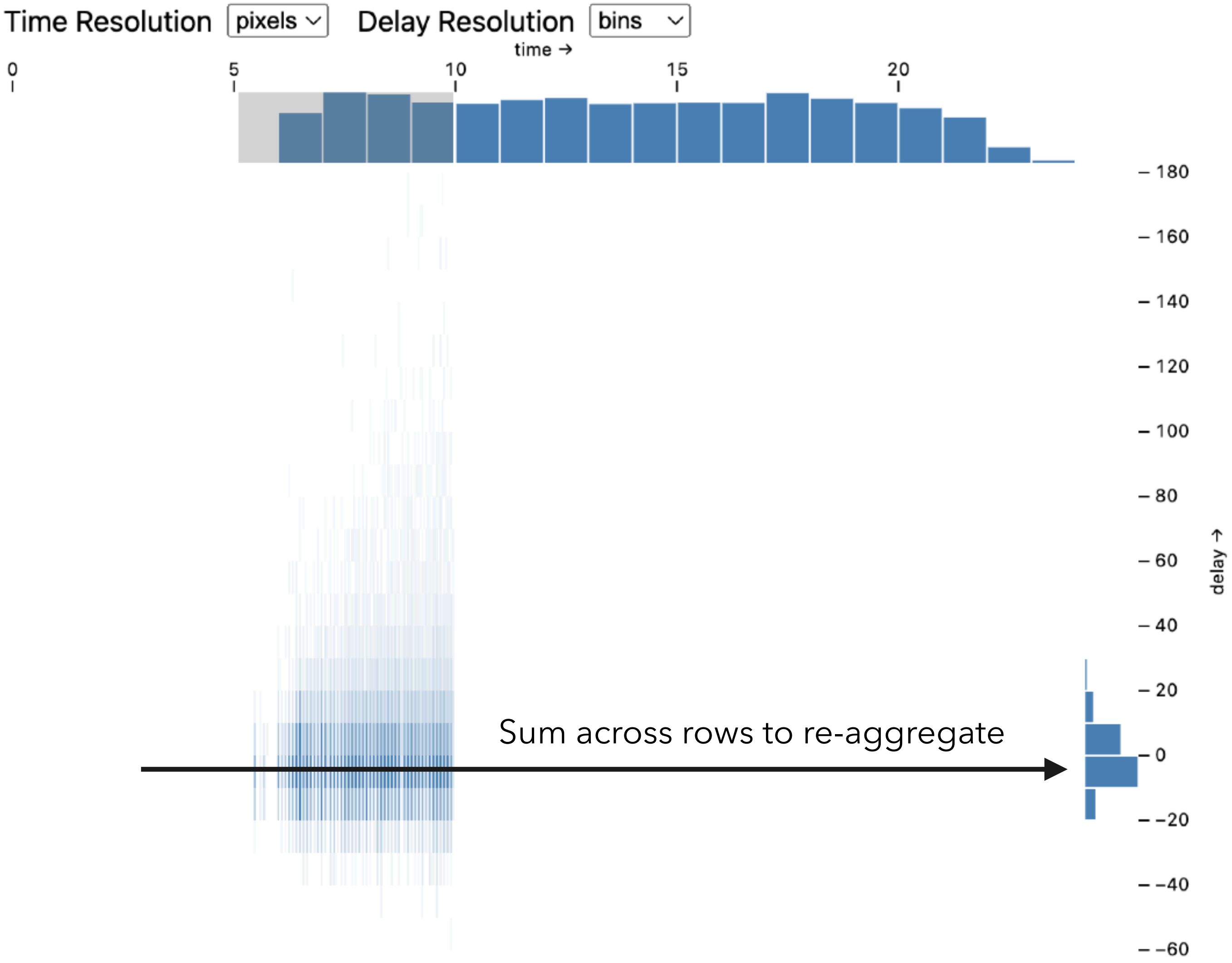
# Flight Delays

250k Records



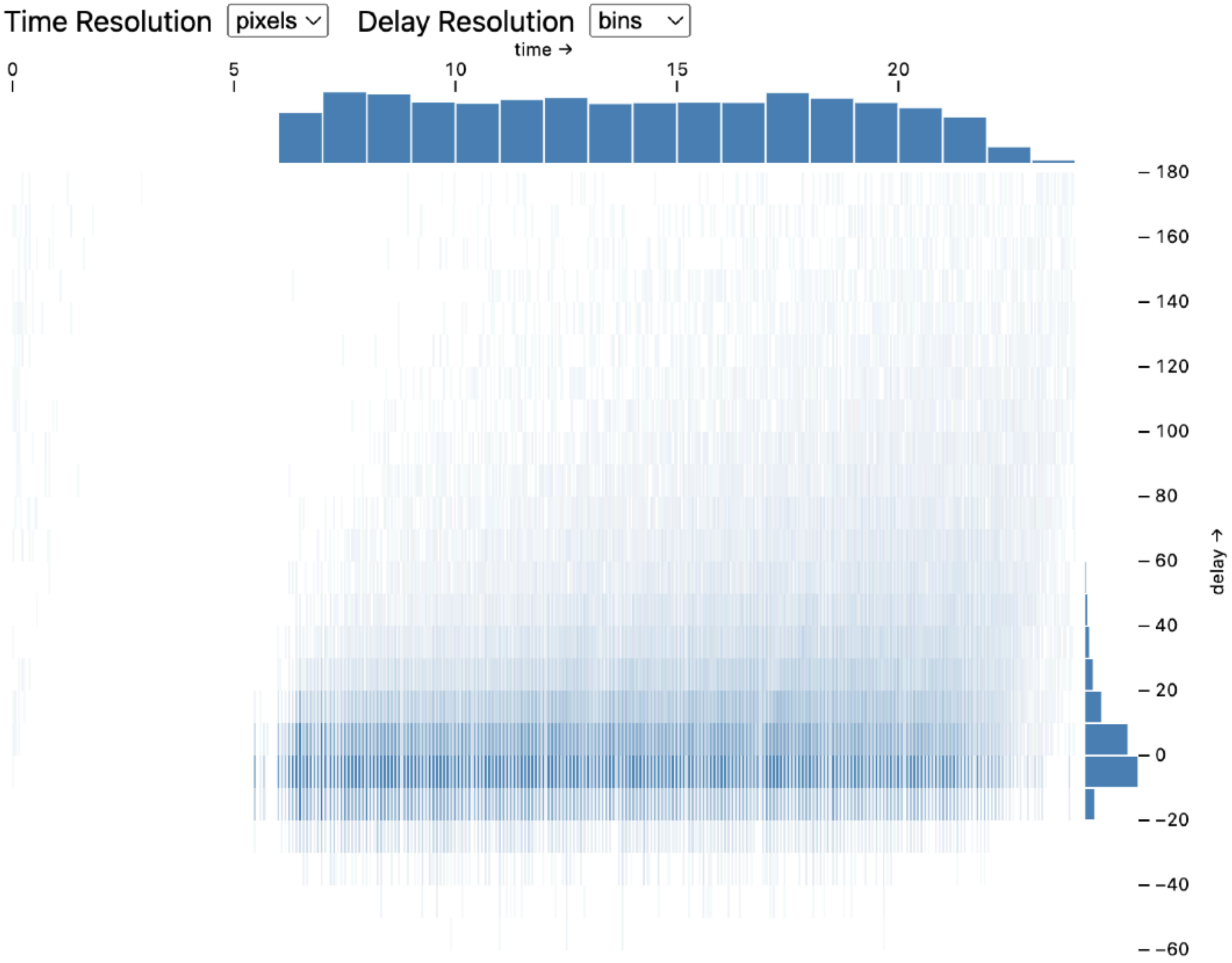
# Flight Delays

250k Records



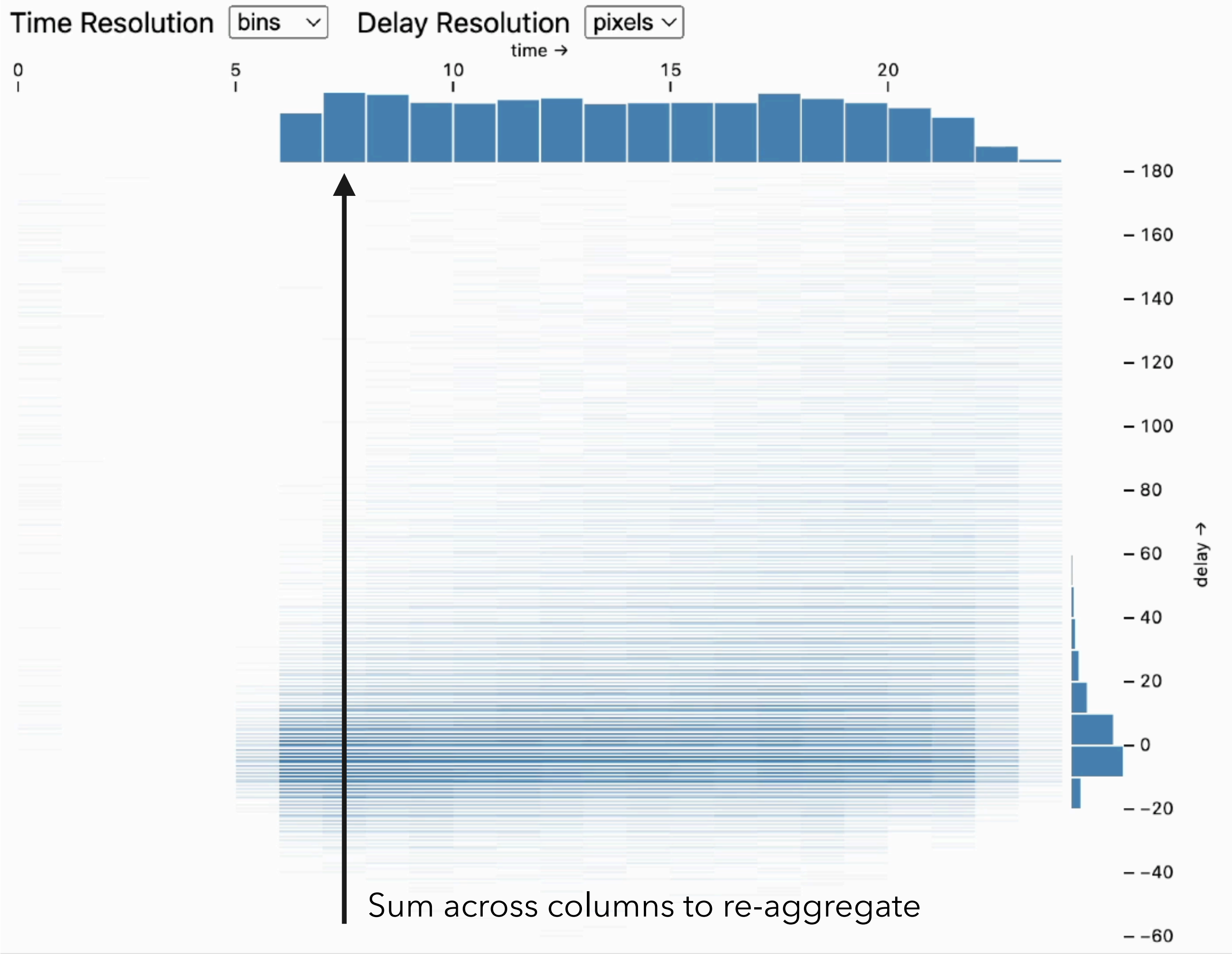
# Flight Delays

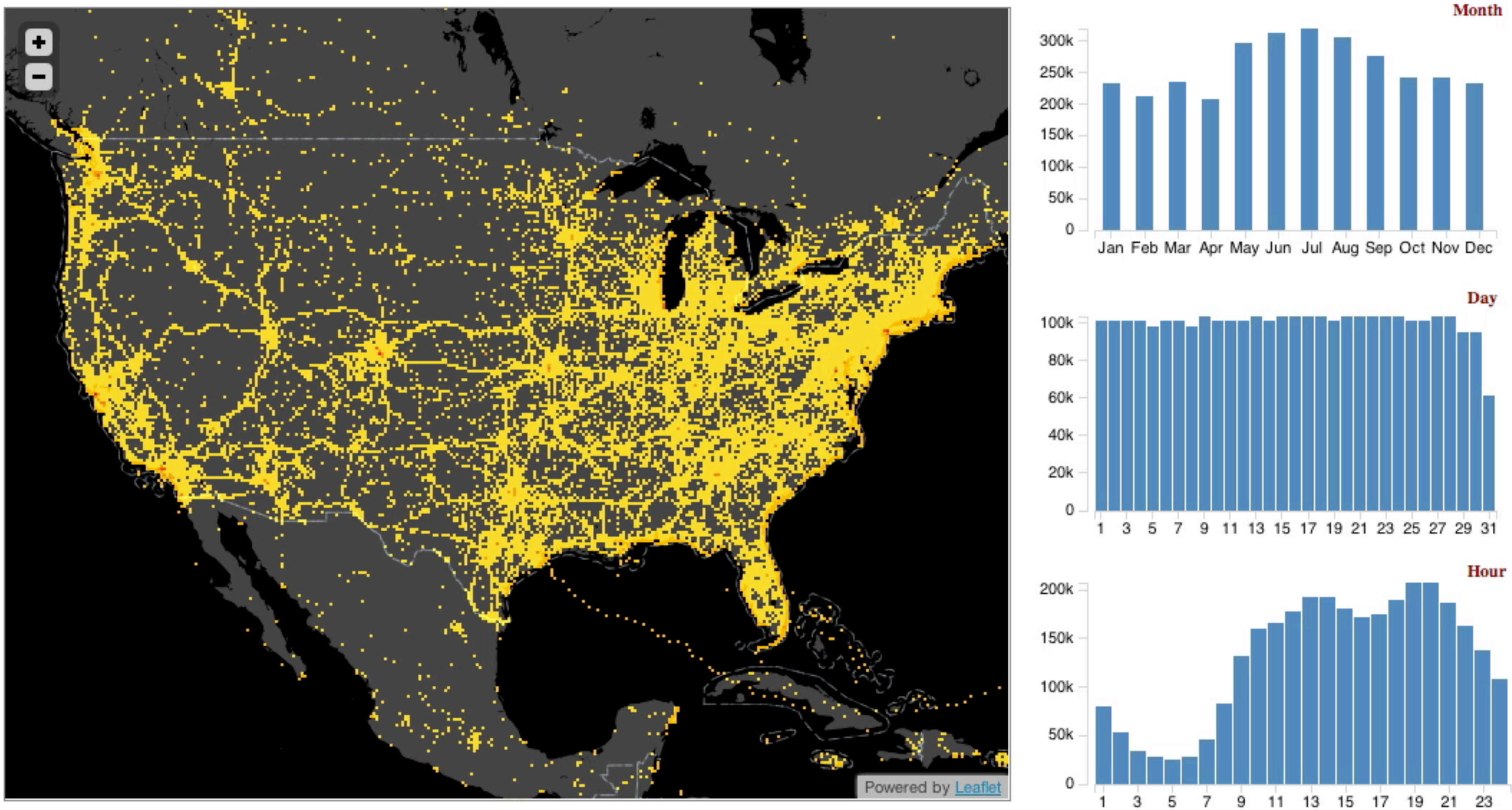
250k Records



**Flight Delays**

250k Records

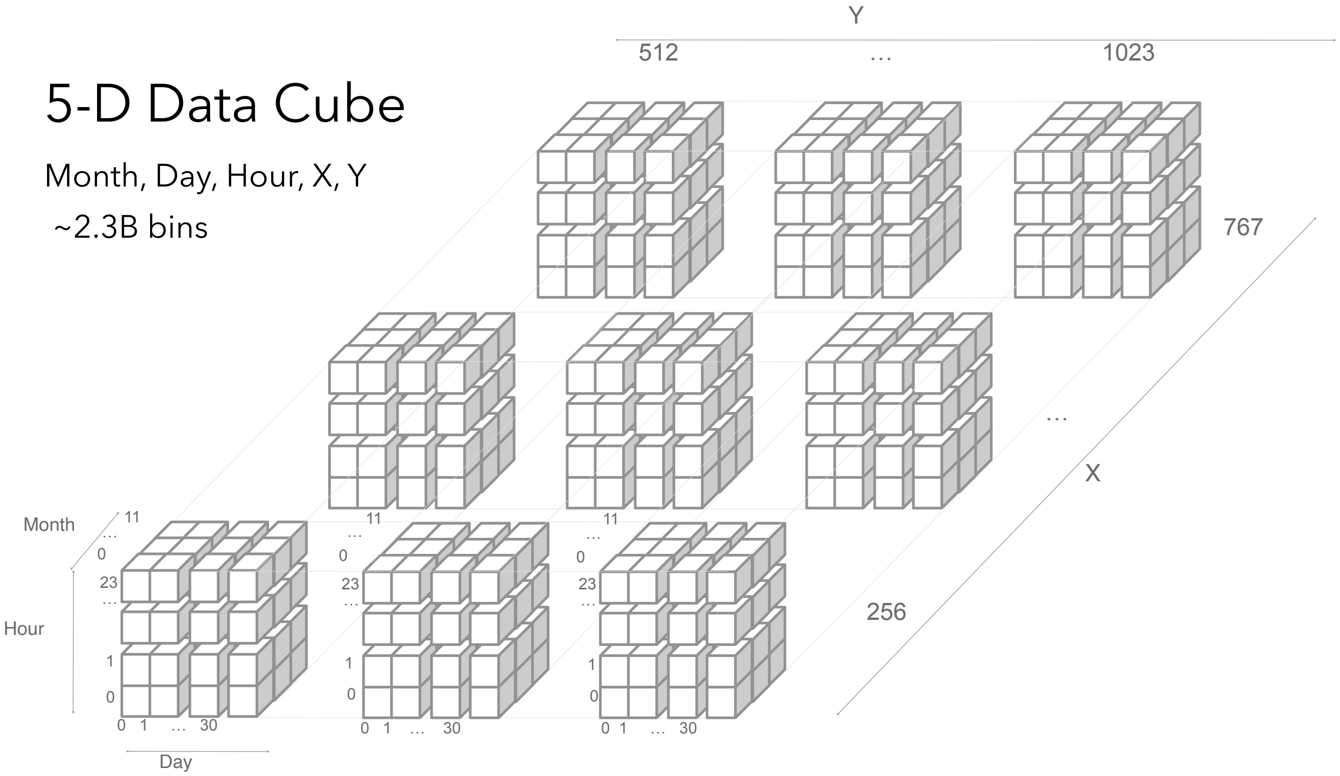



Powered by Leaflet
Month
300k
250k
200k
150k
100k
50k
0
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day
100k
80k
60k
40k
20k
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
Hour
200k
150k
100k
50k
0
1 3 5 7 9 11 13 15 17 19 21 23

# 5-D Data Cube

Month, Day, Hour, X, Y

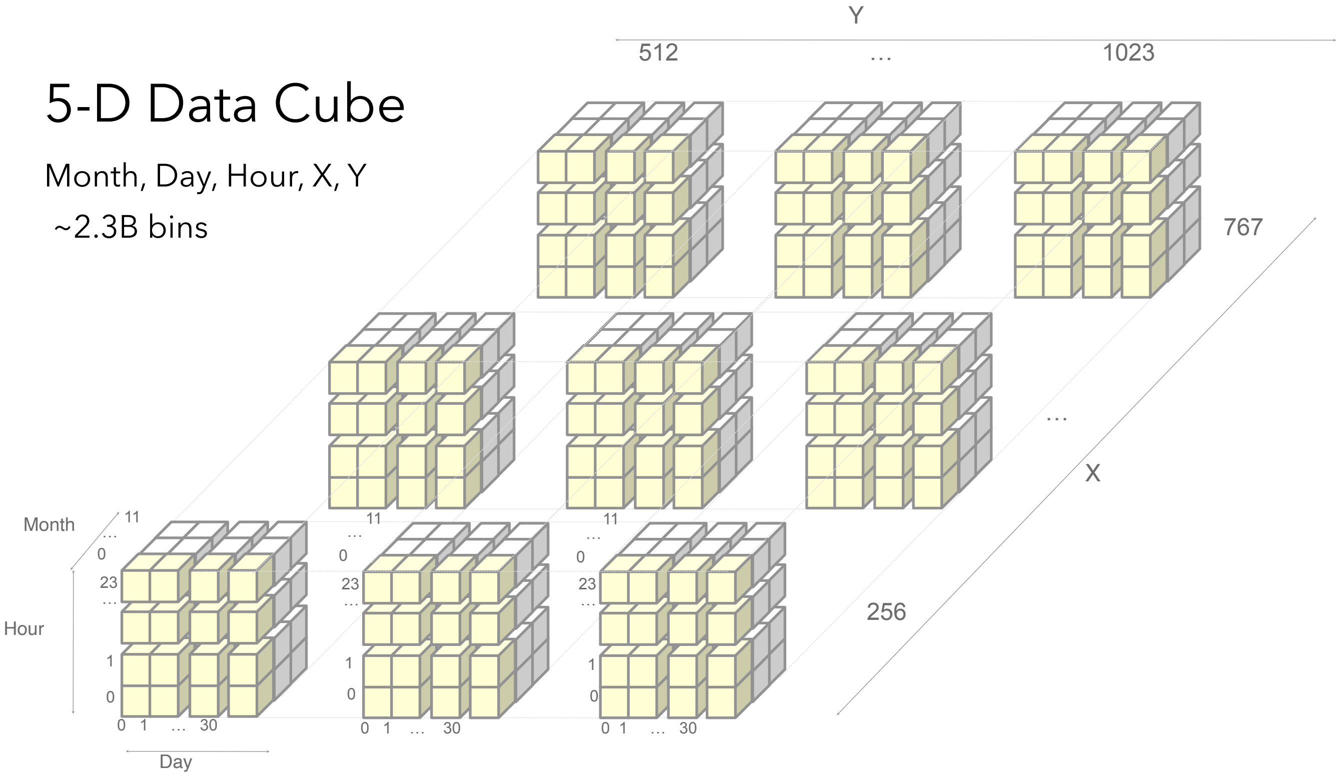
~2.3B bins

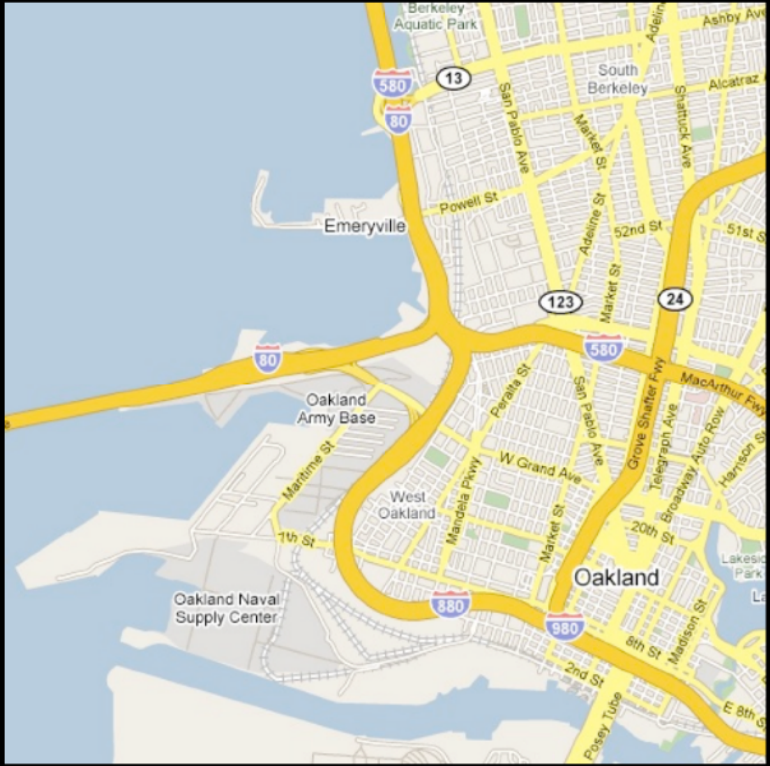


# 5-D Data Cube

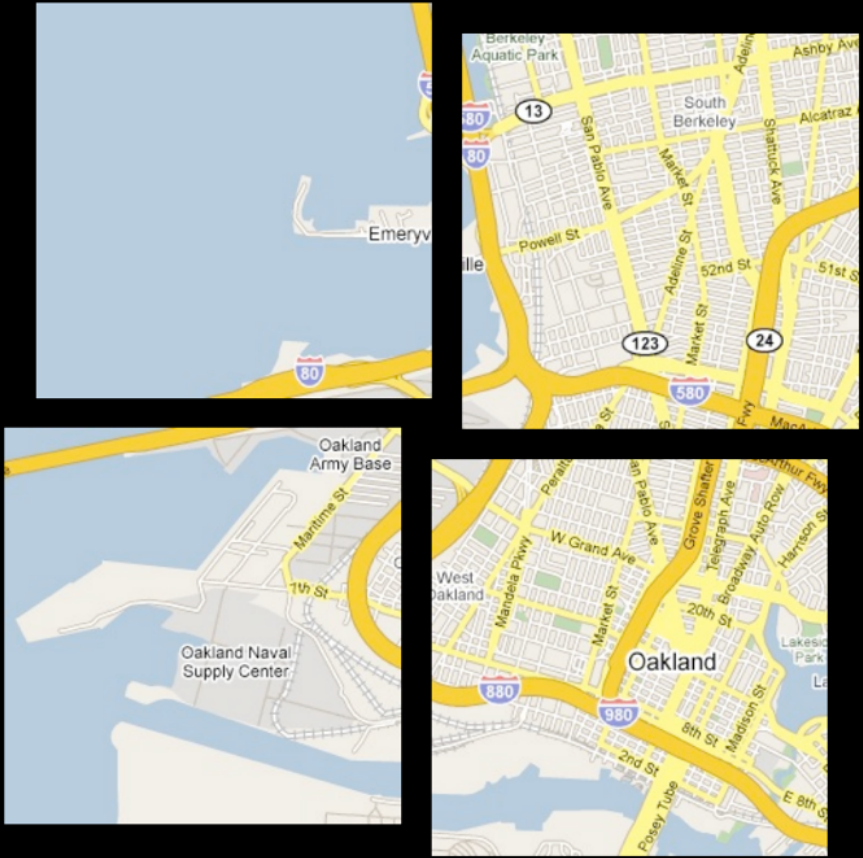
Month, Day, Hour, X, Y

~2.3B bins



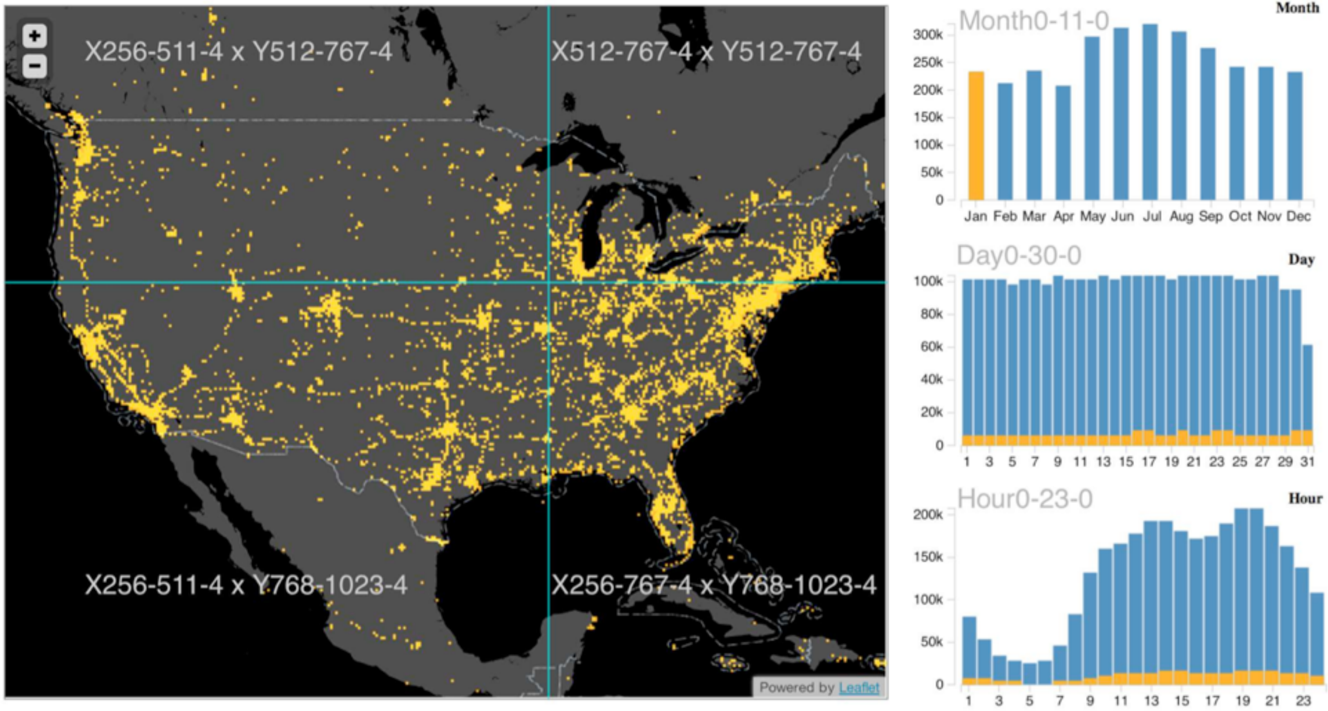


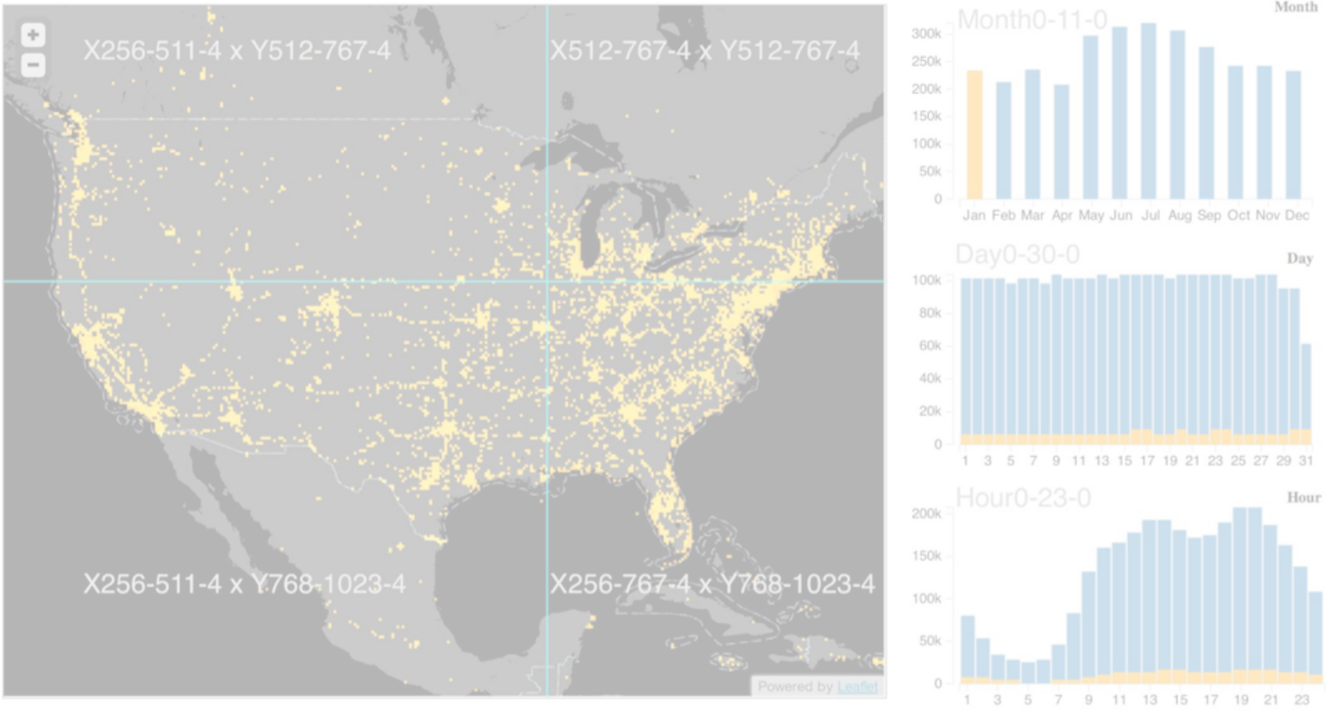
Berkeley
Aquatic Park
580
13
80
South
Berkeley
Ashby Ave
Alcatraz
San Pablo Ave
Market St
Shattuck Ave
Powell St
Emeryville
Adeline St
52nd St
51st
123
24
Market St
80
580
Oakland
Army Base
Peralta St
MacArthur Fwy
Grove Shafter Fwy
San Pablo Ave
Telegraph Ave
Broadway Auto Row
Maritime St
W Grand Ave
West
Oakland
Mandela Pkwy
7th St
Market St
20th St
Oakland
Oakland Naval
Supply Center
880
980
8th St
Madison St
2nd St
Posey Tube
E 8th St


Emeryv
80
Berkeley
Aquatic Park
Ashby Ave
South
Berkeley
Alcatraz
13
580
80
San Pablo Ave
Market St
Shattuck Ave
Powell St
Adeline St
52nd St
51st S
123
24
580
Oakland
Army Base
Maritime St
7th St
Oakland Naval
Supply Center
Peralta
San Pablo Ave
Grove Shafter
Telegraph Ave
Broadway Auto Row
Harrison St
W Grand Ave
West
Oakland
Mandela Pkwy
Market St
20th St
Lakeside
Park
Oakland
880
980
8th St
Madison St
2nd St
E 8th St
Posey Tube

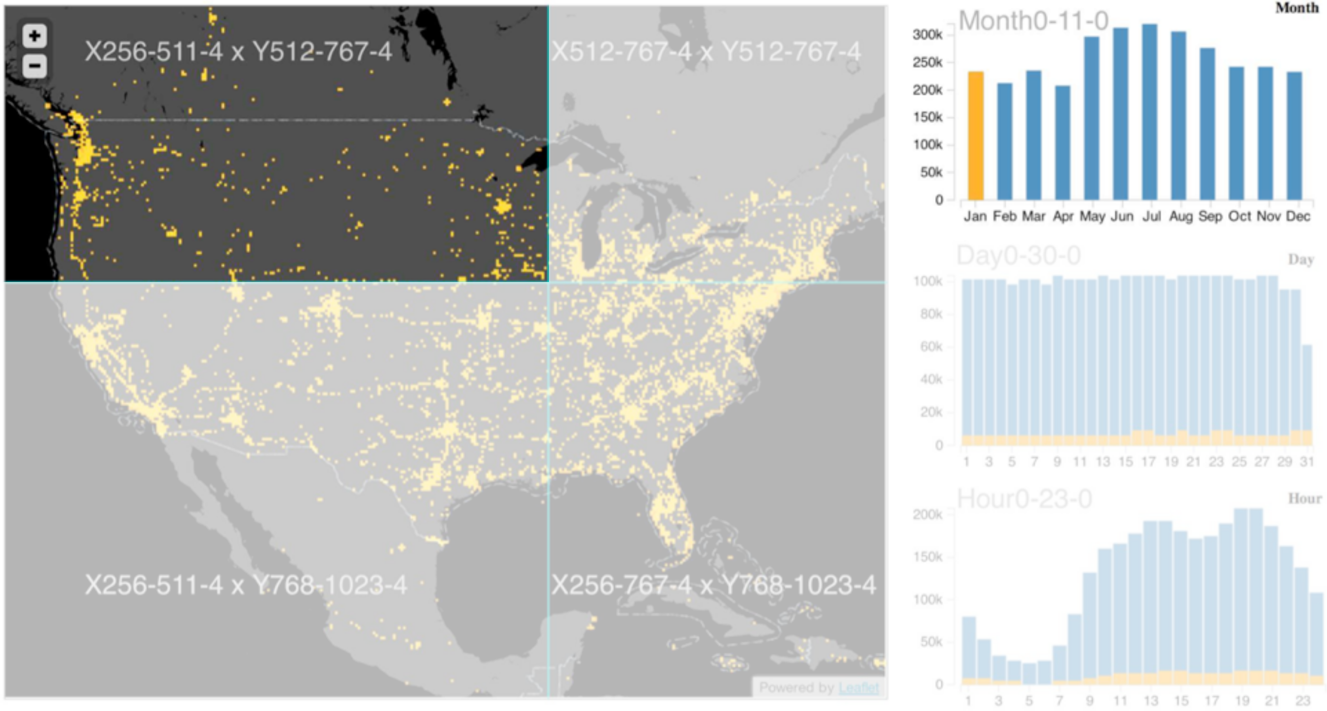
**Multivariate Data Tiles**

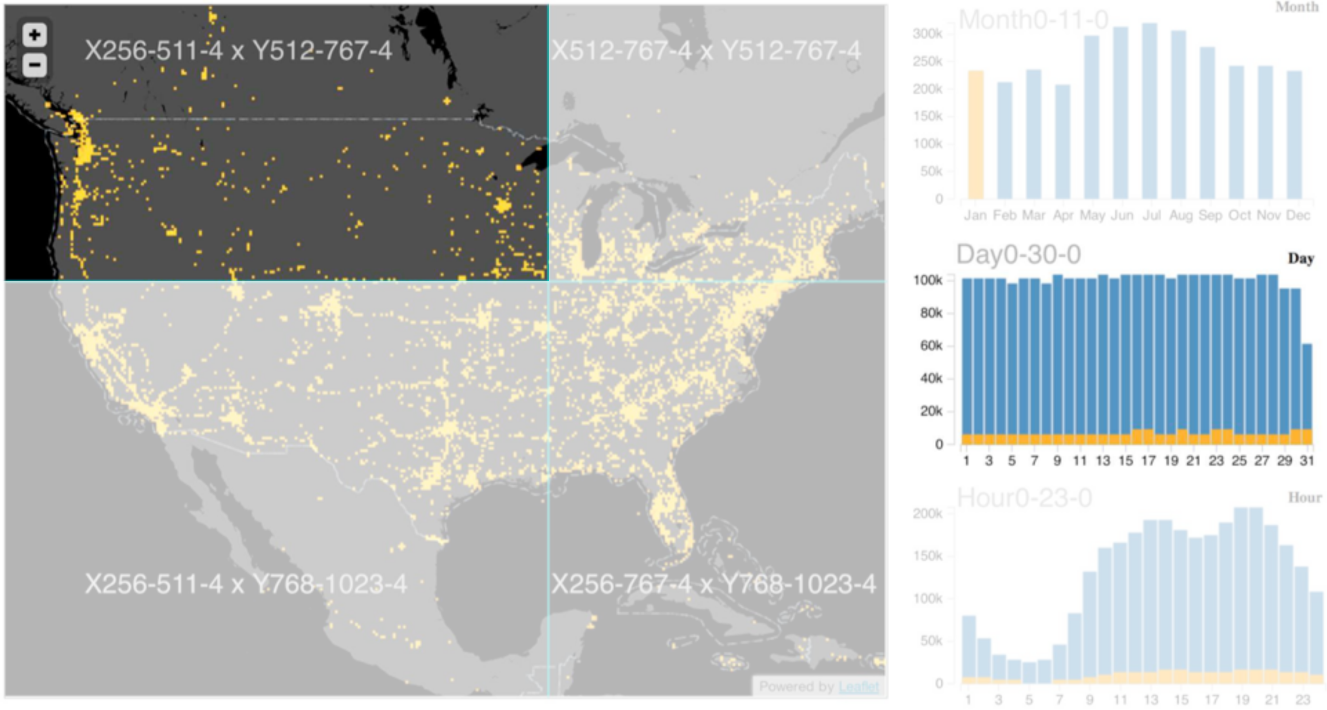
1. Send data, not pixels
2. Embed multi-dim data

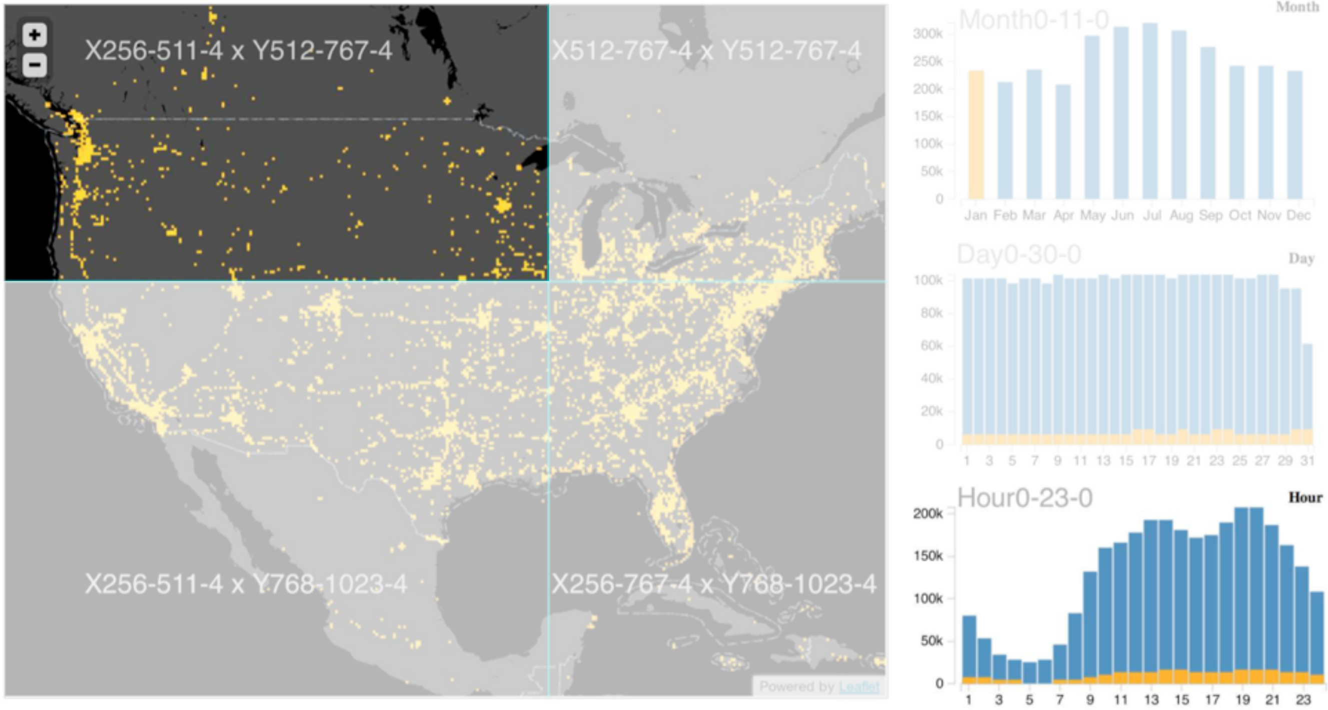

X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month0-11-0
Month
300k
250k
200k
150k
100k
50k
0
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day0-30-0
Day
100k
80k
60k
40k
20k
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
Hour0-23-0
Hour
200k
150k
100k
50k
0
1 3 5 7 9 11 13 15 17 19 21 23

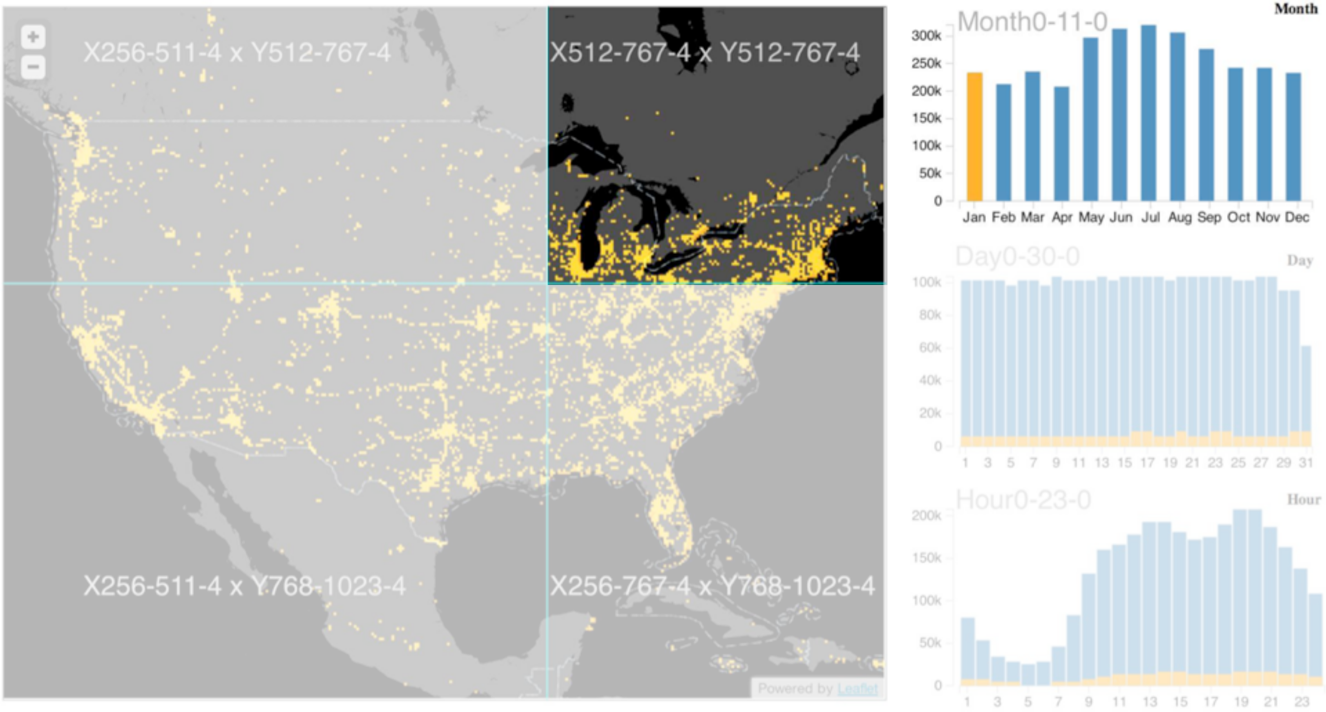


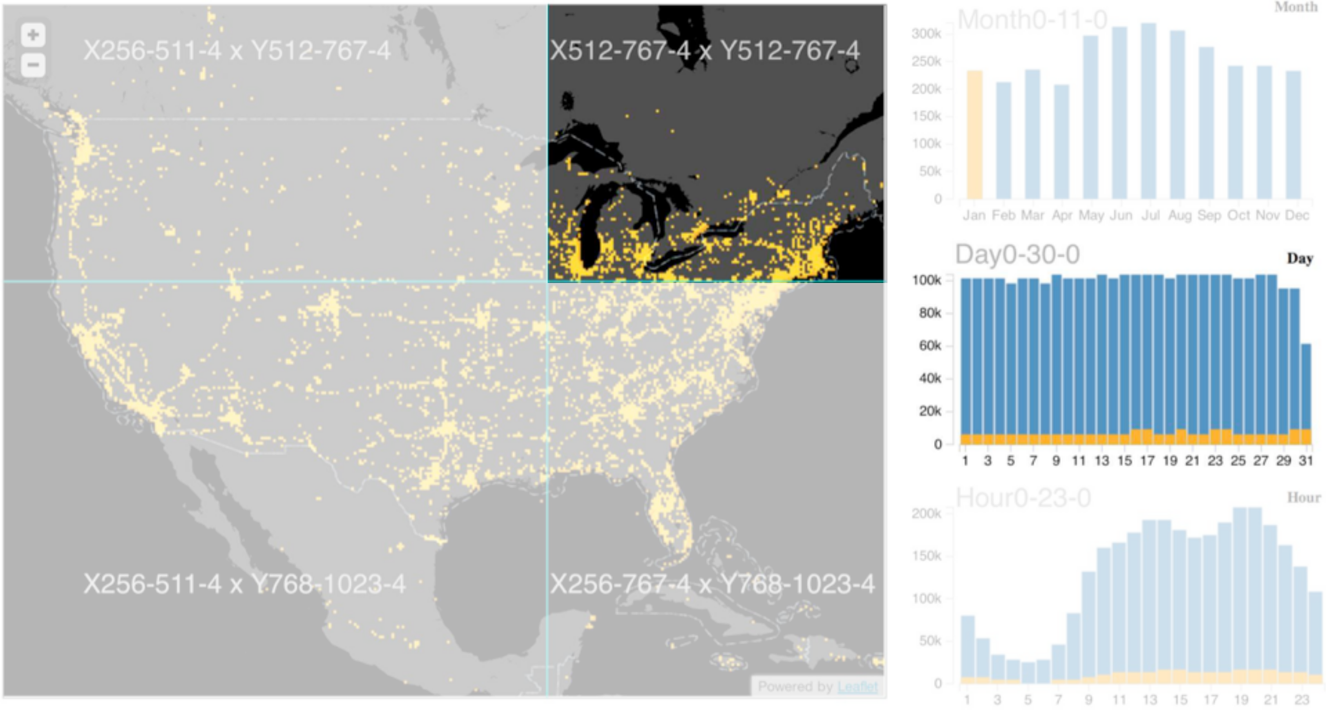
X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month0-11-0
Month
Day0-30-0
Day
Hour0-23-0
Hour


X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month0-11-0
Month
300k
250k
200k
150k
100k
50k
0
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day0-30-0
Day
100k
80k
60k
40k
20k
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
Hour0-23-0
Hour
200k
150k
100k
50k
0
1 3 5 7 9 11 13 15 17 19 21 23

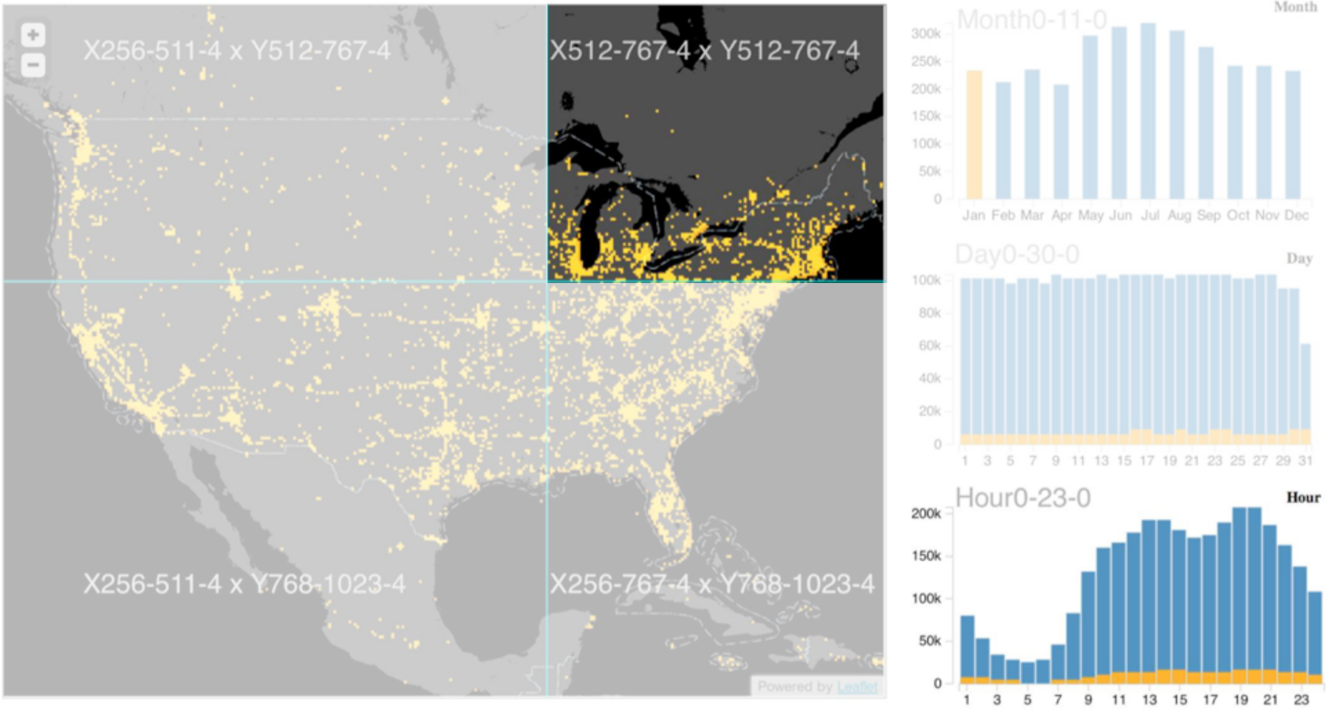

X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month0-11-0
Month
0 50k 100k 150k 200k 250k 300k
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day0-30-0
Day
0 20k 40k 60k 80k 100k
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
Hour0-23-0
Hour
0 50k 100k 150k 200k
1 3 5 7 9 11 13 15 17 19 21 23


X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month0-11-0
Month
300k
250k
200k
150k
100k
50k
0
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day0-30-0
Day
100k
80k
60k
40k
20k
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
Hour0-23-0
Hour
200k
150k
100k
50k
0
1 3 5 7 9 11 13 15 17 19 21 23

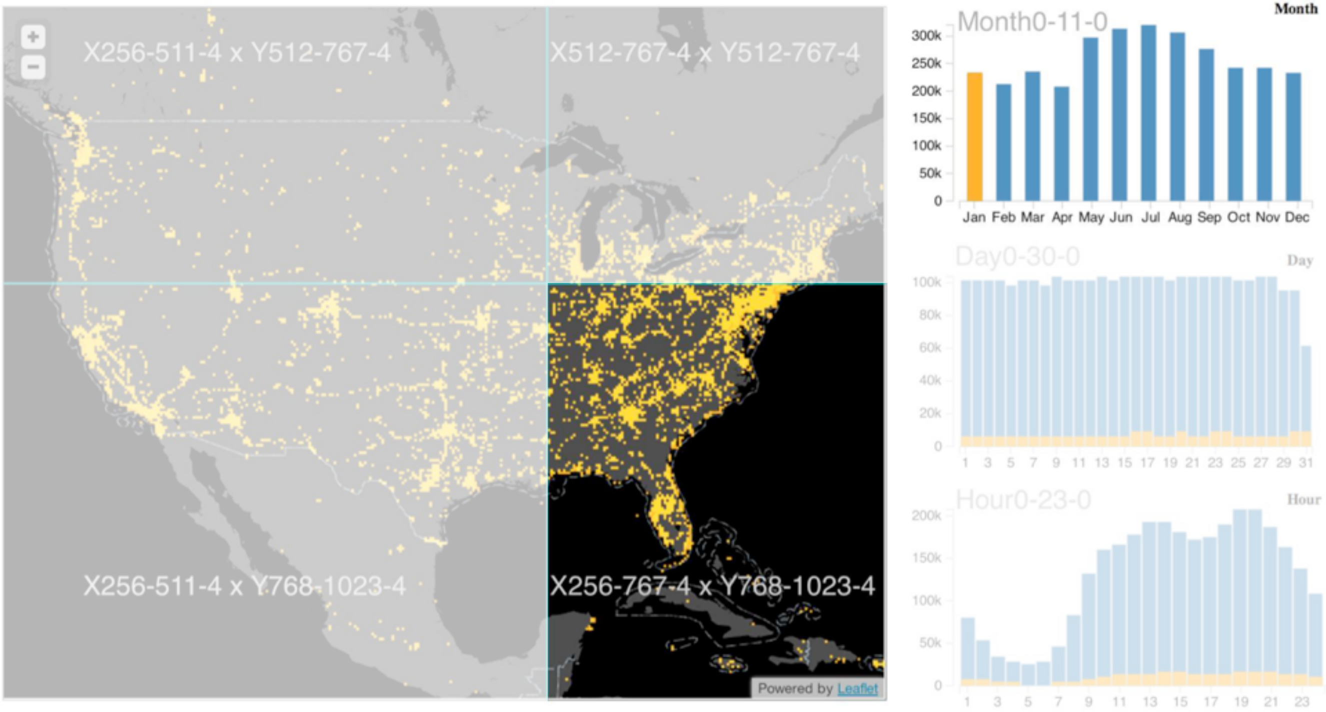

X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month0-11-0
Month
300k
250k
200k
150k
100k
50k
0
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day0-30-0
Day
100k
80k
60k
40k
20k
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
Hour0-23-0
Hour
200k
150k
100k
50k
0
1 3 5 7 9 11 13 15 17 19 21 23

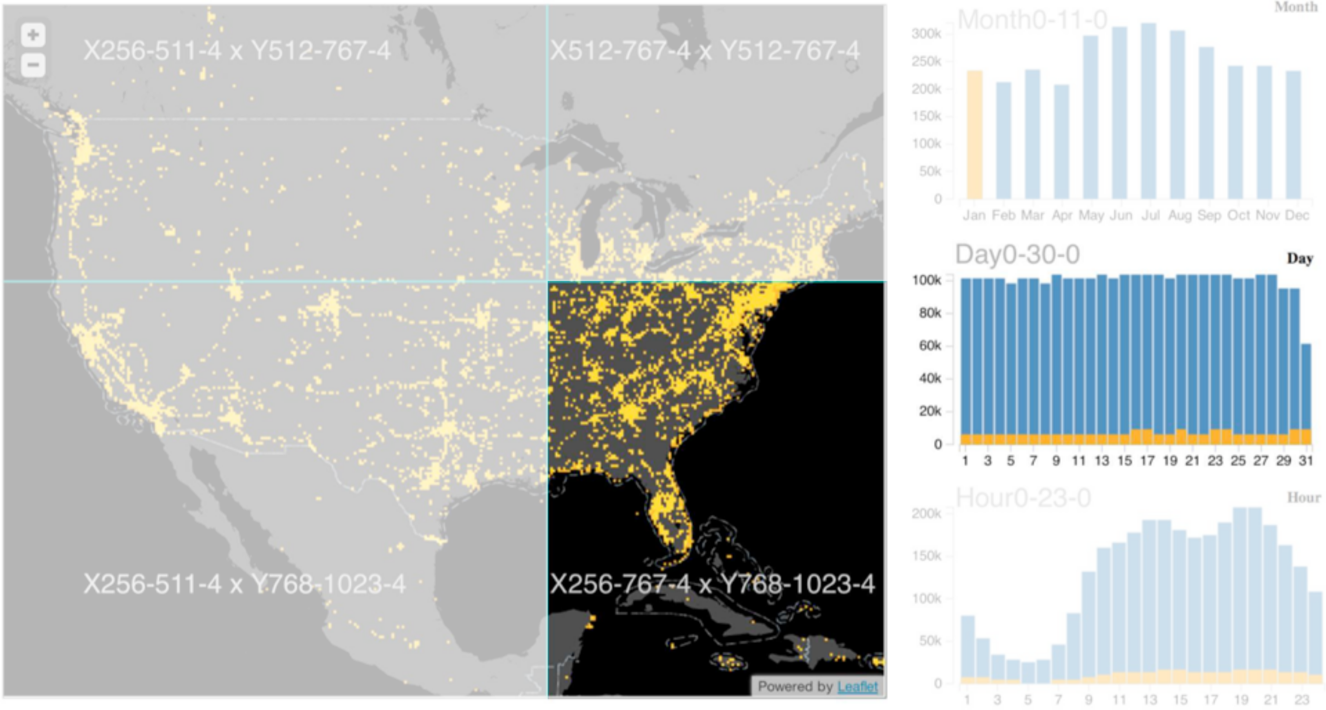


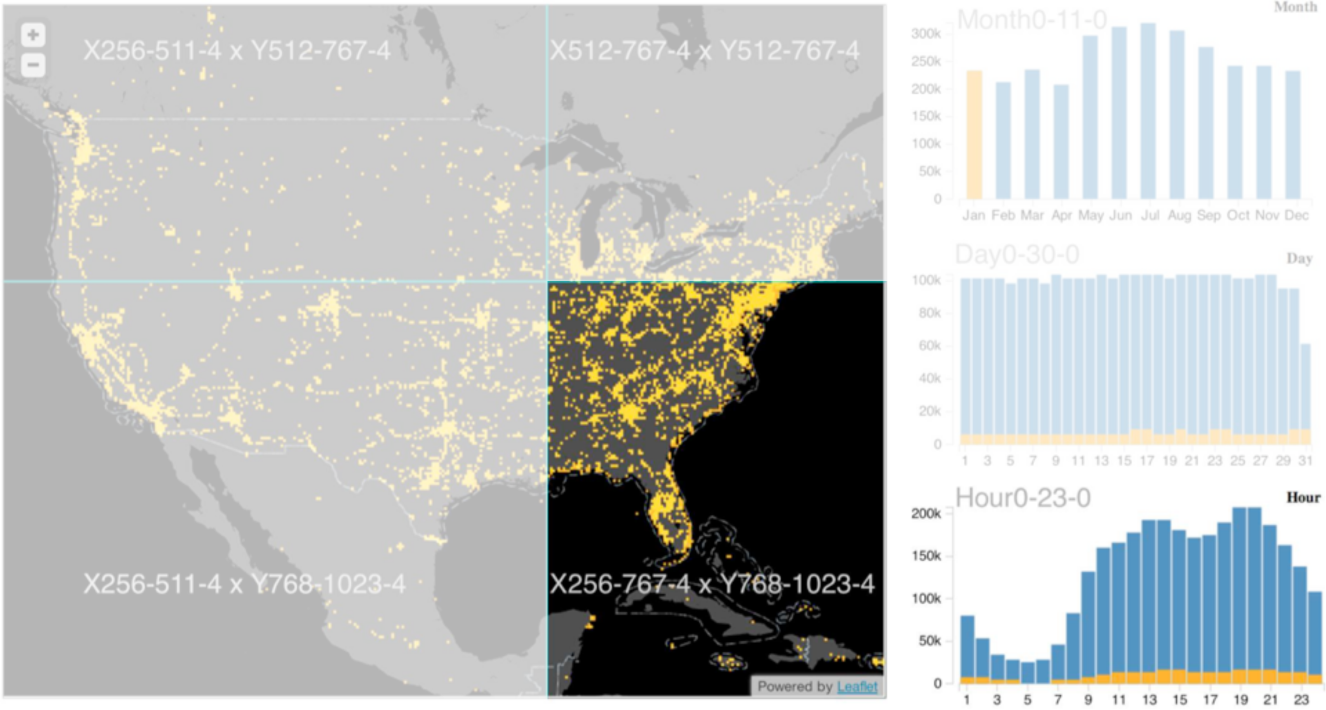
X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month0-11-0
Month
300k
250k
200k
150k
100k
50k
0
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day0-30-0
Day
100k
80k
60k
40k
20k
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
Hour0-23-0
Hour
200k
150k
100k
50k
0
1 3 5 7 9 11 13 15 17 19 21 23

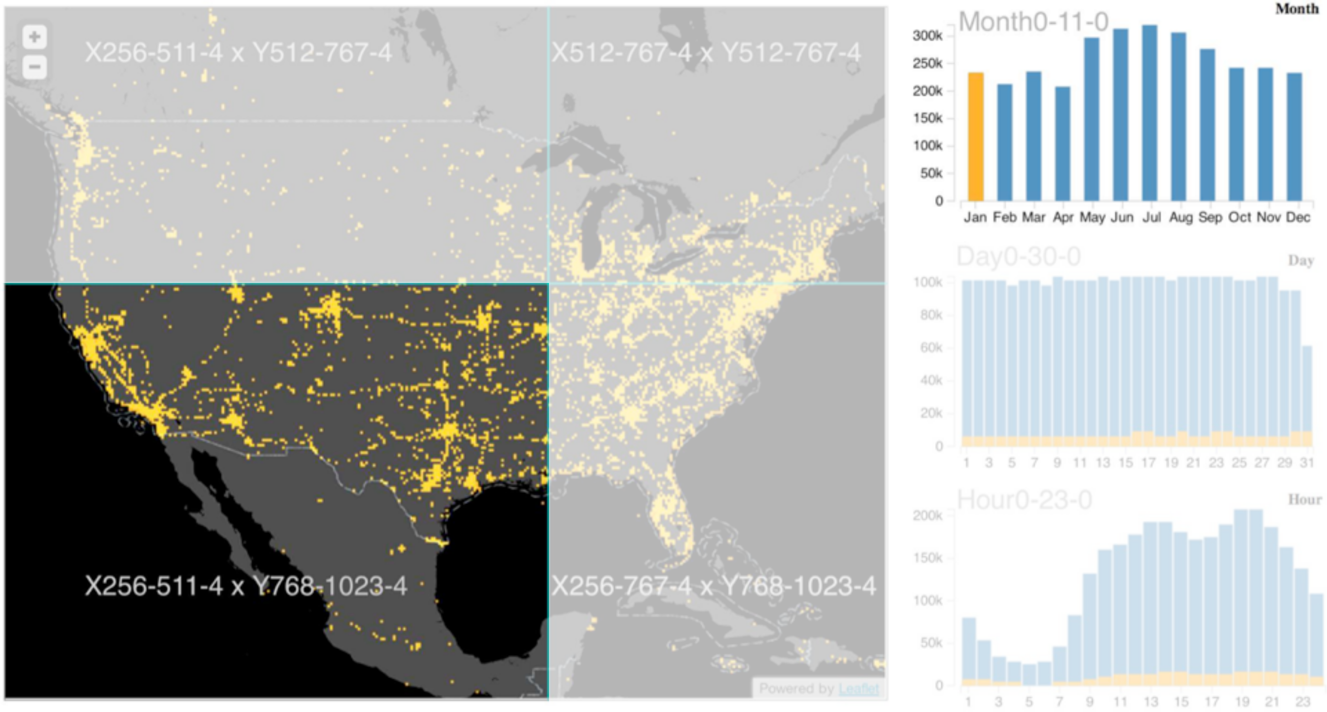


X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month0-11-0
Month
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day0-30-0
Day
Hour0-23-0
Hour

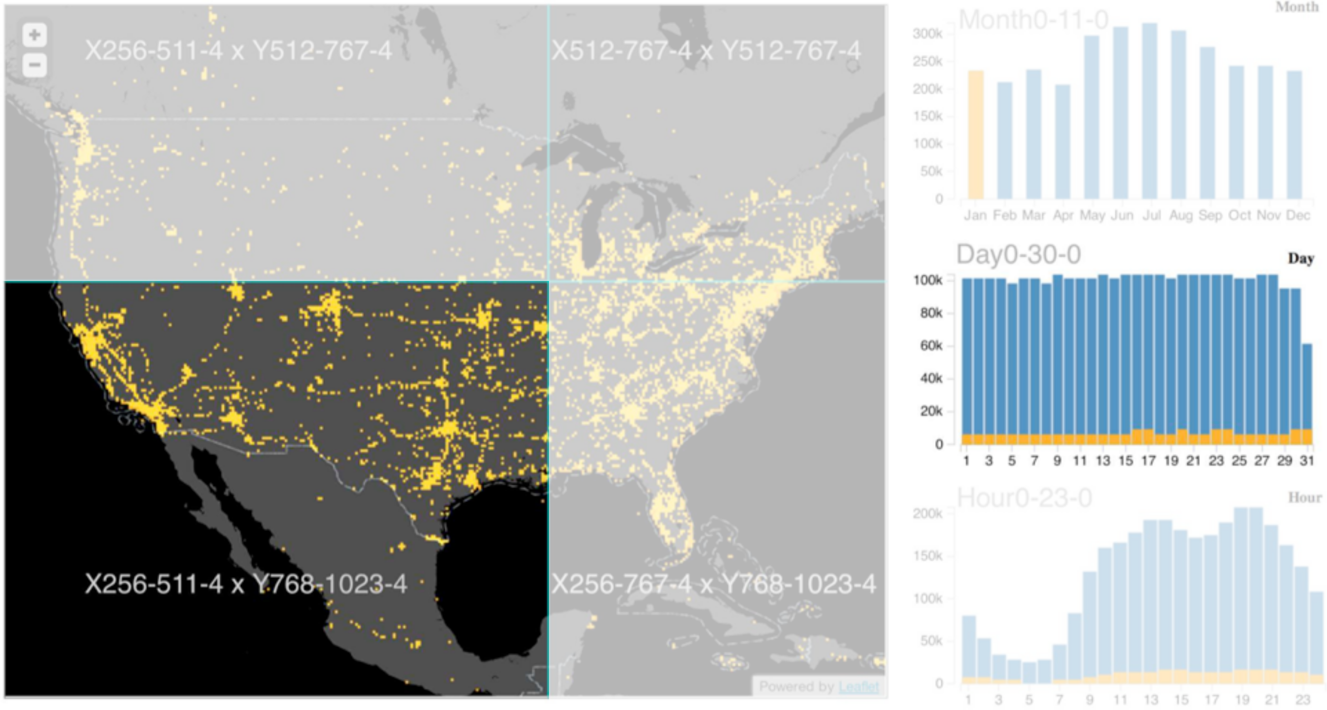

X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month
Month0-11-0
300k
250k
200k
150k
100k
50k
0
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day0-30-0
Day
100k
80k
60k
40k
20k
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
Hour0-23-0
Hour
200k
150k
100k
50k
0
1 3 5 7 9 11 13 15 17 19 21 23


X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month0-11-0
Month
300k 250k 200k 150k 100k 50k 0
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day0-30-0
Day
100k 80k 60k 40k 20k 0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
Hour0-23-0
Hour
200k 150k 100k 50k 0
1 3 5 7 9 11 13 15 17 19 21 23

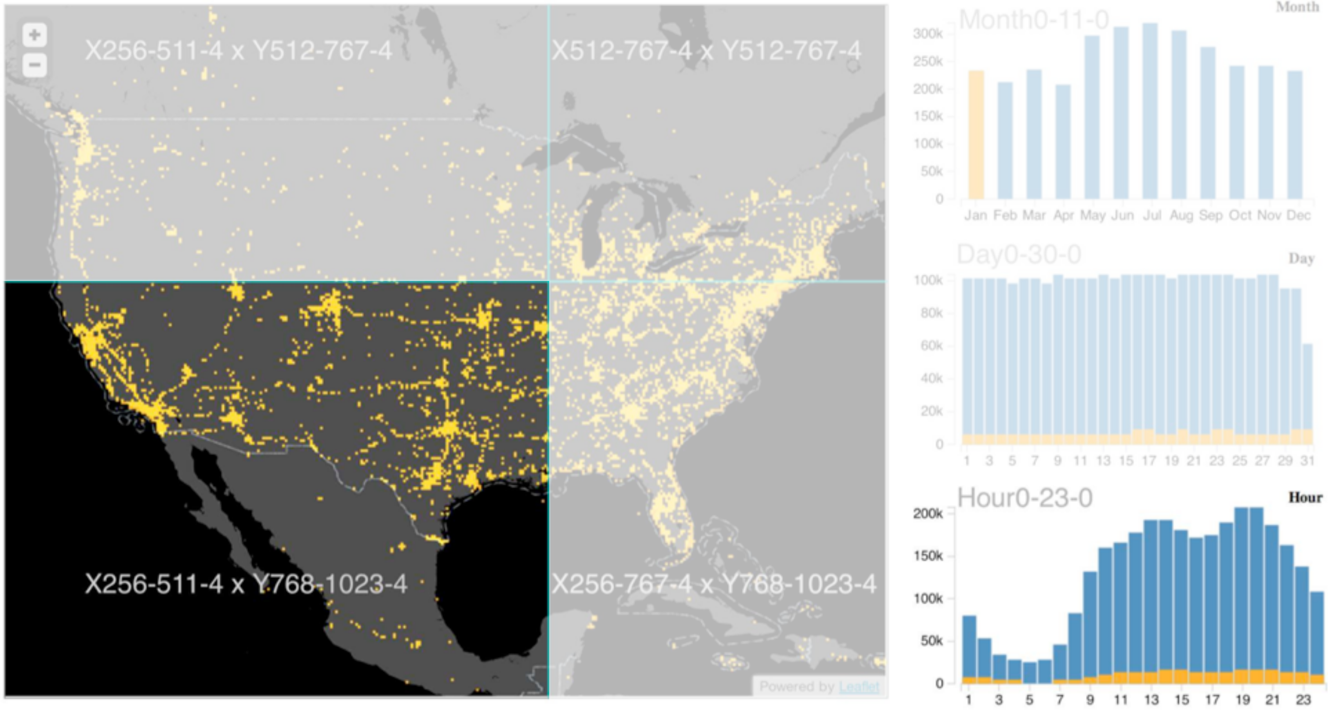

X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month0-11-0
Month
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day0-30-0
Day
Hour0-23-0
Hour



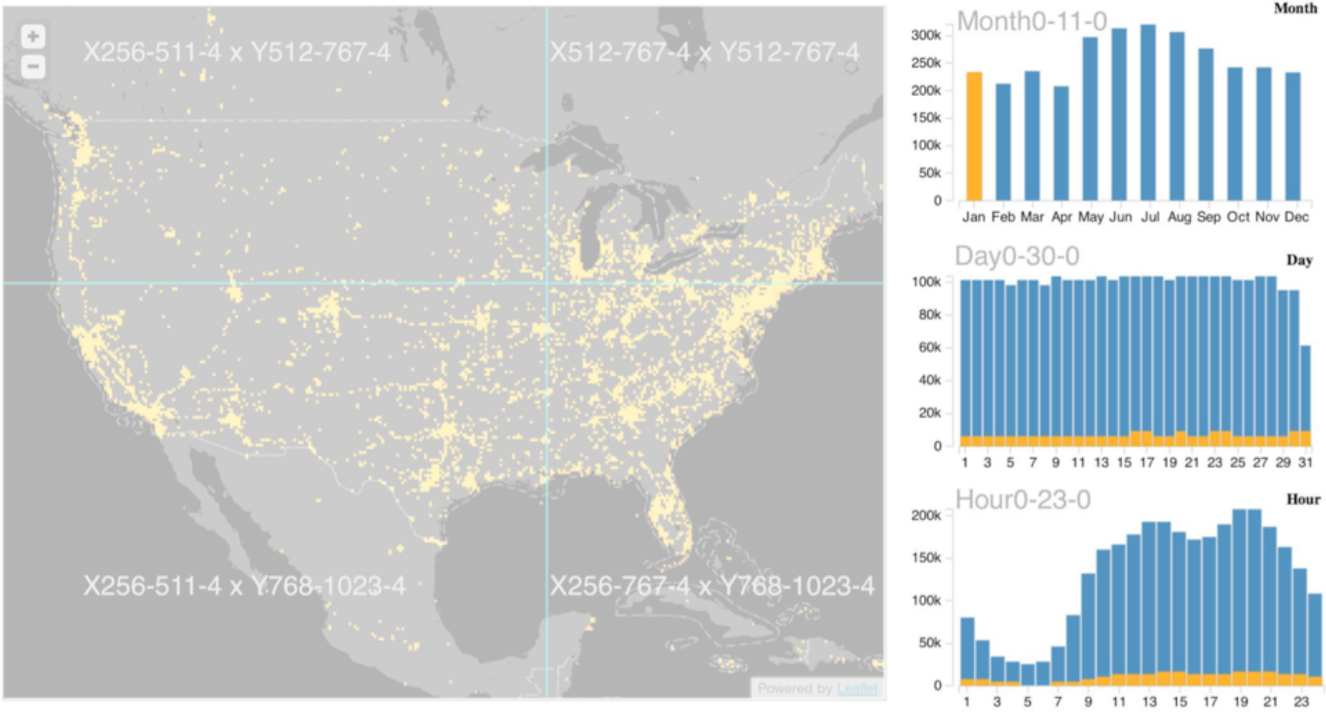
X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month0-11-0
Month
300k
250k
200k
150k
100k
50k
0
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day0-30-0
Day
100k
80k
60k
40k
20k
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
Hour0-23-0
Hour
200k
150k
100k
50k
0
1 3 5 7 9 11 13 15 17 19 21 23

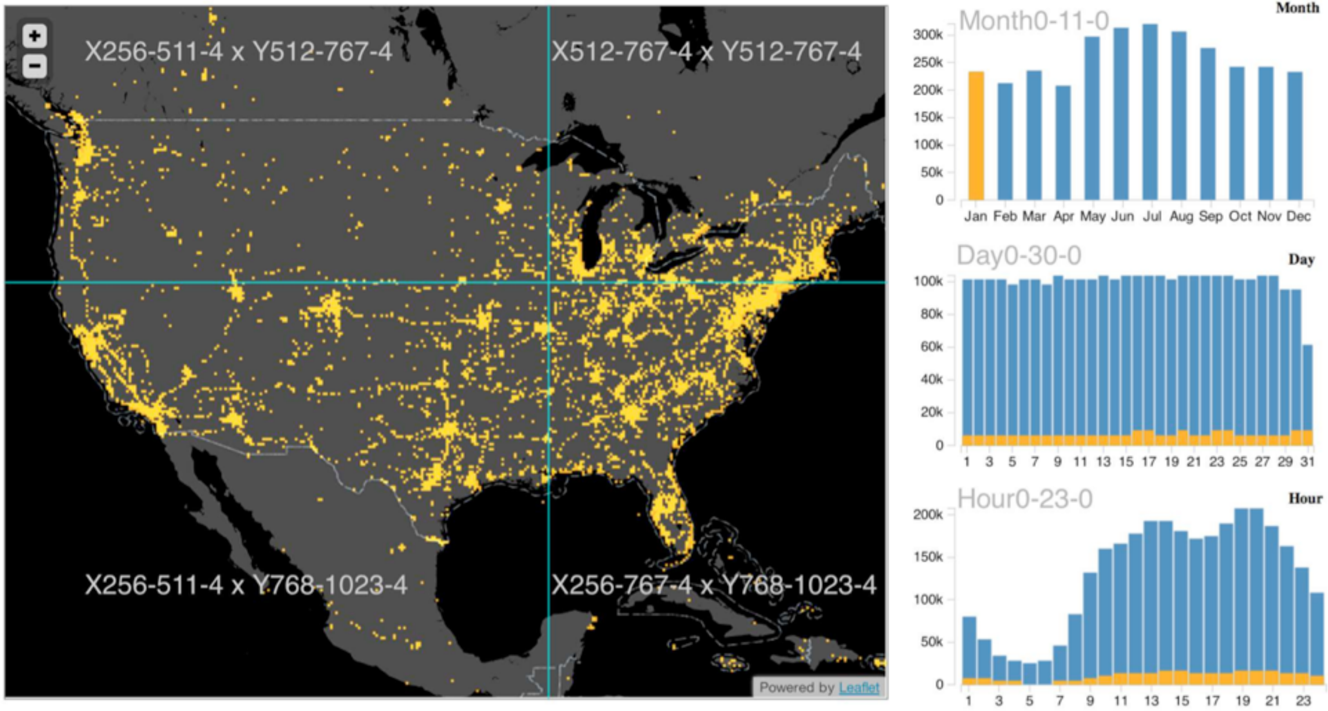


X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month0-11-0
Month
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day0-30-0
Day
Hour0-23-0
Hour

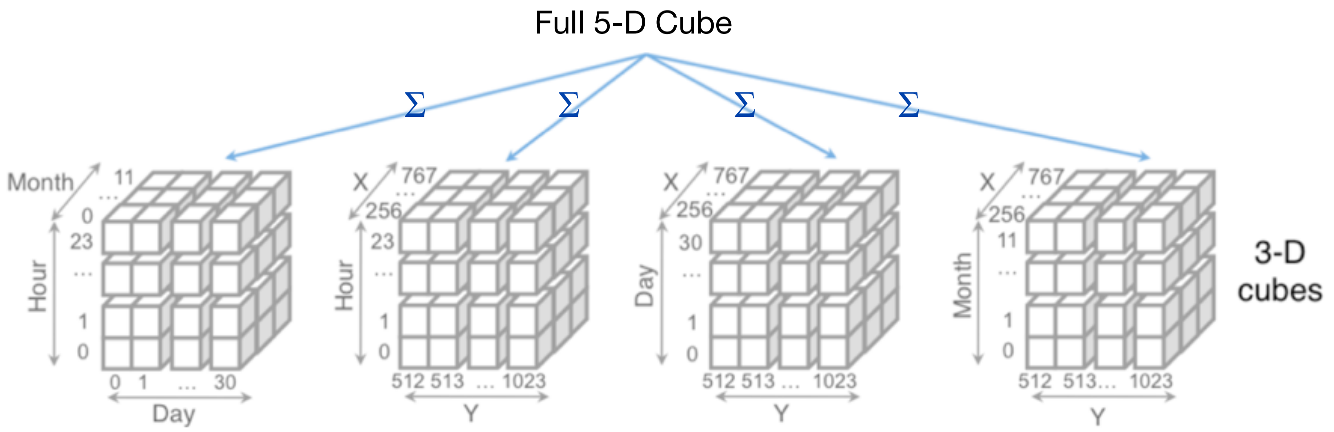


X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month0-11-0
Month
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day0-30-0
Day
Hour0-23-0
Hour


X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month0-11-0
Month
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day0-30-0
Day
Hour0-23-0
Hour

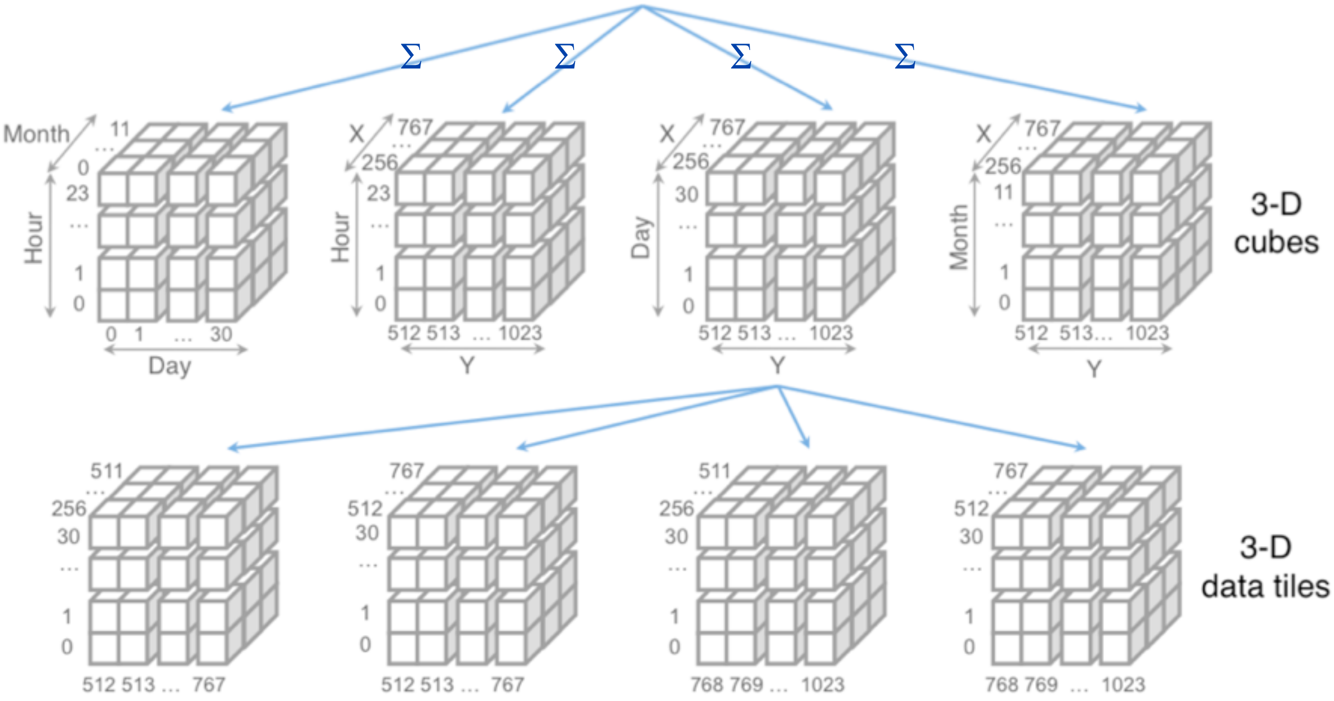

X256-511-4 x Y512-767-4
X512-767-4 x Y512-767-4
X256-511-4 x Y768-1023-4
X256-767-4 x Y768-1023-4
Powered by Leaflet
Month0-11-0
Month
300k
250k
200k
150k
100k
50k
0
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Day0-30-0
Day
100k
80k
60k
40k
20k
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
Hour0-23-0
Hour
200k
150k
100k
50k
0
1 3 5 7 9 11 13 15 17 19 21 23

# Full 5-D Cube

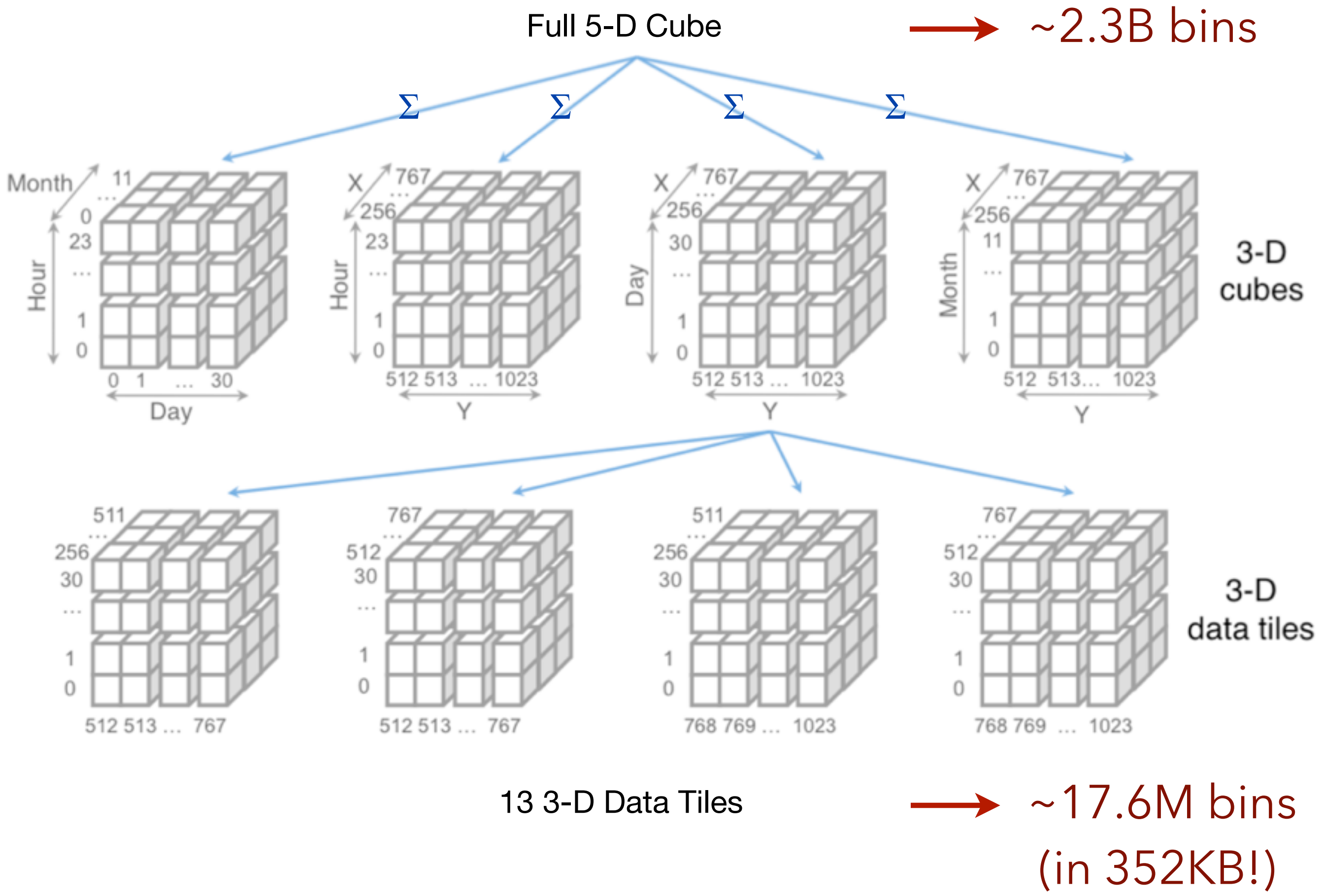



For any pair of 1D or 2D binned plots, the maximum number of dimensions needed to support brushing & linking is **four**.

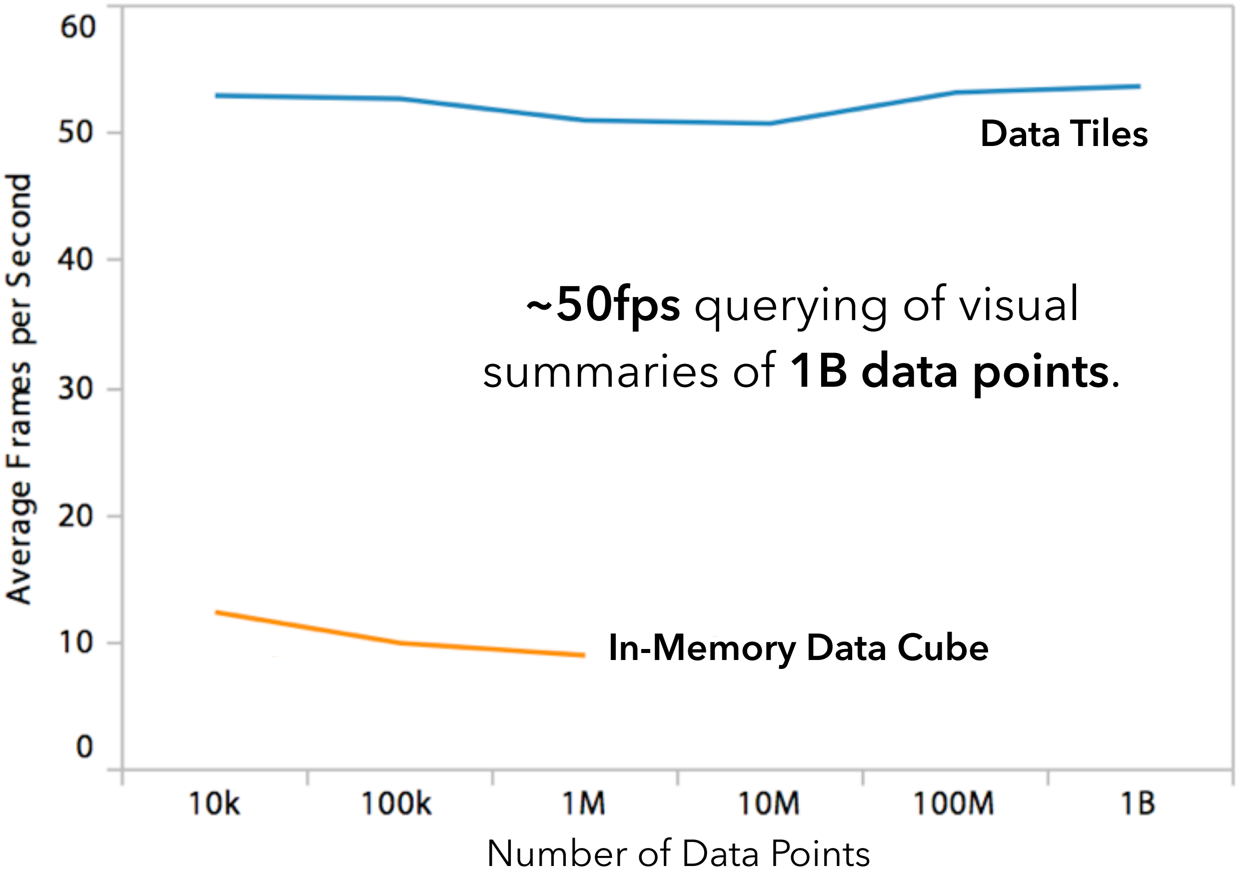
Full 5-D Cube




13 3-D Data Tiles


Full 5-D Cube
~2.3B bins
Σ
Σ
Σ
Σ
Month
Hour
Day
X
Y
3-D
cubes
3-D
data tiles
13 3-D Data Tiles
~17.6M bins
(in 352KB!)

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]

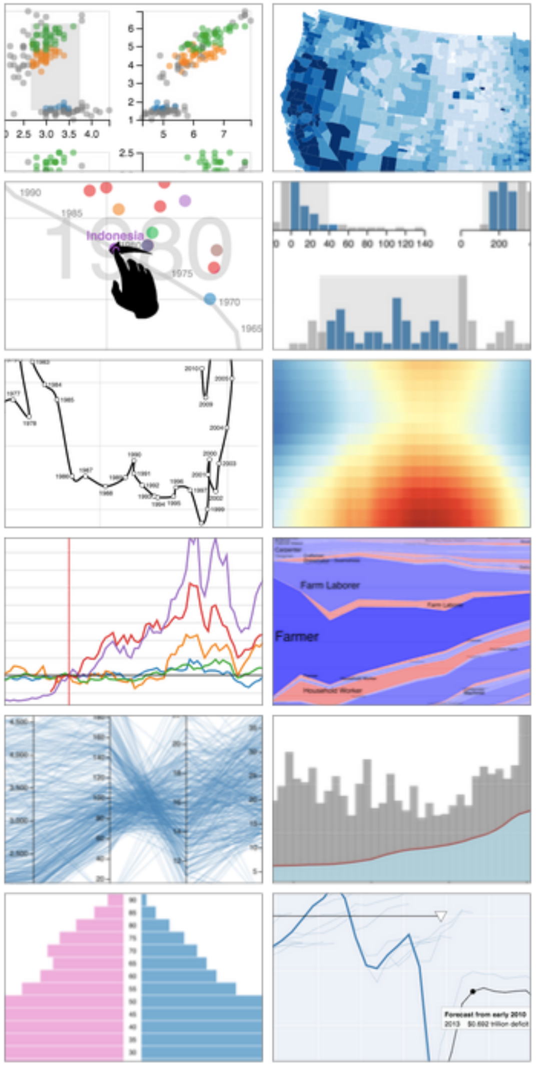
# Sampling Methods

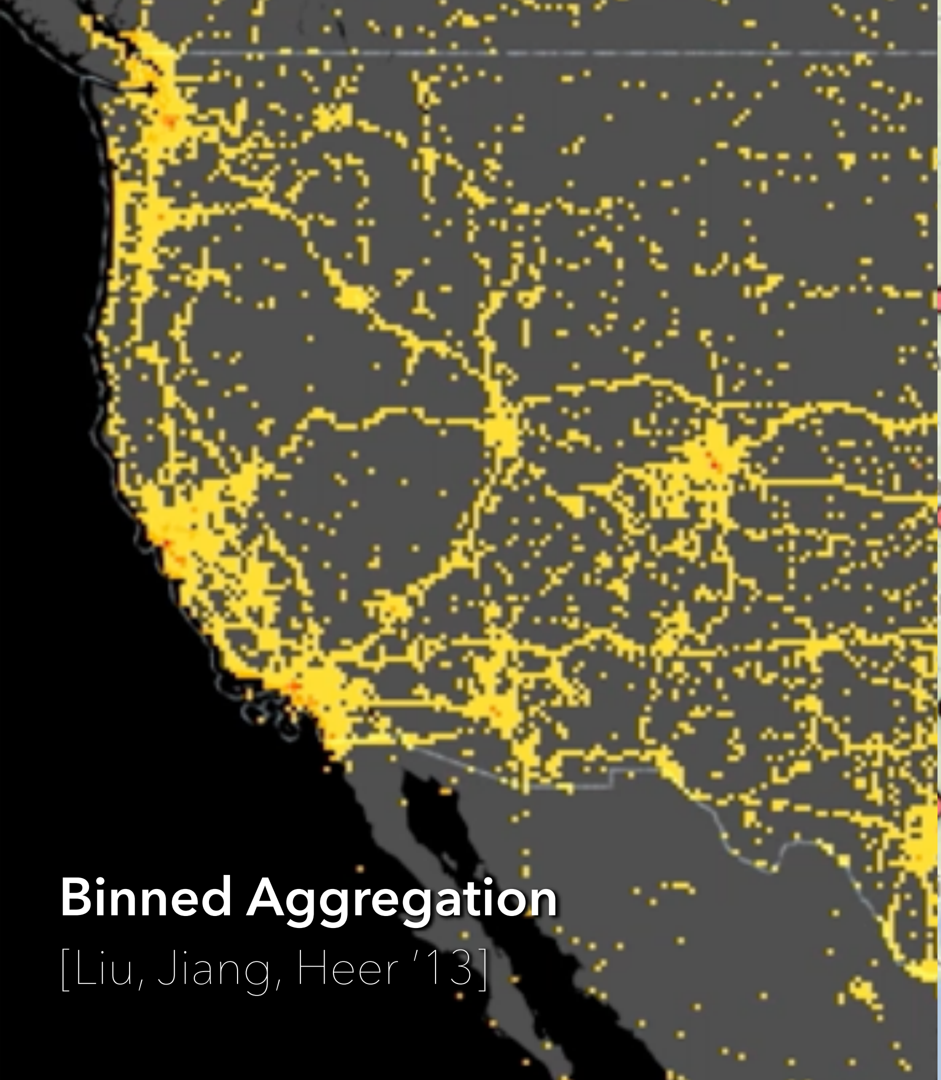
# Common Sampling Methods

**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**

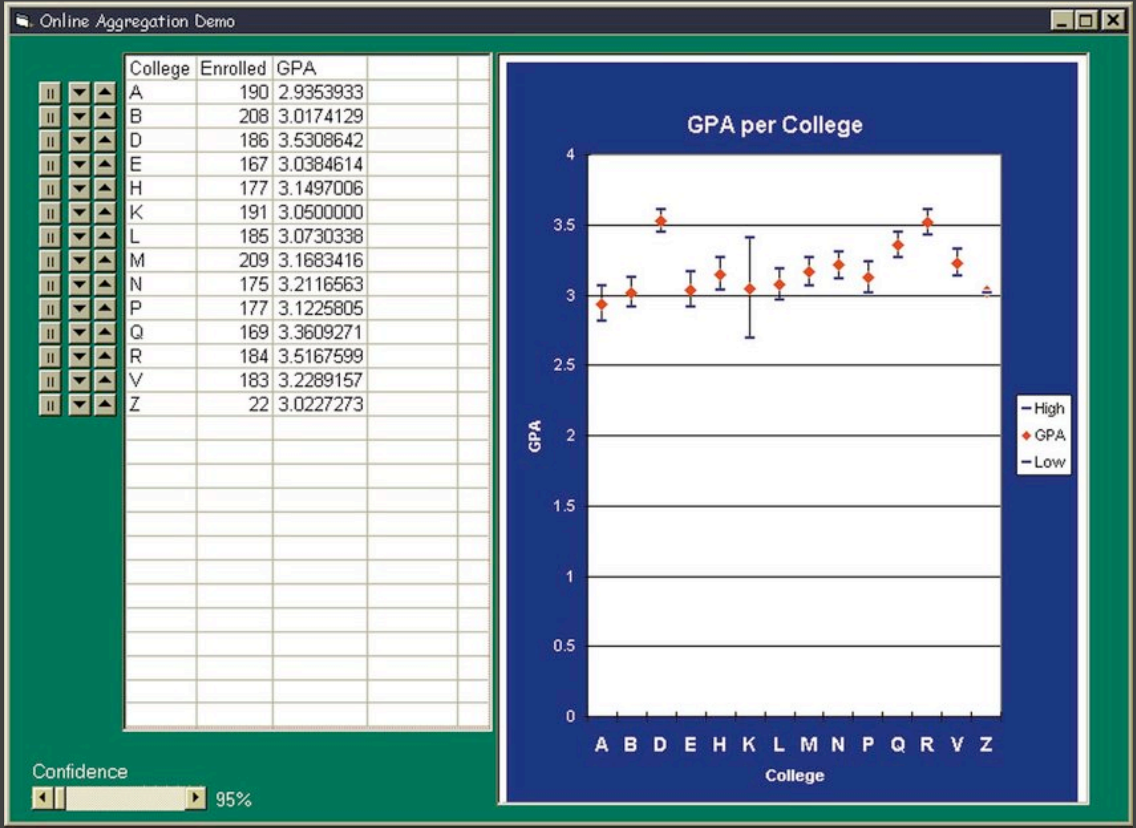
**Google Fusion Tables**

# **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.

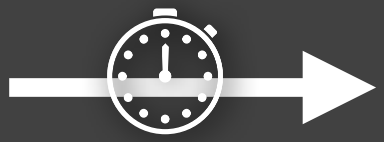


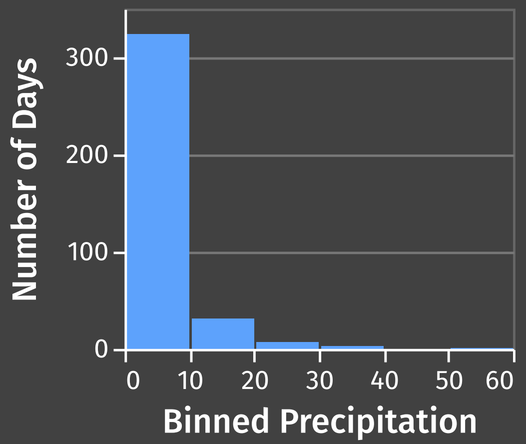
What if data is too large to query in a reasonable time?

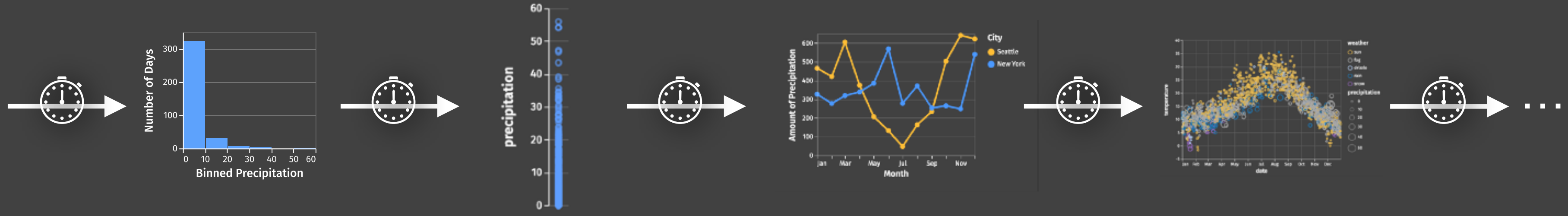
# Trust, but Verify: Optimistic Vis

[Moritz, Fisher, Ding & Wang '17]

Strategies: Query Database, Approximation




Number of Days
300
200
100
0
0
10
20
30
40
50
60
Binned Precipitation




Latencies reduce engagement
and lead to fewer observations.

The Effect of Interactive Latency. Liu, Heer. *IEEE InfoVis 2014*.

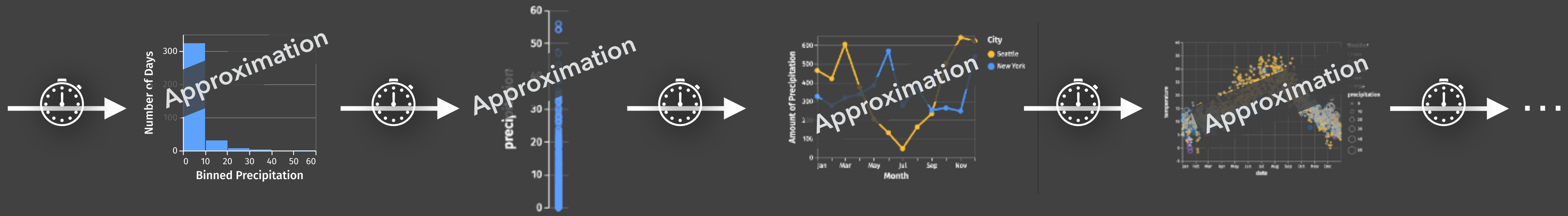
Small chance of error

Very likely to have at least one error

Small chance of error

Small chance of error

Small chance of error




# 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.

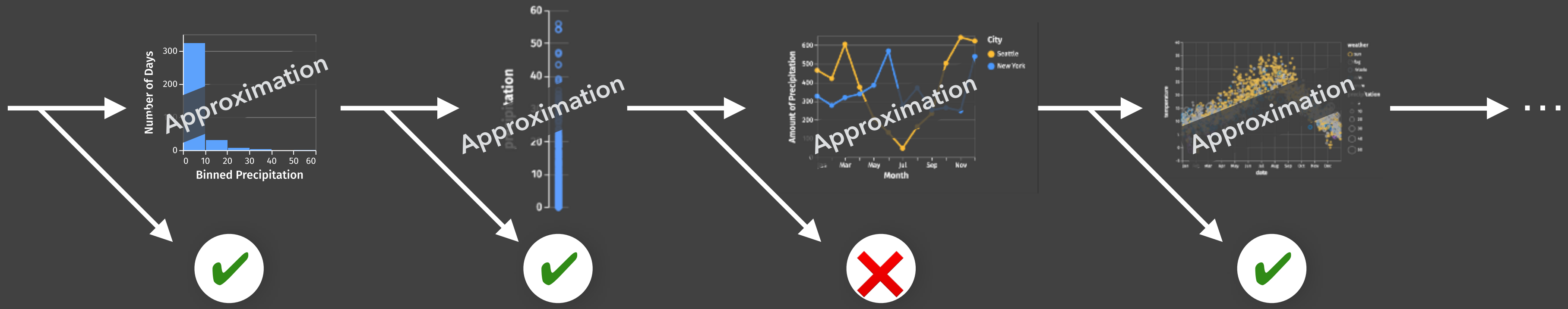

# Optimistic Visualization

Trust but Verify

What if we think of the
issues with approximation as
**user experience** problems?

# Optimistic Visualization

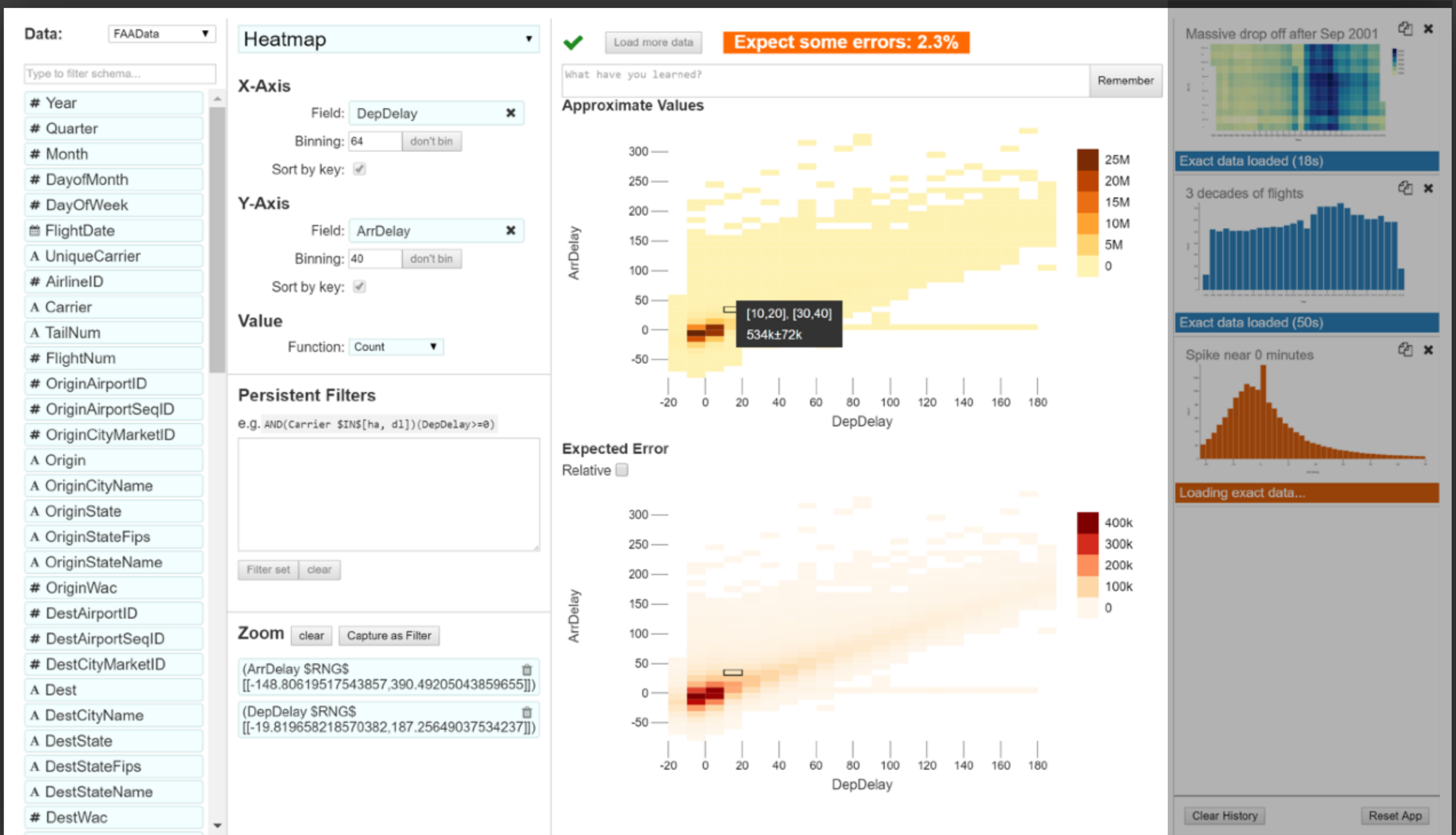
Trust but Verify. Moritz et al. *CHI 2017*.




1. Analysts uses initial estimates.
2. Precise queries run in the background.
3. System confirms results. Analyst detects errors.

Analysts can use approximations and also trust them.

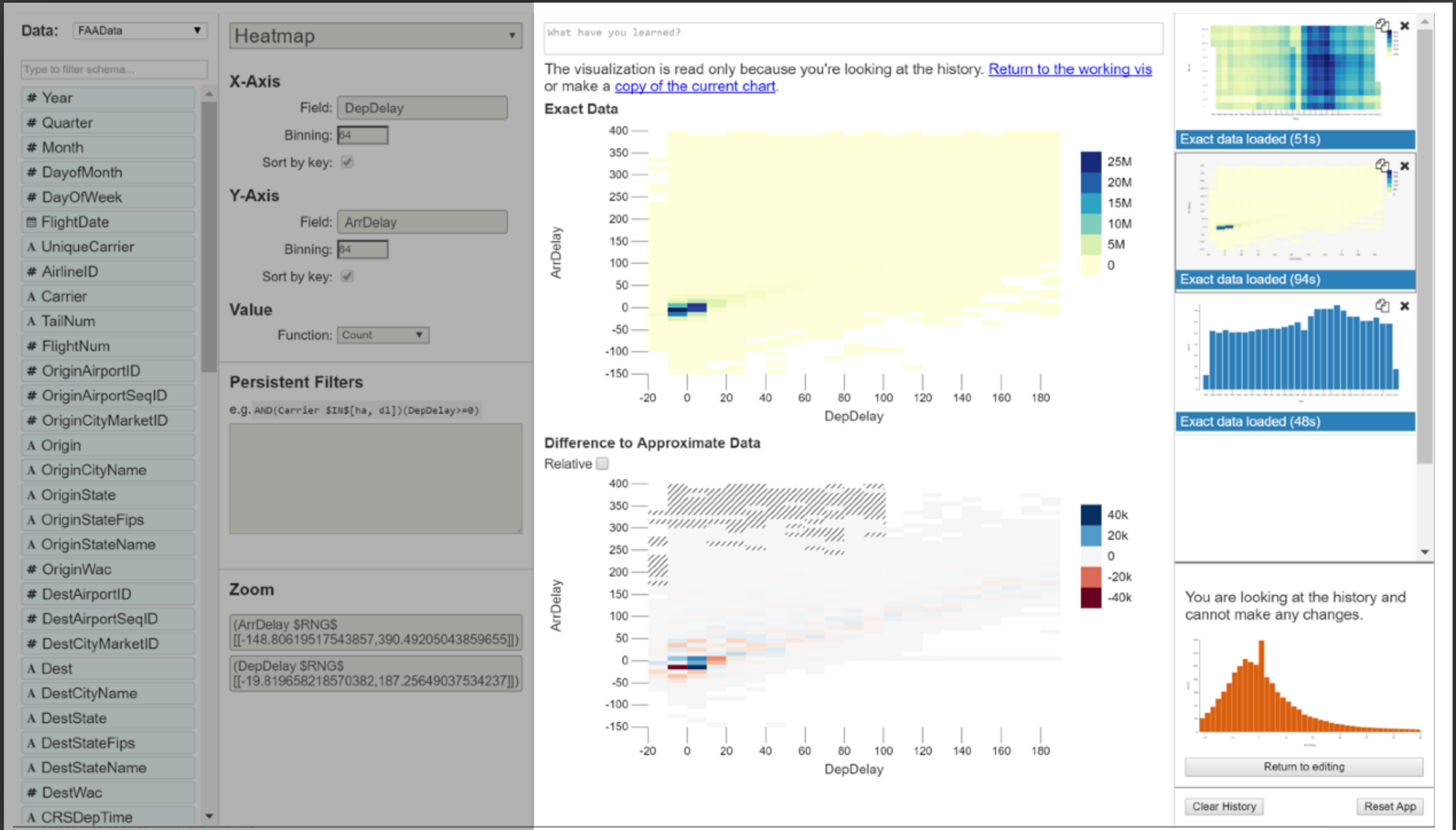
# Optimistic Visualization



# Visualize Uncertainty

# 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.

# Bin > Aggregate (> Smooth) > Plot

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.