Aviz

Visual Analytics Project

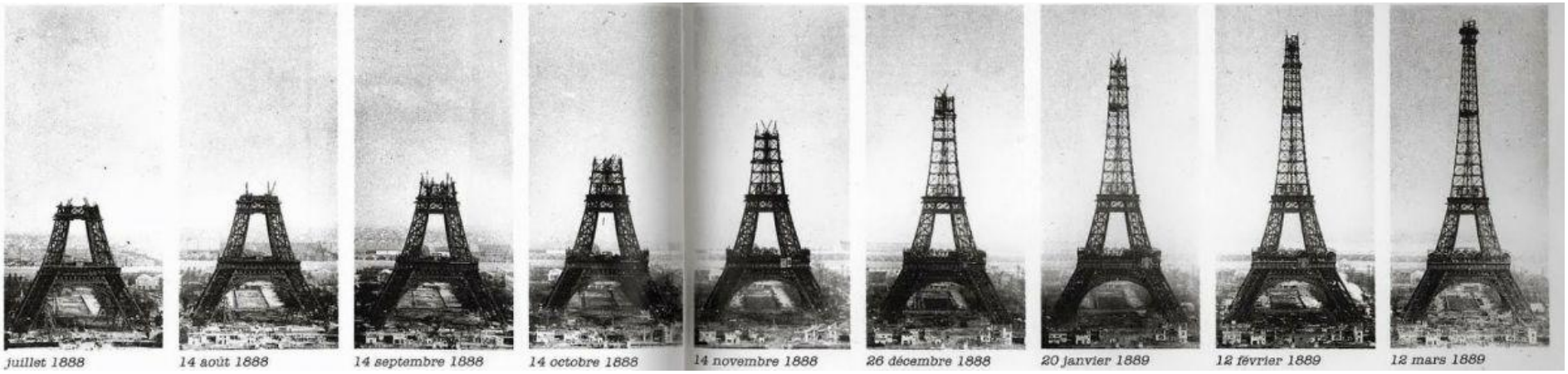
informatics mathematics

Inria

ProgressiVis: a New Computation Paradigm for Scalability in Exploratory Data Analysis

Jean-Daniel Fekete & Christian Poli

INRIA http://www.aviz.fr/~fekete



Data at Scale

- “Confirmatory” Data Analysis, or Analytics, has scaled dramatically in the last 20 years
  - Following Moore’s law
- With the increase of storage and parallel architectures, it continues to scale

- What about “Exploratory” Data Analysis?

Sequential Execution Paradigm

Current computers and programming languages rely on the *sequential execution paradigm*

- Computing f(g(x)) requires computing y=g(x) then f(y)
- Time to completion is time(g(x))+time(f(y))
- This time is not bounded

- **Human cognition requires a latency < 10s**

Using Analytics for Exploration

- Scalable infrastructures provide high-throughput with **high-latency**
  - Big black boxes that run to completion with no time-to-deliver guarantees
- Can we get **interactive-time responses** with scalability?
  - Tinkering with tools can work, but is tedious
  - **Progressive Data Analysis is meant as a solution**

Exploration and Latency

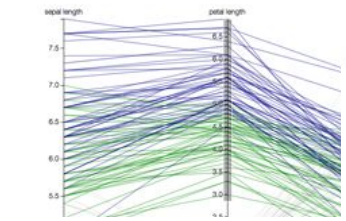
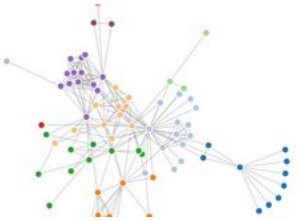
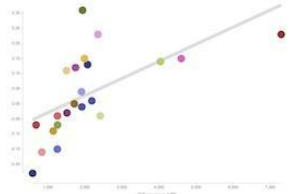
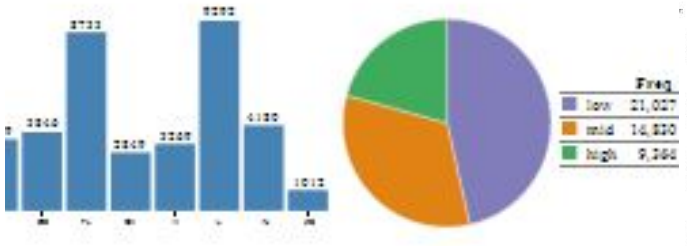
3 types of latency to consider for HCI:

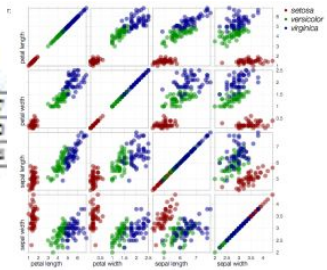
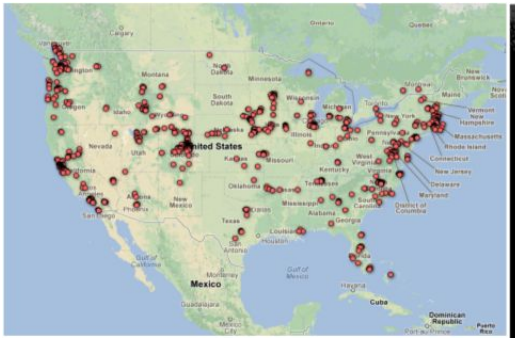
1. *Continuity Preserving Latency*: ~0.1s user feel that the system is reacting instantaneously
2. *Flow Preserving Latency*: ~1s user's flow of thought to stay uninterrupted
3. *Attention Preserving Latency*: ~10s keeping the user's attention focused on the dialogue

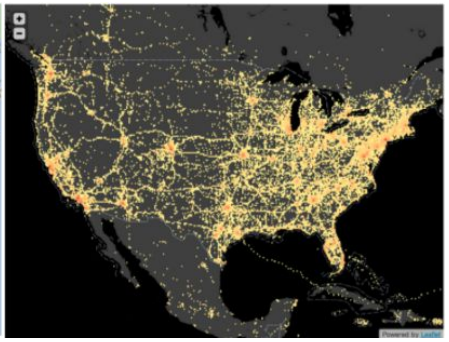
- R. B. Miller. Response time in man-computer conversational transactions. In Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I, AFIPS '68 (Fall, part I), pages 267–277, New York, NY, USA, 1968. ACM.
- J. Nielsen. Response times: The 3 important limits, https://www.nngroup.com/articles/response-times-3-important-limits/
- B. Shneiderman. Response time and display rate in human performance with computers. ACM Comput. Surv., 16(3):265–285, Sept. 1984.

Scaling Visualizations

- Vis. does not scale well
  - Not in number of items
  - Not in number of dimensions
- It needs additional methods such as:
  - Sampling (of items/dim.)
  - Aggregation
  - Dimensionality Reduction
- These methods introduce artifacts
  - Their results should be explored too, to be validated!

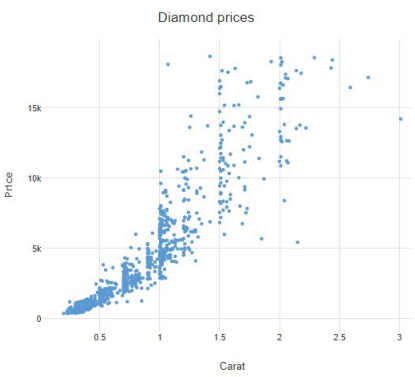





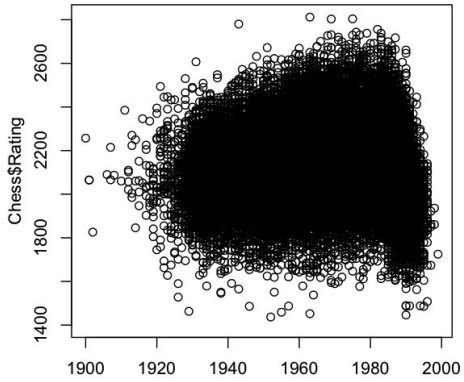



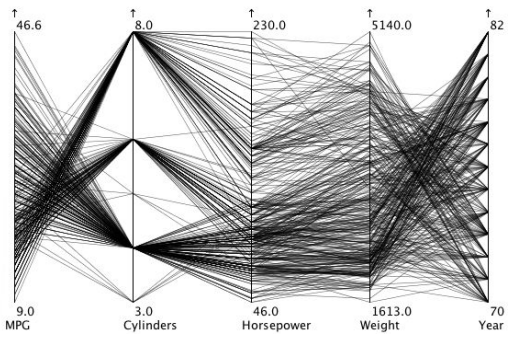


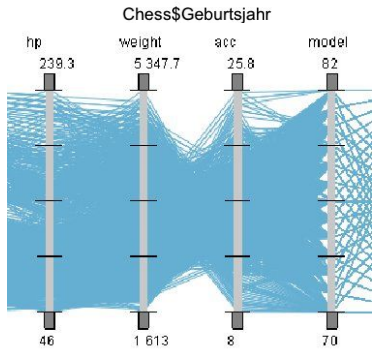
Scaling Visualizations

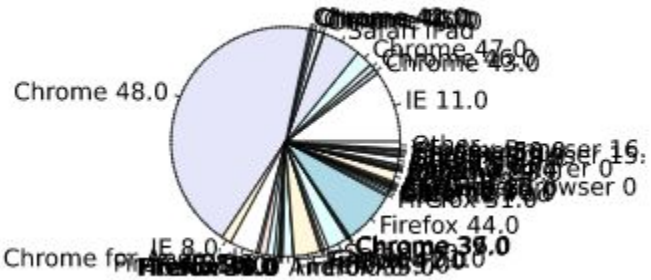
- Many visualization techniques suffer from **overplotting**
  - Scatterplots
  - Parallel Coordinates
  - Even Pie Charts
- Repairing these artifacts require analytical operations







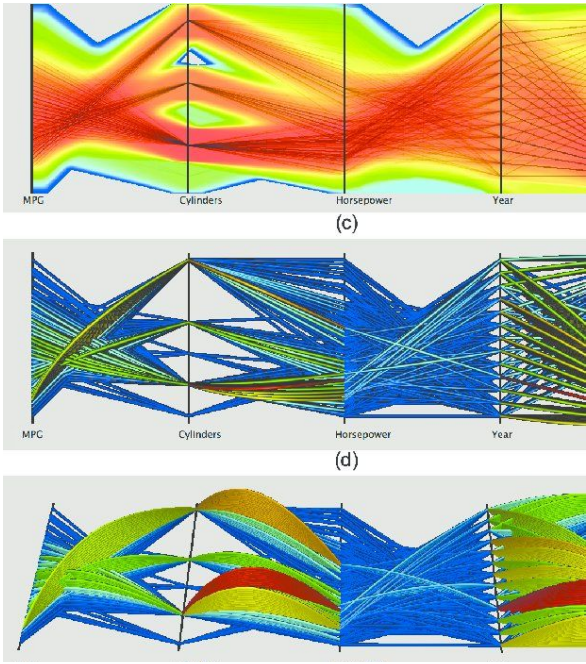


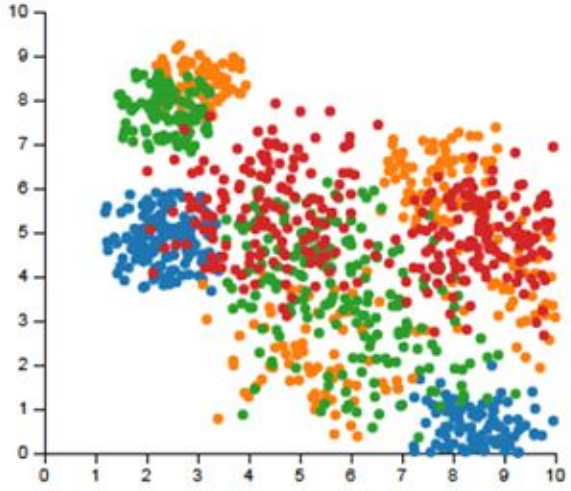


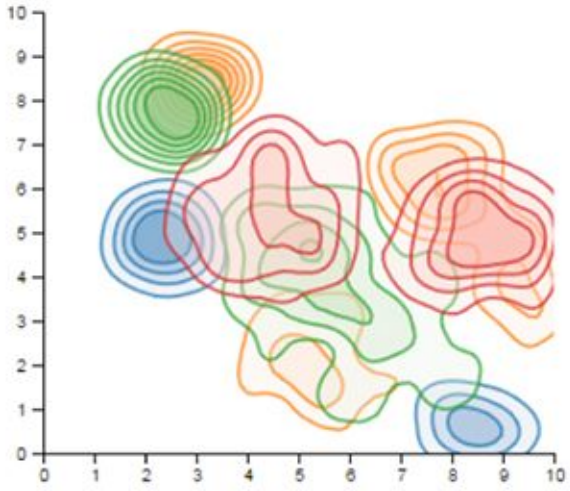
Scaling Visualizations

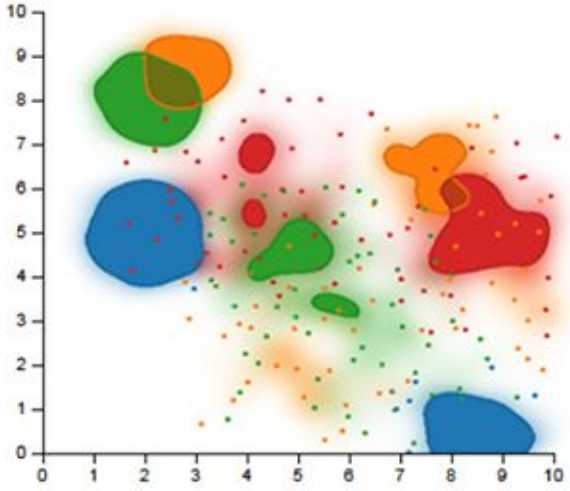
Using more complex visualizations require analytical operations

e.g. histogram computations, aggregations using log functions...



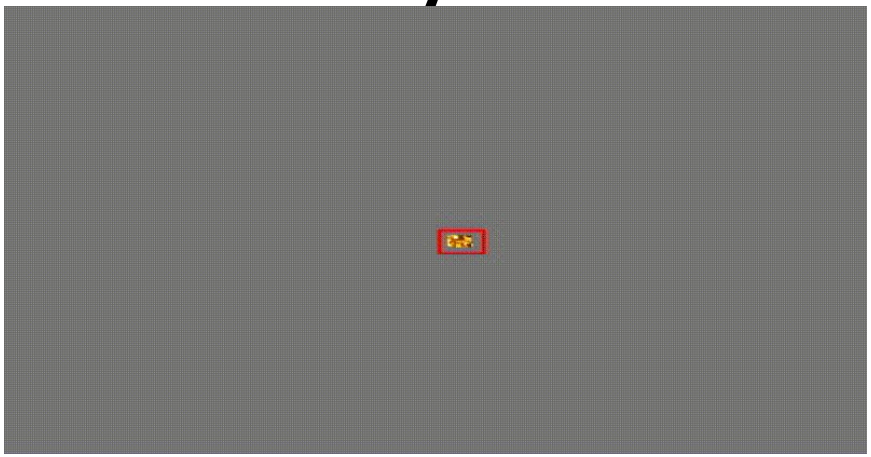




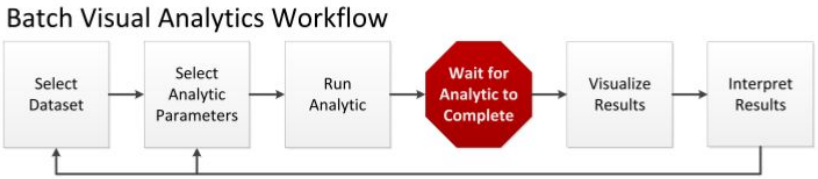


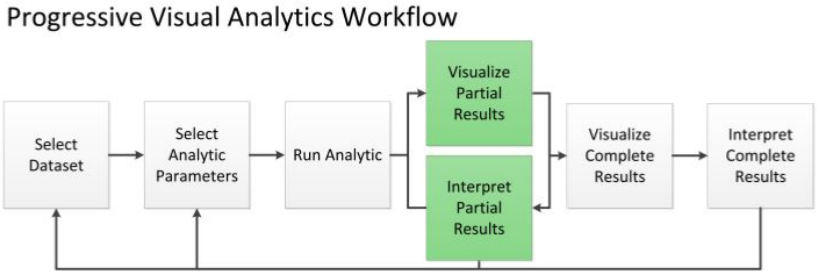
Progressive Data Analysis

- Allow Exploratory tools to work while the computation is being done
- Many articles mention it
- Some systems implement it in ad-hoc ways
- No realistic model to implement it in general



Williams, M.; Munzner, T., "Steerable, Progressive Multidimensional Scaling," in *INFOVIS 2004.*





Charles D. Stolper, Adam Perer, and David Gotz. **Progressive Visual Analytics**. *IEEE TVCG* (Volume 20, Issue 12, 2014).

Progressive Data Analysis

1. Produce improving estimates
   - With **bounded latency**
2. Converge to result
3. Allows Steering
   - Change any parameter
4. Produce measures of:
   - Quality (higher is better)
   - Progress (current position, end position)

Benefits of Progressive Data Analysis

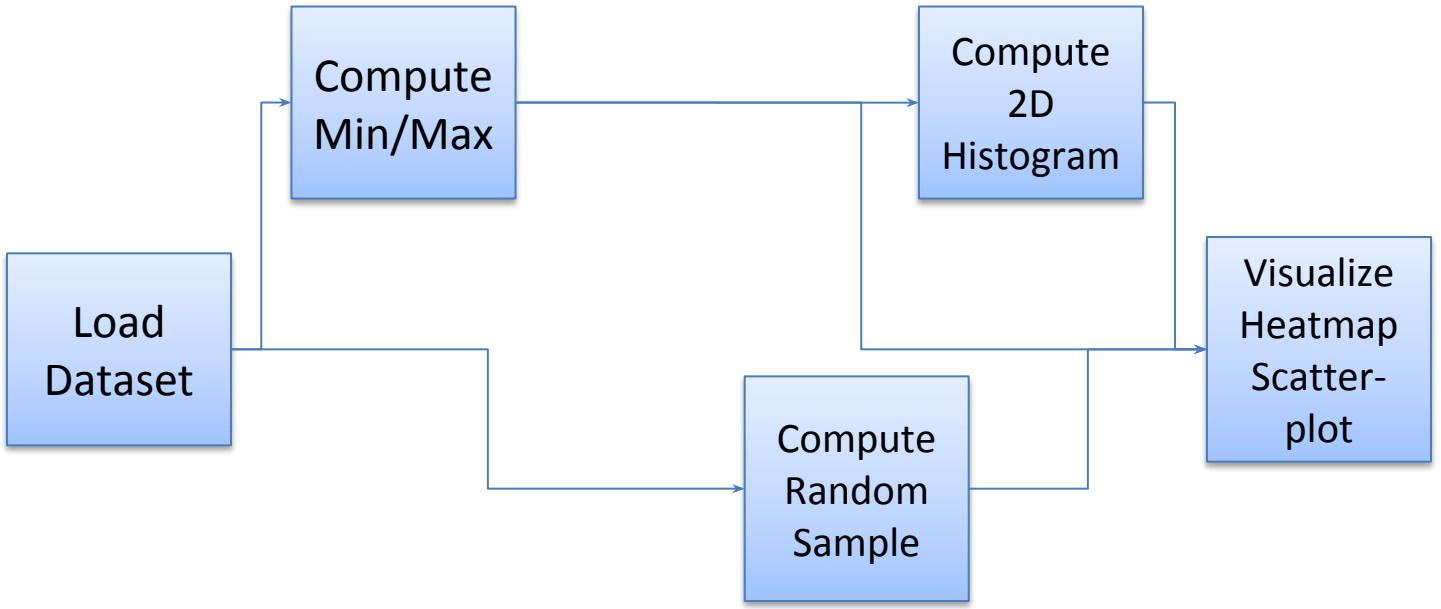
- Scalability
  - Exploration of large datasets and complex algorithms at a human pace
- Early decisions
  - Abort when results are useless or algorithms not well configured
- Algorithm understanding
  - Seeing the results as they are computed helps understand the algorithm behavior (sometimes, more research needed)

ProgressiVis: New Execution Semantics

with Christian Poli and Romain Primet

```
data=CSVLoader('bigfile.csv')
minmax = MinMax(data)
histo = Histogram2D(data, minmax)
sample = Sample(data,  500)
plot = Scatterplot(histo, sample, minmax)
show(plot)
```

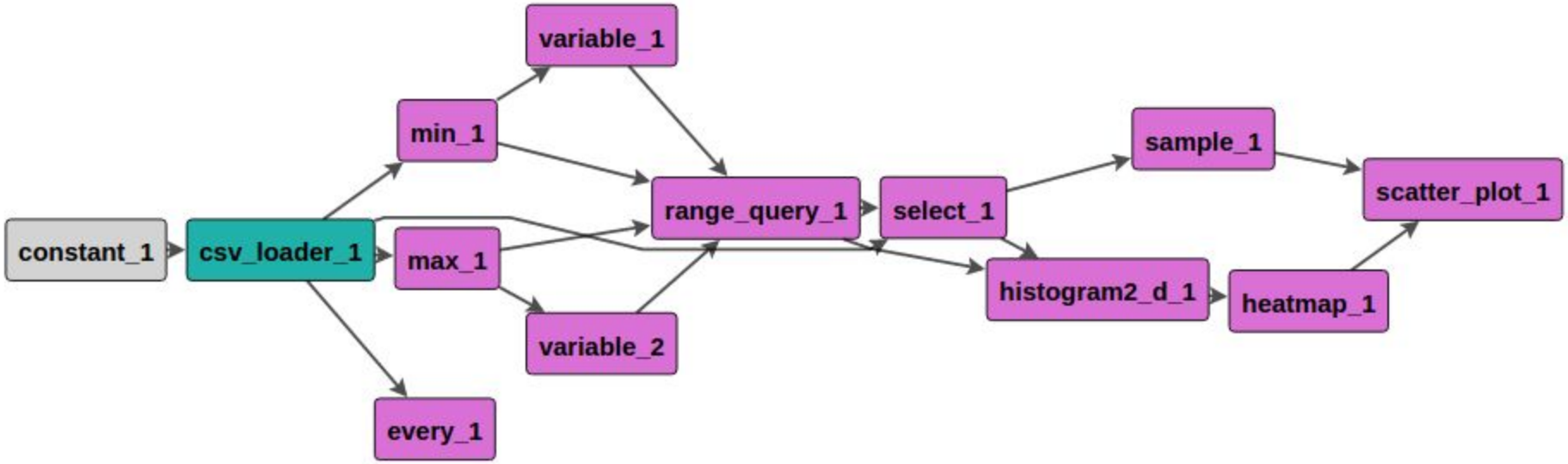
ProgressiVis:
Splitting the Computation in Chunks



ProgressiVis: Splitting the Computation in Chunks

```
URLS = [PREFIX+'yellow_tripdata_2015-01.csv'+SUFFIX,
        PREFIX+'yellow_tripdata_2015-02.csv'+SUFFIX,
        ...]
filenames = pd.DataFrame({'filename': URLS})
cst = Constant(df=filenames)
csv = CSVLoader()
csv.input.filenames = cst.output.df
pr = Every()
pr.input.df = csv.output.df
scatterplot = ScatterPlot('pickup_longitude',
    'pickup_latitude')
scatterplot.create_dependent_modules(csv,'df')
```

Dataflow Graph

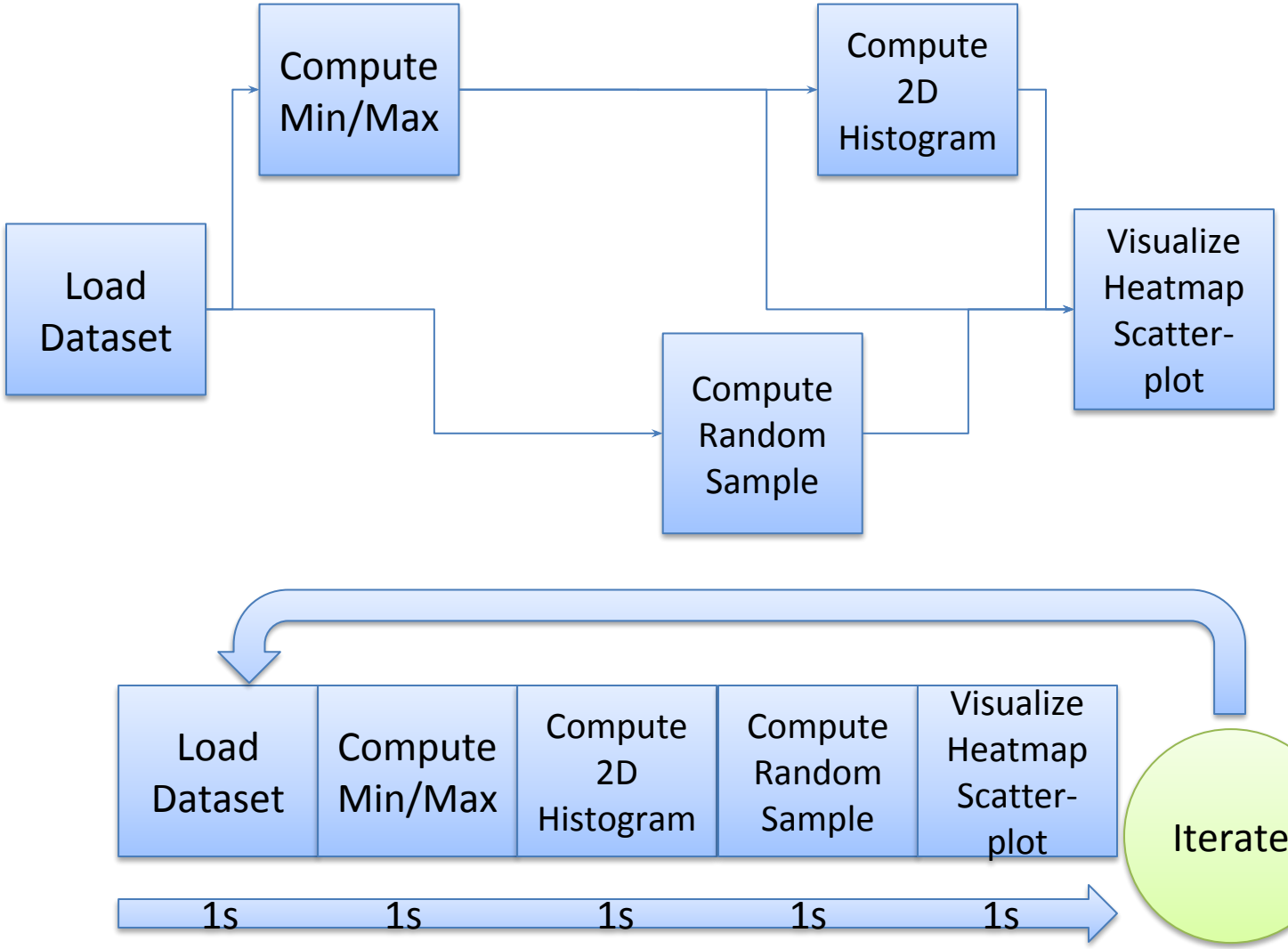


ProgressiVis: Splitting the Computation in Chunks

- Each module is given a quantum to run (~1s)
- At the end of its quantum, it should provide a useful result, even if partial or approximate
- Modules are run in ~~round-robin~~ dependency order until they reach the end of their computation

- Additionally, interaction is possible to steer and modify module parameters!

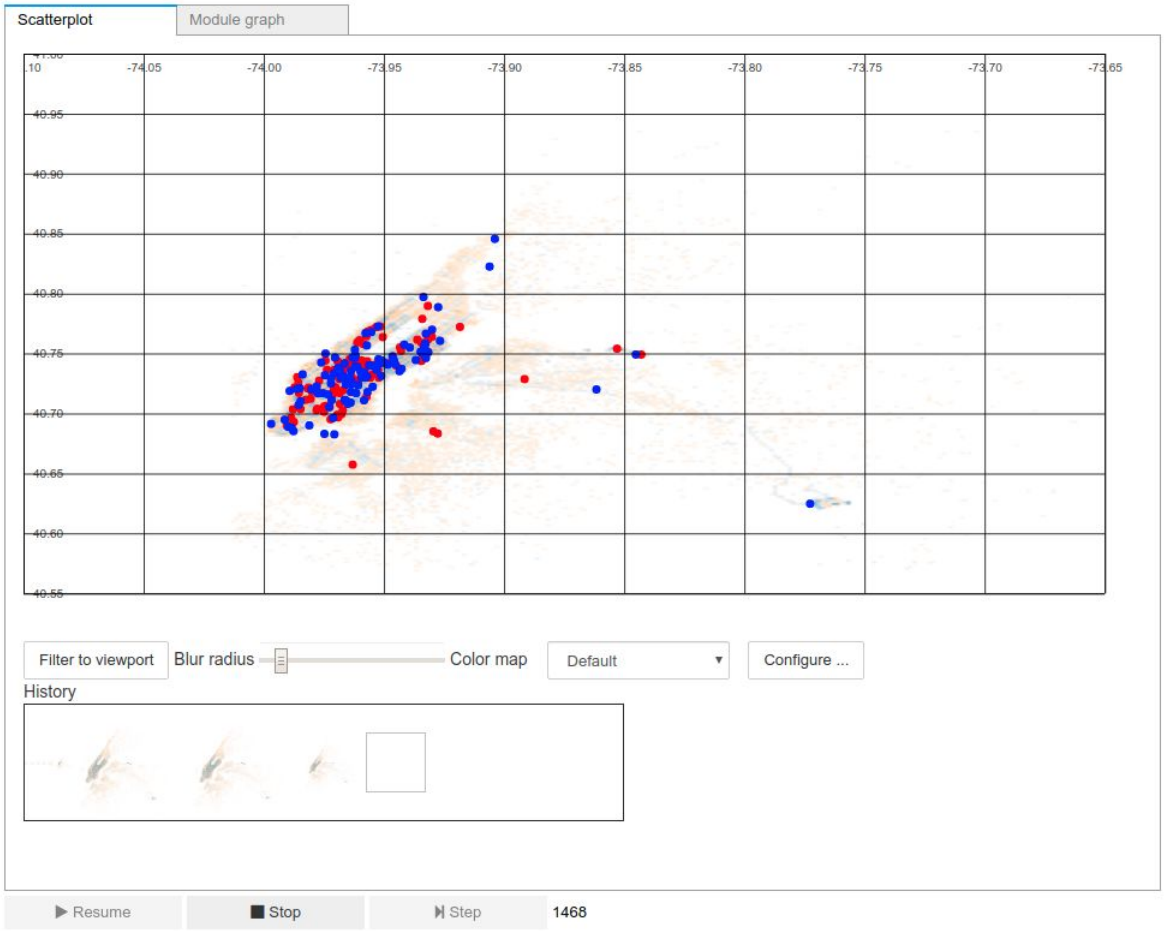
ProgressiVis:
Splitting the Computation in Chunks



Demo

In [3]:
```
import ipywidgets as ipw
tab = ipw.Tab()
tab.children = [sc, gr]
tab.set_title(0, 'Scatterplot')
tab.set_title(1, 'Module graph')
vbox = ipw.VBox([tab, cpanel])
vbox
```

*****



ProgressiVis: *Work In Progress* Internals

- Python toolkit
  - with a built-in web server for control
  - being connected to the Jupyter notebook
- Unified representation of Data to communicate between modules
  - Specific “Change Management” in Modules
- ~~*Modules* are run in a specific thread by a *Scheduler*~~
  - ~~2 modes, normal “round robin” after topological sort of modules~~
- Modules are asynchronous tasks using Python asyncio
  - Each task runs for a bounded amount of time and yield a result
- Time Prediction
  - Algorithms are never parametrized by time, at best by # of “steps” to perform, usually implicitly
  - Each module is told to run a certain number or steps
  - Predicting the number of steps to stay within the quantum requires analyzing (mining) the execution trace.

Data Tables: Problems

- Data Tables as DataFrames like Python/Pandas and R
- EDA requires column-oriented tables
- Pandas cannot grow tables nicely
- ProgressiVis implements a change manager
- Initially, as an additional column _update in DataFrames
  - Not scalable (time to compute changes linear in # of rows)
  - Bugs in Pandas DataFrame subclassing

Data Tables: Solution

- ProgressiVis Tables are column oriented
  - Growable
  - Trade-offs to keep columns contiguous
    - mmap for growable memory with persistence
  - Chunked columns would be better but not well supported by other libraries
  - ProgressiVis allows using other basic implementations for columns (Zarr, blosc, HDF5)
    - but not efficient currently
- Views for slices AND masked/filtered
- Change tracking for the Change Manager

Change Manager

- When re-entering an algorithm with updated data, the algorithm needs to know what has changed upstream
- Min module takes a column (or many)
  - computes the min the first time
  - maintains it when the column changes
  - but what has changed?
- Structure called IndexUpdate (Delta):
  - what has been created, updated, deleted
  - Compressed bitmaps (RoaringBitmap)

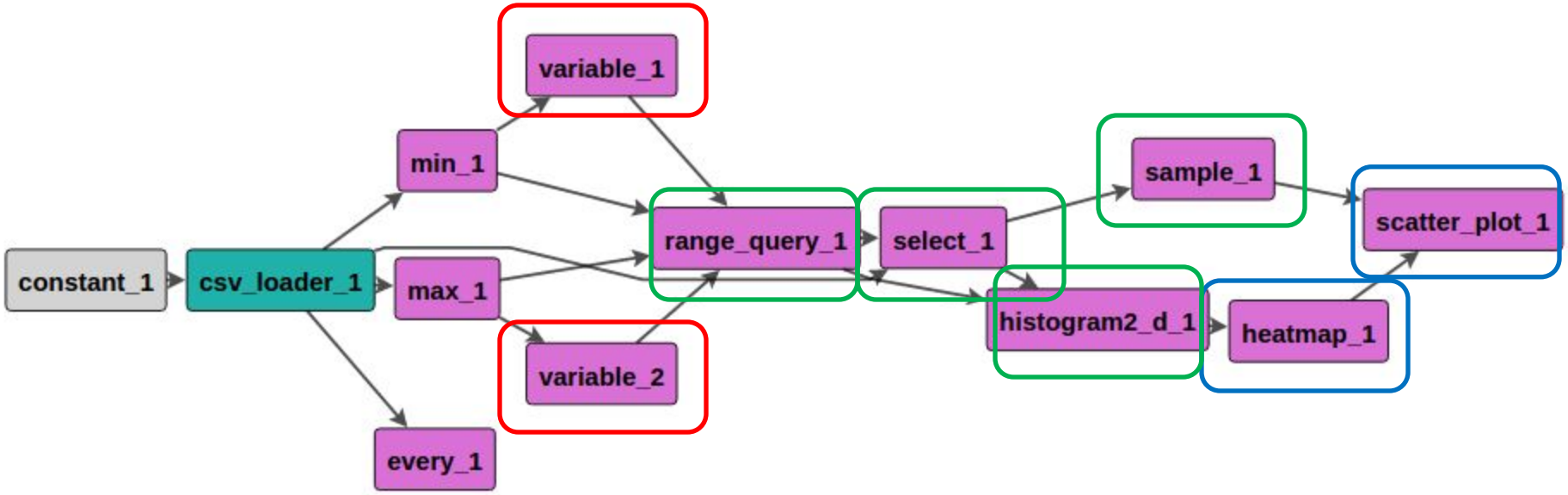
Example: **`Min`** module

```
async def run_step(self, run_number, step_size, howlong):
    dfslot = self.get_input_slot('table')
    dfslot.update(run_number)
    if dfslot.updated.any() or dfslot.deleted.any():
        dfslot.reset()
        self._table.resize(0)
        dfslot.update(run_number)
    indices = dfslot.created.next(step_size)
    steps = len(indices)
    if steps==0:
        return self._return_run_step(self.state_blocked, steps_run=0)
    input_df = dfslot.data()
    op = self.filter_columns(input_df, indices).min(keepdims=True)
    if len(self._table)!=0:
        last = self._table.last()
        for colname in last:
            current_max = op[colname]
            current_max[0] = np.minimum(current_max, last[colname])
    self._table.append(op)
    return self._return_run_step(self.next_state(dfslot), steps_run=steps)
```

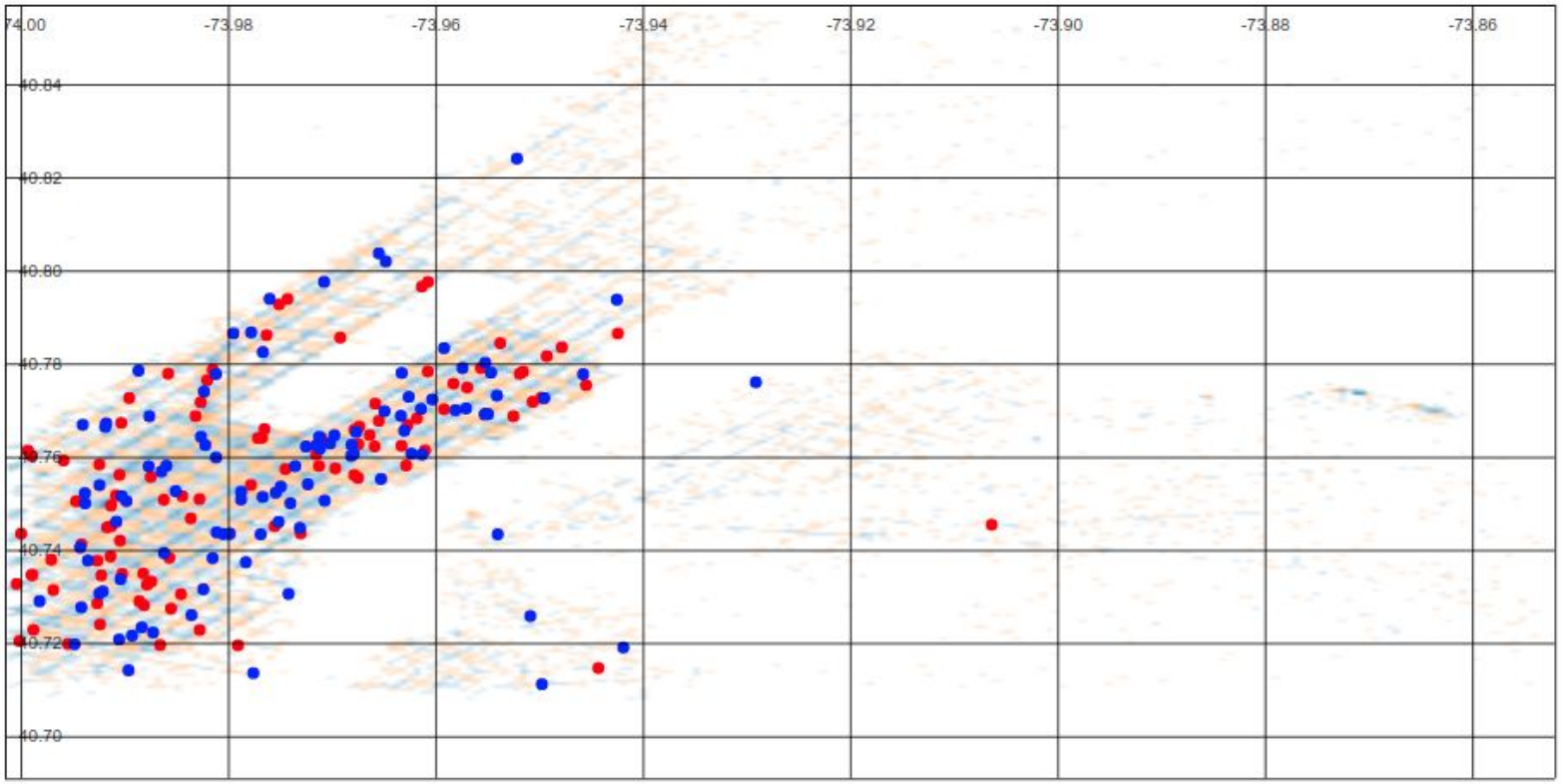
Interaction and Steering

- The Scheduler has a special mode for “direct manipulation”
- 2 special types of modules: *input* and *output*
  - *Inputs* can receive messages from the outside world at any time
    - e.g. `{'query': '-74.20 < pickup_longitude < -73.1'}`
  - *Outputs* show information to the outside world (e.g. visualizations)
- Direct manipulations restrict the execution to modules between the touched inputs and the dependent outputs during 100ms (latency 1)

ProgressiVis: Interaction and Steering

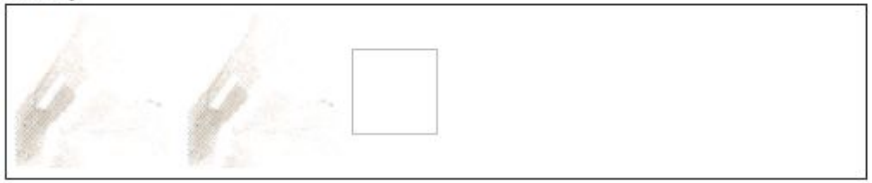


Scatterplot | Module graph



Filter to viewport | Blur radius | Color map | Default | Configure...

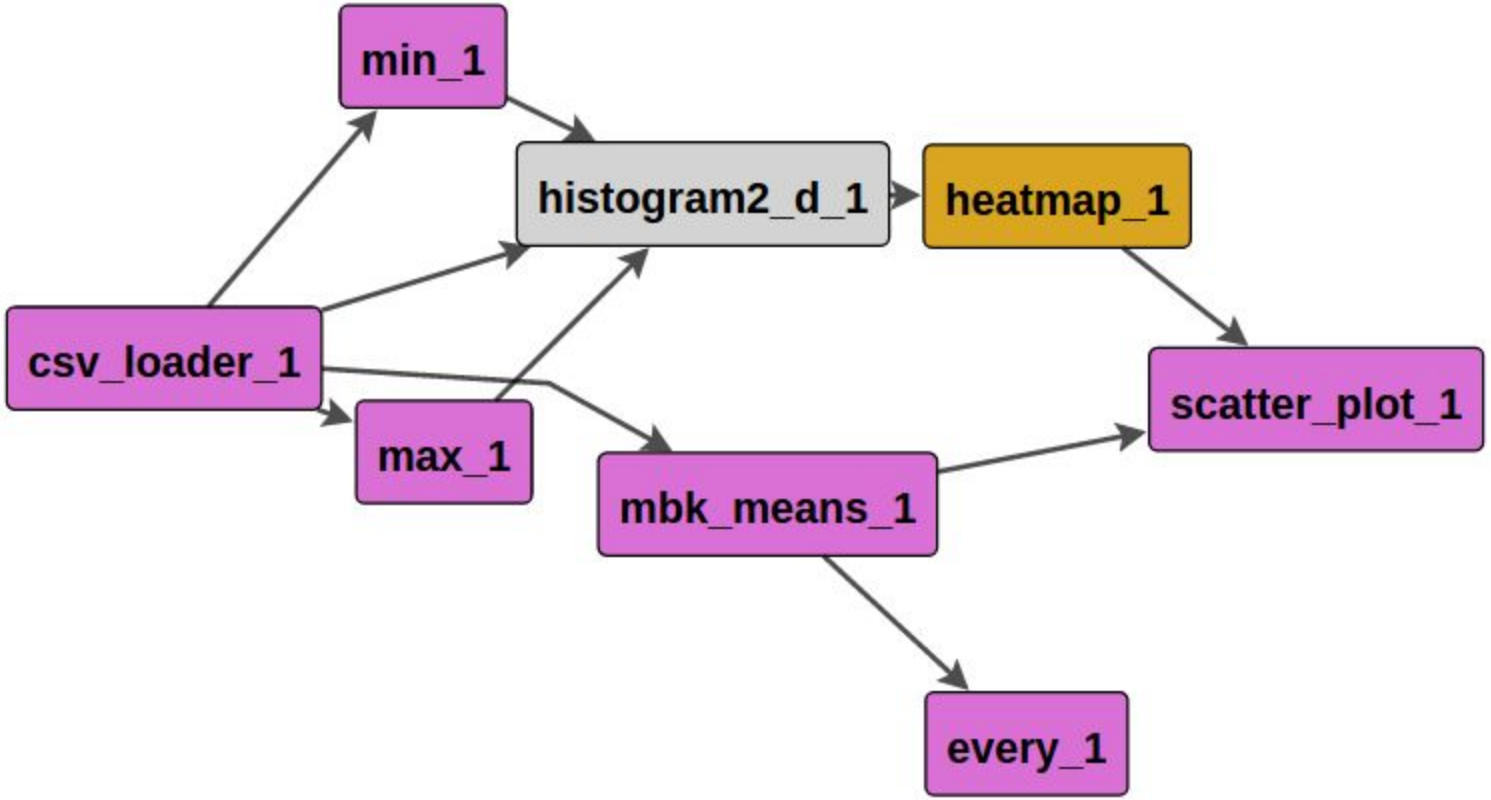
History



K-Means Clustering

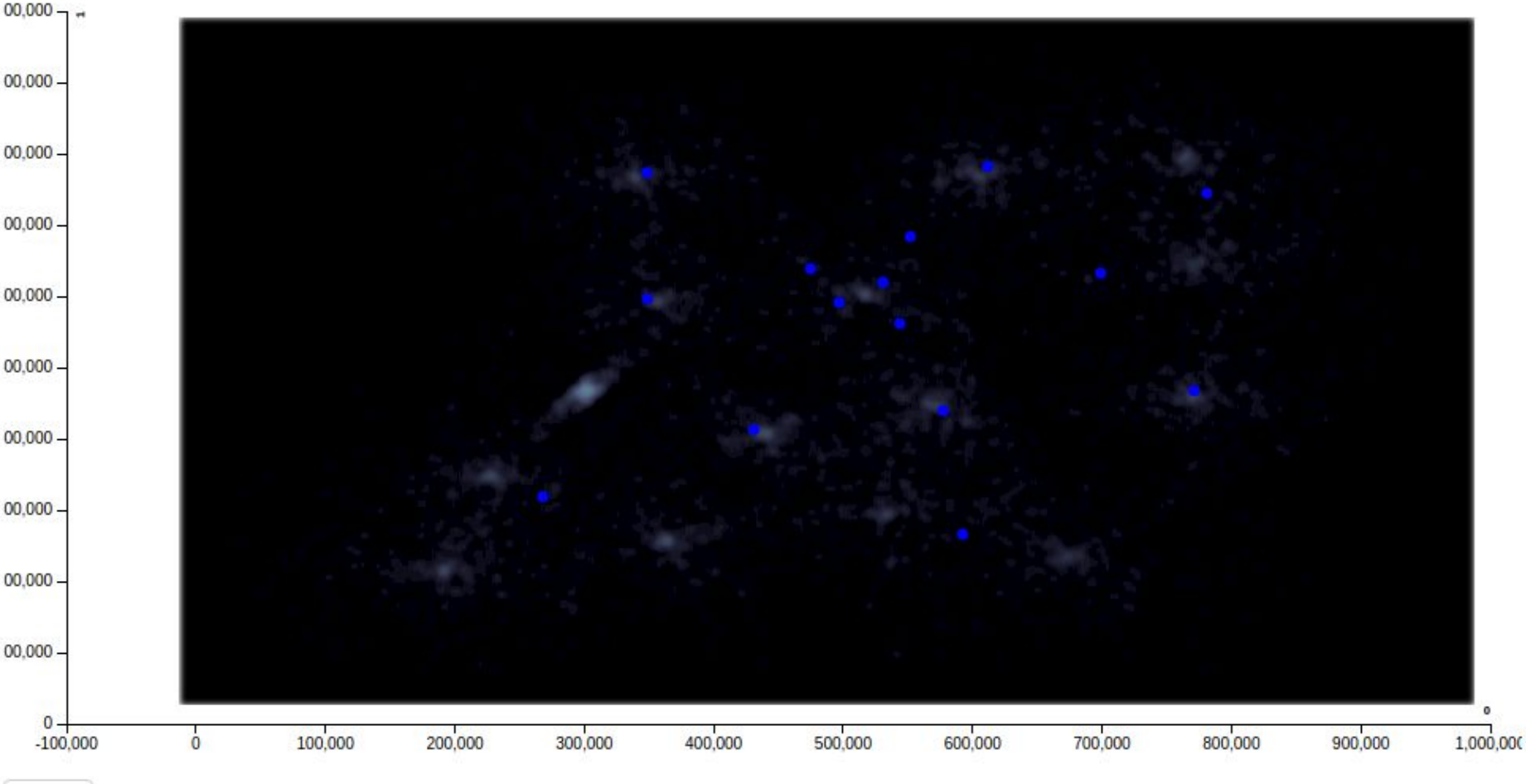
```
data = CSVLoader(get_dataset('cluster:s3'))
mbkmeans = MBKMeans(columns=[0, 1], n_clusters=15)
mbkmeans.input.df = data.output.df
prn = Every()
prn.input.df = mbkmeans.output.df
sp = ScatterPlot(0,1)
sp.move_point = mbkmeans
histogram2d = Histogram2D(0, 1)
histogram2d.input.df = data.output.df
min = Min(columns=[0,1])
max = Max(columns=[0,1])
min.input.df = data.output.df
max.input.df = data.output.df
histogram2d.input.min = min.output.df
histogram2d.input.max = max.output.df
heatmap = Heatmap()
heatmap.input.array = histogram2d.output.df
sp.input.heatmap = heatmap.output.heatmap
sp.input.df = mbkmeans.output.df
```

ProgressiVis:
Interaction and Steering



Scatterplot for Module scatter_plot_1

Scatterplot | Module



Challenges

Progressive Data Analysis raises many questions

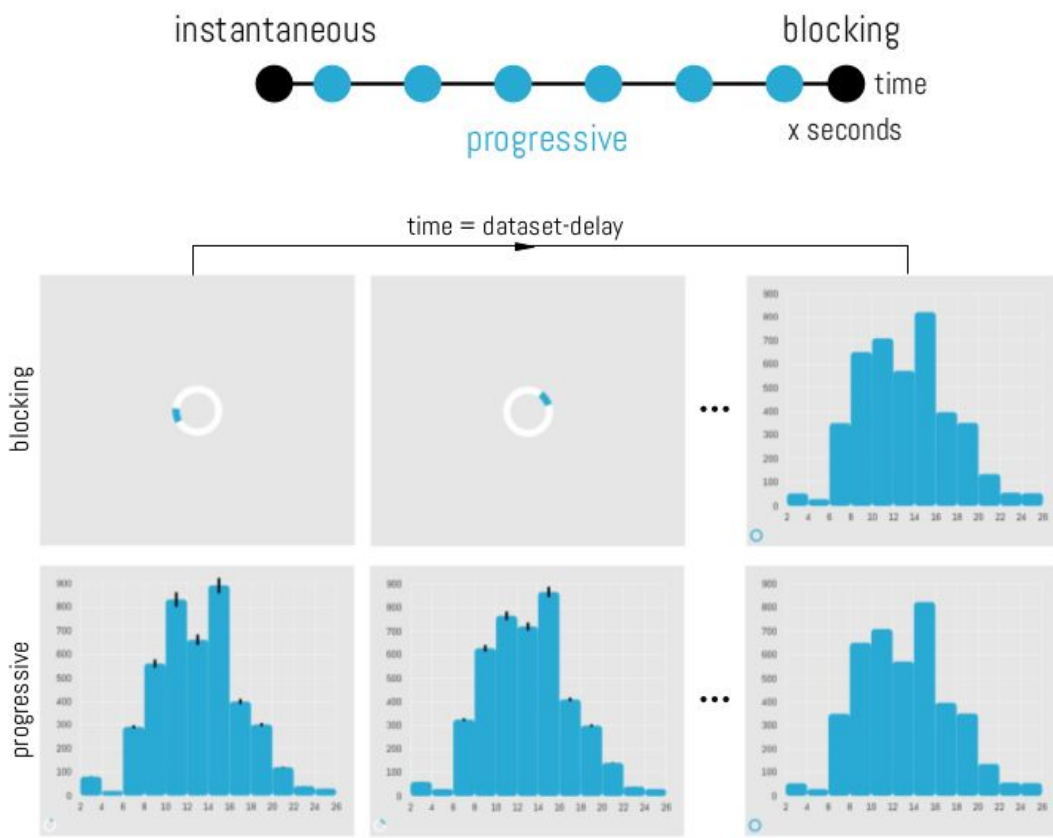
- HCI
- Visualization
- Analytics
- Data management

Challenges in HCI

- Are humans able to cope with progressive data monitoring?
  - Latency is detrimental to exploration: "Our study confirms that an injected delay of half a second per operation adversely affects user performance in exploratory data analysis."
    - Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. IEEE Transactions on Visualization and Computer Graphics, 20(12):2122–2131, Dec2014.
- How can we help analysts make sense of data arriving progressively?

How Progressive Visualizations Affect Exploratory Analysis [Zgraggen et al. TVCG 2017]

- Experiment
- 4 conditions
  - Instantaneous
  - Progressive 6s, 12s
  - blocking
- 3 datasets
- Count insights

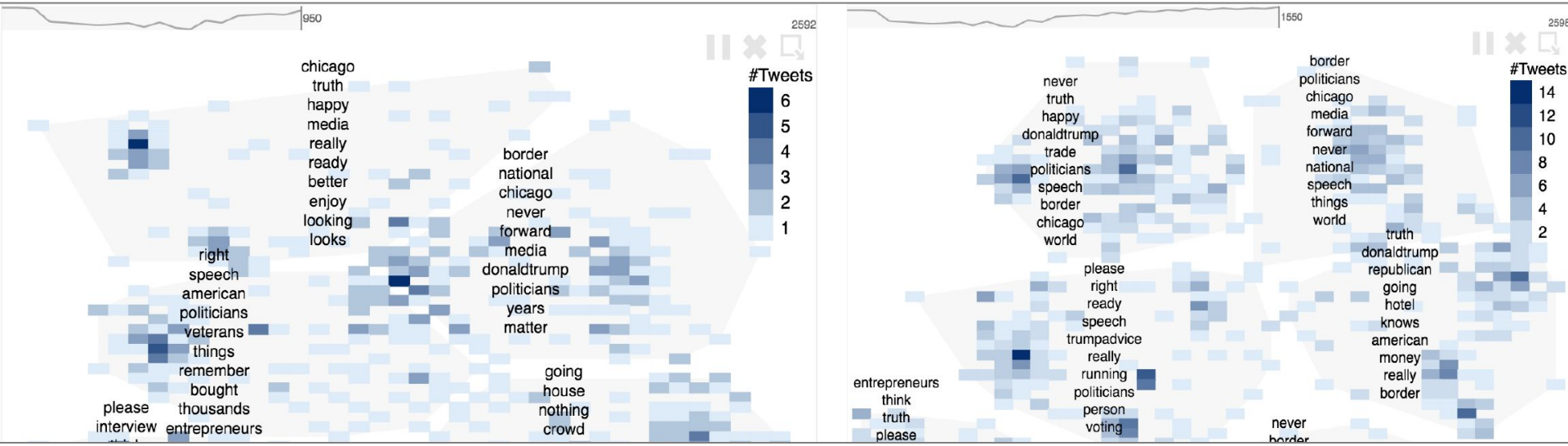


Latency and Exploratory Analysis

- E. Zgraggen, A. Galakatos, A. Crotty, JD Fekete, T. Kraska, How Progressive Visualizations Affect Exploratory Analysis, TVCG 2017
- Experiment with 4 conditions:
  - Instantaneous, Progressive, Latency of 6s and 12s
- Measure # of insights generated by analysts
- Measure coverage explored
- Instantaneous and progressive generate more insights ($p < 0.005$) and more coverage
- Participants liked the progressive condition and disliked the blocking conditions.

Steering the Craft

UI Elements and Visualizations for Supporting Progressive Visual Analytics [Badam et al. 2017]



Sriram Karthik Badam,

Niklas Elmqvist, Jean-Daniel Fekete

Human Computer Interaction Laboratory

UNIVERSITY OF MARYLAND

Aviz

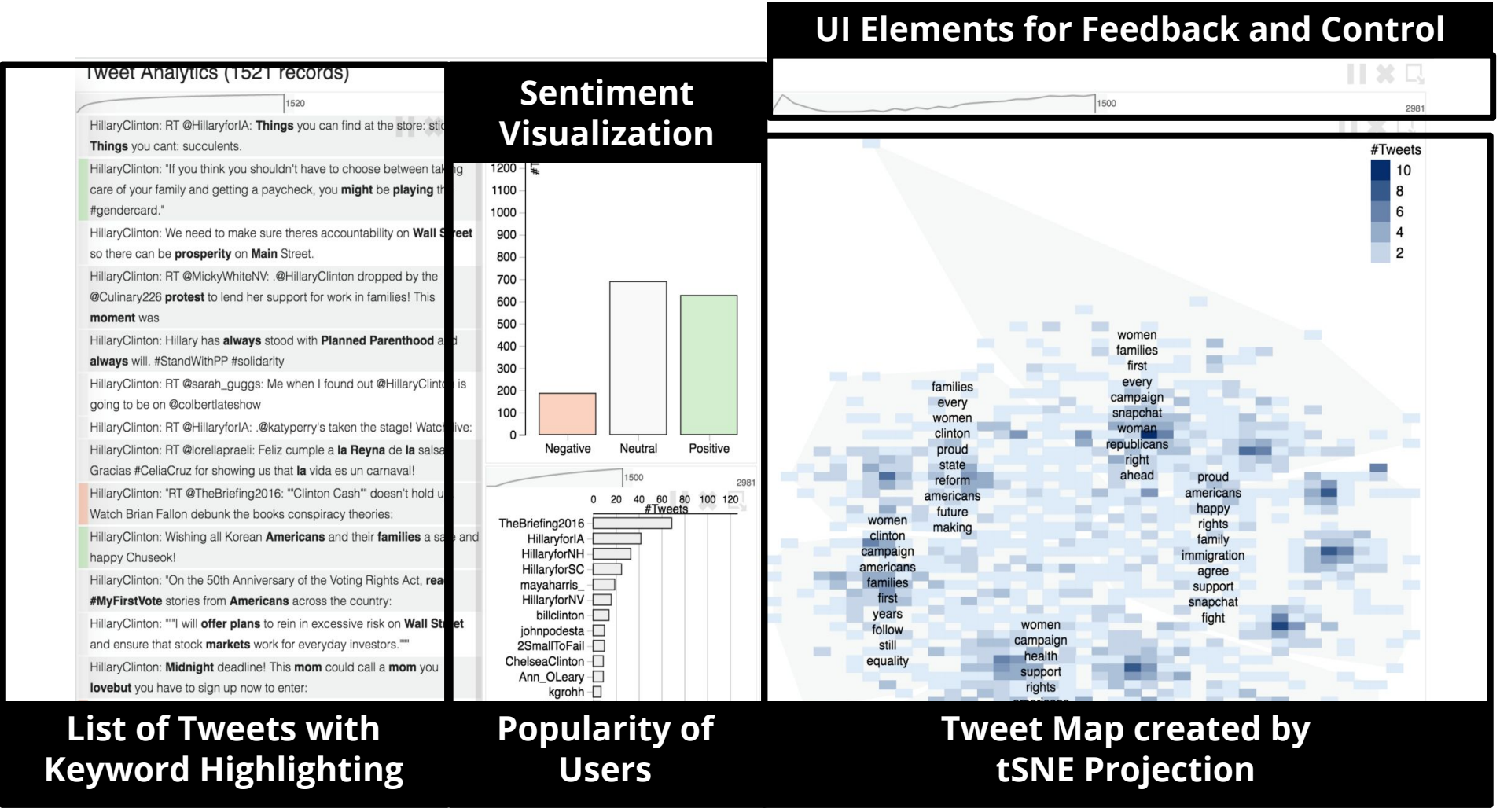
Inria INVENTEURS DU MONDE NUMÉRIQUE

UI Elements and Visualizations for Progressive Visual Analytics

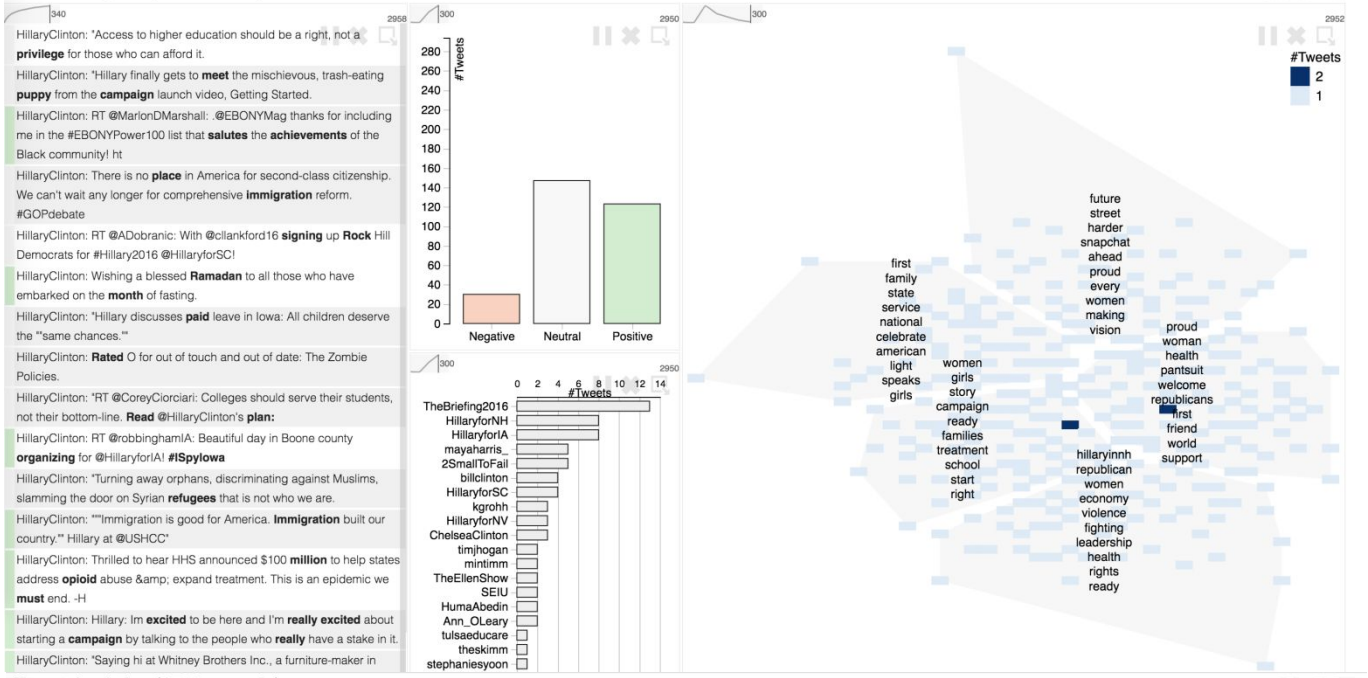
- S. K. Badam, N. Elmqvist, JD Fekete, UI Elements and Visualizations for Supporting Progressive Visual Analytics, Computer Graphics Forum, Volume 36, Issue 3 June 2017 , ages 491–502

- What information should we provide to analysts to benefit from PVA?
  - Early decision
  - Time remaining to complete
  - Is it converging / useful?
  - Monitor mode vs. exploration mode
  - Consistency!

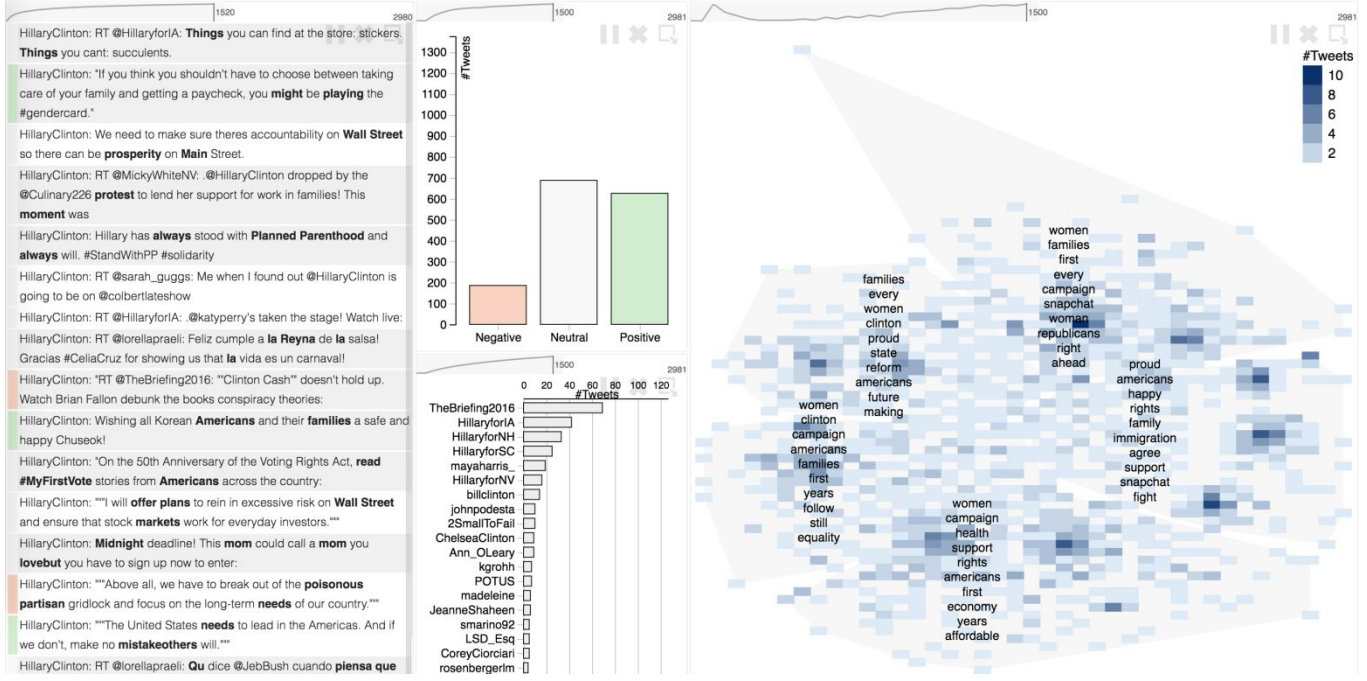
Interface: InsightsFeed for Twitter Data



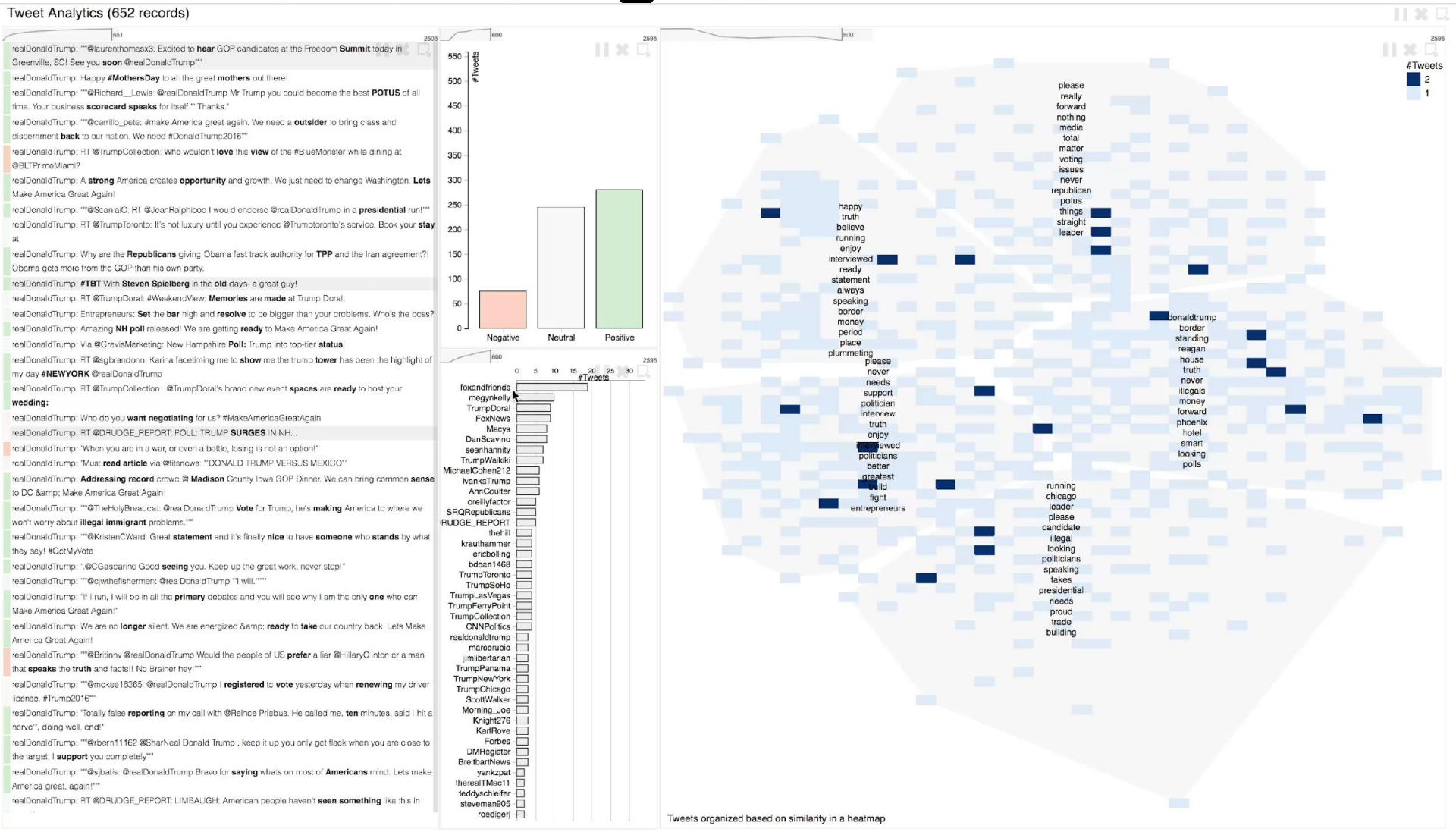
Tweet Analytics (341 records)



Tweet Analytics (1521 records)

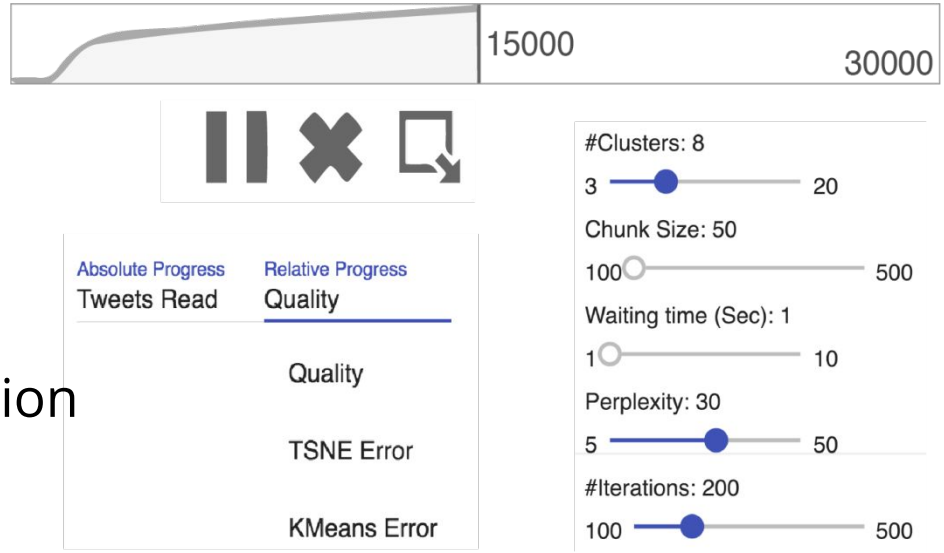


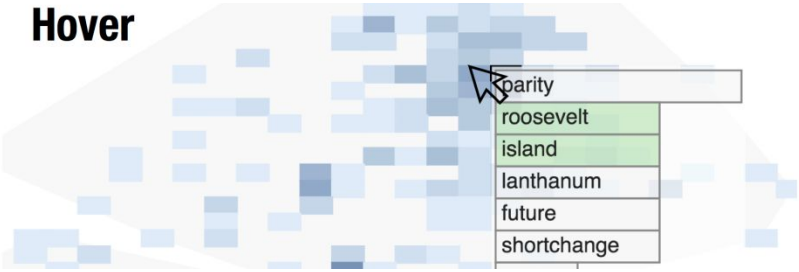
InsightsFeed

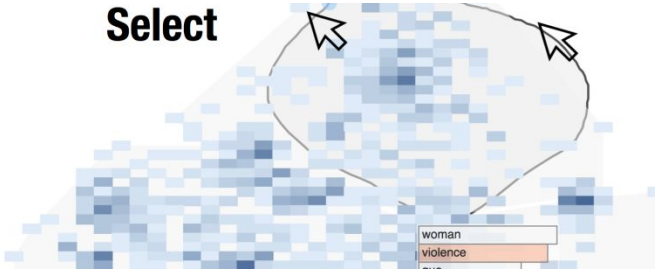


Five UI Elements for PVA

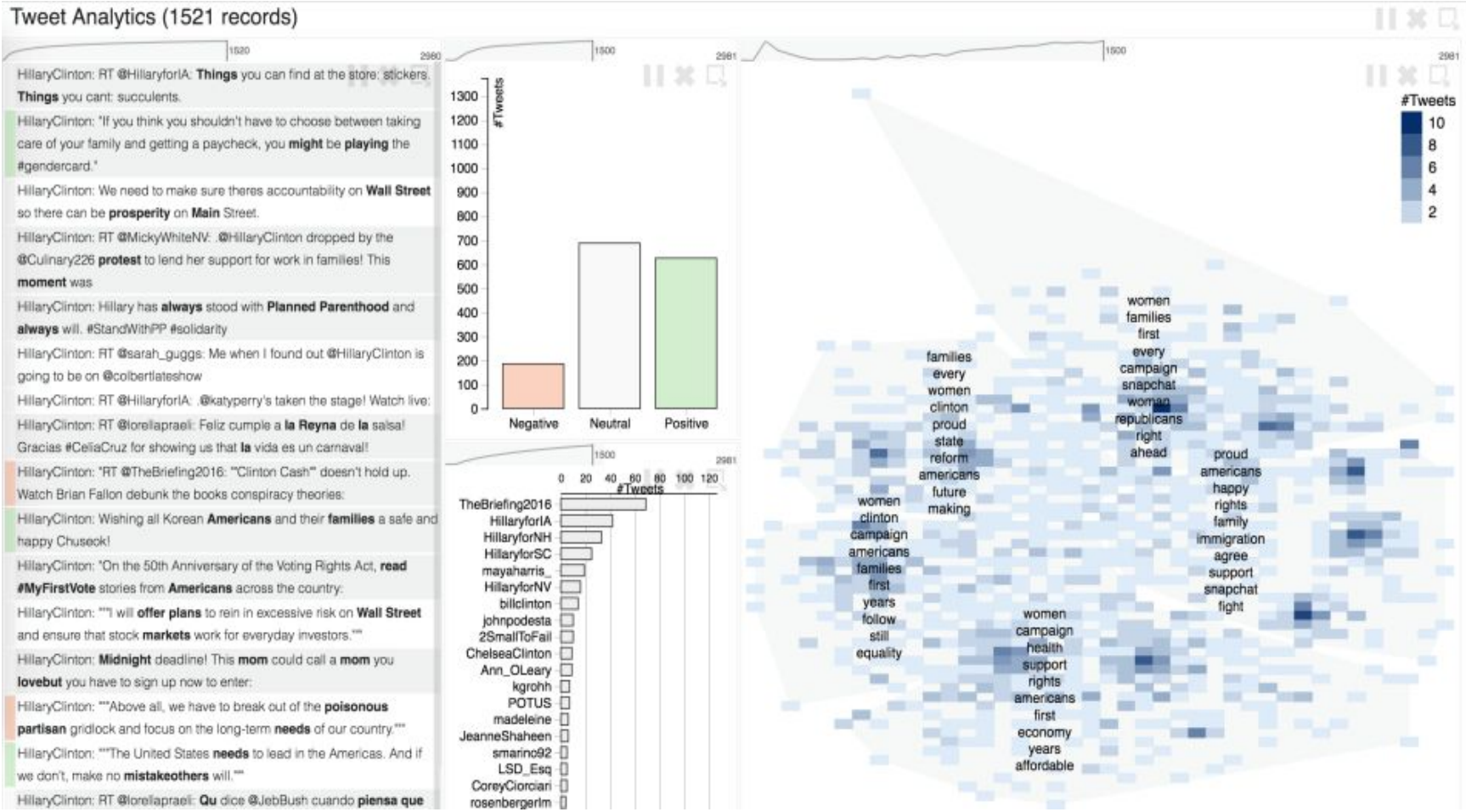
- Progression towards stability
- Enhanced progress bars
  - Quality of computations
- Controlling playback
- Interactivity for visual exploration
- Steering results



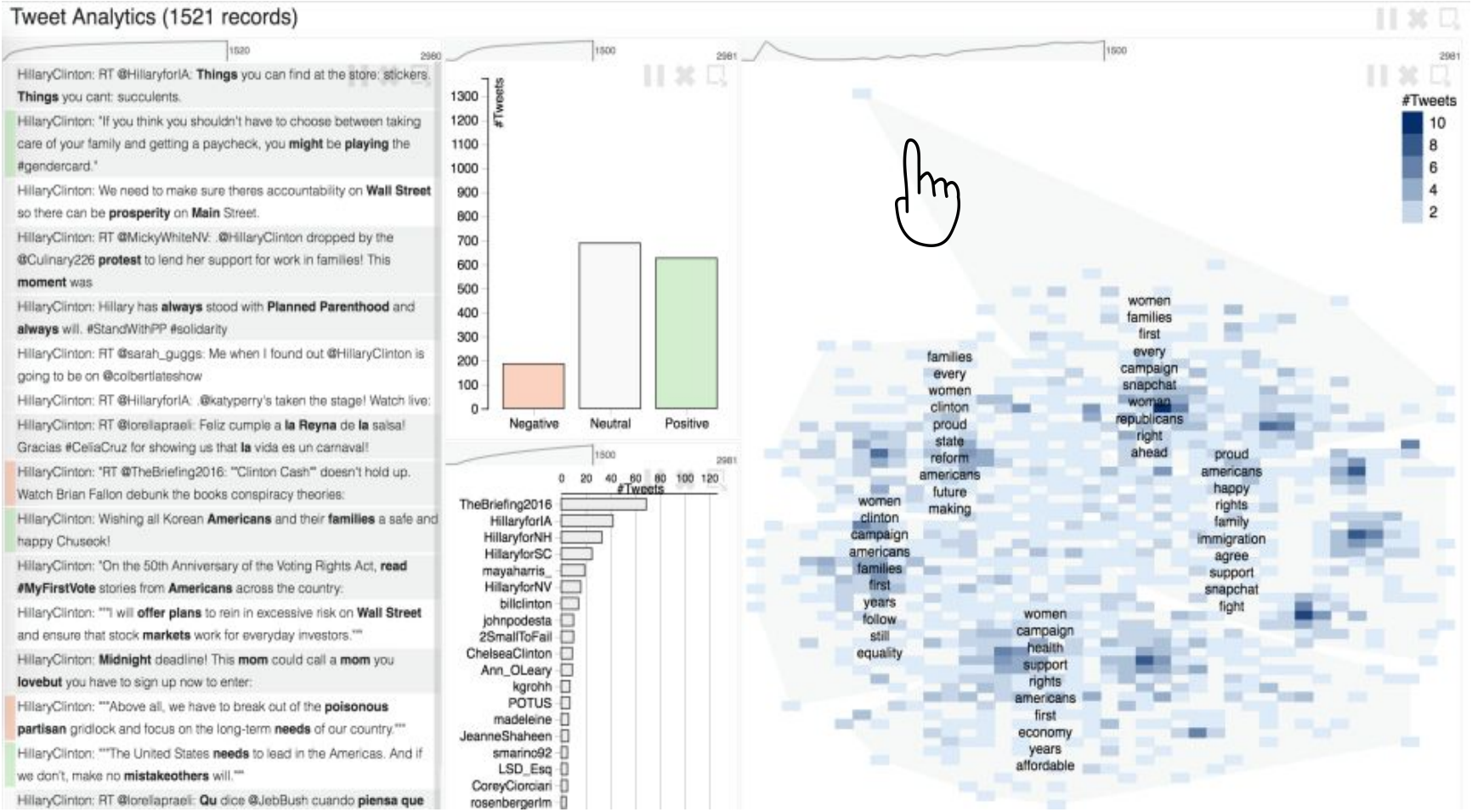




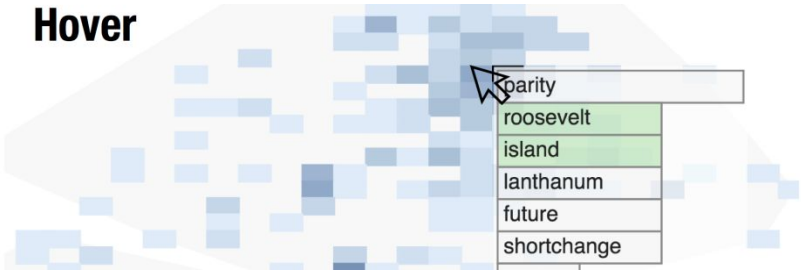
Enhanced Progress Bars

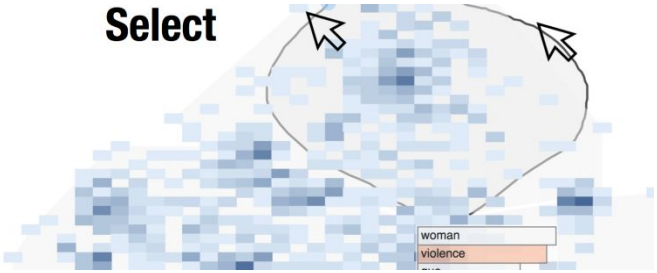


Control Playback

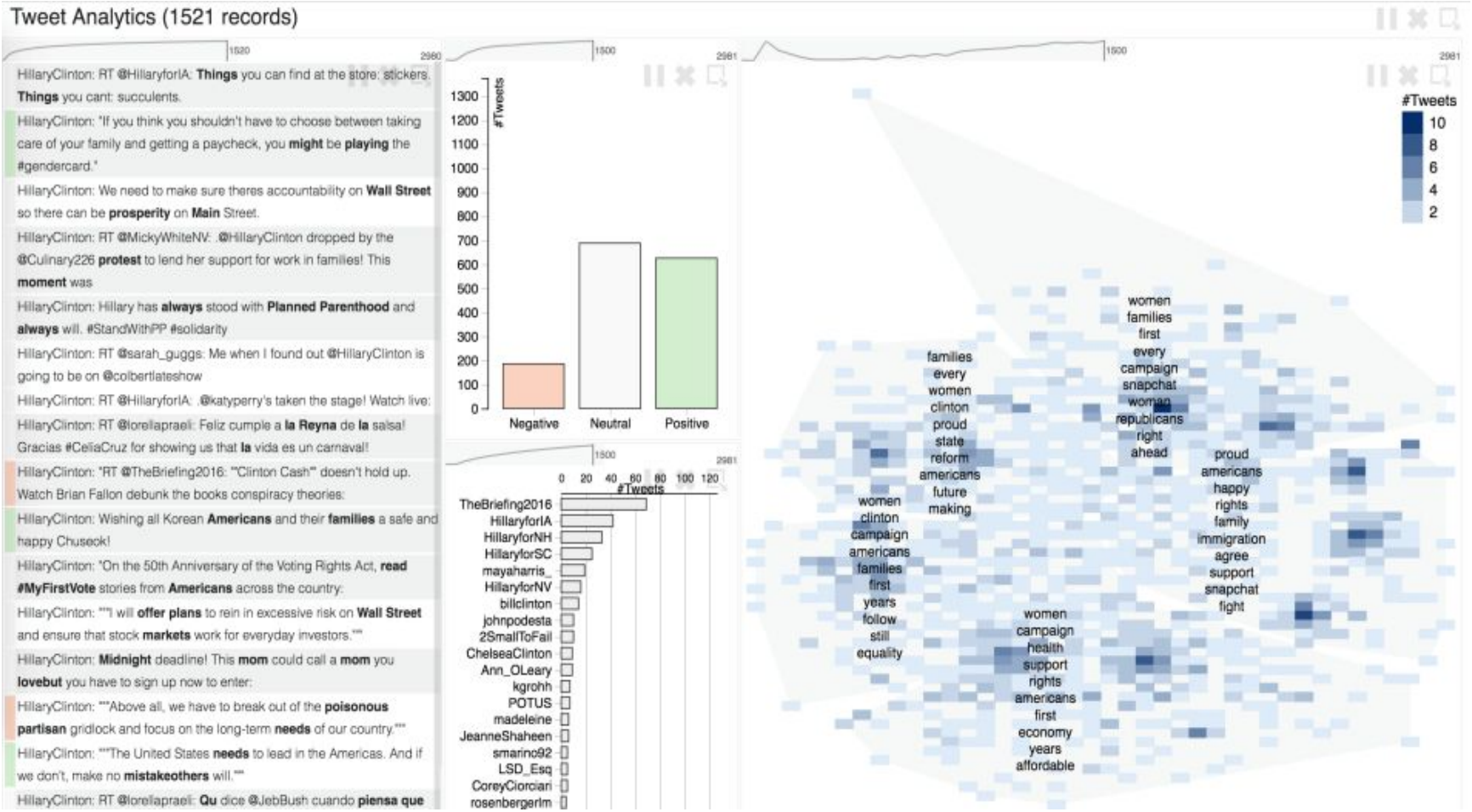


Interactivity for Visual Exploration





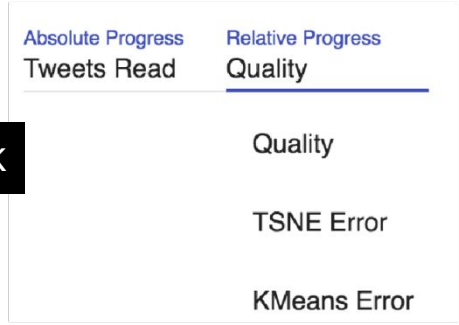
Control Playback

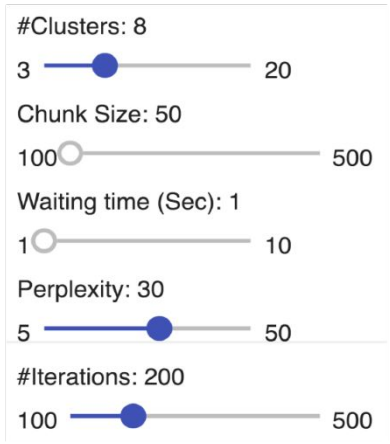


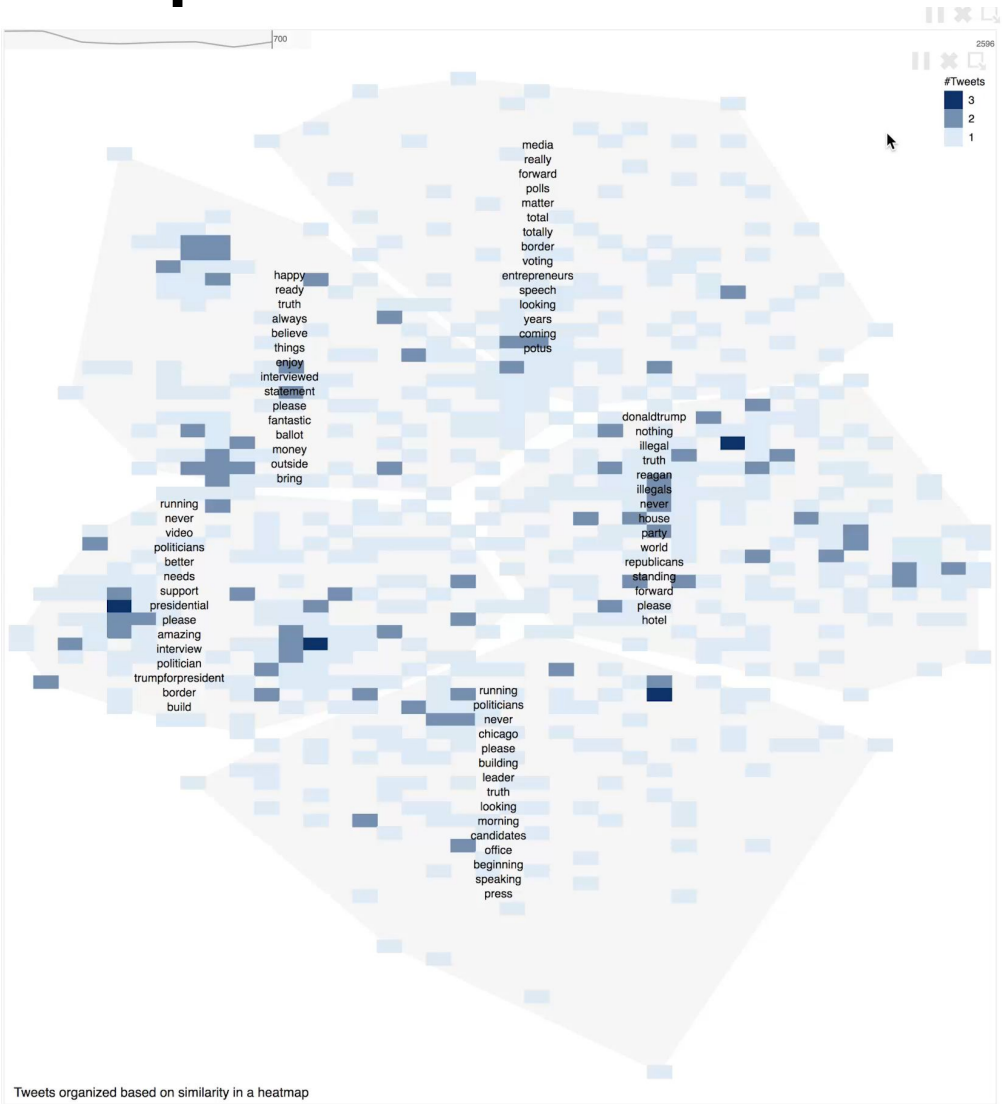
Steering Computations

Feedback

Controls



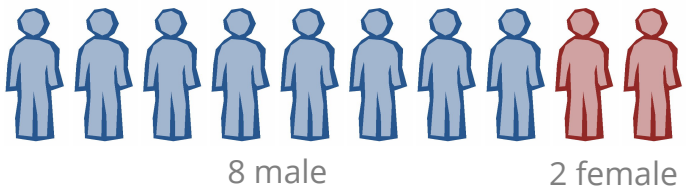




User Evaluation

Goal**:** Understanding **effects** of progressive **UI elements** on visual exploration

10 participants from HCI/visualization research labs



2 interfaces (25 min each)

Progressive Visual Analytics

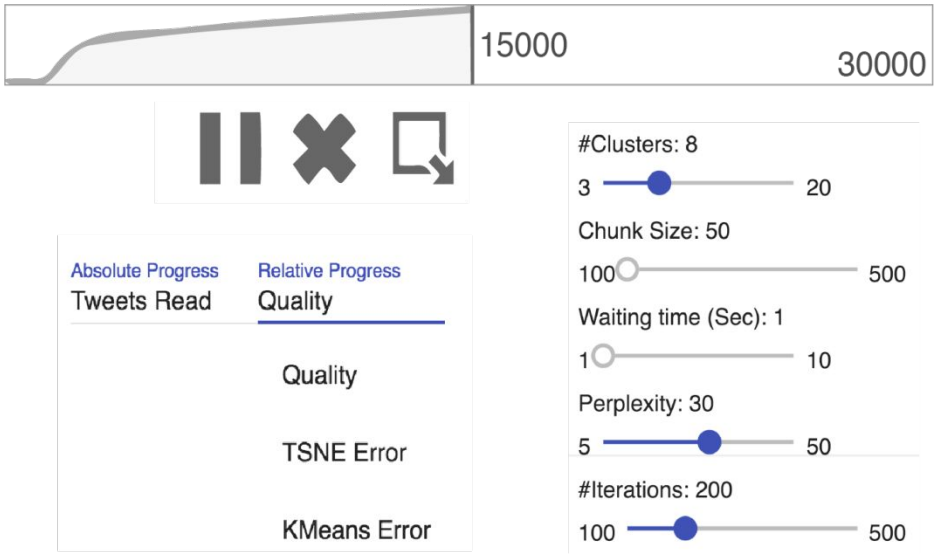
vs. Instantaneous Visual Analytics

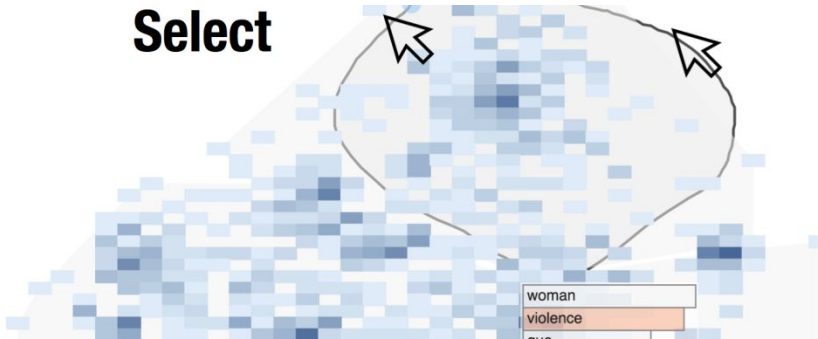
5 questions from two Twitter datasets (Clinton, Trump)

Q1: most frequent sentiment
Q2: popular users
Q3: frequent keywords
Q4: representative tweet
Q5: keywords associated with popular terms

Takeaway: Five UI Elements for PVA

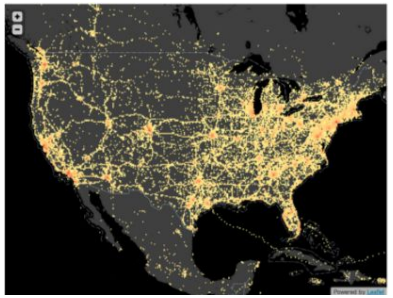
- Progression towards stability
- Enhanced progress bars
  - Quality of computations
- Controlling playback
- Steering results
- Interactivity for visual exploration





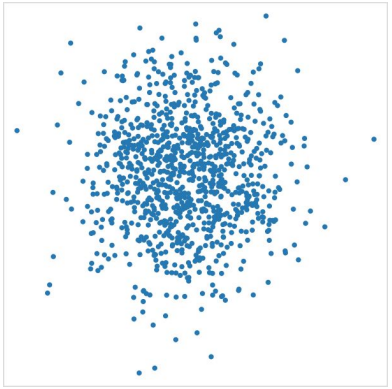
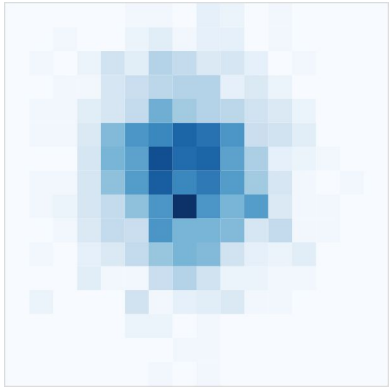
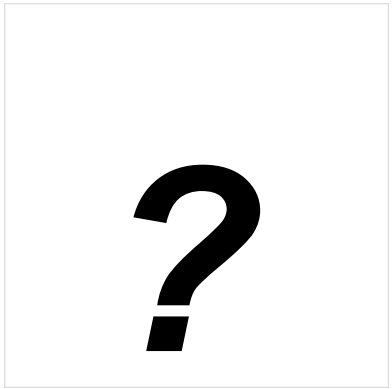
ProgressiVis: Challenges in Visualization

Adapting existing visualization techniques to become progressive



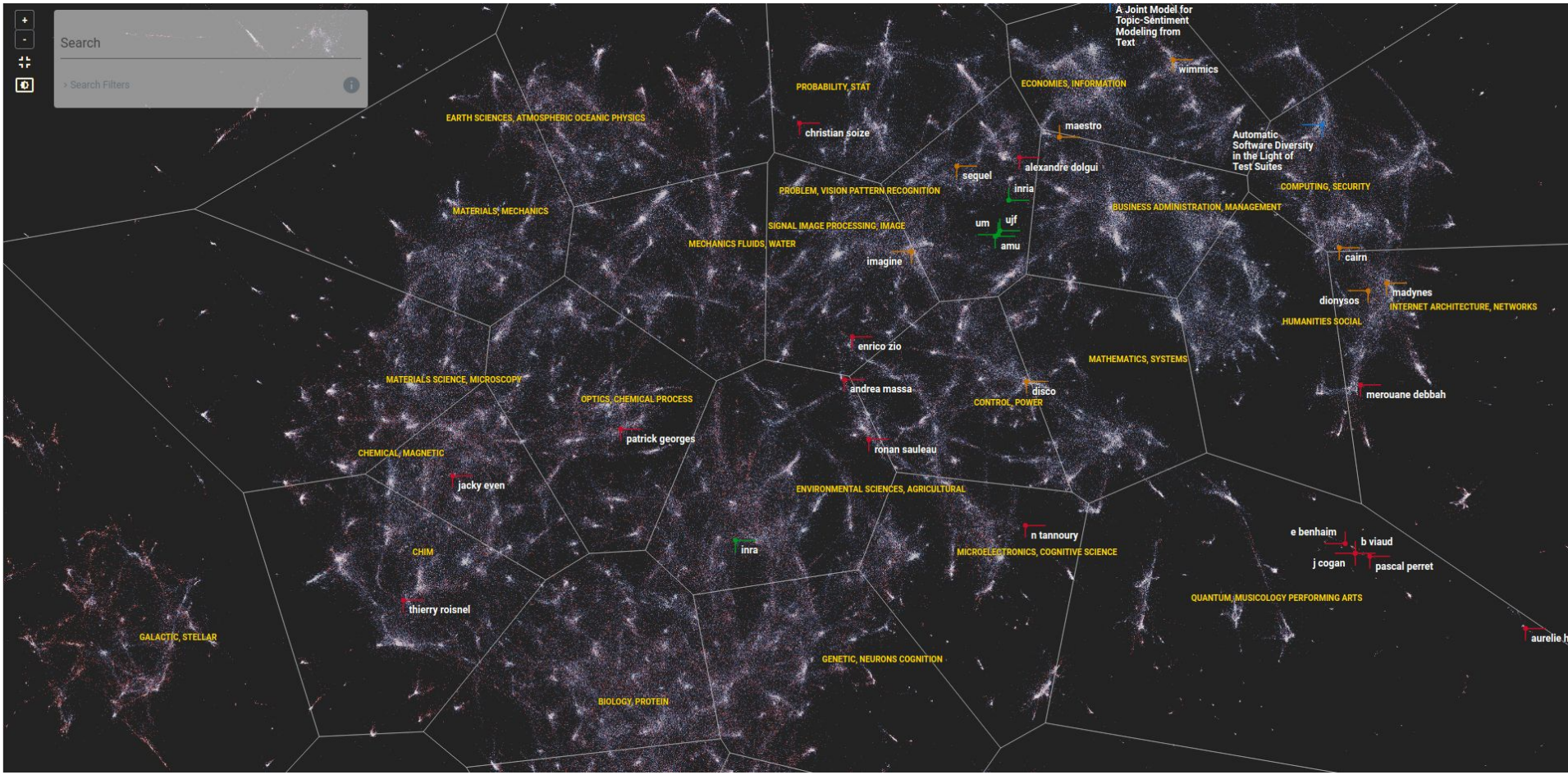
- Managing the scale
  - Scatterplots was an example, using aggregation+sampling (+landmarks)
  - what about the others techniques?
- Managing the incoming changes
  - e.g. differentiate monitoring/exploration modes
- Managing the interaction

Challenges in Visualization

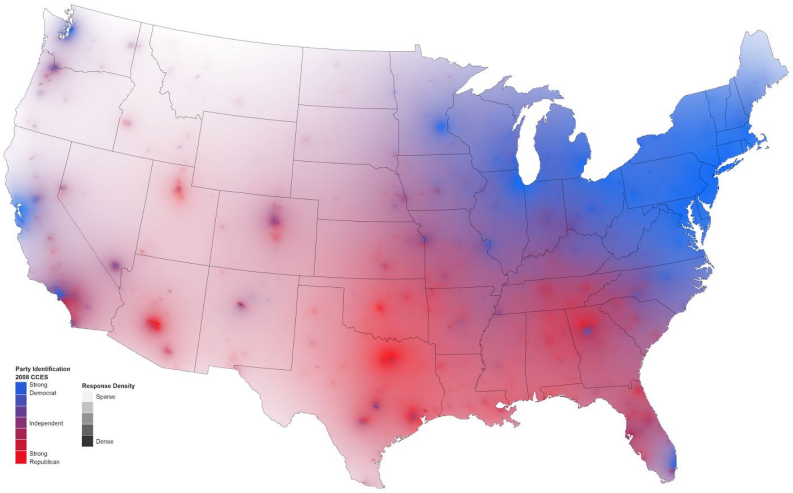
	Scatterplots	*Density Maps*
***Uni**class*		
***Multi**class*		

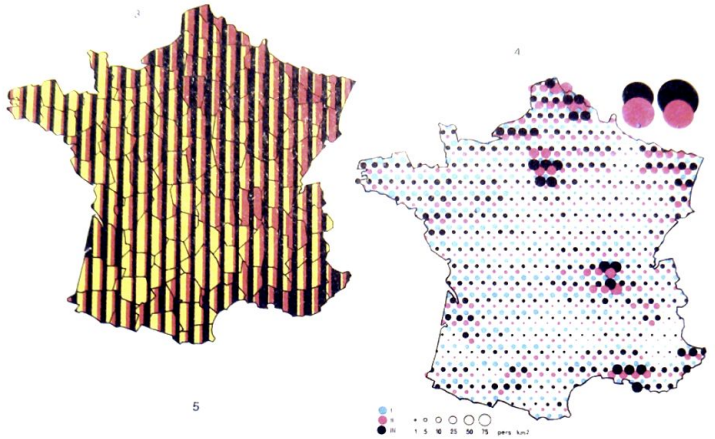
Density maps can scale up conventional scatterplots,
but it is nontrivial to visualize **multiclass** data on a density map.

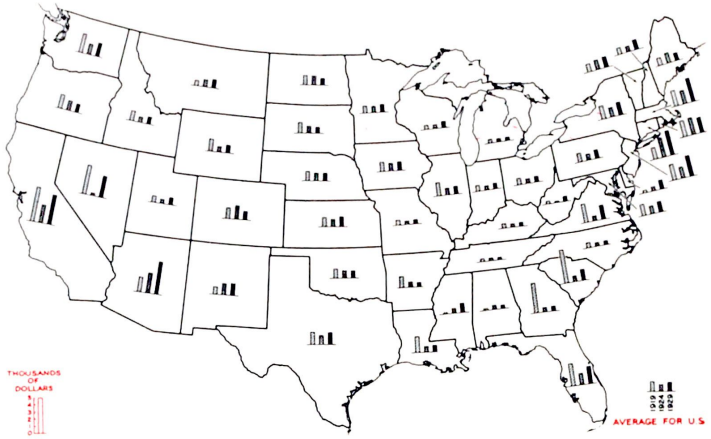
Challenges in Visualization

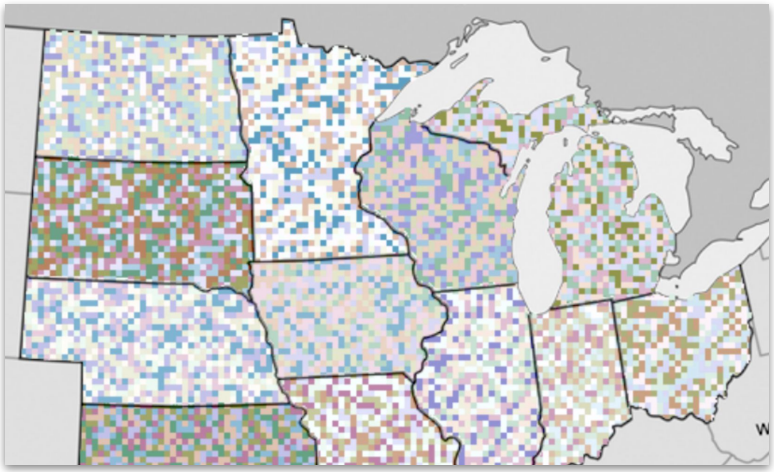


Challenges in Visualization

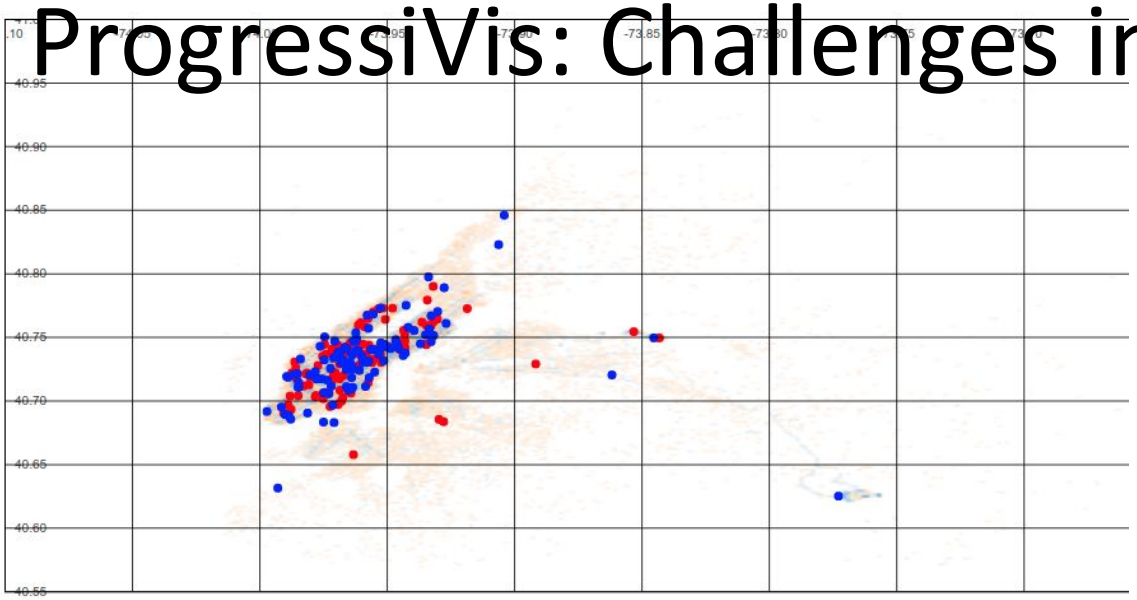








ProgressiVis: Challenges in Visualization



Filter to viewport | Blur radius | Color map | Default | Close props editor

History

Multiclass Density Map Editor

Rebin

Type None

Rescale

Type Cubic Root

Compose

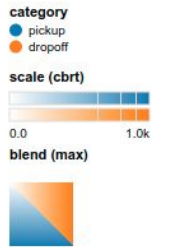
Mix Max

category
pickup
dropoff

scale (cbrt)

0.0 1.0k

blend (max)

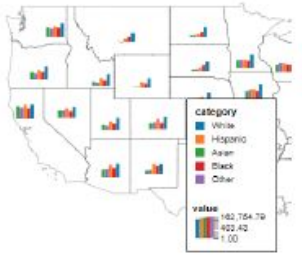


Specification

```
"rebin": // US rebinning
"compose": {
  "mix": "weavingrandom",
  "size": 2}
```

Specification

```
"rebin": // US rebinning
"compose": { "mix": "glyph",
  "glyphSpec": {
    "template": "bars",
    "width": 20, "height": 24}}
```



ProgressiVis: Challenges in Analytics

Integration with Python, Julia, R?

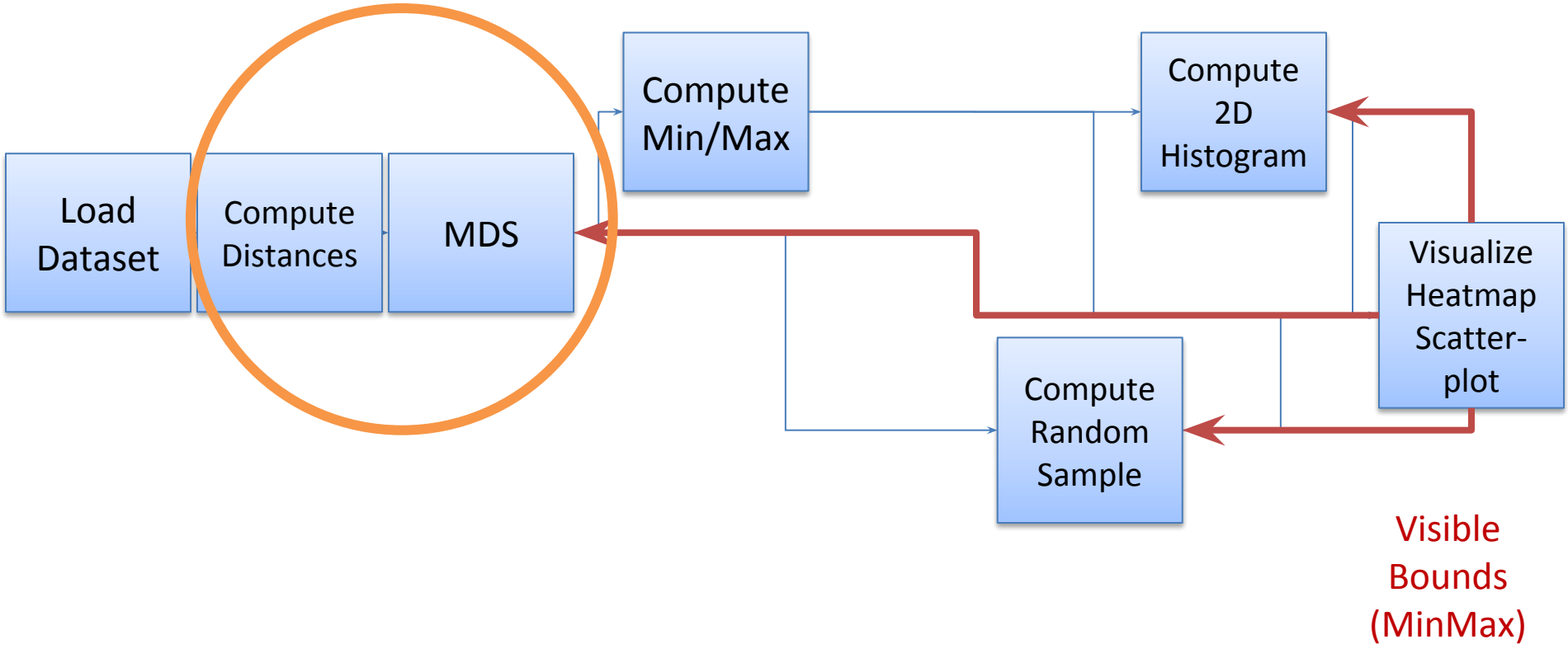
- Breaks the numpy/scipy API semantics
  - not limited to one function call with a ret. value
- Need to rebuild a data analysis stack
  - Progressive Data Loading/saving/access
  - Progressive Computations (fundamental algos.)
  - Progressive Linear Algebra
  - Progressive ML
  - Progressive VIS
  - Progressive HCI

ProgressiVis: Challenges in Analytics

What algorithms can be made progressive?

- The most useful operations can be done in a progressive way
  - Looking at Scikit Learn, about 80% can be made progressive, with various levels of efforts
- Some algorithms are challenging (e.g. hierarchical clustering) but there are possible replacements
- The quantum constrains the algorithms, they sometimes have to be adapted
- Can also use Data-Streaming Approx. methods such as "Data Sketching"

ProgressiVis: Computational Modules

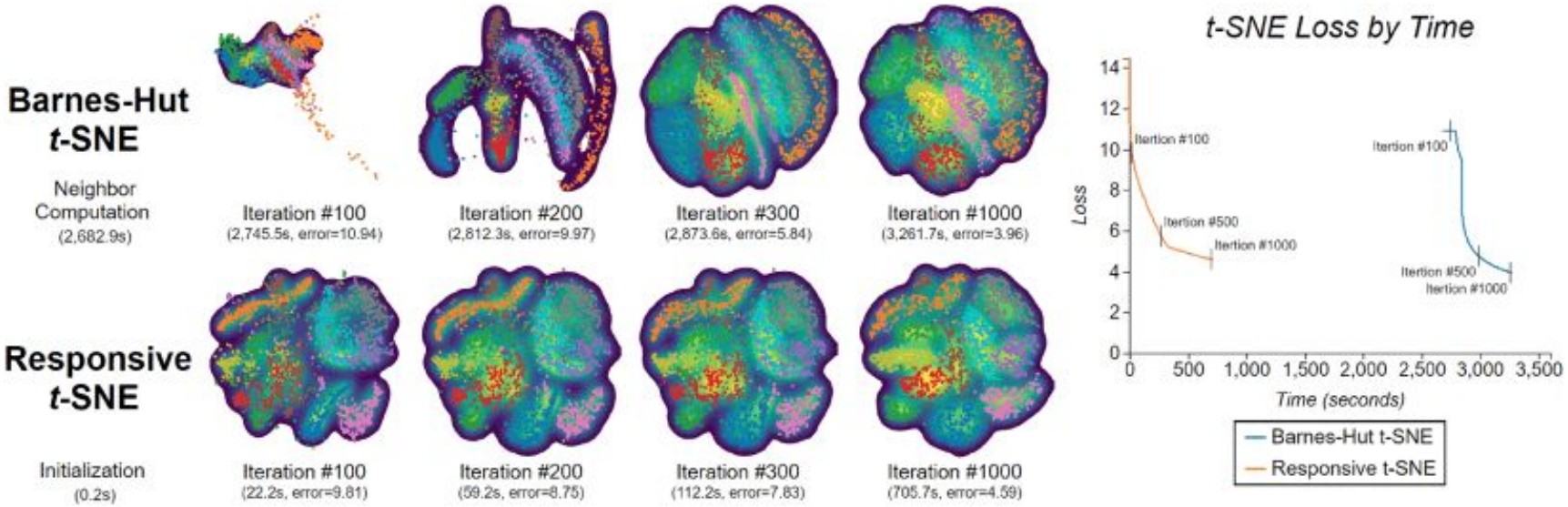


Adapting Distance Computation

- For a table with *n* rows, compute the *n×n* matrix for pairwise distances (e.g. Euclidean)
- Complexity is $O(n^2)$
- The progressive version computes the distances as they come when loaded, or in steps of less than 1s if they are already loaded
- Update the existing matrix of *m×m* with *p* new rows:
  - Compute the distances of (rectangular), *m×p* = T(m×p*)*
  - Compute the distances of *p×p* (square)
  - Assemble the final matrix
- However, approximate nearest neighbors are more adapted to progressive methods since O(n log(n))
  Jaemin Jo, Jinwook Seo, Jean-Daniel Fekete, **A Progressive K-D Tree for Approximate K-Nearest Neighbors,** Data Systems for Interactive Analysis (DSIA 2017) Workshop

Progressive Multidimensional Projections

Based on the (progressive) computation of the k-nearest neighbors



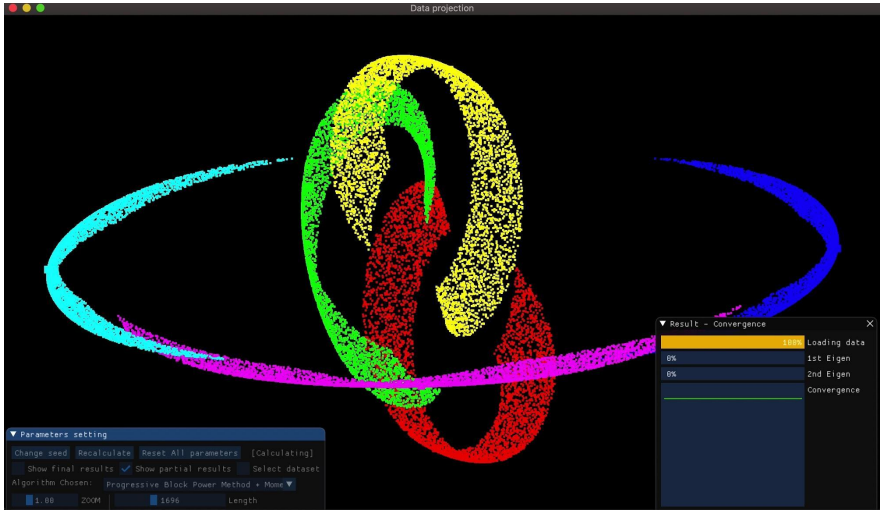
Jaemin Jo, Jinwook Seo, Jean-Daniel Fekete. **PANENE: A Progressive Algorithm for Indexing and Querying Approximate k-Nearest Neighbors**. IEEE Transactions on Visualization and Computer Graphics, IEEE, 2020, 26 (2), pp.1347-1360.

Progressive PCA for Massive Time-Series

Multiple methods to compute PCA.

Three can be used in a progressive setting:

1. Power-Iteration with Momentum
   C. De Sa, B. He, I. Mitliagkas, C. Ré, and P. Xu. Accelerated stochastic power iteration. arXiv preprint arXiv:1707.02670, 2017
2. Randomized PCA
   N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM review, 53(2):217–288, 2011
3. Incremental PCA
   D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking.International journal of computer vision, 77(1-3):125–141, 2008.



ProgressiVis: Challenges in Data Management

- Progressive Loading (data not in the database)
  - when the stored order is correlated with some attribute (e.g. temperature or time), Min/Max computation does not converge quickly
  - Need to shuffle or access data stochastically
- Progressive Queries
  - Online queries/aggregation/joins
- Progressive Computations
  - Stochastic gradient descent *already addressed for some cases*
  - More general computations?

ProgressiVis: Humanized Analytics

- Computation infrastructure are meant to optimize machine resources
- Exploration needs to optimize human resources

- If you're interested in any of the challenge, let me know!

http://github.com/jdfekete/progressivis

Work in progress!