

Progressive visualization

CS594: Big Data Visualization & Analytics

Fabio Miranda

<https://fmiranda.me>

Incremental visualization



- Queries over large-scale data are often not interactive.
 - Slow queries, users must wait for result.
- Potential exploration paths are ignored because it will take too much time to run certain queries.
- Potential solution: query process with incremental visualization.

Enabling interactive visualization

Progressive
analytics

- Analysis is program centric:
 - Analysis will read data, process it and store its results.
 - Analysis will produce unbounded amounts of data in unbounded time.

Progressive
visual analytics

- Visualization is user centric:
 - Visualization will only show a small amount of data.
 - Visualization needs to be interactive.
 - How to enable interactivity?

Progressive
DB

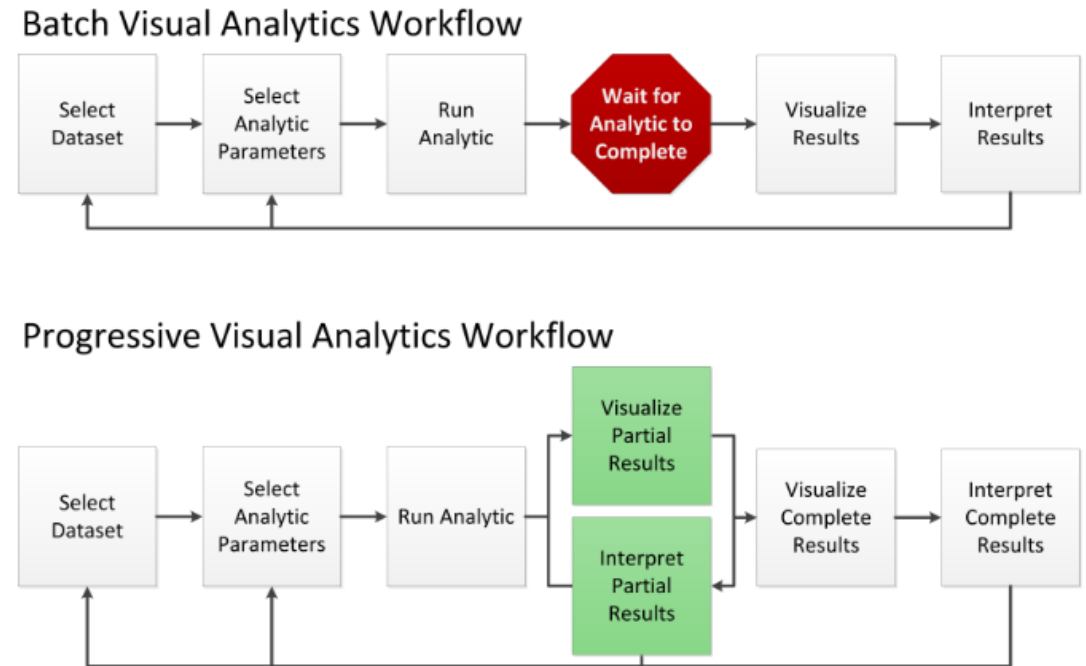
- Database is data centric:
 - DB will store and retrieve unbounded amounts of data in unbounded time.

Progressive analytics

- Progressive analytics are statistical algorithms that produce semantically meaningful partial results that improve in quality over time.
- Many algorithms produce partial results during execution, only a subset produce meaningful partial results.

Progressive visualization

- Progressive analytics: produce more complete results over time.
- User-driven analytics: statistical algorithms that can adapt to input by analysts.
- Progressive visualization: incorporate partial results as they are produced by an analytical algorithm.
 - Enable exploratory tasks while computation is being done.



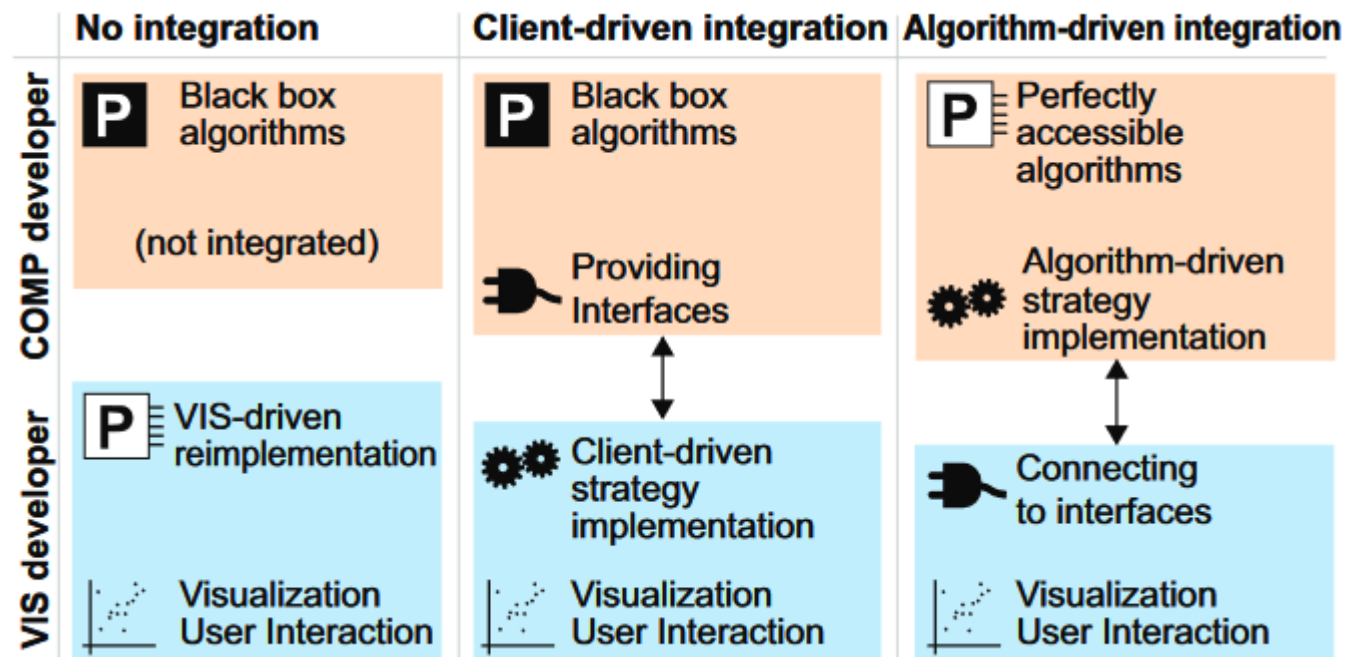
[Stolper et al., 2014]

Progressive visual analytics

- Provide user with meaningful intermediate results. Based on the intermediate results, the user can start with the analysis process.
- Set of strategies for enabling user involvement in ongoing computations of algorithm P . Key idea: execution of an algorithm by a series of smaller steps.
 - Data subsetting: perform computations of P for increasingly larger subsets
 - Complexity selection: perform computations of P for less complex parameter configurations.
 - Divide and combine: subdivide a workload W into n disjoint parts and apply P independently.
 - Dependent subdivision: subdivide P into sequentially dependent steps P .

Progressive visual analytics

Separation between
visualization tool and
computational
environments.



[Muhlbacher et al., 2014]

Progressive visual analytics

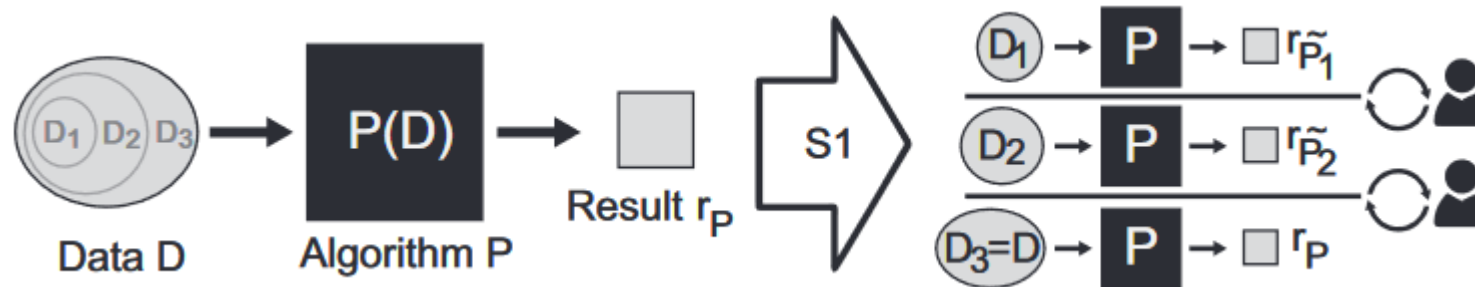
- Muhlbacher et al. describe a set of strategies for achieving user involvement in ongoing computations.
- These strategies can be integrated in interactive visualization software and computational environments by following a set of requirements.

Strategies for achieving user involvement

- S1: data subsetting
- S2: complexity selection
- S3: divide and combine
- S4: dependent subdivision

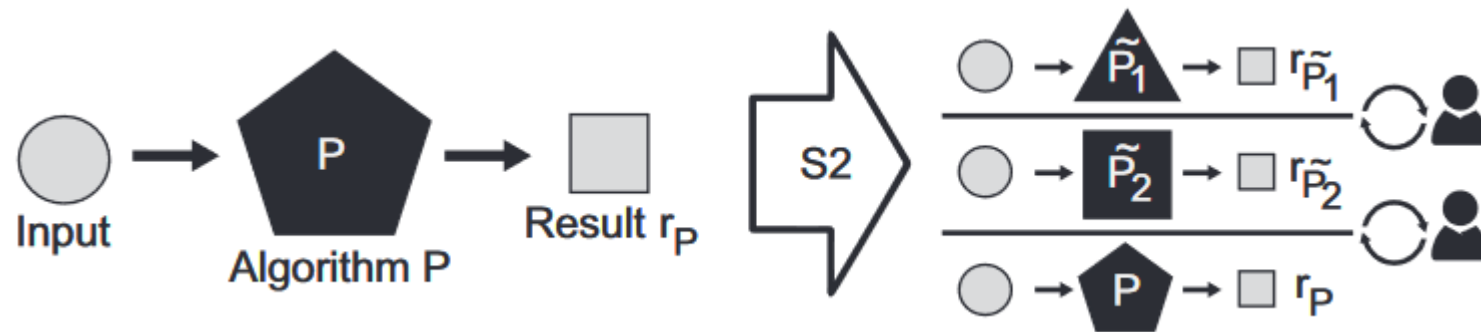
Strategies for achieving user involvement

- S1: data subsetting
 - Perform computations of P for increasingly larger subsets of data records or dimensions of a data table, and enable user involvement after completing every pass.



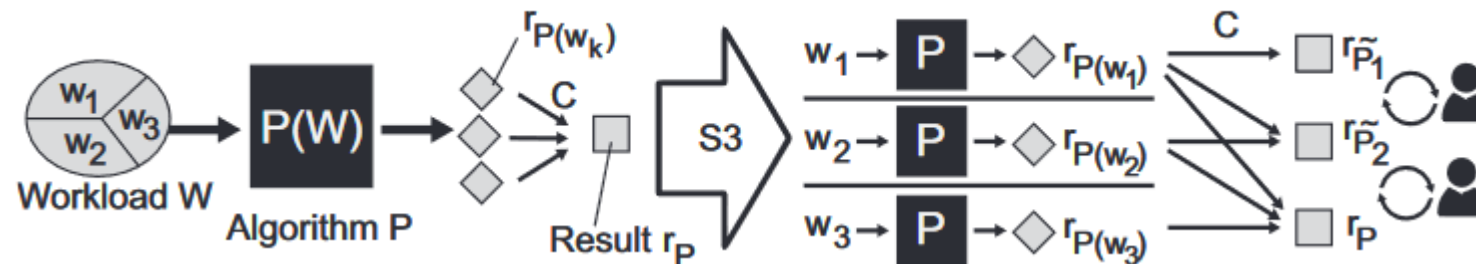
Strategies for achieving user involvement

- S2: complexity selection
 - Perform computation of P for less complex parameter configurations in additional passes before computing P itself.



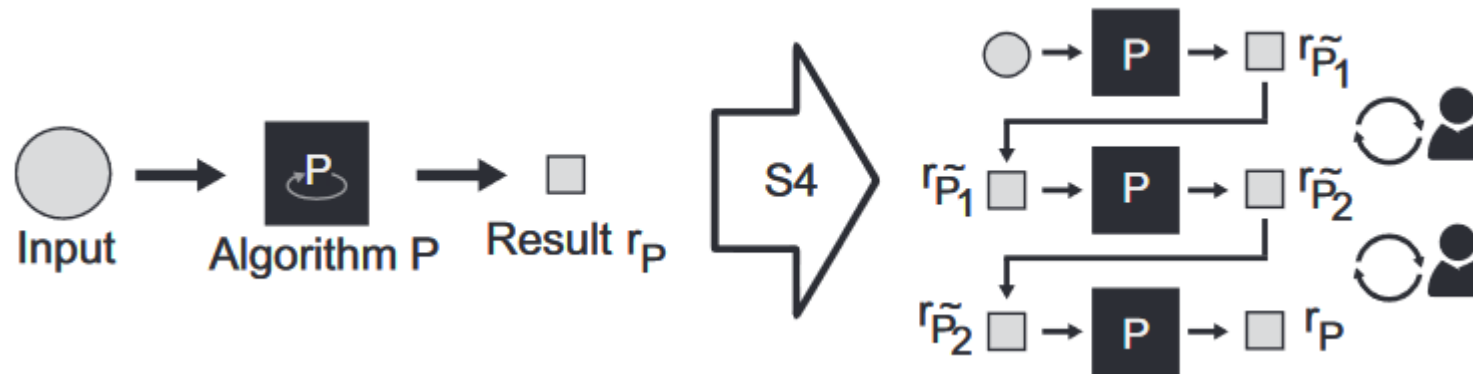
Strategies for achieving user involvement

- S3: divide and combine
 - Subdivide a workload into disjoint parts and apply P independently to each part to generate partial results.



Strategies for achieving user involvement

- S4: dependent subdivision
 - Subdivide P into sequentially dependent steps so that the result of each step is an input to the next step.



Requirements of progressive visual analytics

- Client-driven integration:
 - RC1: exposes complexity parameters (S2)
 - RC2: definable workload
 - RC3: state restorability
- Algorithm-driven integration:
 - GA1: communication granularity
 - GA2: allows callbacks

Requirements of progressive visual analytics

- RC2: definable workload
 - Enable a precise specification of the processed workload.
 - Explicit: number of iterations.
 - Implicit: stopping criterion.
 - Supports S3 (divide and combine) and S4 (dependent subdivision).
- RC3: provides access to all parts of the state as output and accept equivalent information as input.
 - Supports S4 (dependent subdivision).

Requirements of progressive visual analytics

- GA1: communication granularity
 - Specify preference and constraints by the client regarding desired feedback and control rates.
 - Specify level of verbosity.
- GA2: allows callbacks
 - Provide a callback interface to allow a client-side customization of the communication protocol.

Opening the black box

Algorithm (package)	Description	Operates on table	Exposes complexity param. (RC1)	Definable workload (RC2)	State restorability (RC3)	Provides communication from within	Comm. granularity (GA1)	Allows callbacks (GA2)
tsne (tense)	T-SNE dimension reduction	✓	✓	✓	✓	✓ (trace+GA2)	✓	✓
neuralnet (neuralnet)	Neural network	✓	✓	✓	✓	✓ (trace)	✓	✗
optim (stats)	Optimization	-	✓	✓	✓	✓ (trace)	✓	✗
sammon (MASS)	Multi-dimensional scaling	✓	✓	✓	✓	✓ (trace)	✗	✗
vegas (R2Cuba)	Monte Carlo Integration	-	✓	✓	✓	✓ (trace)	✗	✗
kmeans (cluster)	K-means clustering	✓	✓	✓	✓	✗	✗	✗
som (kohonen)	Self organizing map	✓	✓	✓	✓	✗	✗	✗
emcluster (EMCluster)	Expectation max. clustering	✓	✓	✓	✓	✗	✗	✗
rpart (rpart)	Recursive tree construction	✓	✓	✓	✓	✗	✗	✗
regsubsets (leaps)	Best subset feature selection	✓	✓	✓	✓	✗	✗	✗
biglm (biglm)	Linear model	✓	✓	✓	✓	✗	✗	✗
pam (cluster)	Partitioning around medoids	✓	✓	✗	✓	✓ (trace)	✗	✗
acf (stats)	Autocorrelation	✓	✓	✗	✗	✗	✗	✗
ksvm (kernlab)	Support vector machine	✓	✓	✗	✗	✗	✗	✗

Table 1. A survey of frequently used R algorithms regarding the fulfillment of the identified requirements and guidelines in favor of a tight integration.

[Muhlbacher et al., 2014]

Clustering

- Unsupervised learning, usually for large-scale data with no labels.
- Many uses:
 - Useful summary of a large dataset, representing N records using only $k \ll N$ clusters.
 - Exploratory data analysis, enabling us to understand the underlying sub-structure of the data.
 - We can improve performance of model-based prediction by learning separate models for each cluster.
 - We can use clustering to detect anomalies, by finding points that are far from any cluster center.
 - We can use clustering to handle massive streaming data by maintaining only the cluster summaries in memory.

Progressive analytics: K-means

- Each iteration: assignment each data point to a cluster.
- Analysts may inspect current state at any time, aware that clusters or distributions may change.

1. Initialize **cluster centroids** $\mu_1, \mu_2, \dots, \mu_k \in \mathbb{R}^n$ randomly.

2. Repeat until convergence: {

For every i , set

$$c^{(i)} := \arg \min_j \|x^{(i)} - \mu_j\|^2.$$

For each j , set

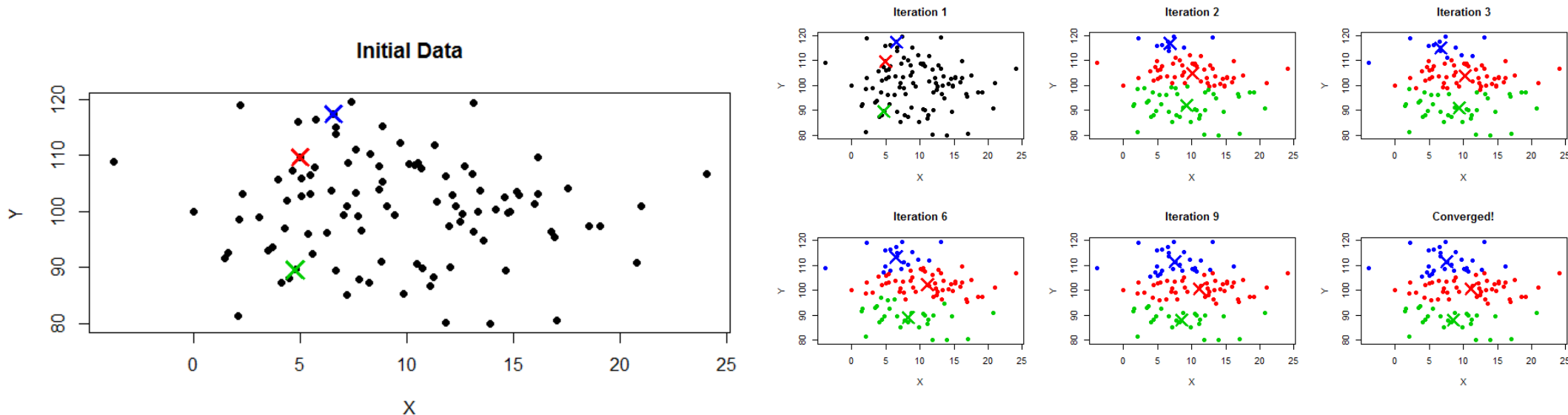
$$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$

}

[Chris Piech, 2013]

Progressive analytics: K-means

- Each iteration: assignment each data point to a cluster.
- Analysts may inspect current state at any time, aware that clusters or distributions may change.



Progressive analytics: K-means

- R1: exposes complexity parameters
- R2: definable workload
- R3: state restorability

Dimensionality reduction



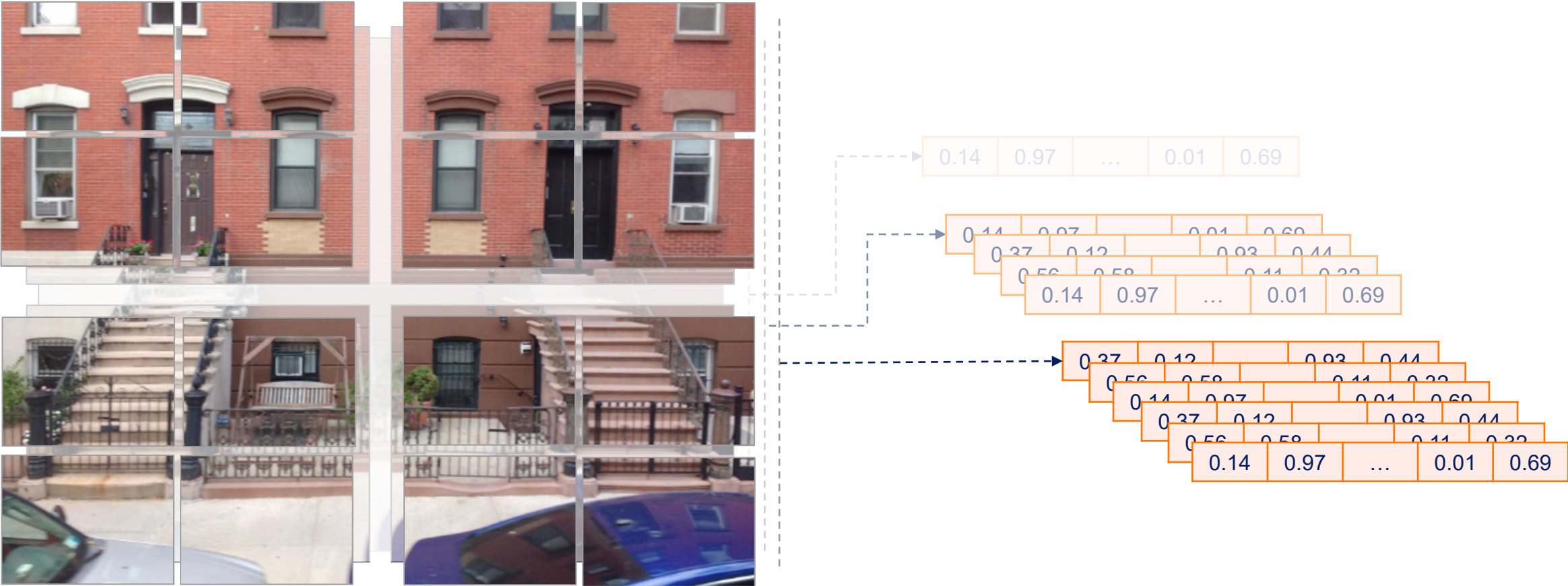
- Transformation from high-dimensional space into a low-dimensional space, maintaining some meaningful properties of original data.
- Many uses:
 - Fewer dimensions mean less computation.
 - Fewer dimensions, less storage space required.
 - Removes redundant features and noise.
 - Underlying structure of the data.
 - Useful for visualization.

tSNE

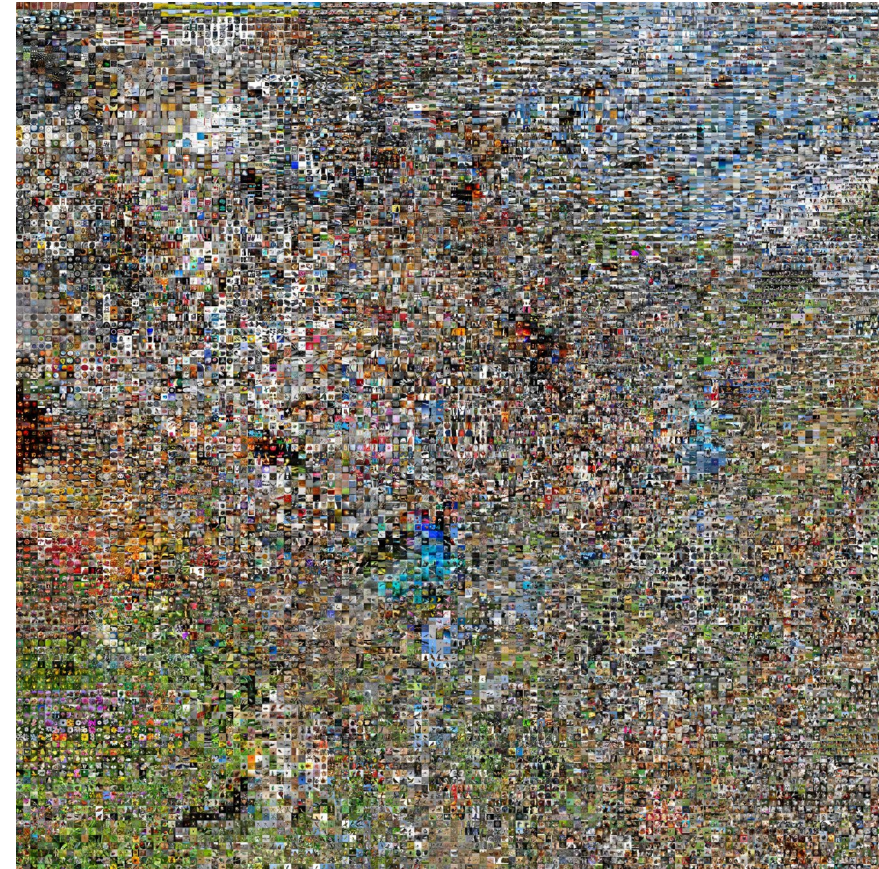
- Multidimensional projection method.
- Most real-world datasets are high dimensional.
 - High dimensional data vis is hard.
- Dimensionality reduction:
 - Input: points in nD .
 - Output: points in mD , with $m < n$.

Close points should remain close, far points should remain distant

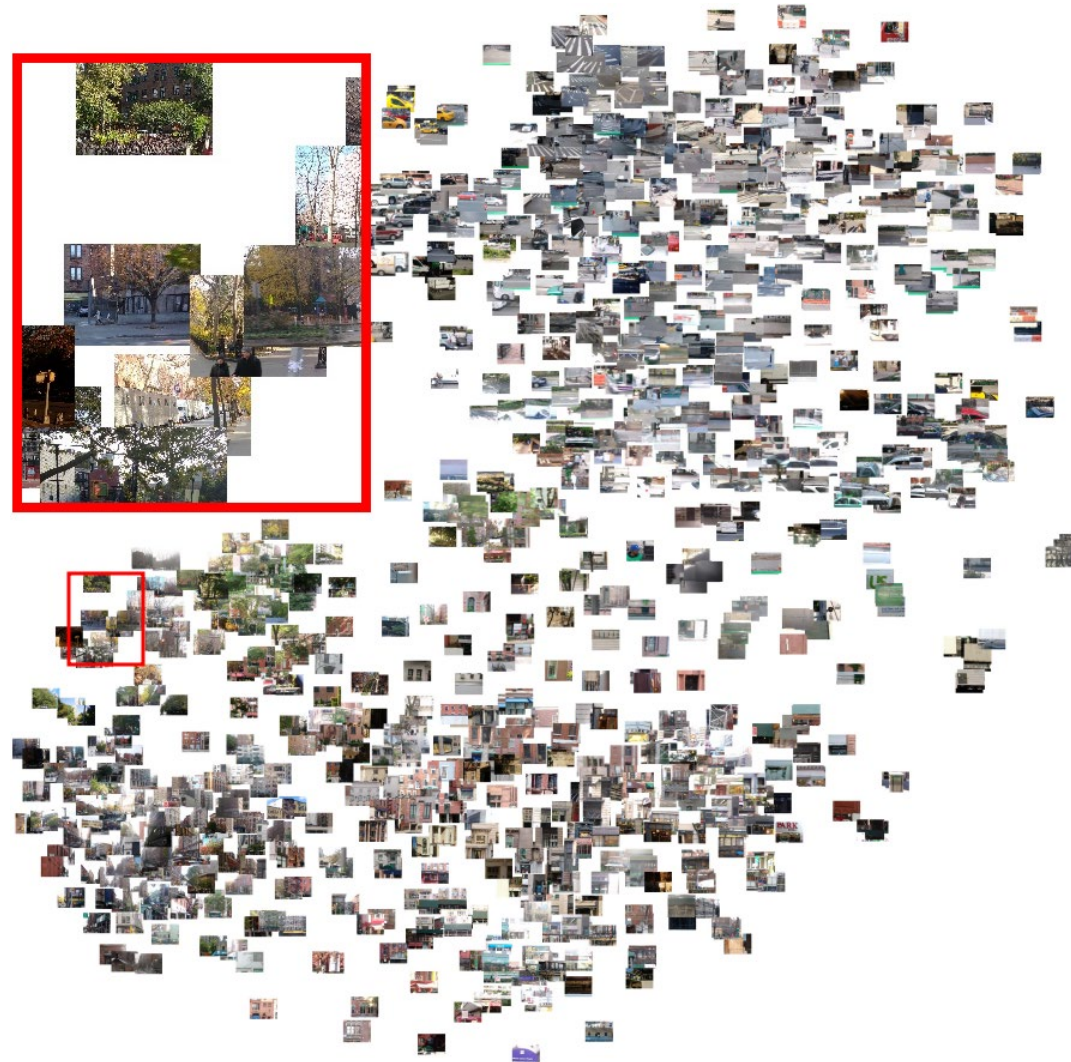
Image embeddings



tSNE



tSNE



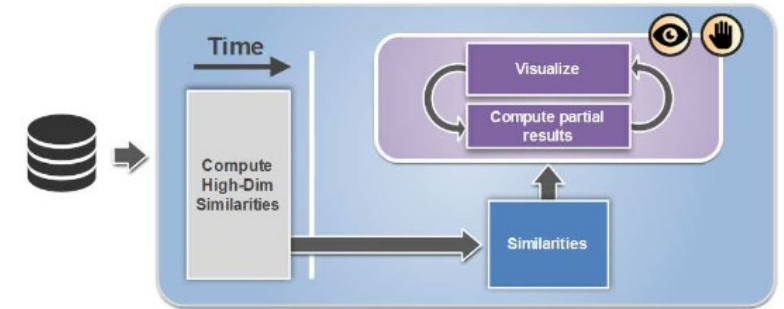
tSNE



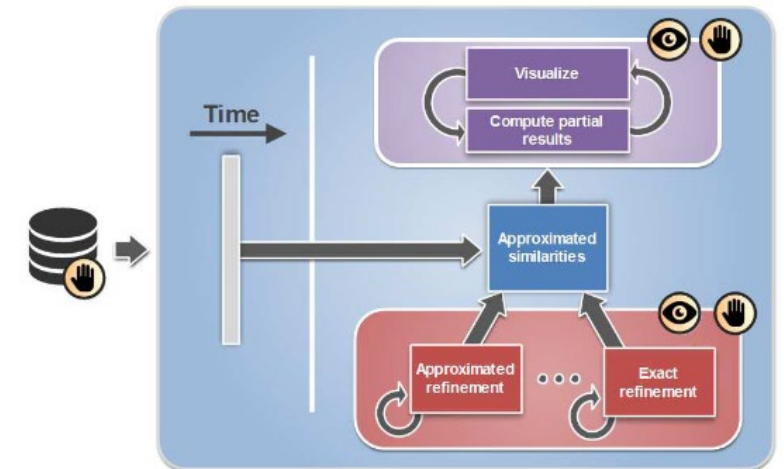
- tSNE: tool for dimensionality reduction of high-dimensional data.
- Main idea:
 - Similarity between data points in the high-dimensional space treated as distribution P .
 - Similarity between data points in the low-dimensional space Q .
 - Achieve a representation (embedding) in the low dimension where Q faithfully represents P .
- Steps:
 1. Computation of similarities ($O(n^2)$ or $O(n \log n)$ if using Barnes-Hut-SNE).
 2. Minimization of cost function (divergence between P and Q).

Progressive tSNE: A-tSNE

- tSNE is well suited for progressive visual analytics:
 - Intermediate results can be interpreted while they change over time.
- Shortcoming: computation of similarities (step 1) is slow and does not create meaningful intermediate results.
- A-tSNE introduces approximate similarities:
 - Compute similarities using approximated neighbors (approximated KNN)
 - Iterate to converge.



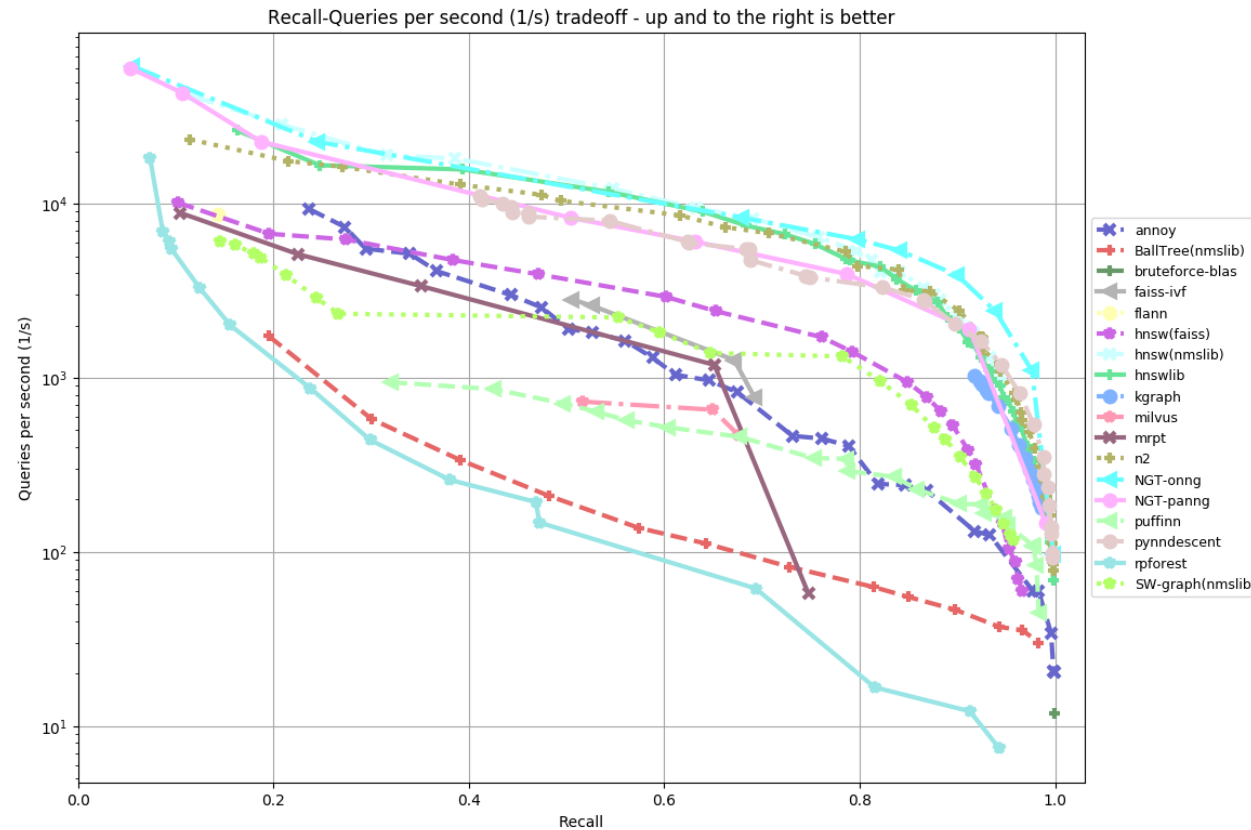
(a) Progressive Visual Analytics workflow for tSNE.



(b) Progressive Visual Analytics workflow for A-tSNE.

[Pezzotti et al., 2016]

Benchmarking nearest neighbors



<https://github.com/erikbern/ann-benchmarks/>

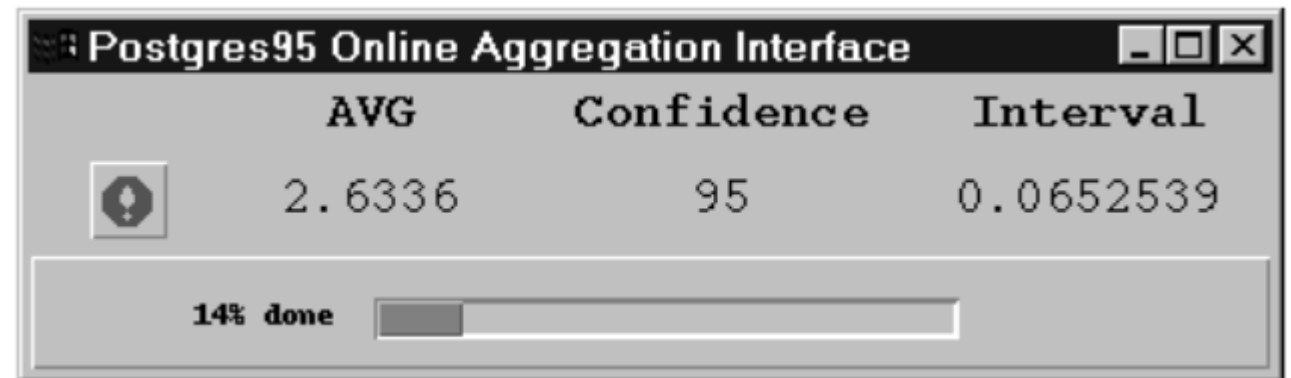
Progressive tSNE: A-tSNE

- R1: exposes complexity parameters
- R2: definable workload
- R3: state restorability

Progressive DB: aggregation

- Problems with aggregation:
 - Performed in batch mode; a query is submitted, the system processes a large volume of data over a long period of time, and final answer is returned.
 - User is forced to wait without feedback, while querying is being processed.
 - Aggregate queries are used to get a rough picture of a large volume of data, and yet computed with high precision.

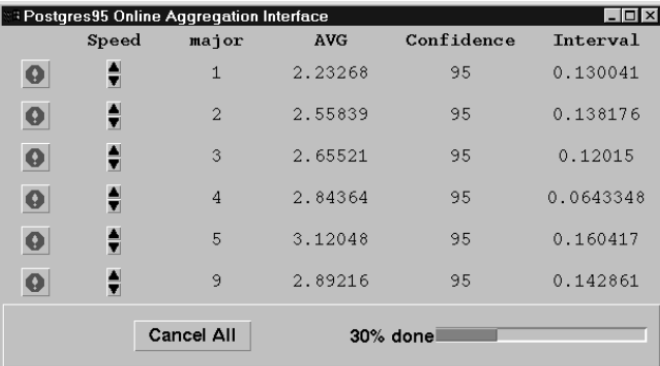
```
SELECT AVG(final_grade)
FROM grades
WHERE course_name = 'CS186';
```



[Hellerstein et al., 2014]

Online aggregation

- Permits users to observe the progress of aggregation.
- Goals:
 - Minimize time required to produce useful estimate of the final answer.
 - Aggregates should be updated at regular rates.
 - Fairness / partiality: users should control relative rate at which different running aggregates are updated.



The screenshot shows a window titled "Postgres95 Online Aggregation Interface". It contains a table with the following data:

	Speed	major	AVG	Confidence	Interval
ⓘ	▲▼	1	2.23268	95	0.130041
ⓘ	▲▼	2	2.55839	95	0.138176
ⓘ	▲▼	3	2.65521	95	0.12015
ⓘ	▲▼	4	2.84364	95	0.0643348
ⓘ	▲▼	5	3.12048	95	0.160417
ⓘ	▲▼	9	2.89216	95	0.142861

At the bottom of the window, there is a "Cancel All" button and a progress bar labeled "30% done".

Online aggregation

- Two important considerations:
 - In what order to access the data?
 - How to compute confidence interval?

Online aggregation: access order

- Statistically meaningful estimates are available only if records are retrieved in random order.
- Concretely: avoid access methods in which the attribute values of a tuple affect the order.
 - Index scan
 - Sampling from indices
 - Heap scan

Online aggregation: confidence interval

- Naïve solution: simply sum partial results.
- Deviation of the average of random numbers sampled from a set: Hoeffding's inequality or central limit theorem.

```
running_avg(float current)
{
    static int count, sum;
    sum+=current;
    count++;
    return (sum/count);
}
```

```
running_interval(float current, Column c)
{
    static int count;
    static int upper = c.max;
    static int lower = c.min;
    count++;
    return ((1.36*(upper - lower)) / sqrt(count));
}
```

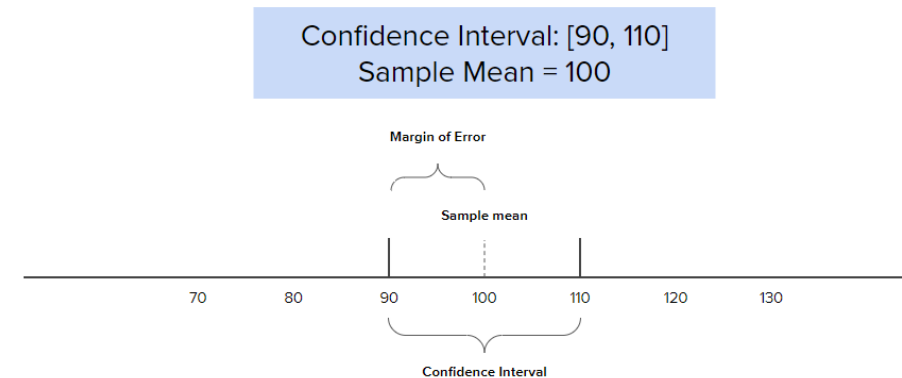
Central limit theorem

- Distribution of the sum of n independent random variables X is approximately normal when n is large:
$$X_1 + X_2 + \dots + X_n = S_n \sim N(\mu, \sigma^2)$$
- Given a population with a finite mean μ and variance σ^2 , the sampling distribution of the mean approaches a normal distribution $N(\mu, \sigma^2/N)$ as N (sampling size) increases.

Sampling distribution of the mean is the distribution of the means of samples of the dataset

Central limit theorem

- Mean is 100.
- 95% confidence that the sample mean lies between 90 and 110.



Central limit theorem

- This means we can estimate the mean, and std. deviation:

$$\bar{x} = \text{mean}(\text{sample})$$

$$\sigma = \frac{\text{stddev}(\text{sample})}{\sqrt{N}}$$

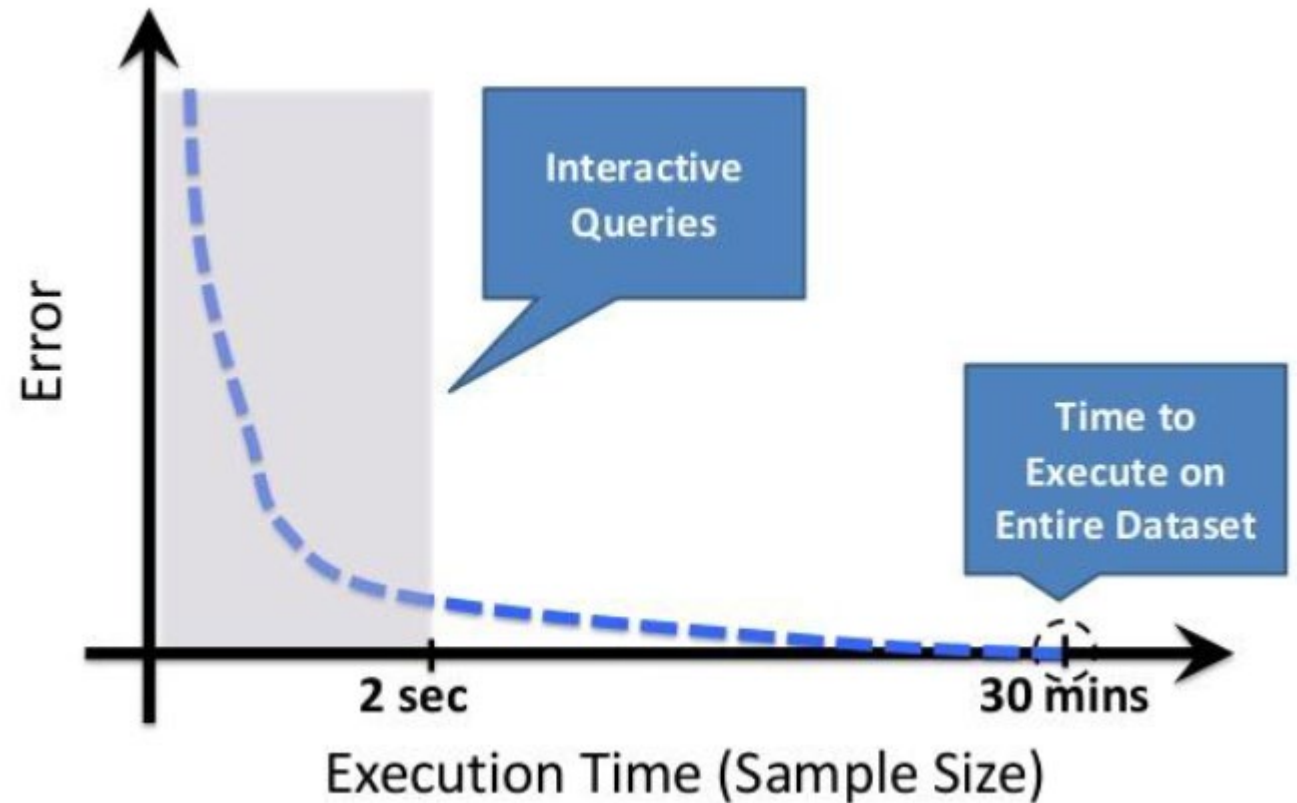
~ 96% confidence interval: $[\bar{x} - 2\sigma, \bar{x} + 2\sigma]$

Online aggregation

- Applications:
 - Large-scale data analysis
 - Data mining
 - Visualization
 - ...

Approximate computing

- Key observation: applications can tolerate quick, approximate answers over data, as they are often exploration queries.
- Trade-off: small error for up to orders of magnitude in efficiency.



Far from easy



- Hadoop:
 - Processing 10TB on 100 machines can take an hour.
- In-memory computing:
 - Processing 10TB on 100 machines will take minutes.

Existing solutions



BlinkDB

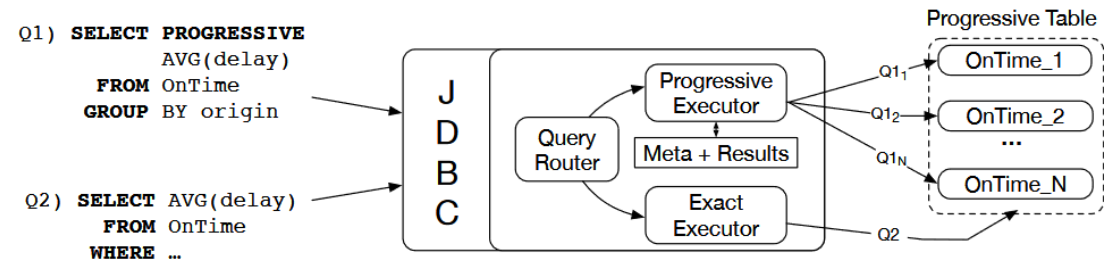
- Allows users to trade off accuracy for response time and provide users with meaningful bounds on accuracy.

```
SELECT COUNT(*)  
FROM Sessions  
WHERE Genre = 'western'  
GROUP BY OS  
ERROR WITHIN 10% AT CONFIDENCE 95%
```

```
SELECT COUNT(*)  
FROM Sessions  
WHERE Genre = 'western'  
GROUP BY OS  
WITHIN 5 SECONDS
```

Progressive DB

- Makes use of online aggregation to provide interactive response times for arbitrary aggregate queries.
- Partitions tables into chunks holding data such that each partition is small enough to be queried by an analytical query within a given query latency.
- Combined result of all sub-queries from decomposable aggregation functions (sum, count, average).



[Berg et al., 2018]

Progressive DB

