SEEDB: Efficient Data-Driven Visualization Recommendations to Support Visual Analytics

Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, Neoklis Polyzotis

Motivating Example

Visualization: Either to get a sense of data, or to report the results

- In this work, we focus on the first one

Census data: age, education, marital-status, sex, race, income, hours-worked, etc.

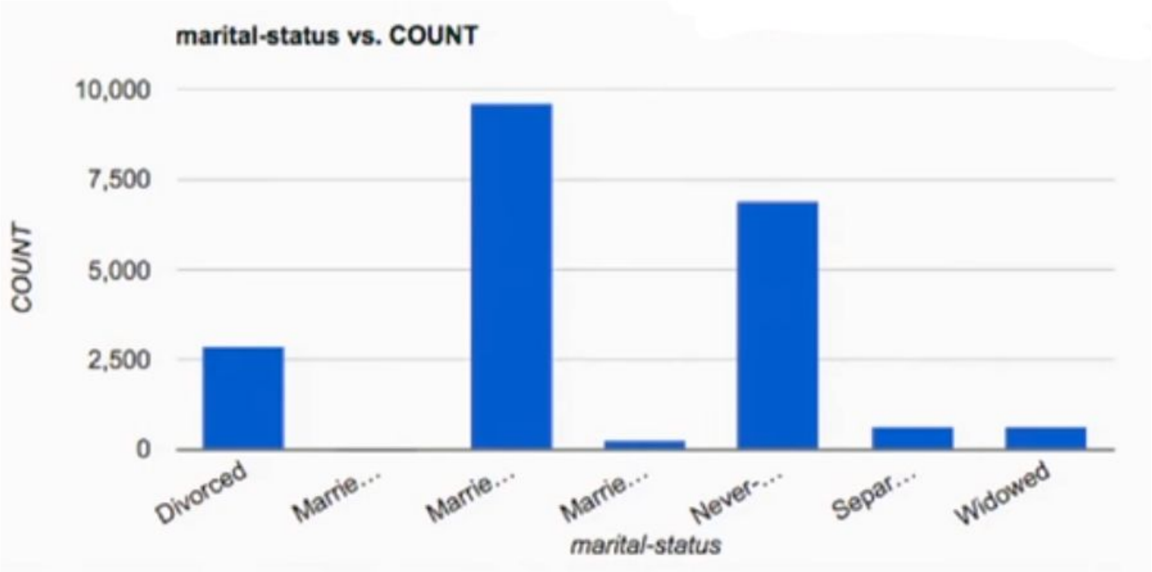
- A: #attributes in the table

Task: Socioeconomic statuses of adults who have *never-been-married*

Visualizing Census Data

Histograms: O(A)

Multi-attribute: $O(2^A)$



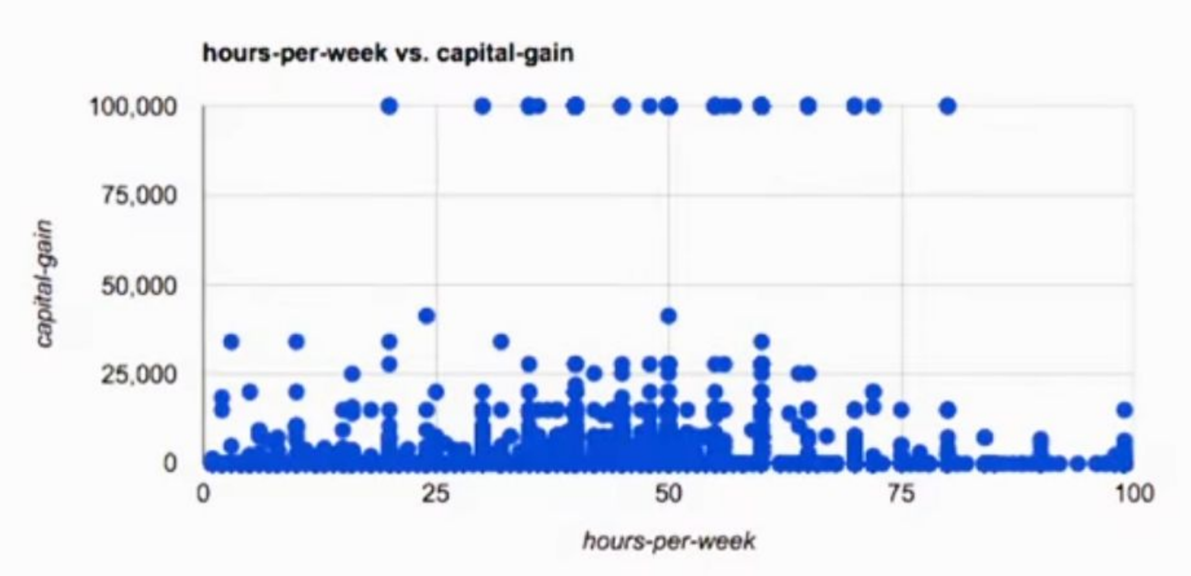
Visualizing Census Data

Pairwise Scatterplot: $O(A^2)$

Efficient at

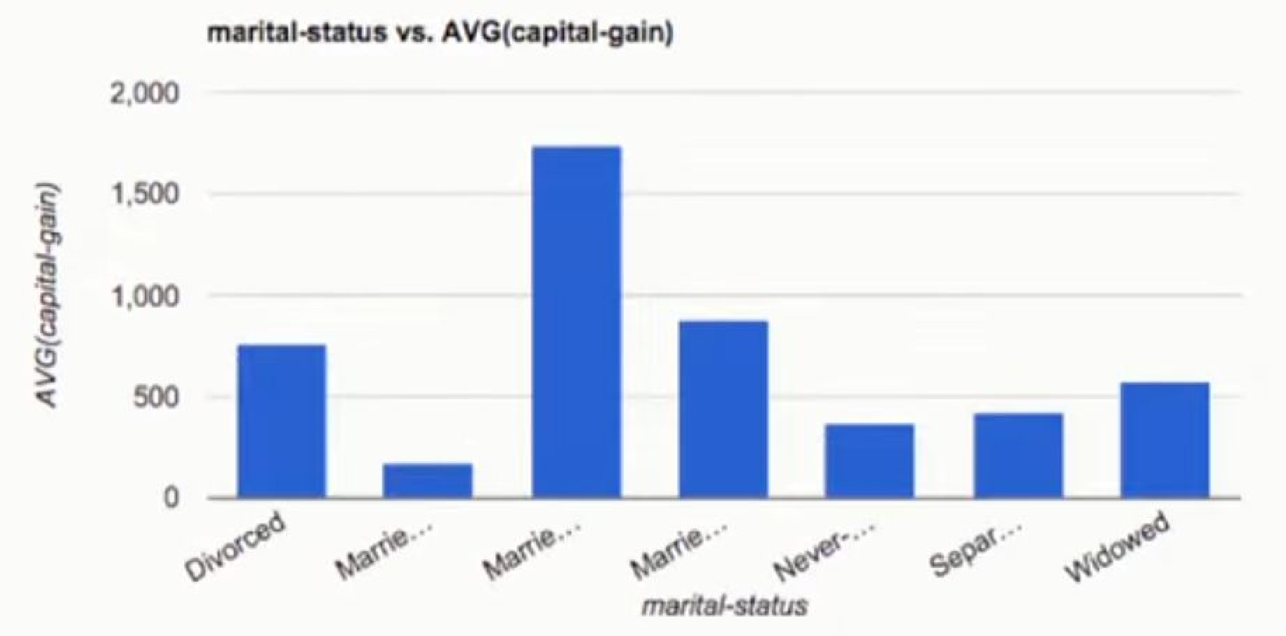
Showing trends

And correlations



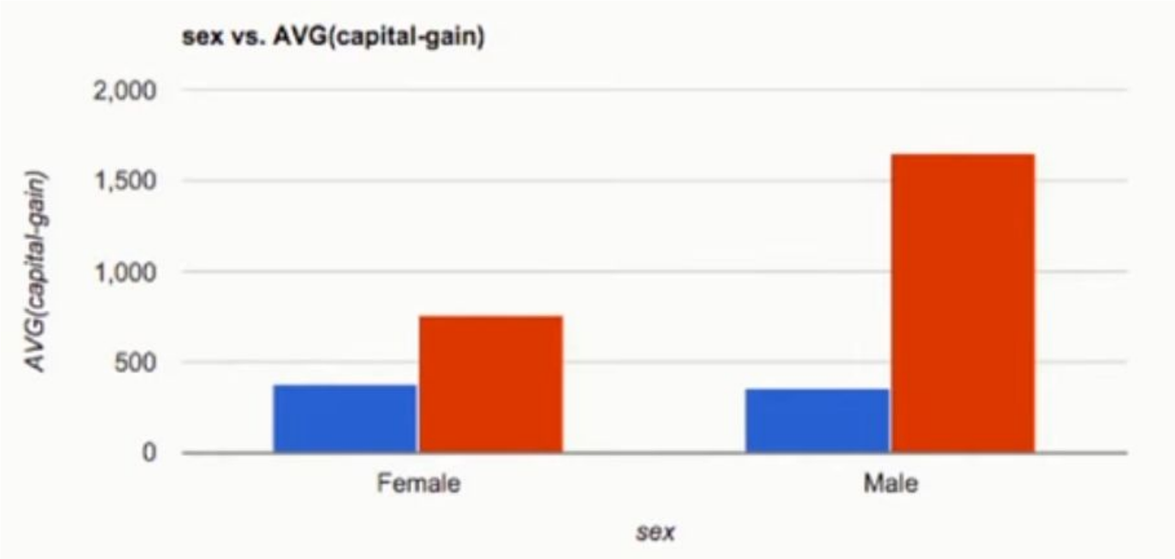
Visualizing Census Data

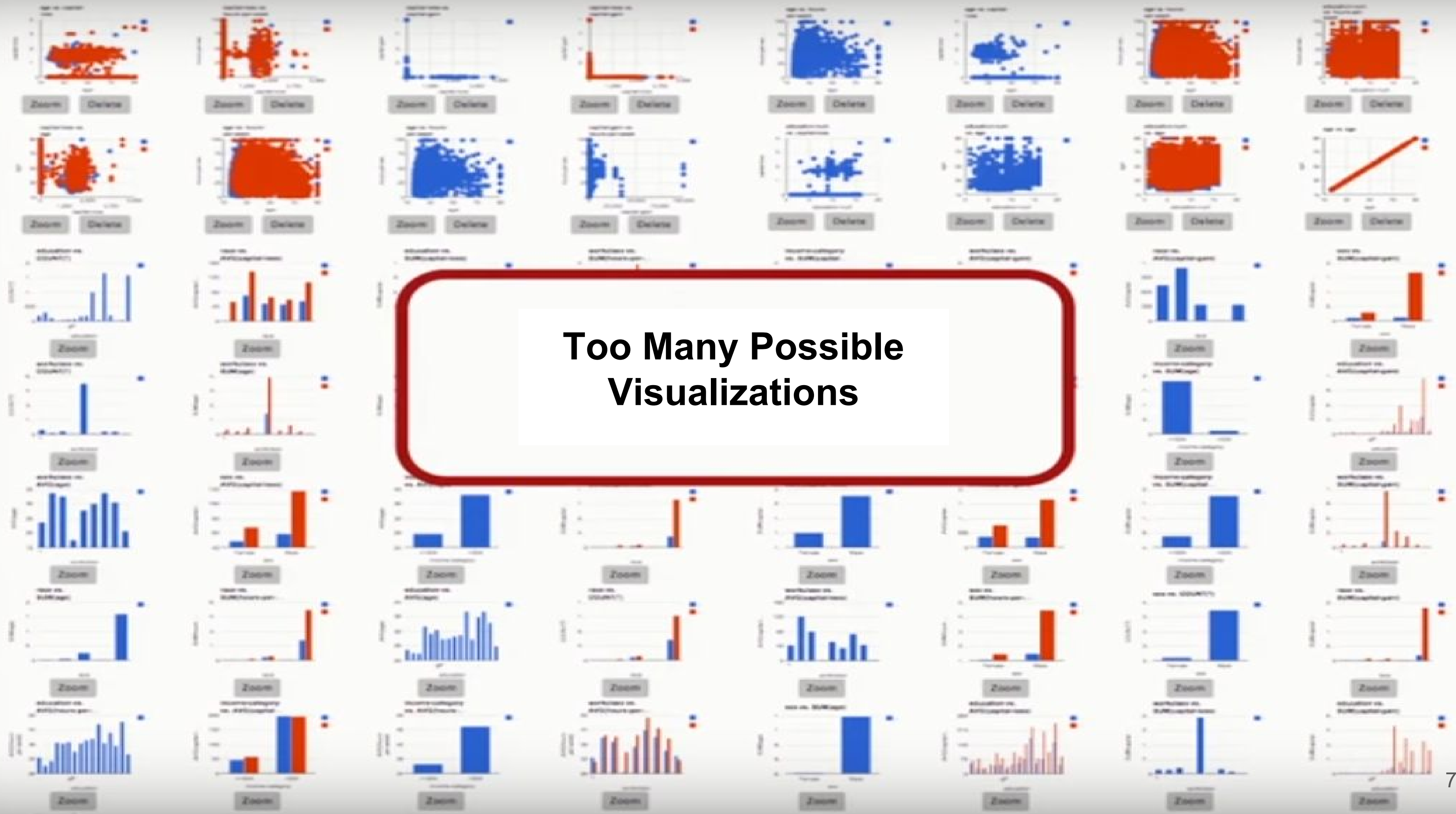
Aggregate view $O(A^2)$



Visualizing Census Data

Comparative visualizations: $O(A^2)$

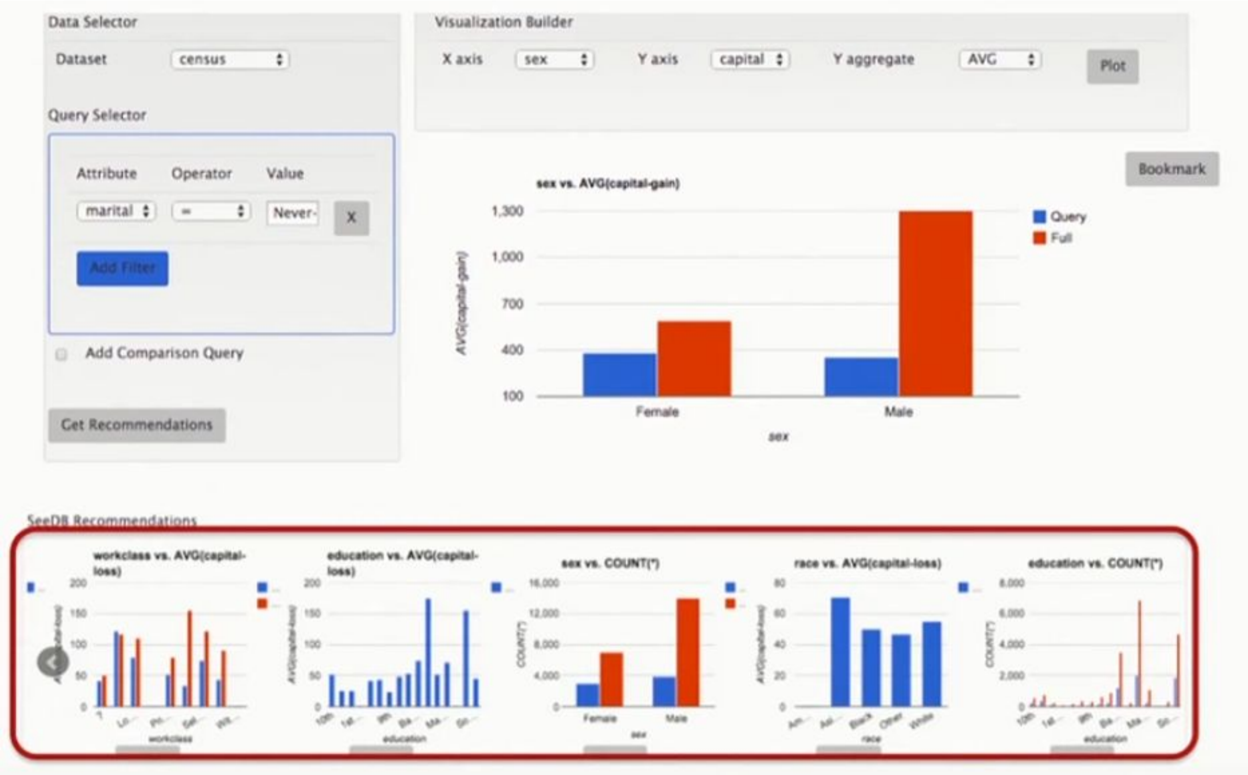




Too Many Possible Visualizations


Zoom
Delete

SeeDB: A Visualization Recommender



Recommending Visualizations

I. How to find relevant visualizations?

Interestingness or utility metric

II. How to make recommendations efficiently?

- Scale to large number of rows
- Curse of dimensionality
- Interactive time scales

How to find relevant visualizations?

Visualization Utility

Utility depends on data (distribution), query, metadata, context, user preferences, aesthetics

U = f(data, query, metadata)

SeeDB Visualizations

Bar charts / Aggregate Visualizations

- Aggregates essential for large datasets
- Large fraction of common visualizations

Scatterplots in the near future

SeeDB Visualizations

- V_i = (d: dimension, m: measure, f: aggregate)

  X-axis Y-axis

- AGGREGATE + GROUP BY queries

SELECT d, f(m) FROM table GROUP BY d WHERE selection_predicate

> “The greatest value of a picture is when it forces us to notice what we never expected to see.”
>
> Tukey, Exploratory Data Analysis 1977

What is unexpected (different from expected trends) is interesting

Deviation-based Utility Metric

Q: SELECT * from WHERE marital-status = 'Never-married'

Table: D, Data selected by Q: Q_D

{d}: race, work-type, sex, etc.

{m}: capital-gain, capital-loss, hours-per-week

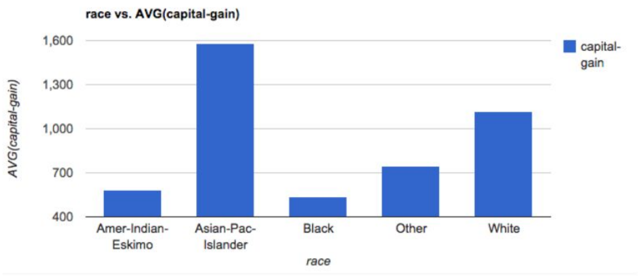
{f}: COUNT, SUM, AVG

Computing Expected Trend vs. Actual Trend

V_i: Race vs. AVG(capital-gain)

SELECT race, AVG(capital-gain) FROM census GROUP BY race

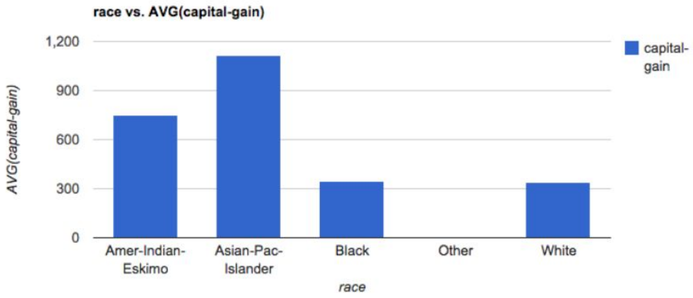
P[V_i(D)] Expected Distribution



Vi: Race vs. AVG(capital-gain)

SELECT race, AVG(capital-gain) FROM census GROUP BY race WHERE marital-status = 'Never-married'

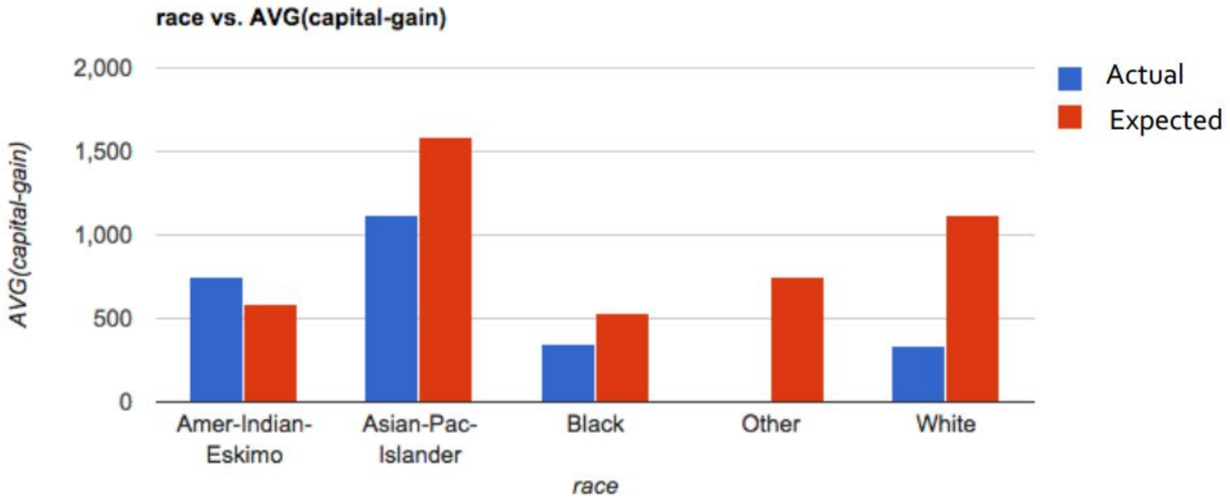
P[Vi(D)] Actual Distribution



Computing Utility

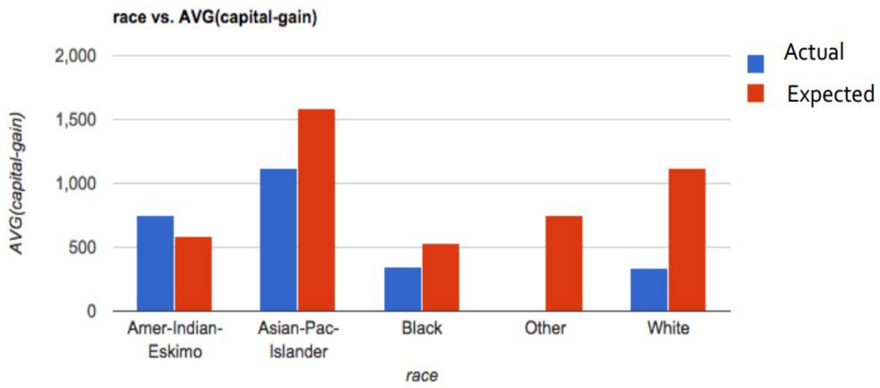
Vi: $P[V_i(d)]$ (expected), $P[V_i(D_Q)]$ (actual)

$$U(V_i) = \Delta(P[V_i(QD)] , P[V_i(D)])$$

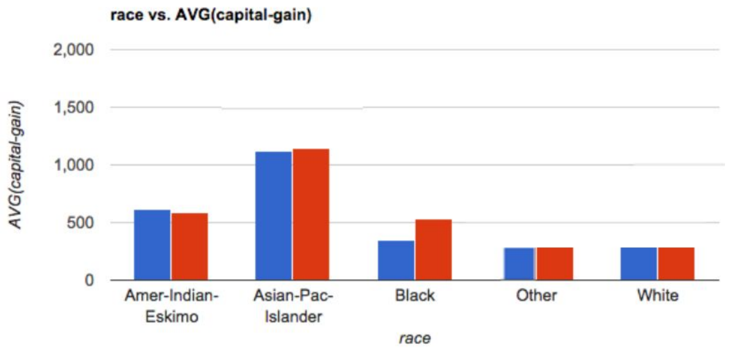


Utility Visualization

High Utility Visualization



Low Utility Visualization



How to make recommendations efficiently?

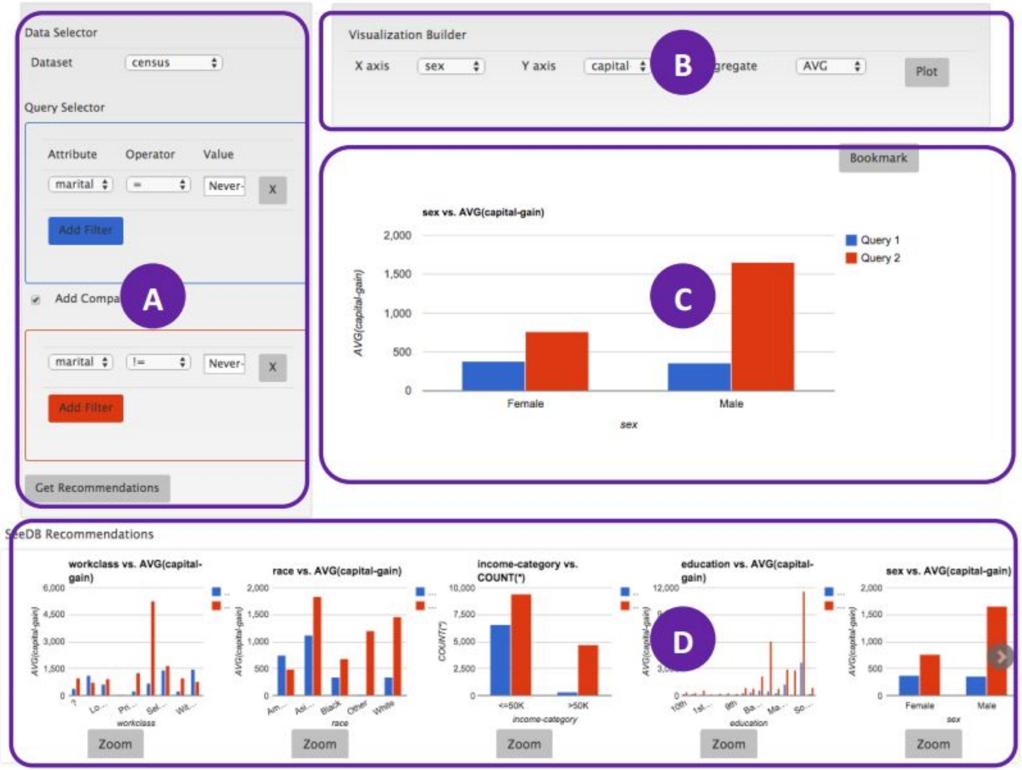
SeeDB FRONT-END

(A) dataset selector

(B) visualization builder

(C) visualization display pane

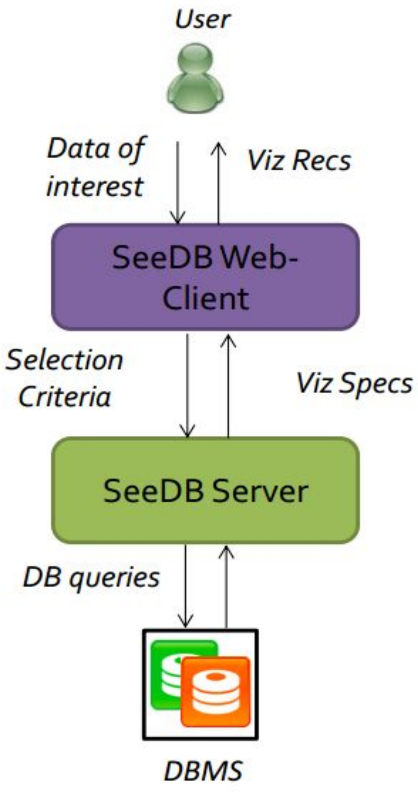
(D) a recommendations plugin



SeeDB Architecture

Middleware on top of DB

Data processing delegated to DBMS



Challenges

D = # dimensions, M = # measures, F = # aggregate functions

D * M * F potential visualizations

- 2 * D * M * F queries to the DBMS
- Each query (potentially) scans full dataset
- Computation wasted on low-utility views

How do we make recommendations efficiently?

Goals:

- Interactive latencies
- No wasted resources on low-utility views

Strategies:

1. Run-time pruning framework
2. Systems-level optimizations

Run-time Pruning Framework

- Identify views with low-utility early, weed out
- Running estimates of utility based on samples

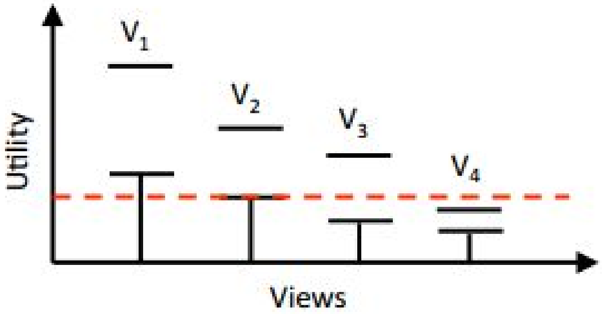
Techniques:

- Confidence Interval-based Pruning
- Multi-Armed Bandit Pruning

Confidence Interval

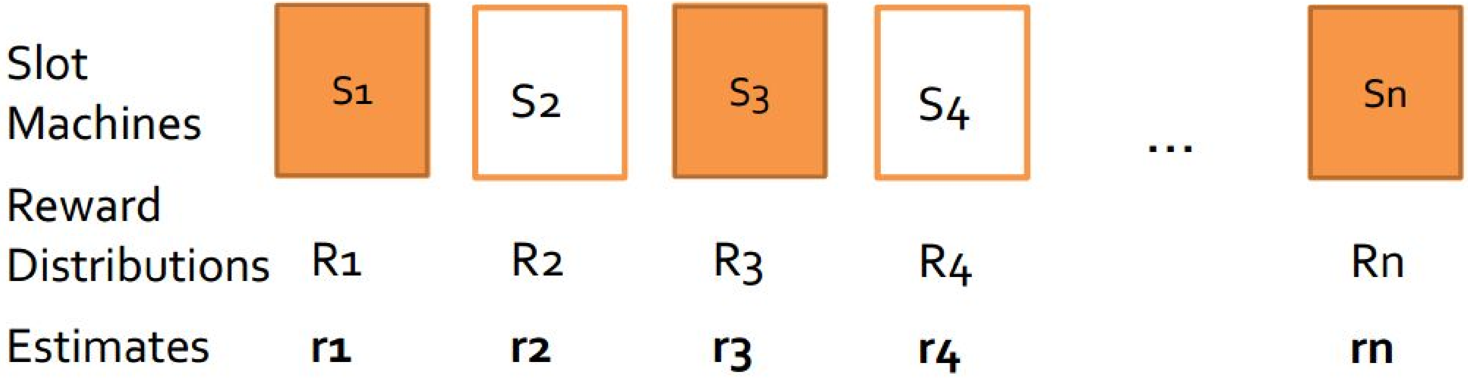
Confidence intervals on the estimates of utility based on different samplings

Throw away visualizations based on these intervals



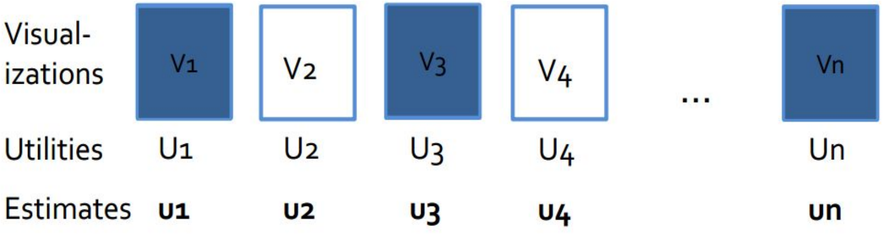
Multi-Armed Bandit

Which machines to play and in what order to maximize reward?



Multi-Armed Bandit Pruning

Rank views by utility



Compute various Δ_i

Δ_1 : the difference between the highest mean and the k + 1st highest mean

Δ_n : the difference between the lowest mean and the kth highest mean

Successive Accepts and Rejects algorithm:

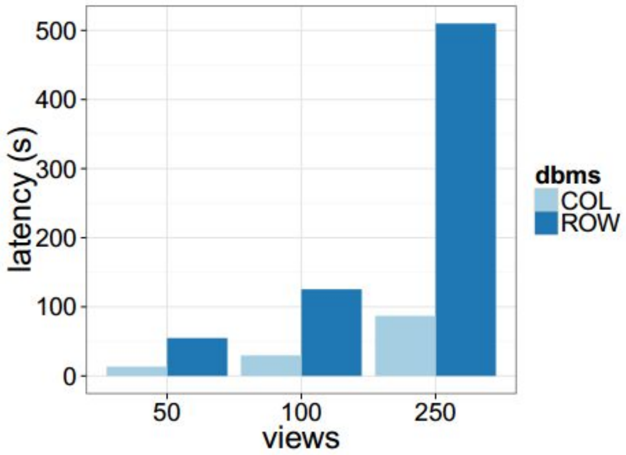
- If $\Delta 1$ is greater than Δn, the view with the highest mean is “accepted”
- If Δn is greater, the view with the lowest mean is discarded

Systems-level optimizations

Each visualization = 2 SQL queries

Latency > 100s

Minimize number of queries and scans



Systems-level optimizations

- Combine aggregate queries on Q_D and D
- Combine multiple aggregates

$(d_1, m_1, f_1), (d_1, m_2, f_1) \longrightarrow (d_1, [m_1, m_2], f_1)$

- Combine multiple group-bys

$([d_1, m_1, f_1), (d_2, m_1, f_1) \longrightarrow ([d_1, d_2], m_1, f_1)$

- Parallel Query Execution

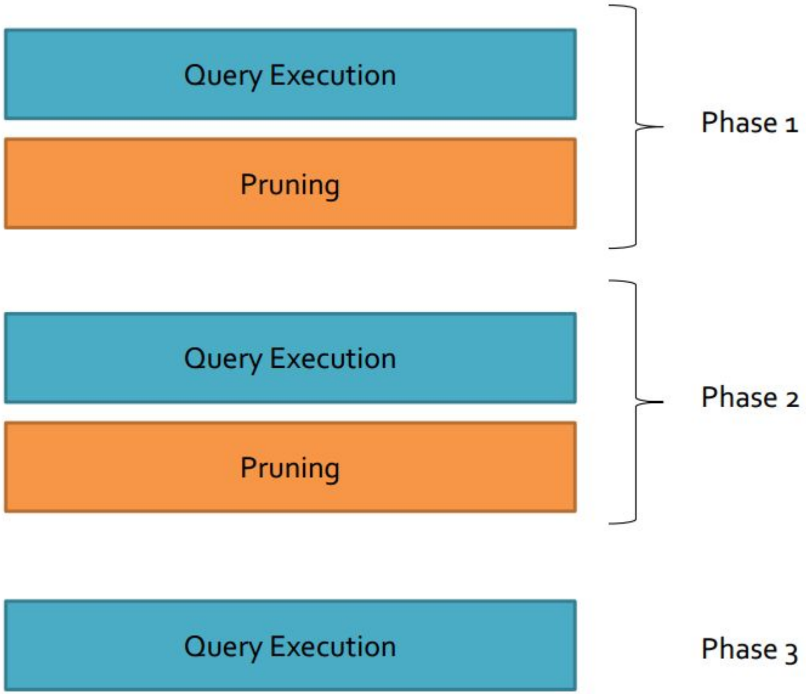
Combining Multiple Group-bys

- Too few group-bys leads to many table scans

- Too many group-bys hurt performance
- \# groups = Π (# distinct values per attributes)

- Optimal group-by combination ≈ bin-packing
- Bin volume = log S (max number of groups)
- Volume of items (attributes) = log (|ai|)
- Minimize # bins s.t. Σi log (|ai|) <= log S

Interleaving Optimizations



Evaluation

Performance Study

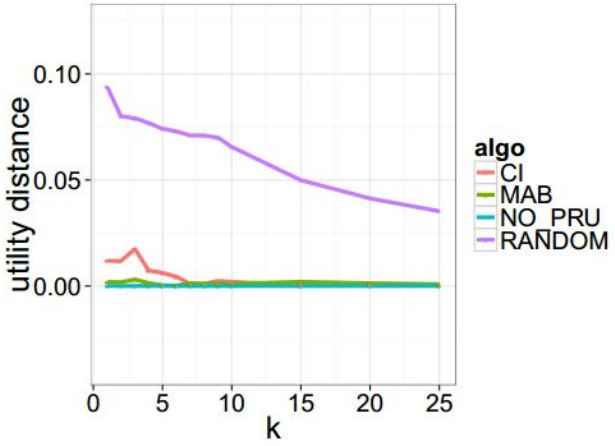
- Latency, accuracy
- Variety of synthetic and real datasets

User Study

- Controlled Study
- Trends, Interactions, Surveys

Pruning Optimizations

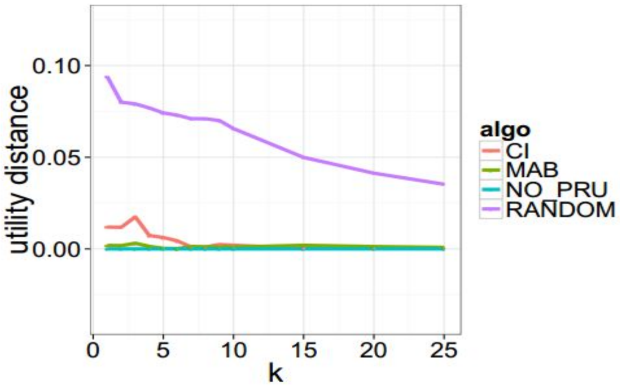
Utility Distance = Δ (Utility of real top-k, Utility of SeeDB top-k)



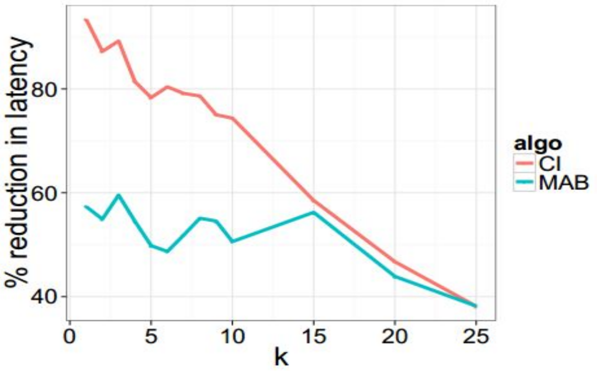
Utility Distance for BANK

Pruning Optimizations

Pruning can reduce latency by 50 - 90% w/o significant hit in accuracy



Utility Distance for BANK

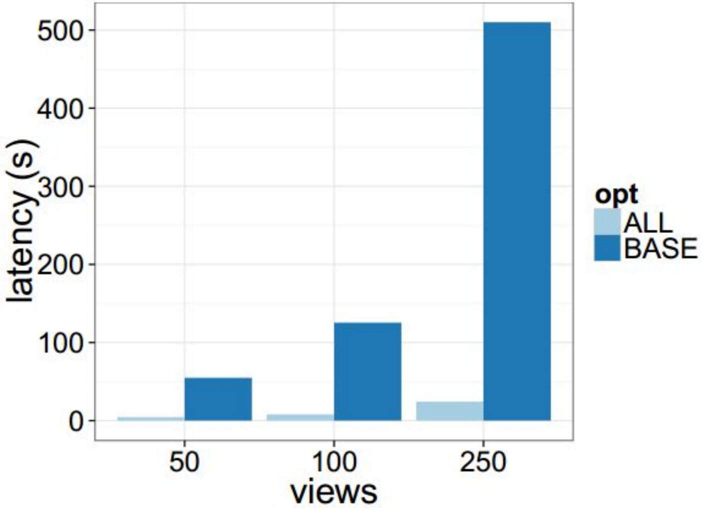


Latency Reduction for BANK

Systems-level Optimizations

Combination of systems optimizations reduce

latency 25X to ~ 10s



Putting it together

SeeDB returns results in < 4s for small and medium-sized data

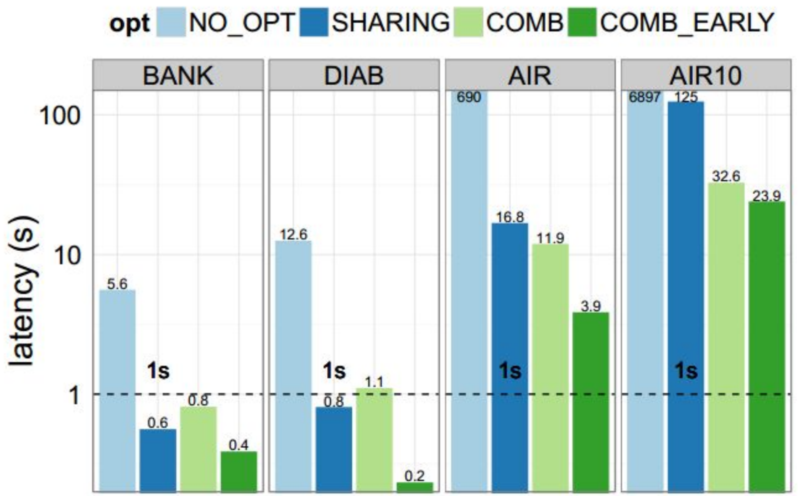
Customer Loan dataset: 40K

Hospital data about diabetic patients: 100K

Airline delays dataset: 6M

Airline dataset scaled 10X: 60M

Caching, indexing, etc. in future work



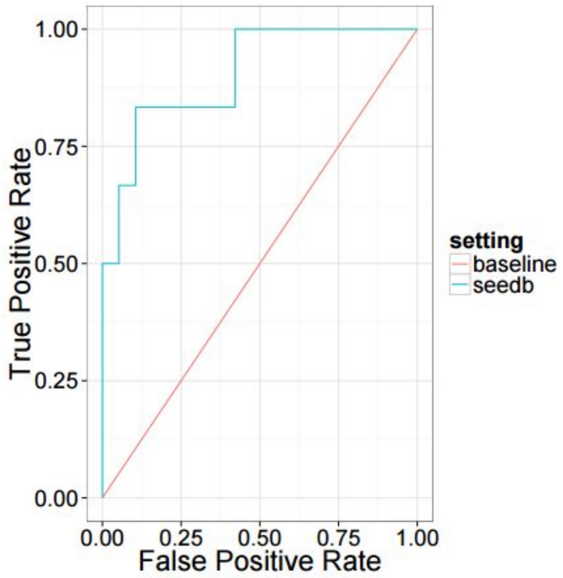
User Study

Validating Deviation-based Utility

5 data analysis experts with the Census dataset

Task: studying the effect of marital status on socioeconomic indicators

Classify each visualization as interesting or not interesting *in the context of the task*



User Study

SEEDB vs. Manual Visualization Tool

(i) Find interesting visualizations faster
(ii) Find more interesting visualizations
(iii) Prefer using SEEDB to a manual tool

16 participants (5 female, 11 male), all graduate students with prior data analysis experience and visualization experience

Housing and Movies datasets

User Study

They were asked to

- use the bookmark button to flag any visualizations they deemed interesting in context of the task
- think aloud during the study
- fill a tool-specific survey

Analytical tasks were open-ended

Capped each analysis session at 8 minutes

User Study

Total number of aggregate visualizations created in the SEEDB condition is higher

The bookmark rate for SEEDB (0.42) is 3X larger

	total_viz	num_bookmarks	bookmark_rate
MANUAL	6.3 ± 3.8	1.1 ± 1.45	0.14 ± 0.16
SEEDB	10.8 ± 4.41	3.5 ± 1.35	0.43 ± 0.23

Prefer tool with recommendations: 100%

Recommendations Rating: 79% (>= helpful)

- Deviation-based metric performs well

Summary

- Visualization recommender for analytics
- Deviation-based utility to capture relevance
- Run-time pruning can provide a 4X speedup and systems-level optimizations 25X
- Users prefer a tool with recommendations
- Ongoing work: incorporate other relevant metrics (other distance metrics, context etc.)

Thanks!

Any Questions?