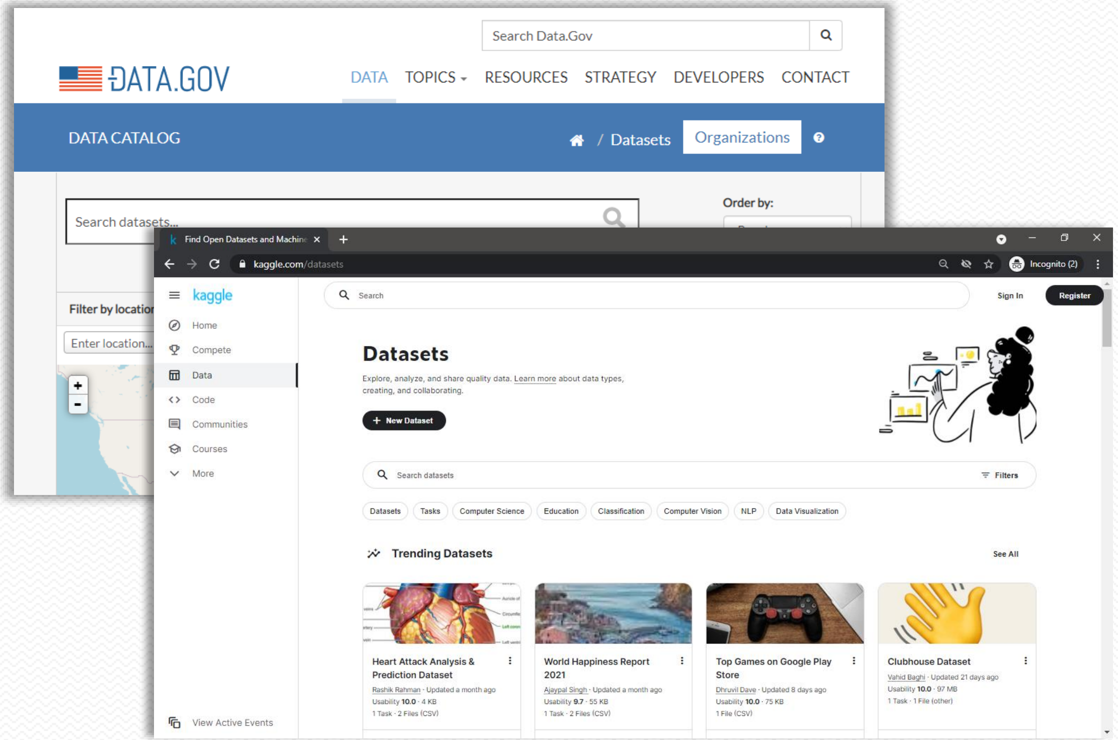
Correlation Sketches for Approximate Join-Correlation Queries

DSIT
Database Systems

Maria Kostopoulou

Relational Data Augmentation

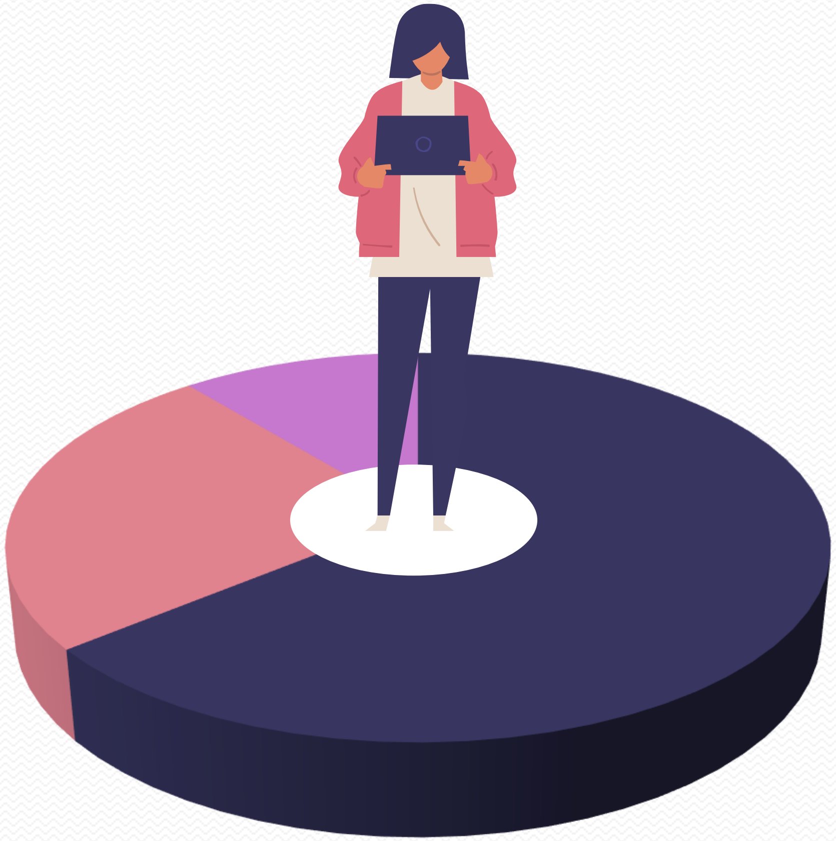


Growing availability nowadays of structured datasets (Web tables and open-data portals to enterprise data)

Relational Data Augmentation

- ✓ Enhance data analytics
- ✓ Improve performance of ML models

Dataset Discovery - Challenges



Datasets spread over many repositories

To find relevant datasets must:
1) Look for reasonable variables
2) Check joinability with target dataset
3) Confirm new columns are correlated with targeted ones
4) Build new model to confirm performance improvement

Searching for relevant datasets it's not an easy task!

Join-Correlation

$\mathcal{T}_X$

K_X	X
2021-01	6.0
2021-02	4.0
2021-03	2.0
2021-04	3.0
2021-05	0.5
2021-06	4.0
2021-07	2.0

$\mathcal{T}_Y$

K_Y	Y
2021-01	5.5
2021-01	4.5
2021-02	3.9
2021-02	2.0
2021-03	4.0
2021-03	1.0
2021-04	4.0

$\mathcal{T}_{X \bowtie Y}$

$K_{X \bowtie Y}$	$X_{X \bowtie Y}$	$Y_{X \bowtie Y}$
2021-04	3.0	4.0
2021-03	2.0	2.5
2021-02	4.0	3.0
2021-01	6.0	5.0

Correlation of columns X and Y

after applying join/aggregation on Tx and Ty tables

Challenges

- Join, aggregation and correlations :

expensive + impossible to compute on query-time

- Cannot pre-compute them
- Cannot sample without applying join

  (Values need to aligned using join key)

What are the alternatives for efficient query evaluation over large dataset collections?

Correlation Sketches

$L_{\langle K_X, X\rangle}$

$h(k)$	$h_u(k)$	x_k
bac52e98	0.48	2.0
16dab449	0.34	2.0
26f79756	0.47	3.0
4da33cf5	0.34	6.0

$L_{\langle K_Y, Y\rangle}$

$h(k)$	$h_u(k)$	y_k
16dab449	0.34	2.5
bd5a7c1f	0.89	3.0
26f79756	0.47	4.0
4da33cf5	0.34	5.0

$L_{\langle X \bowtie Y\rangle}$

$h(k)$	x_k	y_k
16dab449	2.0	2.5
26f79756	3.0	4.0
4da33cf5	6.0	5.0

Instead of using full datasets → use **Correlation Sketches**

A fast join-correlation estimation at query time!

Build data synopses $L\langle K_X, X\rangle$ and $L\langle K_Y, Y\rangle$ of $\langle K_X, X\rangle$ and $\langle K_Y, Y\rangle$

2 different hashing functions

h(x) :

- ✓collision-free hash function
- ✓randomly and uniformly maps key values $k \in K_X$ into distinct integers → tuple identifiers in $L\langle K_X, X\rangle$
- ✓32-bits MurmurHash3 function

$\boldsymbol{h_u(x)}$:

- ✓maps integers $h(k)$ to real numbers in the range [0, 1], uniformly at random
- ✓**the tuples corresponding to the n smallest h_u values are the ones included in the sketch**
- ✓introduces dependence that increases the probability of $L\langle K_X, X\rangle$ and $L\langle K_Y, Y\rangle$ including the same keys
- ✓Fibonacci hashing function

Ranking Correlated Columns

Top-k Join-Correlation Queries:

Given a column$Q$ and a join column KQ from a query table TQ, find the top-k tables TX in a dataset collection such that TX is joinable with TQ on KQ and has the highest after-join correlations between a column $C \in$ TX and Q.

❖ False Positives results:

Poorly-correlated data may seem more correlated than they actually are

Ranking with uncertain estimates

$$max \sum_{i=1}^{k} (|\hat{r}_{Q \lhd \rhd Ci}| \times (1 - risk(Q, C_i)))$$

- $|\hat{r}_{Q \lhd \rhd Ci}|$: estimated correlation computed on $L_{Q \lhd \rhd Ci}$
- $risk(Q, C_i)$: function that returns a number in the range [0, 1] and measures the dispersion of the correlation estimates using $L_{Q \lhd \rhd Ci}$, such as standard error or confidence interval length.

Correlation Sketches – Creation Methods

Based on paper 2 methods are implied for the creation of Correlation Sketches:

Method #1:

Select n samples of pairs $\langle h(k), xk \rangle$ with minimum values of $hu\ (k)$

$$L\langle KX\ ,X \rangle = \{\langle h(k), xk \rangle : k \in min(k, hu\ (k))\}$$

where min a function that returns a set containing the keys k with the n smallest values of $hu\ (k)$.

Method #2:
Tree-based Algorithm – Extension of KMV family

Algorithm 1 (KMV Computation)

```
h: hash function from domain of dataset to {0, 1, ..., M}
L: list of k smallest hashed values seen so far
maxVal(L): returns the largest value in L

for each item x in the dataset do
    v = h(x)
    if v ∉ L then
        if |L| < k then
            insert v into L
        else if v < maxVal(L) then
            insert v into L
            remove largest element of L
        end if
    end if
end for
```

Which method is more effective? 

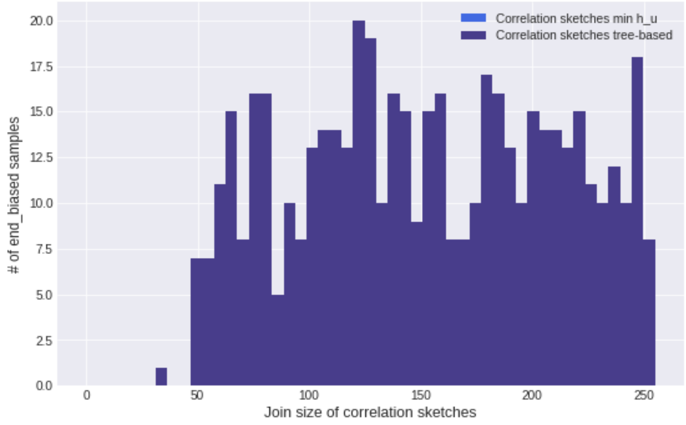
Correlation Sketches – Chose Method

Execution Time

Approach	*Sec*
K minimum values of $h_u(\mathrm{x})$	0.0899
Tree-based algorithm – kmv extension	0.5907

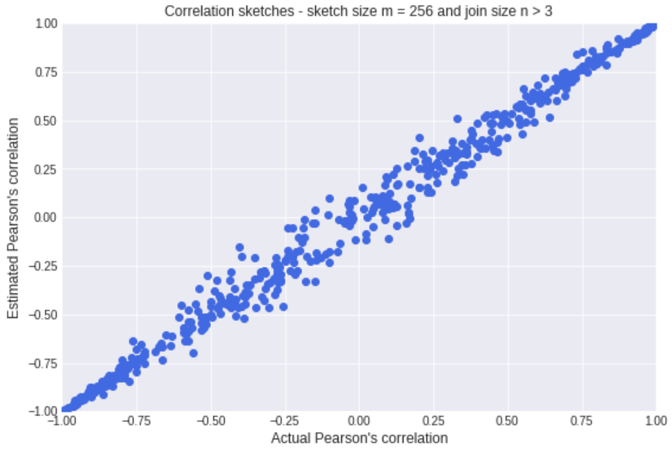
Method #1 runs 15% faster.

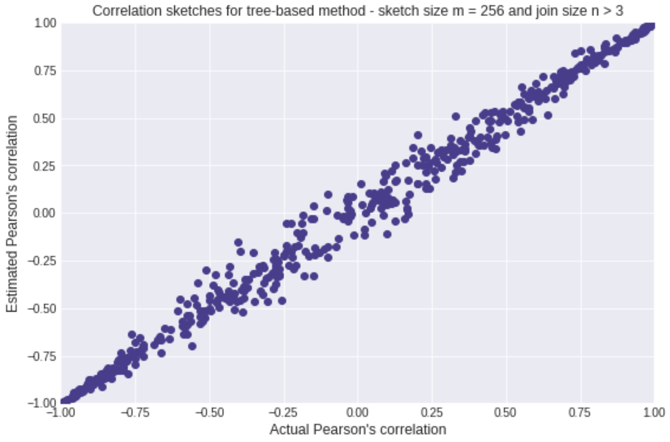
Join Sizes Produced



Both methods produce same join sizes.

Actual VS Estimated Correlation

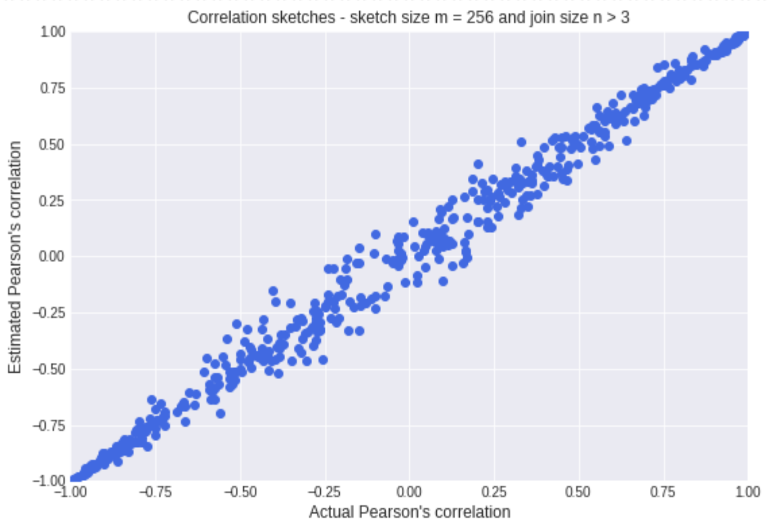


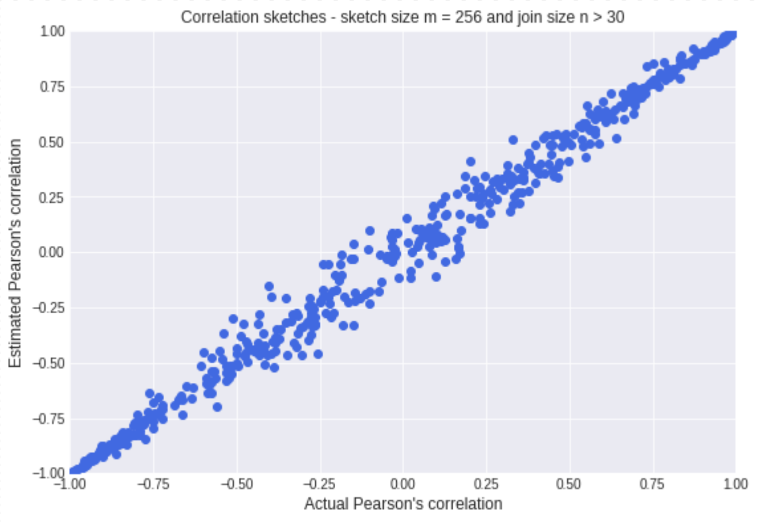


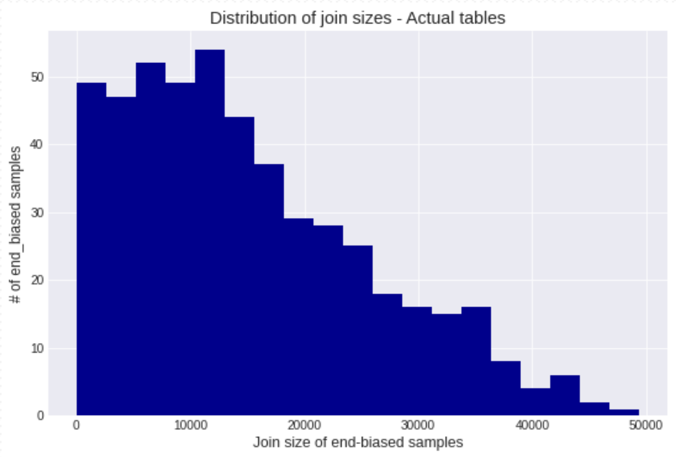
Both methods perform satisfiably well against actual correlations.

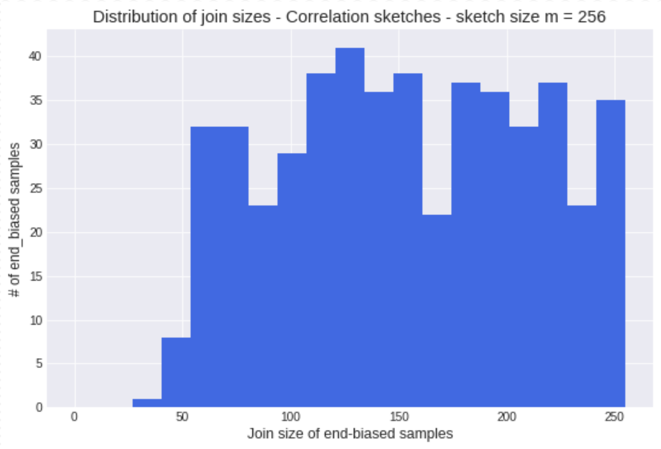
Choose Method #1 –
K minimum values of $h_u(\mathrm{x})$

Estimation accuracy – Correlation Sketches





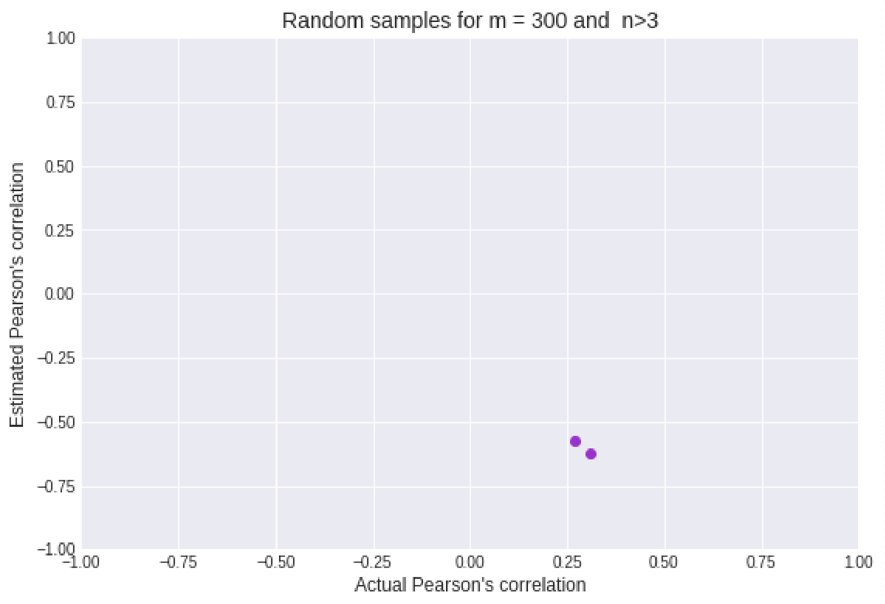




- *Correlation Sketches of size 256*
- *Pearson's correlation estimation*
- *Data drawn from bivariate distribution*

Pair tables Tx = ⟨KX,X⟩, Ty = ⟨KY,Y⟩
- rows uniformly at random (0, 50.000)
-Ty uniform random sample of Tx

Estimation accuracy – Random Sampling



66

- *Random Sampling fails at join-correlation estimation*
- *Values are not aligned based on join values*

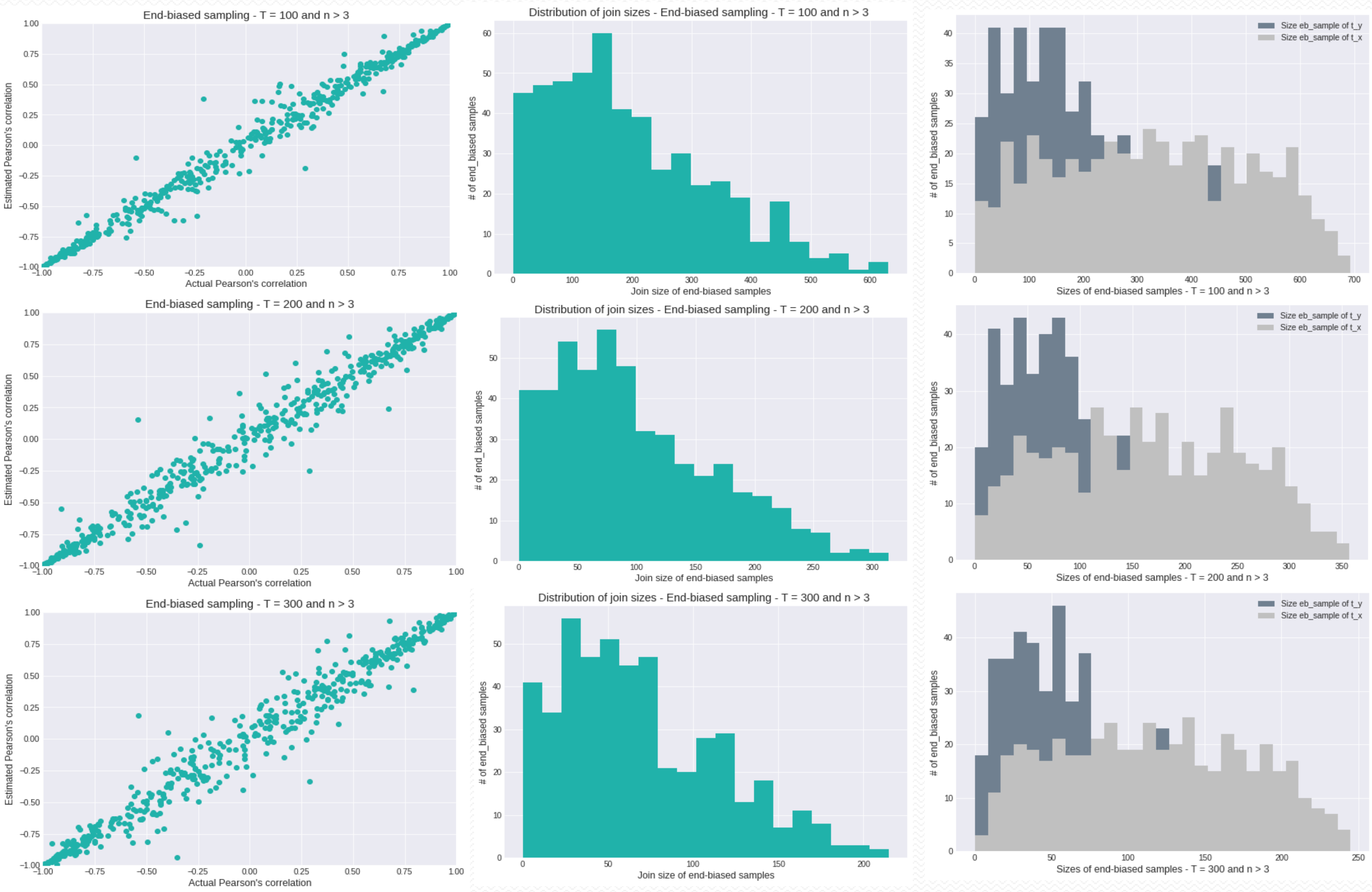
End-biased Sampling

This technique builds on end-biased histograms and uses single dimension histogram to build a compact summary for every given attribute of a table.

<u>End-biased samples creation steps:</u>

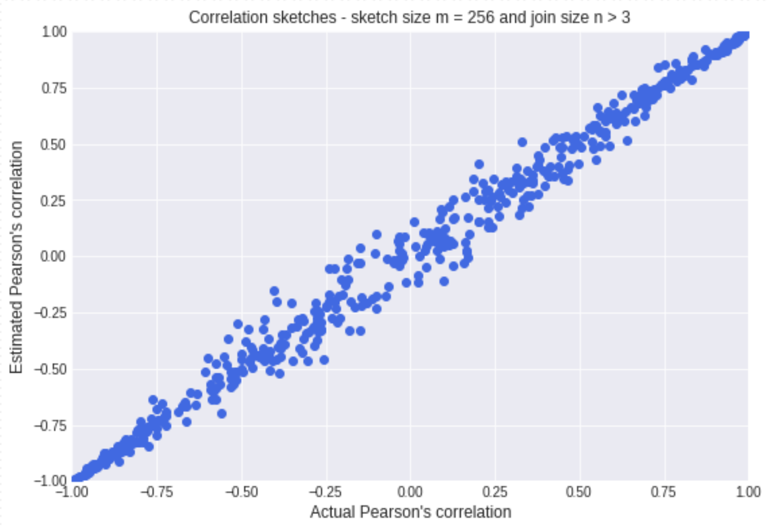
Given an attribute of a table,
1) Calculate the full list of the repeat counts/ frequencies for each value of the attribute
2) Apply rule:
- If frequency f_v > threshold T
  Add tuple in sample
- If frequency f_v < threshold T
  Compute probability p_v = f_v / T
  Apply 2-universal hash function h(v)
  If h(v) < p_v
    Add tuple in sample

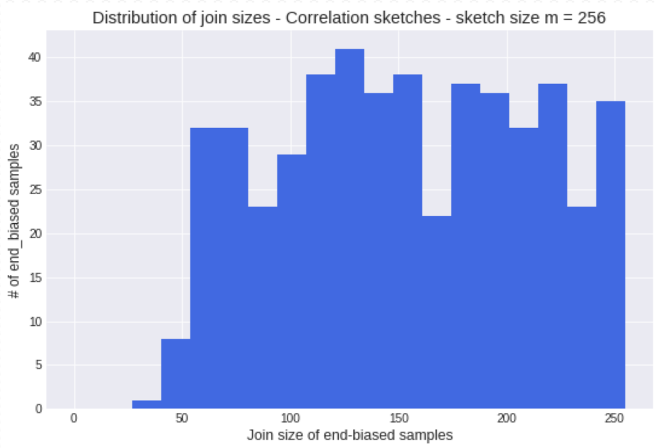
Estimation accuracy – End-biased Sampling

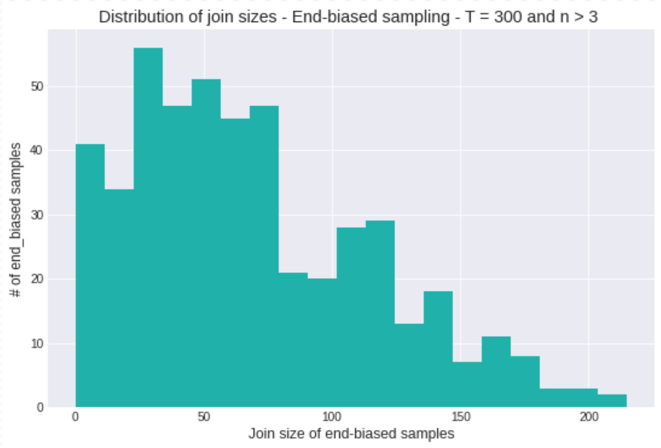


The threshold T is a parameter we can use to trade off accuracy and sample size.

Correlation Sketches VS. End-biased Sampling

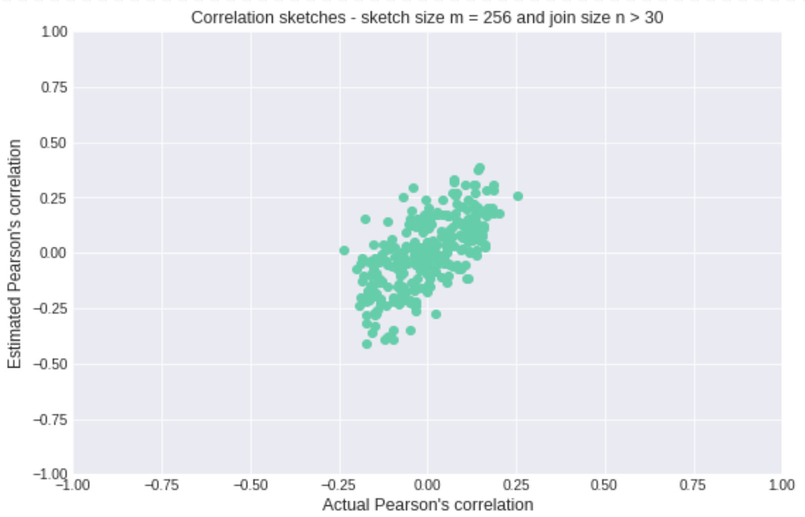


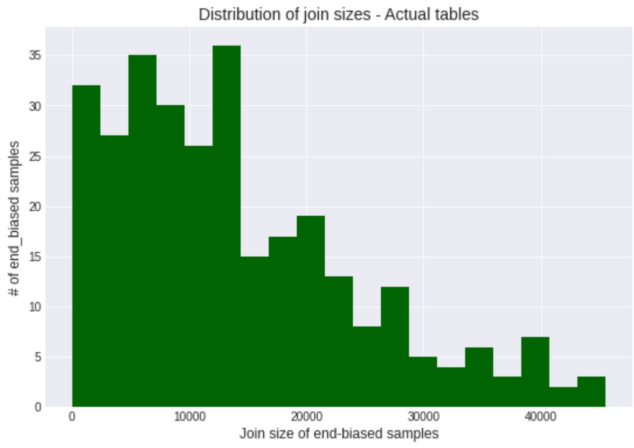


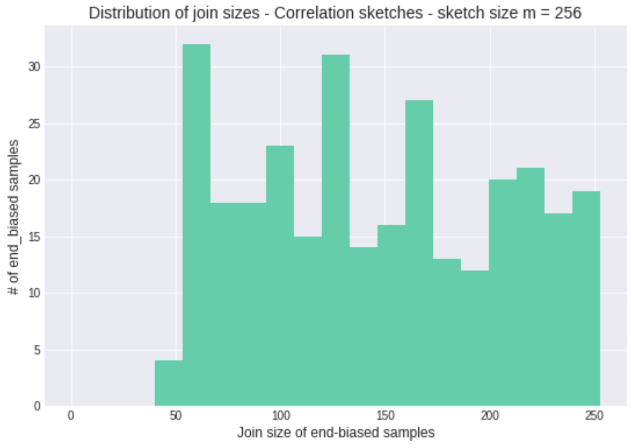


Correlation Sketches outperform End-biased Sampling.

Estimation accuracy – Correlation Sketches





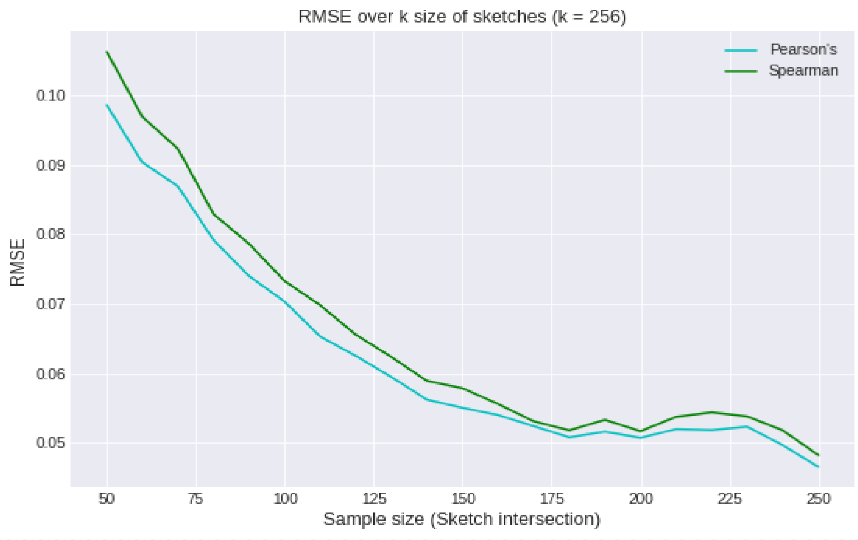


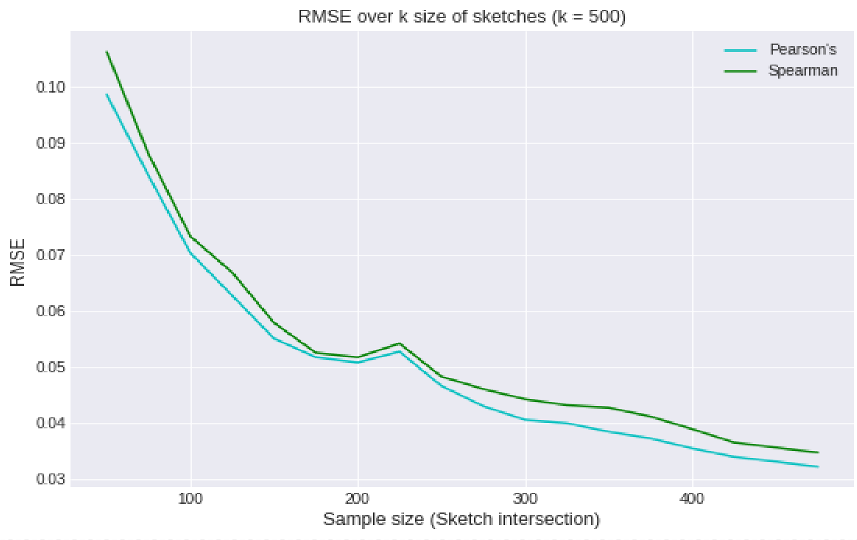
- *Correlation Sketches of size 256*
- *Pearson's correlation estimation*
- *Data drawn from mixture of 2 bivariate distributions*

Pair tables Tx = ⟨KX,X⟩, Ty = ⟨KY,Y⟩
- rows uniformly at random (0, 50.000)
-Ty uniform random sample of Tx

Correlation accuracy – RMSE

Data drawn from bivariate distribution





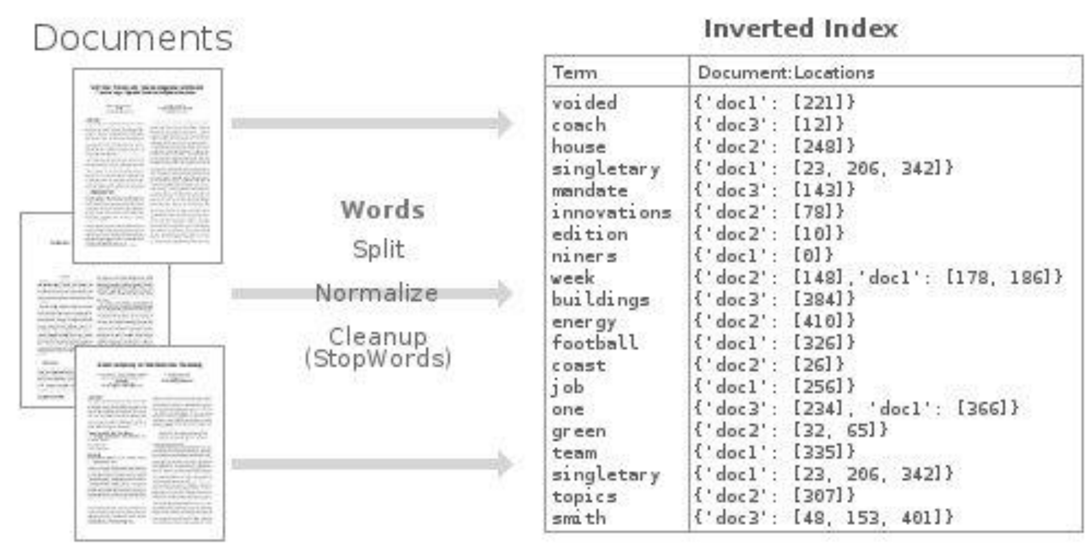
Data drawn from mixture of 2 bivariate distribution

In all cases we confirm the trend:

RMSE converges to minimum value as size of correlation sketches increases

Inverted Indexing

The inverted index is a database index storing a mapping from content (such as words) to its locations in a database (set of documents).
Allows fast full-text search.



Applied invert indexing before top-k join-correlation query to corpus set for join values overlapping > 5000

Instead of searching among 499 tables → 286 tables
43% reduction of tables in corpus set

Top-k join-correlation query

Column_name	Pearson_actual	Pearson_estimated	Fisher_z	Bootstrap	Random_scoring	Jaccard	Jaccard_estimation	Join_actual	Join_estimated
0	0.871135	0.841683	0.779968	0.808595	0.133051	0.753120	0.585139	37656	189
284	0.036431	0.290216	0.225322	0.188484	0.002459	0.057015	0.047035	5394	23
415	0.036232	0.173109	0.134401	0.113115	0.046299	0.058705	0.047035	5545	23
450	0.031627	0.229370	0.182550	0.145580	0.128940	0.056625	0.055670	5359	27
341	0.030356	0.058685	0.043001	0.031760	0.010322	0.059109	0.034343	5581	17
...	...	...	...	...	...	...	...	...	...
297	0.000534	0.054514	0.043823	0.030865	0.051138	0.057932	0.060041	5476	29
257	0.000502	0.077301	0.060820	0.051927	0.037014	0.057876	0.051335	5471	25
198	0.000408	0.515655	0.382513	0.356782	0.175722	0.057437	0.036437	5345	18
117	0.000273	0.113098	0.087808	0.071063	0.048924	0.056053	0.047035	5135	23
340	0.000097	0.006140	0.005054	0.004071	0.004163	0.056468	0.073375	5345	35

RMSE (estimated Pearson’s , actual Pearson’s) = 0.1869

RMSE (Fisher Z , actual Pearson’s) = 0.1456

RMSE (Bootstrap , actual Pearson’s) = 0.1206

✓ *RMSE decreases with risk-averse scoring framework*

- *Correlation Sketches of size 256*
- *Data drawn from bivariate distribution (rows fixed maximum size of 50.000)*

Top-k join-correlation query

Column_name	Pearson_actual	Pearson_estimated	Fisher_z	Bootstrap	Random_scoring	Jaccard	Jaccard_estimation	Join_actual	Join_estimated
RRP_positive	0.999821	0.997166	0.934475	0.992986	0.161896	1.000000	1.0	2106	256
demand_pos_RRP	0.220856	0.289603	0.271396	0.257156	0.193116	1.000000	1.0	2106	256
demand	0.217538	0.245725	0.230277	0.216596	0.171818	1.000000	1.0	2106	256
max_temperature	0.165484	0.081391	0.076274	0.070610	0.051588	1.000000	1.0	2106	256
demand_neg_RRP	0.078815	0.230132	0.215664	0.211690	0.219165	1.000000	1.0	2106	256
frac_at_neg_RRP	0.077955	0.233192	0.218532	0.214989	0.204205	1.000000	1.0	2106	256
min_temperature	0.070619	0.009721	0.009110	0.008345	0.002823	1.000000	1.0	2106	256
solar_exposure	0.061808	0.005615	0.005262	0.004873	0.004144	0.999525	1.0	2105	256
RRP_negative	0.038931	0.068977	0.064640	0.055298	0.057890	1.000000	1.0	2106	256
rainfall	0.028642	0.009379	0.008789	0.008478	0.000587	0.998575	1.0	2103	256

RMSE (estimated Pearson's , actual Pearson's) = 0.08230

RMSE (Fisher Z , actual Pearson's) = 0.07840

RMSE (Bootstrap , actual Pearson's) = 0.07389

✓ *RMSE decreases with risk-averse scoring framework*

- *Real data*
- *Query set : RRP, recommended retail price in AUD$ / MWh*

Running Time

	Complete Tables			Correlation Sketches		
(sec)	Join	Pearson's	Spearman	Join	Pearson's	Spearman
Mean	0.0455	0.0007	0.0057	0.0027	0.0004	0.0011
Std	0.0276	0.0007	0.0041	0.0007	0.0003	0.0002
75%	0.0631	0.0008	0.0079	0.0027	0.0004	0.0011
95%	0.0889	0.0011	0.0138	0.0039	0.0006	0.0016
99%	0.1099	0.0026	0.0171	0.0046	0.0007	0.0020

✓ *Overall Correlation Sketches run faster, specially for the cases :*

- *Larger tables (percentile 99%)*
- *For more complex estimators (Spearman)*

THANK YOU