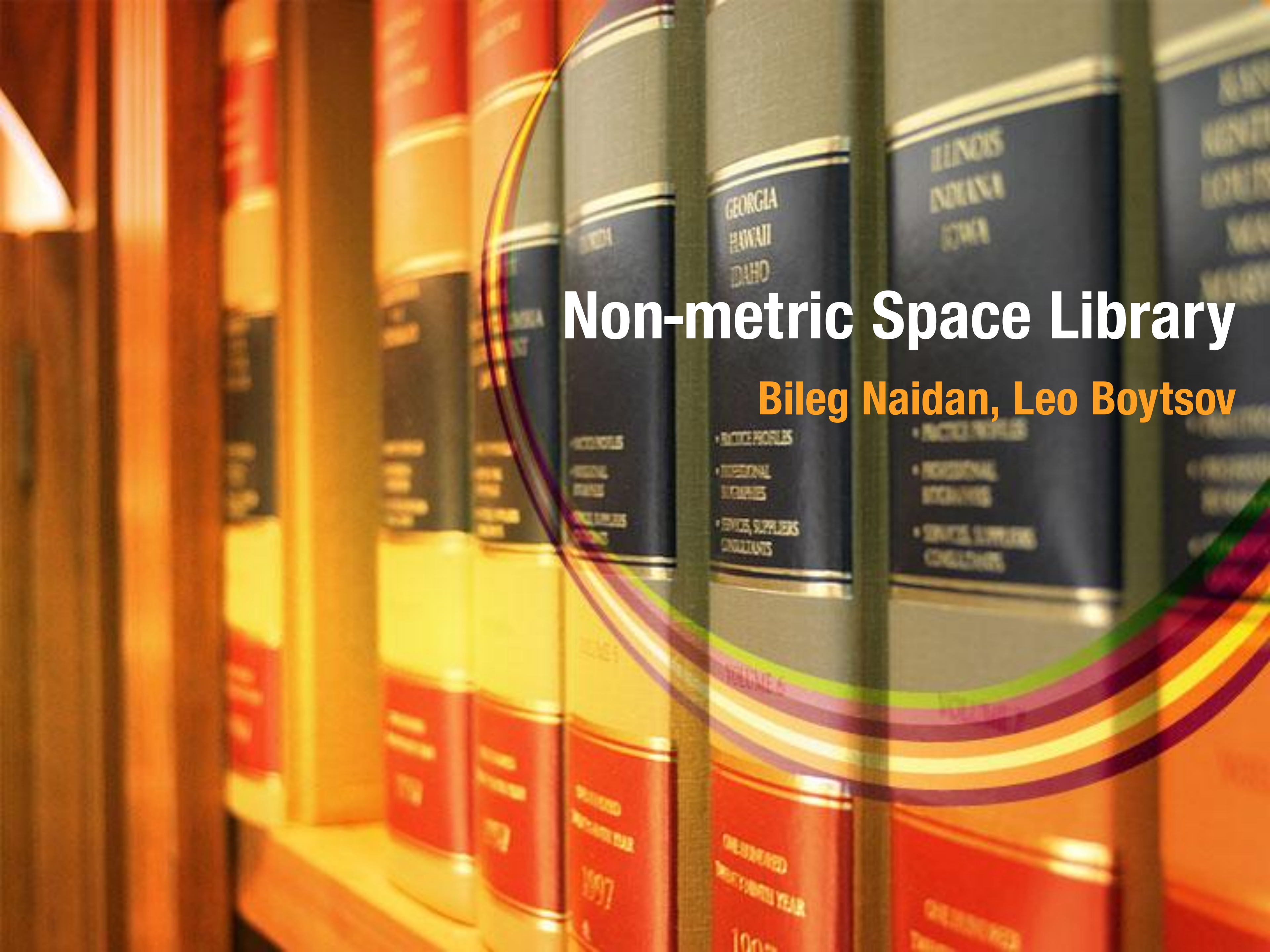
GEORGIA
HAWAII
IDAHO
ILLINOIS
INDIANA
IOWA



Non-metric Space Library

Bileg Naidan, Leo Boytsov

Novelty

- Focus on approximate searching.
- Automatic evaluation (relative position error, recall, number of points closer than NN-neighbor, etc).
- Focus on efficiency and real-world performance.
- Design influenced by Metric Spaces Library: yet, it was reworked and simplified.
- New methods and data sets.

Efficiency: Programming Language

- C++ programs are fast.
- Legacy C-code can be ported rather easily.
- It is easy to use Single Instruction Multiple Data (SIMD) operations.

Efficiency: Not every Distance is Hard

- Many real data sets are (intrinsically) low-dimensional.
- Inexact nature of searching often permits to approximate a complex distance function with a simple one.
- For example, through dimensionality reduction techniques such as PCA or random projections.

Efficiency: How Many Distances per Second?

128 elements, single thread, core-i7, 3.4 Ghz

- **L1** 9.6 millions
- **L2** 9.1 millions
- **Itakura-Saito** 190 thousand
- **KL-divergence** 530 thousand

Efficiency: How Many Distances per Second?

128 elements, single thread, core-i7, 3.4 Ghz

- **L1** 9.6 millions
- **L2** 9.1 millions
- **Itakura-Saito** 190 thousand
- **KL-divergence** 530 thousand

Slow distances!

Efficiency: Optimizing Euclidian Distance

$$L_2(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$$

Let's use SIMD instructions:
one instruction multiplies/adds 4 numbers!

Efficiency: Optimizing KL-divergence

$$KL(x, y) = \sum_i x_i \log\left(\frac{x_i}{y_i}\right) =$$

$$= \sum_i x_i \log x_i - \sum_i x_i \log y_i$$

- Precompute logs at index time.
- In addition, use SIMD at query time.

Efficiency:
How Many Optimized Distances per Second?

128 elements, single thread, core-i7, 3.4 Ghz

• **L1**	27 millions
• **L2**	33 millions
• **Itakura-Saito**	8.3 million
• **KL-divergence**	28 million

Efficiency:
How Many Optimized Distances per Second?

128 elements, single thread, core-i7, 3.4 Ghz

- **L1** 27 millions
- **L2** 33 millions
- **Itakura-Saito** 8.3 million
- **KL-divergence** 28 million

3.5x faster!

Efficiency:
How Many Optimized Distances per Second?

128 elements, single thread, core-i7, 3.4 Ghz

- **L1** 27 millions
- **L2** 33 millions
- **Itakura-Saito** 8.3 million
- **KL-divergence** 28 million

3.5x faster!

40x faster!

Efficiency:
How Many JS-Divergences per Second?

$$\frac{1}{2}\sum_i \left[x_i \log x_i + y_i \log y_i - (x_i + y_i) \log \frac{x_i + y_i}{2} \right] =$$

$$= \frac{1}{2}\sum_i [x_i \log x_i + y_i \log_i] -$$

$$- \frac{(x_i + y_i)}{2} \sum_i \left[\log \frac{1}{2} + \log \max(x_i, y_i) + \log\left(1 + \frac{\min(x_i, y_i)}{\max(x_i, y_i)} \right) \right]$$

- Precompute logs at index time
- Discretize and approximate the last log

Efficiency:
How Many JS-Divergences per Second?

128 elements, single thread, core-i7, 3.4 Ghz

• **unoptimized**	0.2 million
• **precomputed logs**	0.6 million
• **discretized log**	1.1 million
• **SIMD operations**	3.9 million

Efficiency:
How Many JS-Divergences per Second?

128 elements, single thread, core-i7, 3.4 Ghz

- **unoptimized** 0.2 million
- **precomputed logs** 0.6 million
- **discretized log** 1.1 million
- **SIMD operations** 3.9 million

20x faster!

Efficiency:
How Many Distances per Second?

128 elements, core-i7, 3.4 Ghz, **8 cores**, **10GB/sec** memory

~ 2.5 million distance computations/sec

memory becomes a bottleneck!

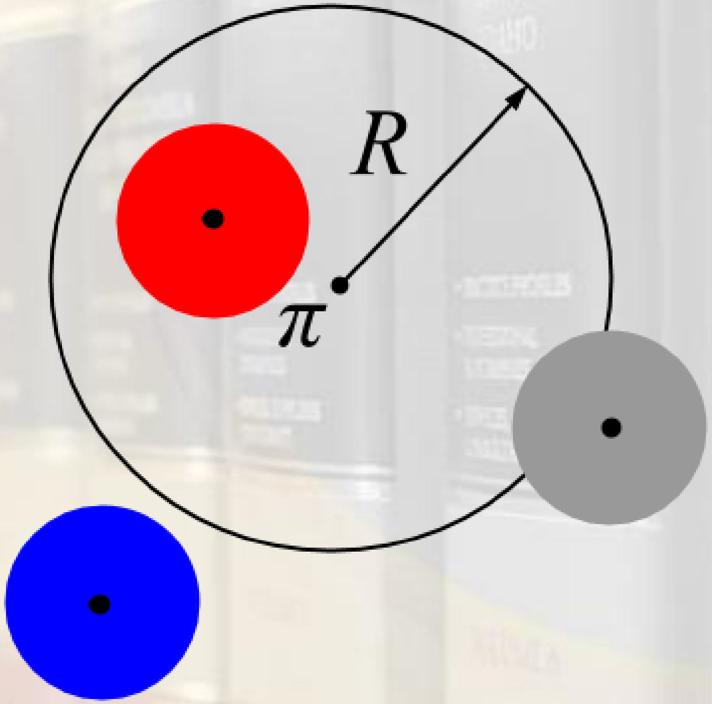
Design I: Simplifications

- Don’t care about storing indices on disk – more rapid development.
- We have a single binary that covers all methods and spaces (both integer and floating-point distances).
- Factory pattern: adding a new method/space doesn’t require changing shared code and/or makefiles!
- Similar interface for NN and range queries: same code can be used.

Design II (Search Oracles)

Most importantly, metric space access methods can work in non-metric spaces, if we replace the triangle inequality based pruning with a more generic search oracle.

Search Oracle
(three types of query balls)

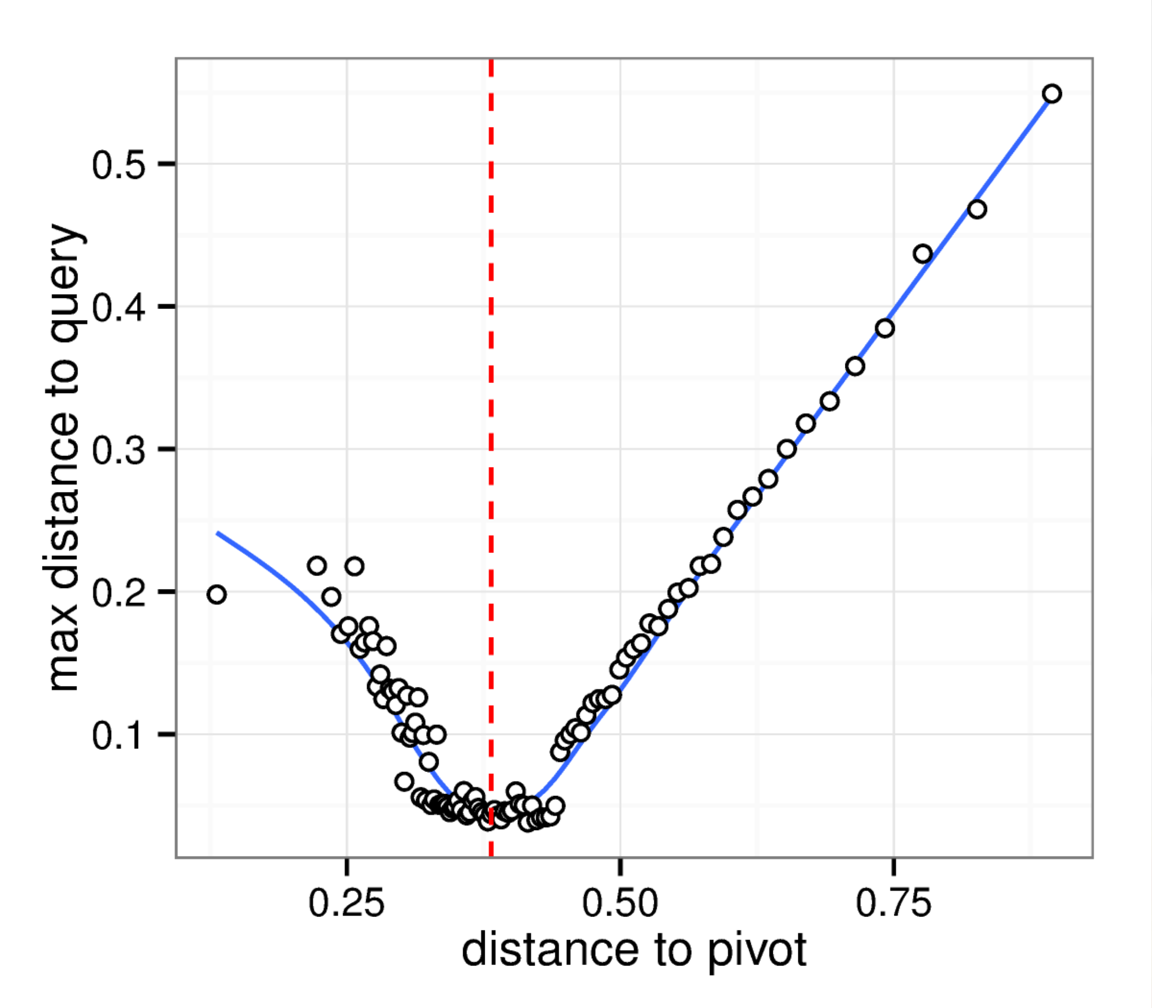


In metric spaces , the triangle inequality allows us to distinguish among three types of query balls!

What about Non-metric Spaces?

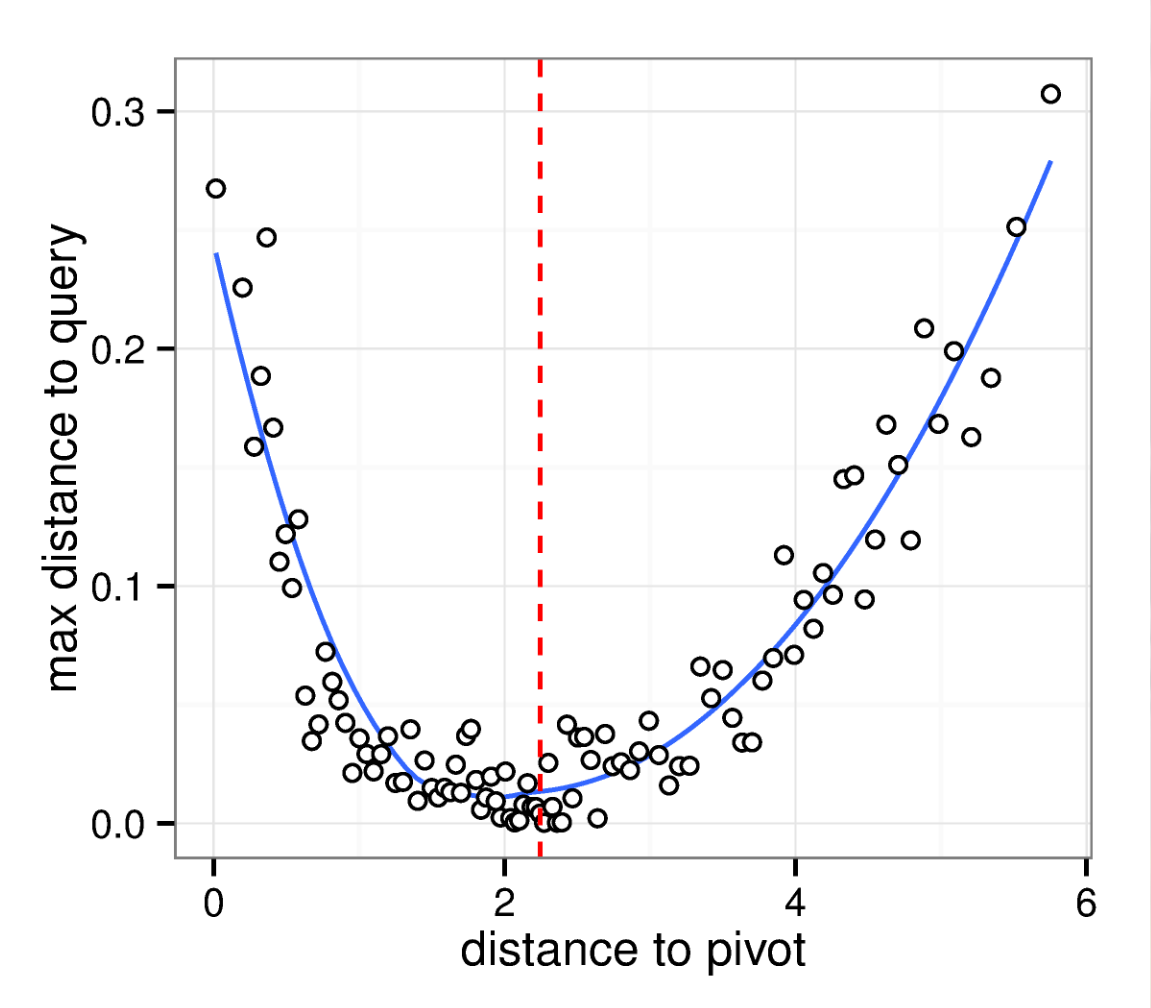
- We have a classification problem, the decision function can be learned.
- One can use sampling, which is old idea (Zezula et al. 1998, Amato et al. 2003).
- A decision function can be approximated using, e.g., a piecewise linear function (Chavez & Navarro, 2003).
- We tried both and found naïve sampling to be inferior, details can be found in our NIPS 2013 paper.

Search Oracle (Learned by Sampling)



Euclidian distance
Colors data set

Search Oracle (Learned by Sampling)



KL-divergence
RCV-8, Cayton 2007

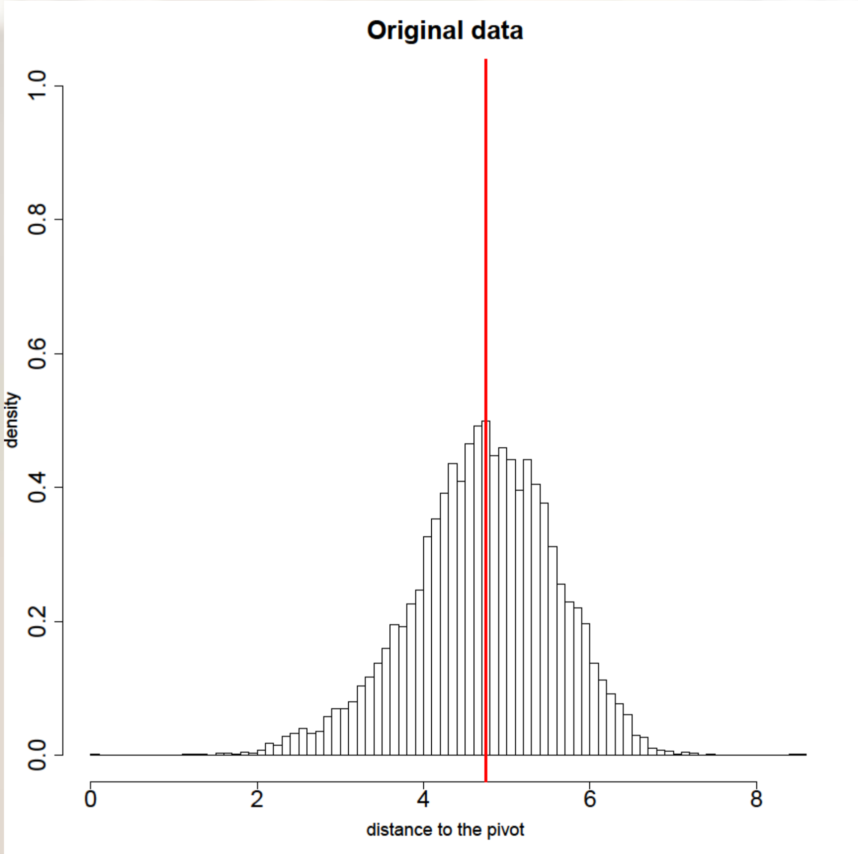
Evaluation

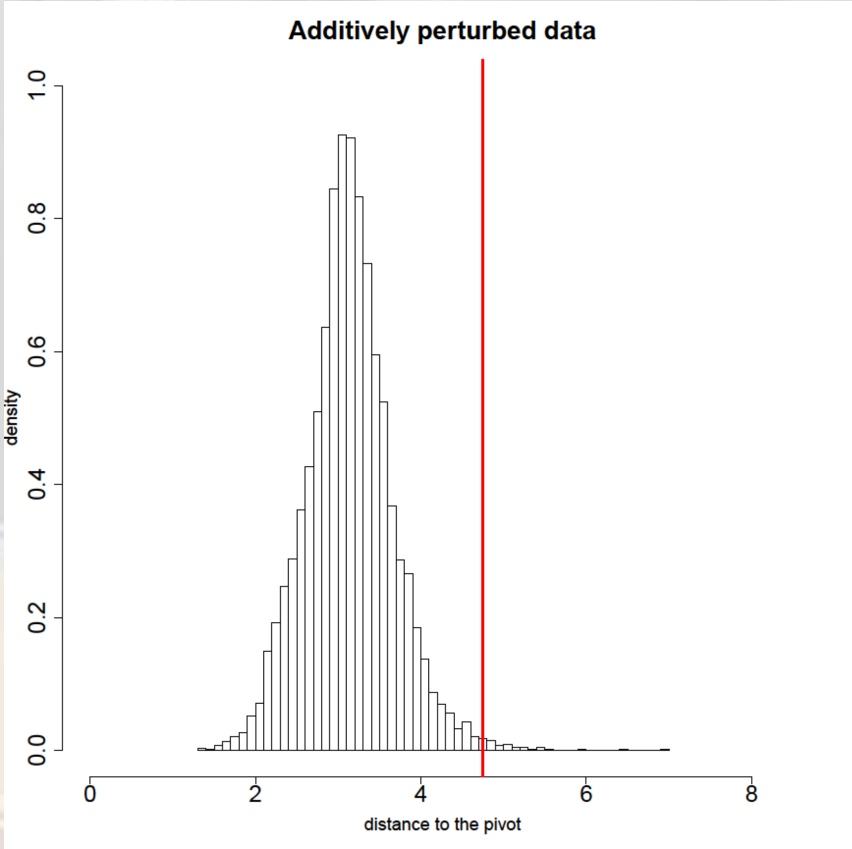
- Methods should return a set of found object ids – necessary for automatic evaluation.
- Effectiveness metrics should be computed automatically.
- Exhaustive search is expensive – compute ones for several methods.

Evaluation

- Best, when we have real queries.
- If not, bootstrap-like automatic test procedures can randomly divide data into indexable data and query sets.
- One should not search for queries that are already indexed!
- One should not create queries by applying additive noise to data points!

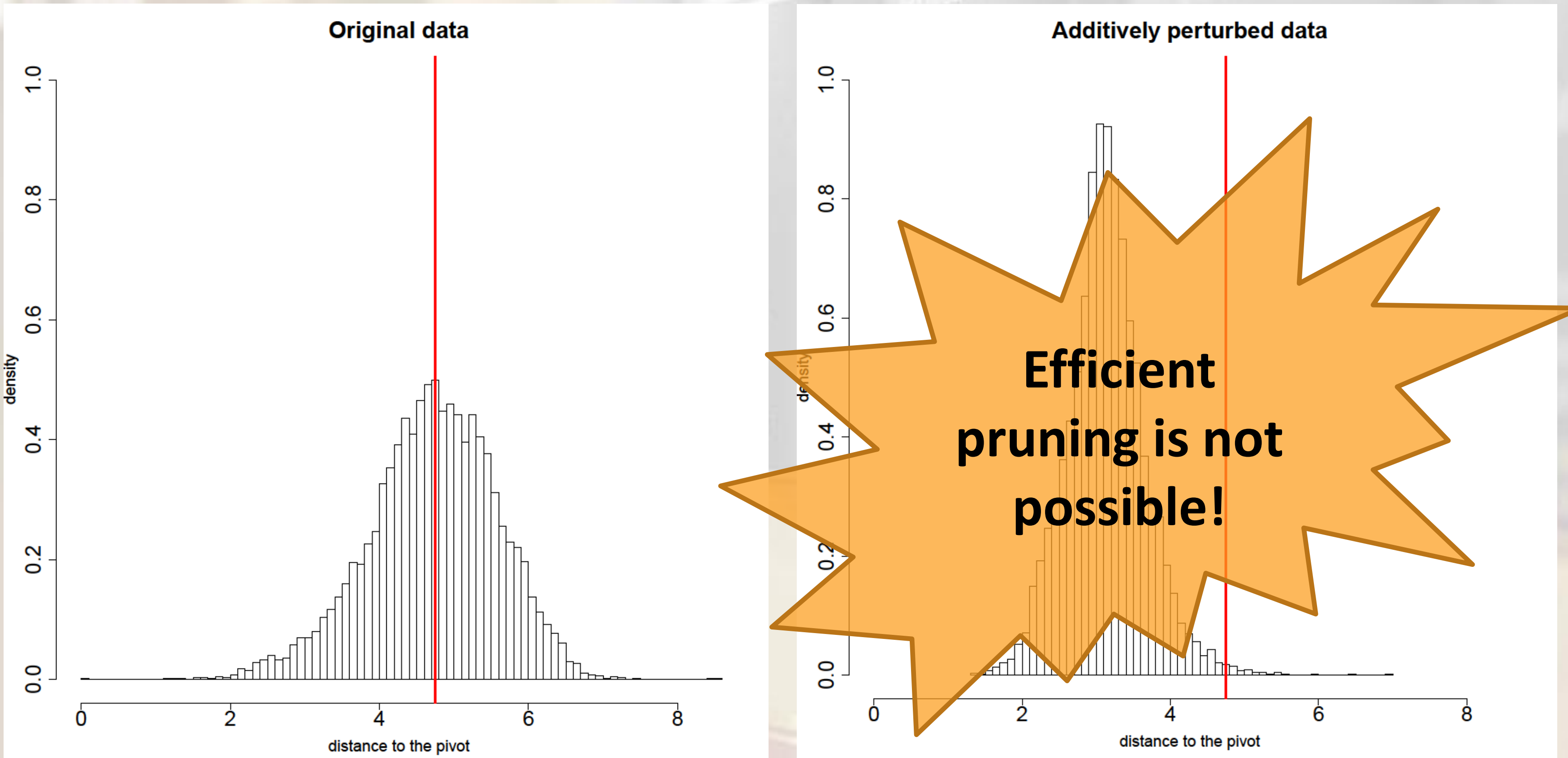
Additive Noise: KL-divergence





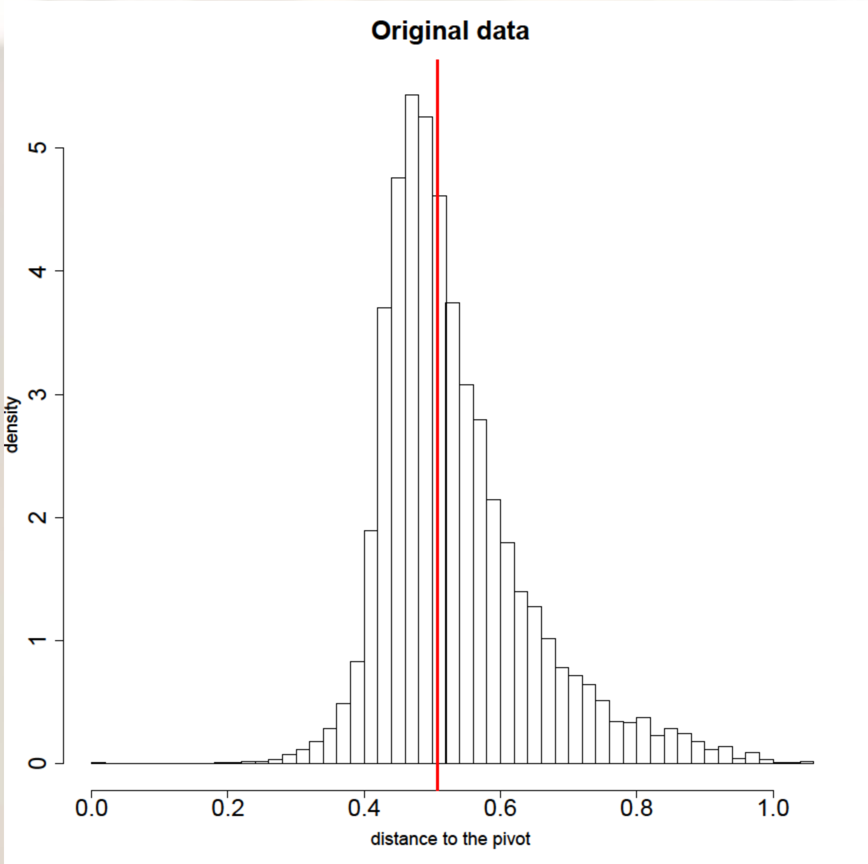
KL-divergence: distribution of distances to a pivot.
The red line denotes the median distance to the pivot in unmodified data.

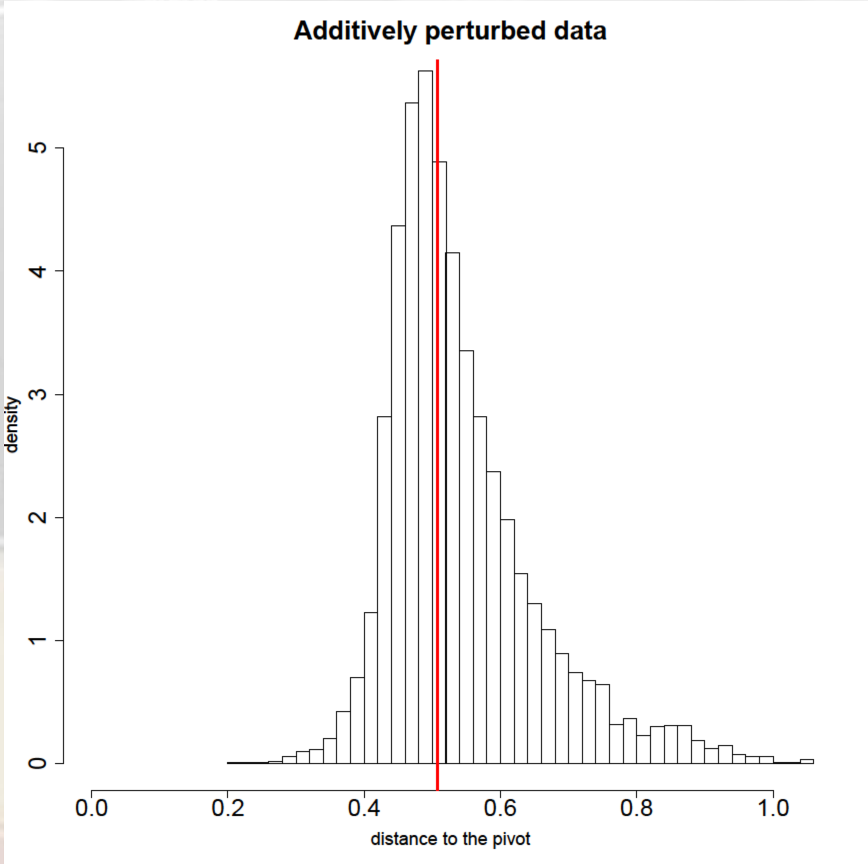
Additive Noise: KL-divergence



KL-divergence: distribution of distances to a pivot.
The red line denotes the median distance to the pivot in unmodified data.

Same Amount of Additive Noise: L2

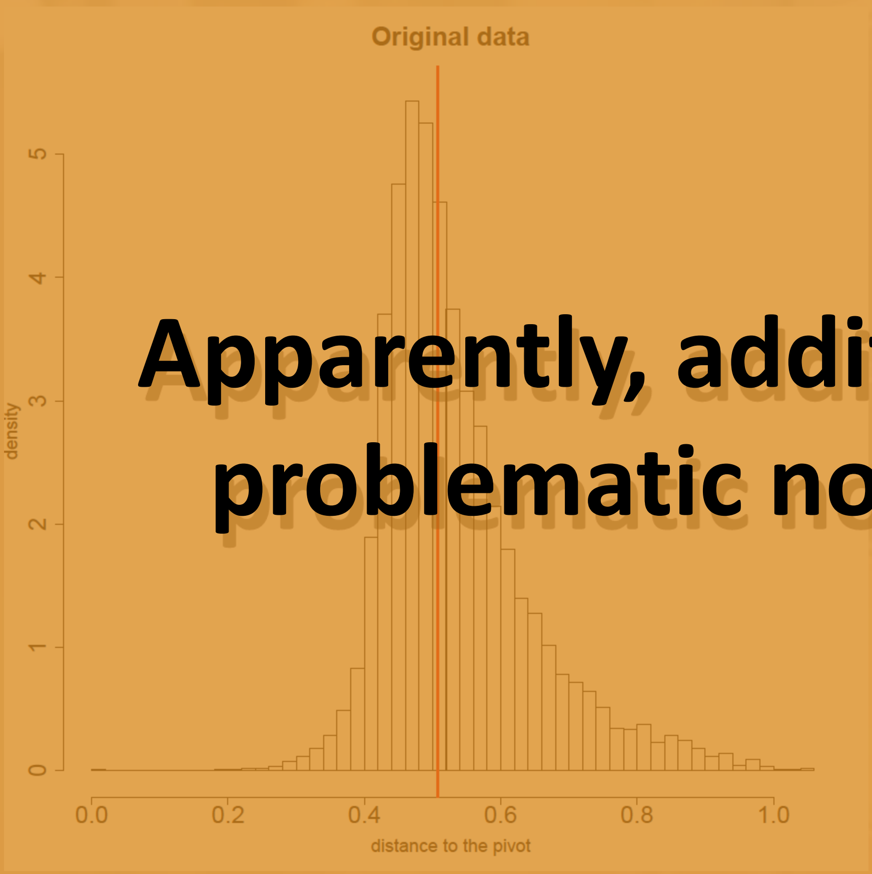


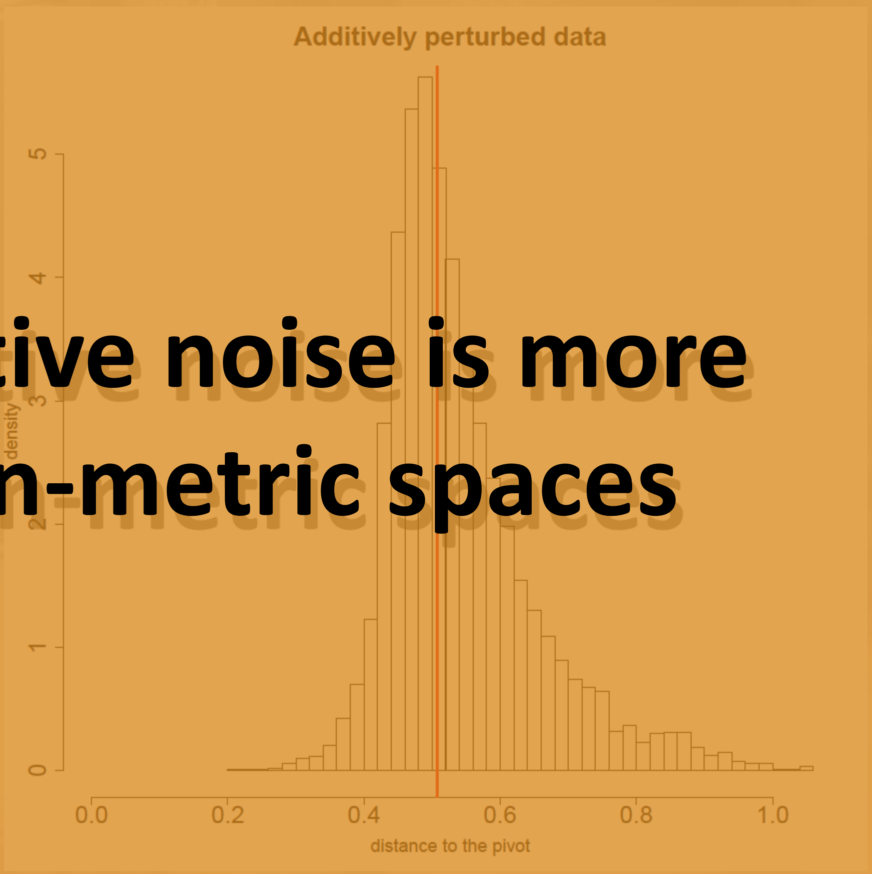


Euclidian distance: distribution of distances to a pivot.

The red line denotes the median distance to the pivot in unmodified data.

Same Amount of Additive Noise: L2

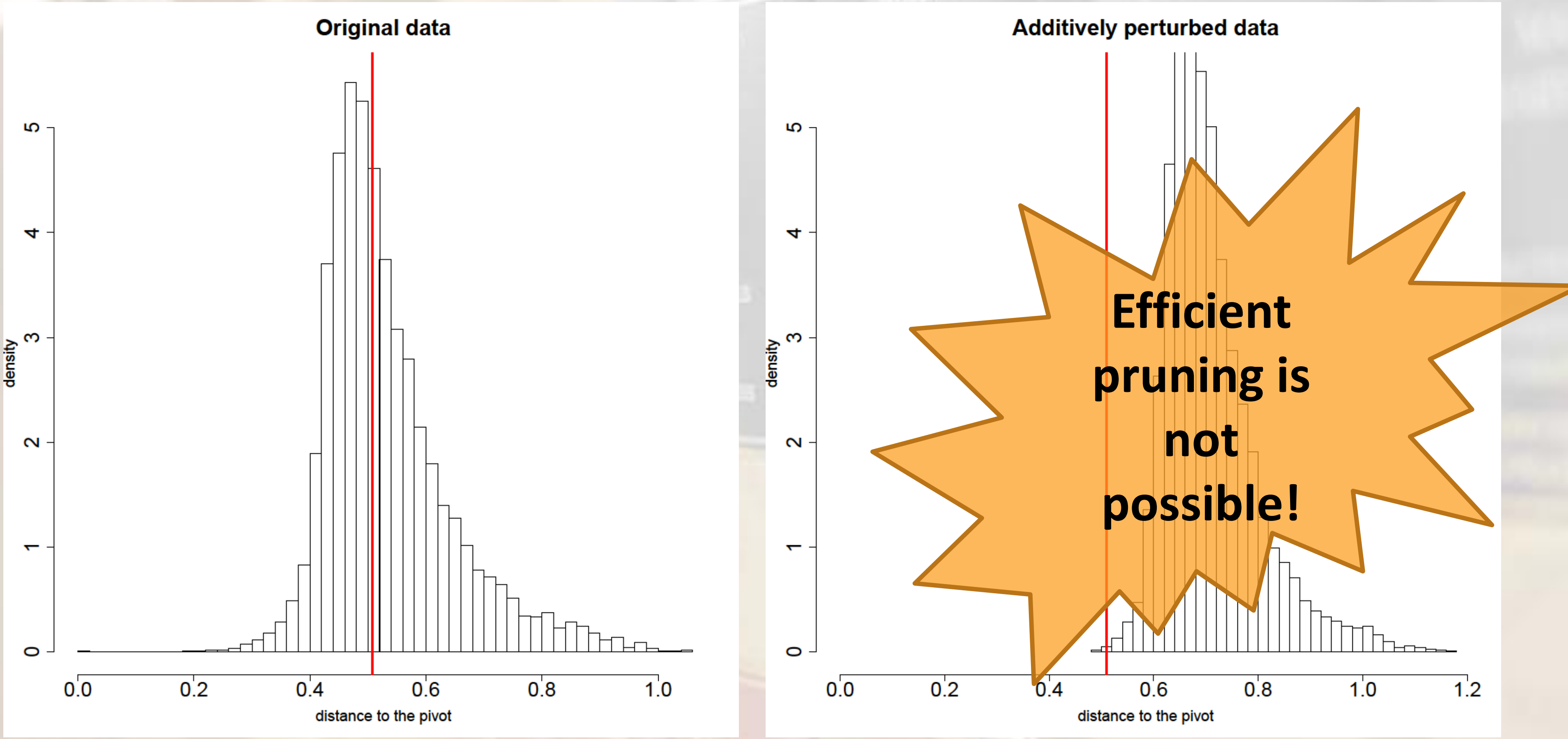




Apparently, additive noise is more problematic non-metric spaces

Euclidian distance: distribution of distances to a pivot.
The red line denotes the median distance to the pivot in unmodified data.

Add more noise:
Is Euclidean Distance Robust?



The red line denotes the median distance to the pivot in unmodified data.

Implemented Methods

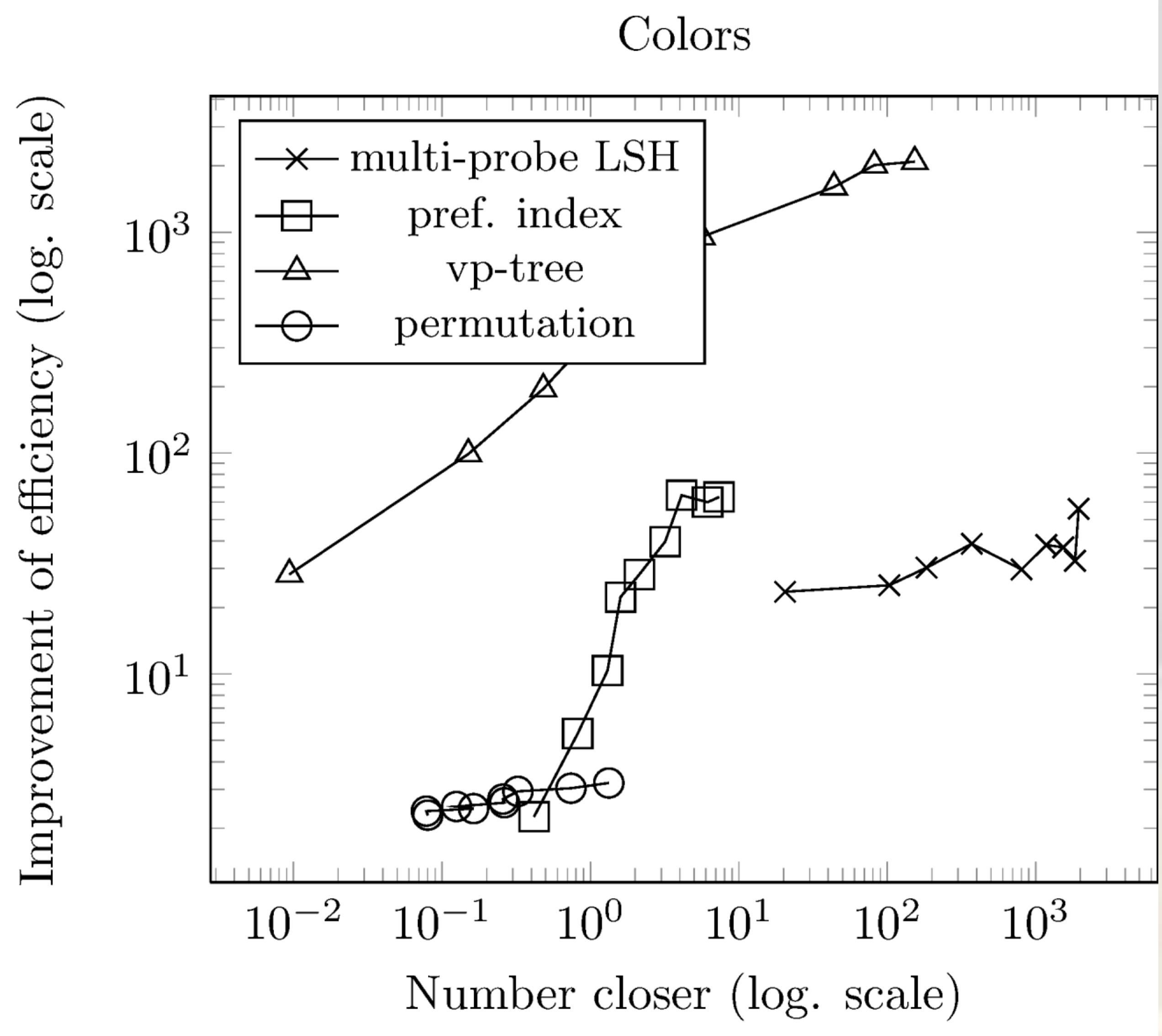
Metric	Non-Metric
VP-tree	VP-tree (with learned oracles)
GH-tree	BB-tree (for Bregman divergences)
List of clusters	Permutation index (regular and incremental sorting)
Spatial approximation tree	Permutation prefix index
LSH (classic and multi-probe)	Permutations indexed with VP-tree

Implemented Methods

Metric	Non-Metric
VP-tree	VP-tree (with learned oracles)
GH-tree	BB-tree (for Bregman divergences)
List of clusters	Permutation index (regular and incremental sorting)
Spatial approximation tree	Permutation prefix index
LSH (classic and multi-probe)	Permutations indexed with VP-tree

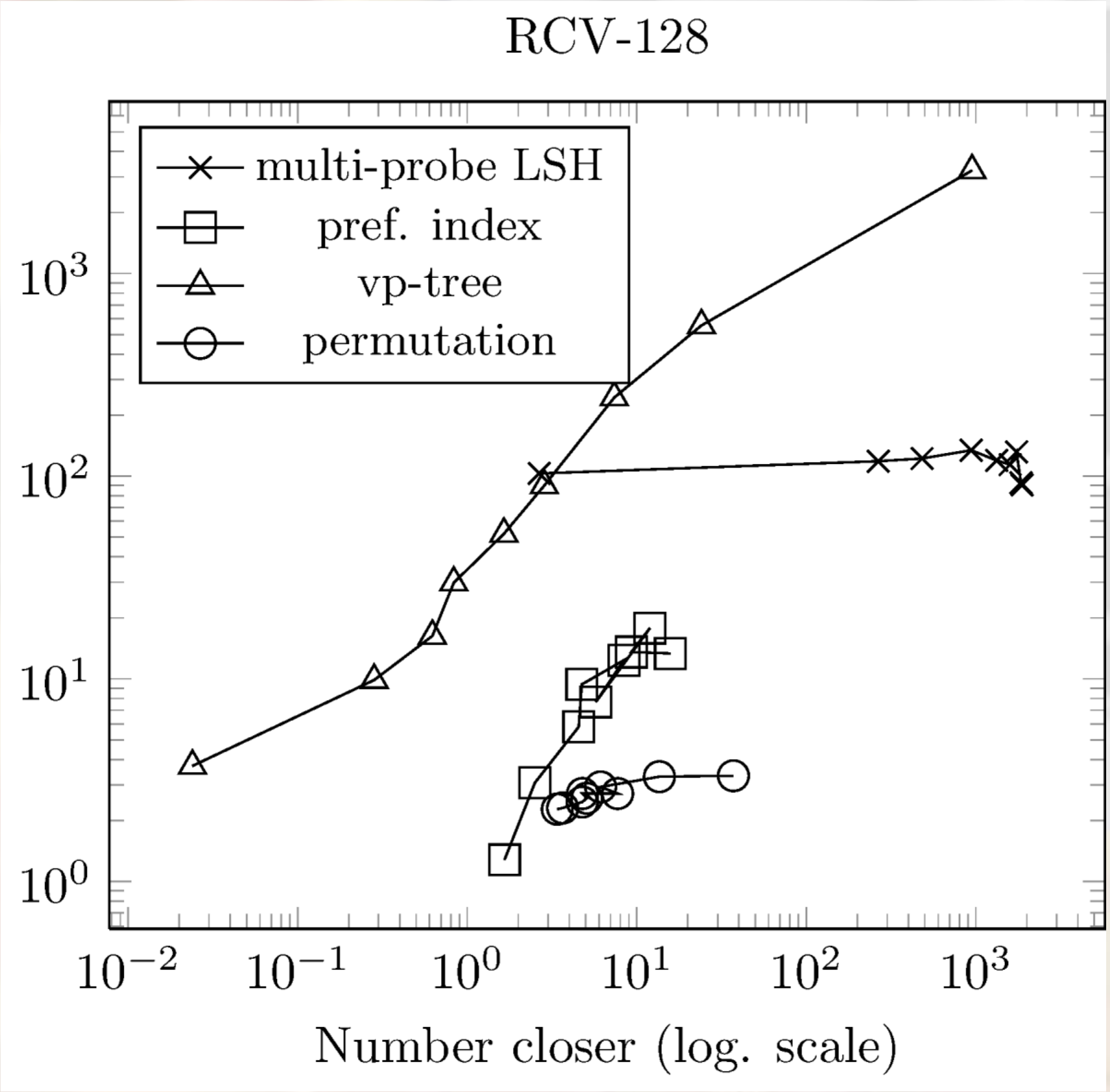
Our library is about 50% non-metric!

Nearest Neighbor (L2)



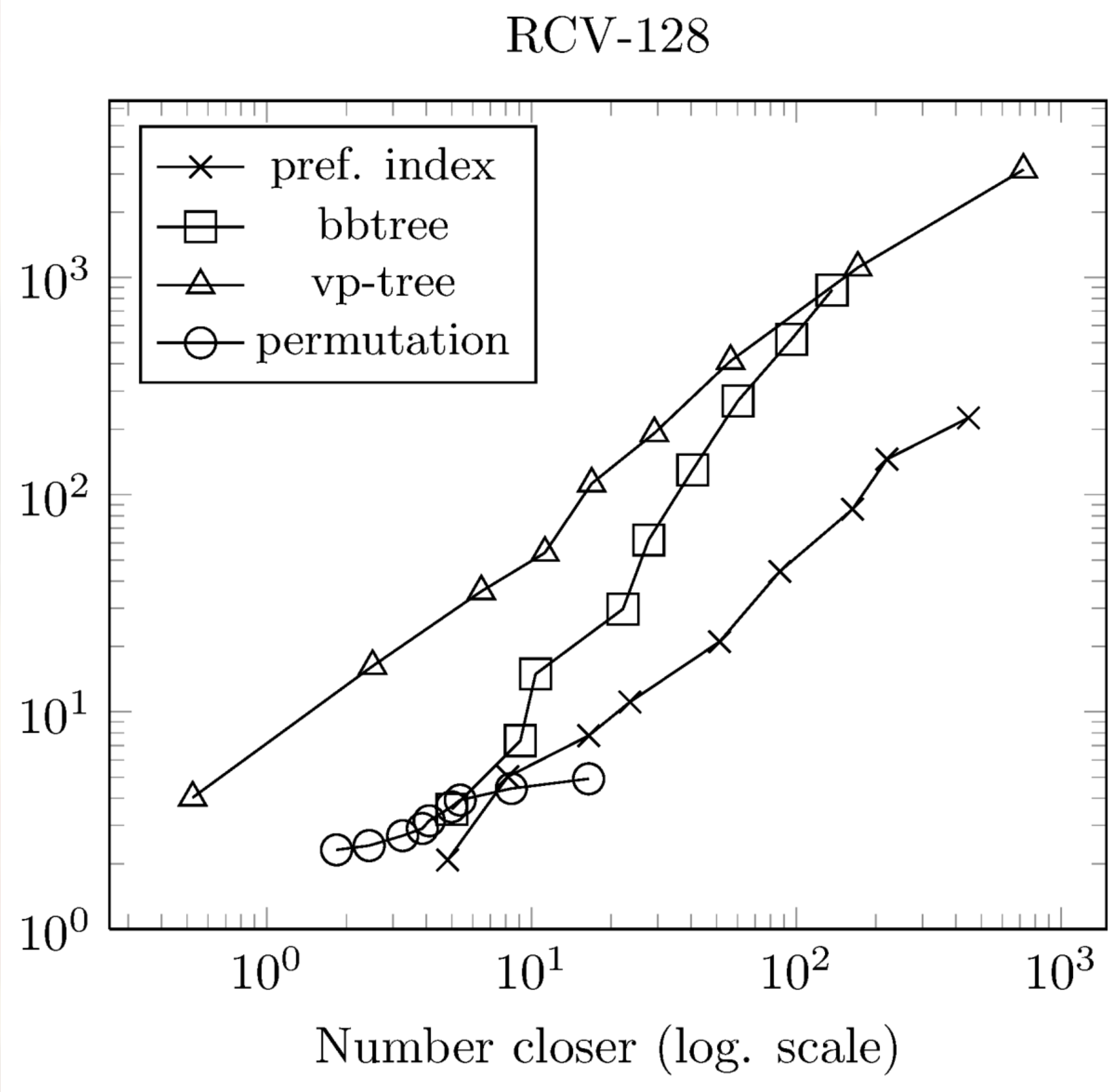
Vector space 112 elements

Nearest Neighbor (L2)



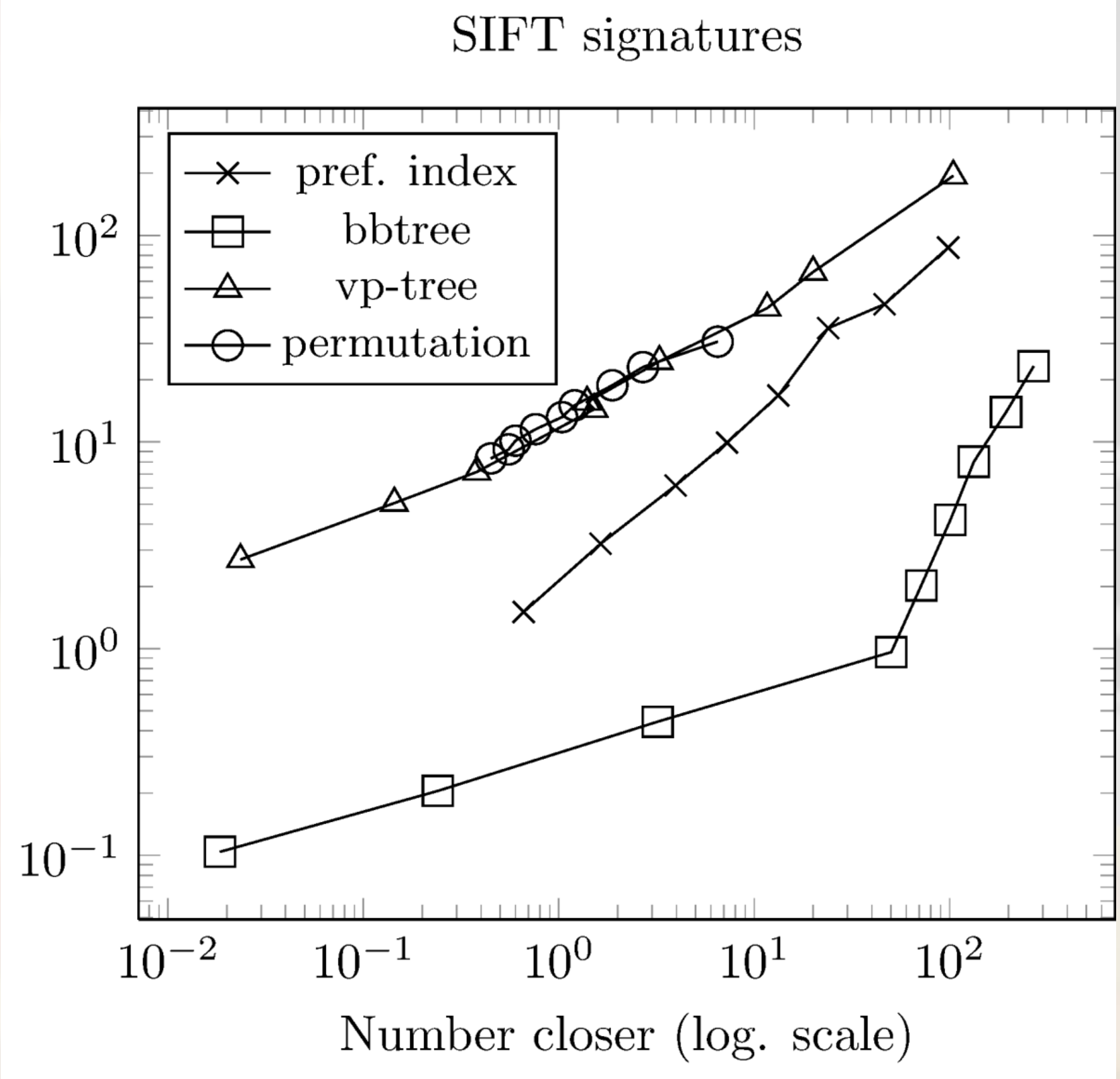
Vector space 128 elements

Nearest Neighbor (KL-divergence)



Vector space 128 elements

Nearest Neighbor (KL-divergence)



Vector space 1111 elements

Future Work

- Implement additional methods for non-metric spaces.
- Better search oracles (our resampling is naïve)
- Add new spaces (we want to have very efficient distance functions).
- More test sets, especially with human judgments.

New Methods to Implement

- TriGen (Skopal, 2007)
- Permutation-based locality sensitive hashing (Tellez, Chavez, 2010)
- Small-word approaches (Malkov et al 2012; Houle and Nett, 2013)
- VA-file and the R-tree for Bregman divergences (2009)
- LSH for symmetrized divergences (Yadong Mu, Shuicheng Yan, 2010)
- Ptolemaic indexing (Hetland et al, 2013)

Concluding Notes

- Software and data are available online: https://github.com/searchivarius/NonMetricSpaceLib
- It is still work in progress.
- The design is not set in stone, we can change it.
- Future additions are welcome (we would be happy to acknowledge them).
- We can jointly produce a very through experimental study (e.g., for ACM Computing Surveys).

Thank you!

Questions?