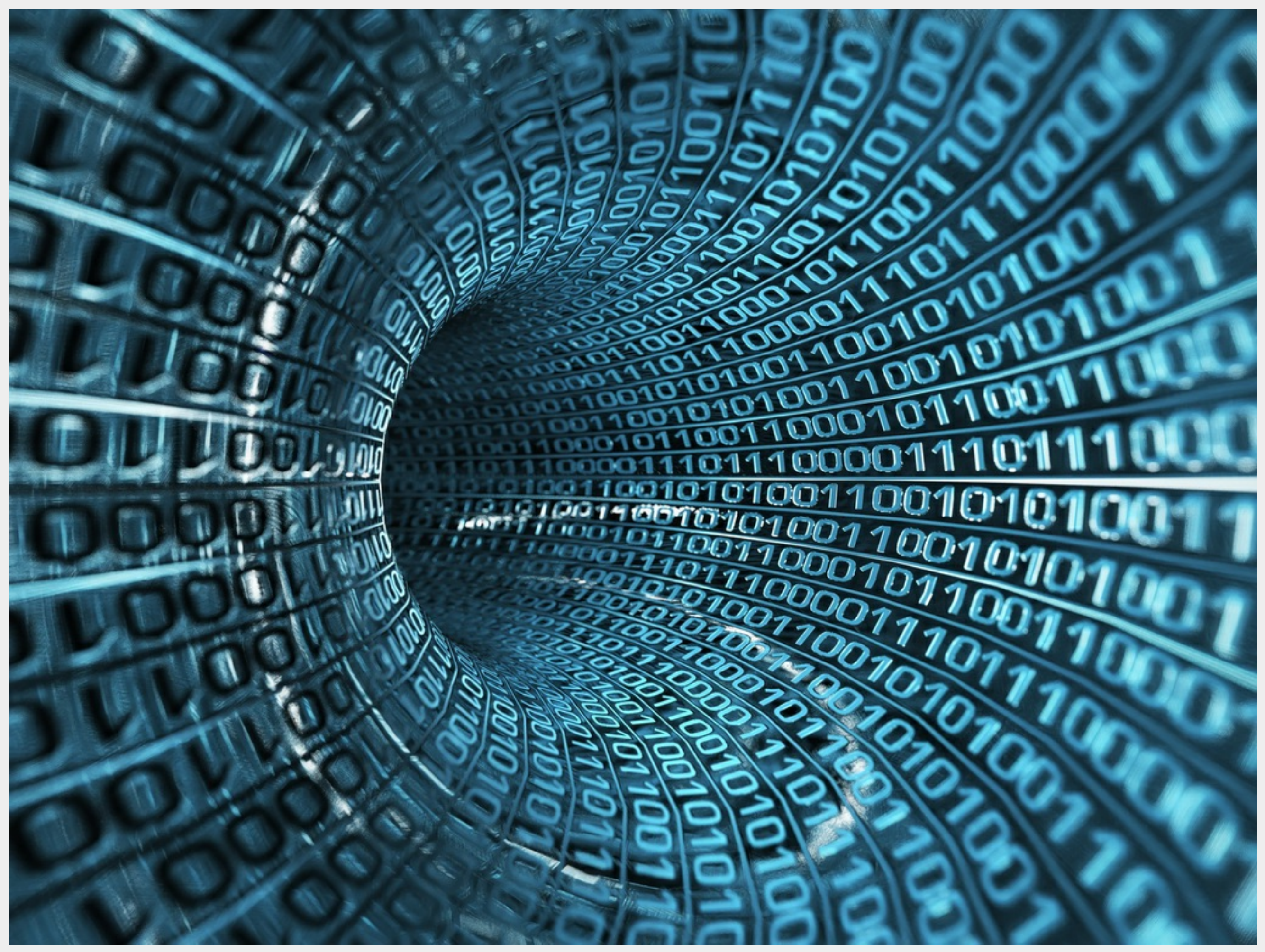
STREAMING DATA ANALYSIS



Rob Doherty / @robdoherty2

STREAMING DATA ANALYSIS

- Introduction to the problem domain
- A few key data structures
- Implementation Considerations

WHAT IS STREAM PROCESSING?

Examples?

WHAT IS STREAM PROCESSING?

Examples?

- Sensor data

WHAT IS STREAM PROCESSING?

Examples?

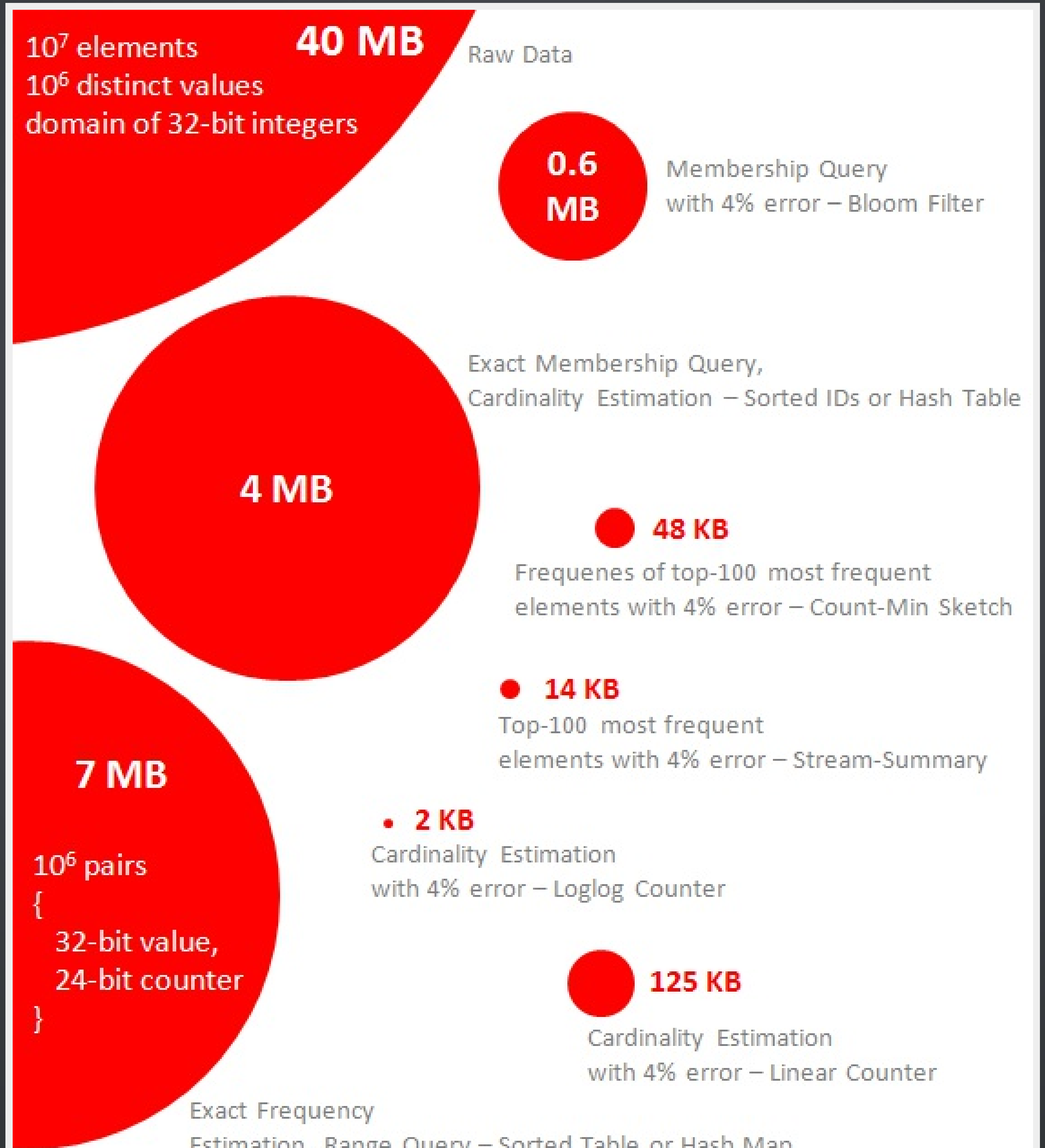
- Sensor data
- Image data

WHAT IS STREAM PROCESSING?

Examples?

- Sensor data
- Image data
- Internet and Web traffic

PROBABILISTIC DATA STRUCTURES FOR WEB ANALYTICS AND DATA MINING



WHAT IS STREAM PROCESSING?

- Data arrives in a stream or streams, if not processed immediately or stored, it is lost

WHAT IS STREAM PROCESSING?

- Data arrives in a stream or streams, if not processed immediately or stored, it is lost
- Most data streaming algorithms *summarize* the stream in some way

WHAT IS STREAM PROCESSING?

- Data arrives in a stream or streams, if not processed immediately or stored, it is lost
- Most data streaming algorithms *summarize* the stream in some way
- Most involve heavy use of *hashing*

WHAT IS STREAM PROCESSING?

- Data arrives in a stream or streams, if not processed immediately or stored, it is lost
- Most data streaming algorithms *summarize* the stream in some way
- Most involve heavy use of *hashing*
- Many involve use of *sketching*

A FEW PROBABILISTIC DATA STRUCTURES

A FEW PROBABILISTIC DATA STRUCTURES

- Set Membership

A FEW PROBABILISTIC DATA STRUCTURES

- Set Membership

  Bloom Filter

A FEW PROBABILISTIC DATA STRUCTURES

- Set Membership

  Bloom Filter

- Cardinality Estimation

A FEW PROBABILISTIC DATA STRUCTURES

- Set Membership

  Bloom Filter

- Cardinality Estimation

  LogLog, HyperLogLog

A FEW PROBABILISTIC DATA STRUCTURES

- Set Membership

  Bloom Filter

- Cardinality Estimation

  LogLog, HyperLogLog

- Frequency Estimation

A FEW PROBABILISTIC DATA STRUCTURES

- Set Membership

  Bloom Filter

- Cardinality Estimation

  LogLog, HyperLogLog

- Frequency Estimation

  Count Min Sketch, CountMean Min Sketch

A FEW PROBABILISTIC DATA STRUCTURES

- Set Membership

  Bloom Filter

- Cardinality Estimation

  LogLog, HyperLogLog

- Frequency Estimation

  Count Min Sketch, CountMean Min Sketch

- Heavy Hitters (Top-K)

A FEW PROBABILISTIC DATA STRUCTURES

- Set Membership

  Bloom Filter

- Cardinality Estimation

  LogLog, HyperLogLog

- Frequency Estimation

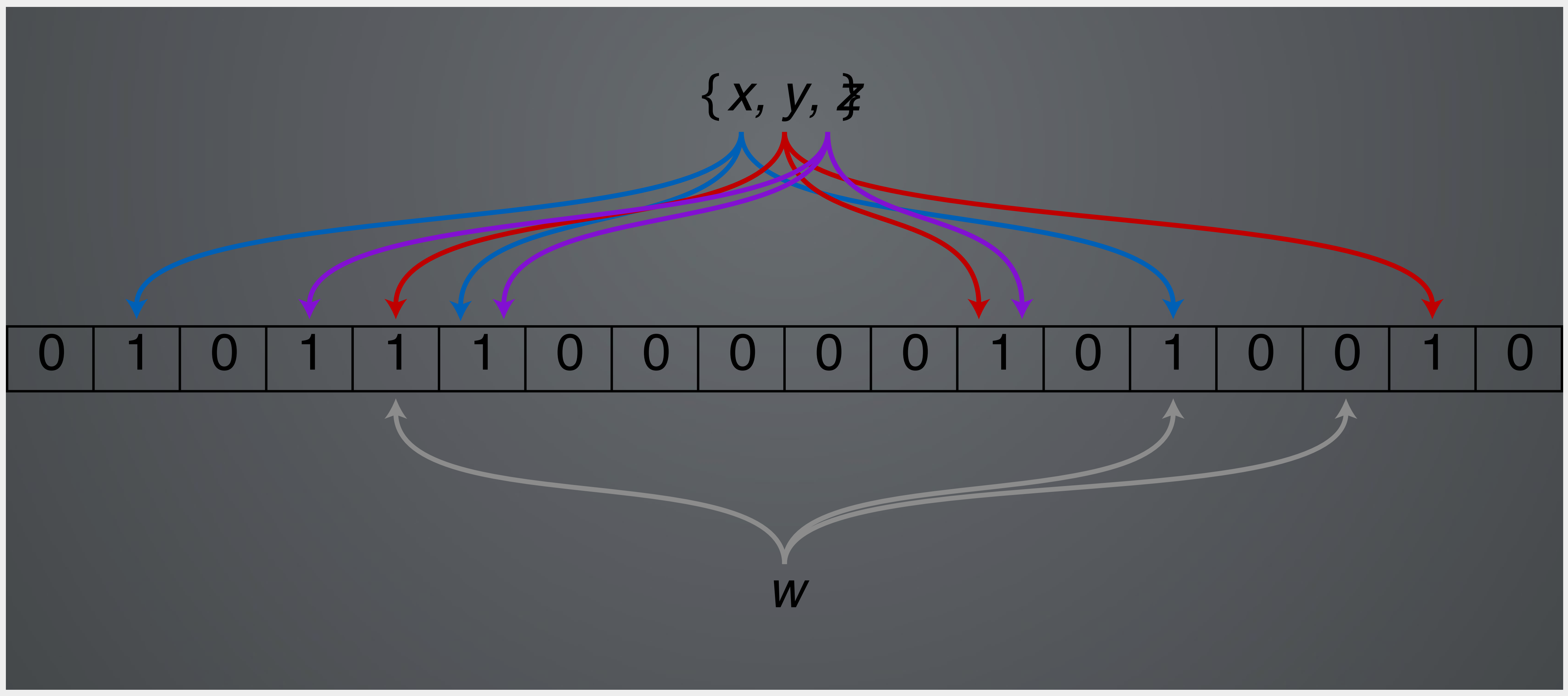
  Count Min Sketch, CountMean Min Sketch

- Heavy Hitters (Top-K)

  Count Min Sketch, Stream Summary

BLOOM FILTER

- Used to test whether an element is a member of a set
- False positive matches are possible, but false negatives are not



0	1	0	1	1	1	0	0	0	0	0	1	0	1	0	0	1	0

BLOOM FILTER

Algorithm

BLOOM FILTER

Algorithm

- Create empty Bloom filter is a bit array of m bits, all set to 0

BLOOM FILTER

Algorithm

- Create empty Bloom filter is a bit array of m bits, all set to 0
- Define k different hash functions

BLOOM FILTER

Algorithm

- Create empty Bloom filter is a bit array of m bits, all set to 0
- Define k different hash functions
- To add an element, feed it to each of the k hash functions to get k array positions

BLOOM FILTER

Algorithm

- Create empty Bloom filter is a bit array of m bits, all set to 0
- Define k different hash functions
- To add an element, feed it to each of the k hash functions to get k array positions
- Set the bits at all these positions to 1

BLOOM FILTER

Algorithm

- Create empty Bloom filter is a bit array of m bits, all set to 0
- Define k different hash functions
- To add an element, feed it to each of the k hash functions to get k array positions
- Set the bits at all these positions to 1
- To query, feed it to each of the k hash functions to get k array positions. If any of the bits at these positions is 0, the element is definitely not in the set

BLOOM FILTER

- Bloom Filter Demo

LOGLOG & HYPERLOGLOG

LOGLOG & HYPERLOGLOG

- Hash each element in the data set and represent as a binary string

LOGLOG & HYPERLOGLOG

- Hash each element in the data set and represent as a binary string
- Expect that about one half of strings will start with 1, one quarter will start with 01, and so on

LOGLOG & HYPERLOGLOG

- Hash each element in the data set and represent as a binary string
- Expect that about one half of strings will start with 1, one quarter will start with 01, and so on
- Denote the number of leading zeros as a rank

LOGLOG & HYPERLOGLOG

- Hash each element in the data set and represent as a binary string
- Expect that about one half of strings will start with 1, one quarter will start with 01, and so on
- Denote the number of leading zeros as a rank
- If the maximum number of leading zeros observed is n, an estimate for the number of distinct elements in the set is 2^{n}

LogLog & HyperLogLog

LogLog & HyperLogLog

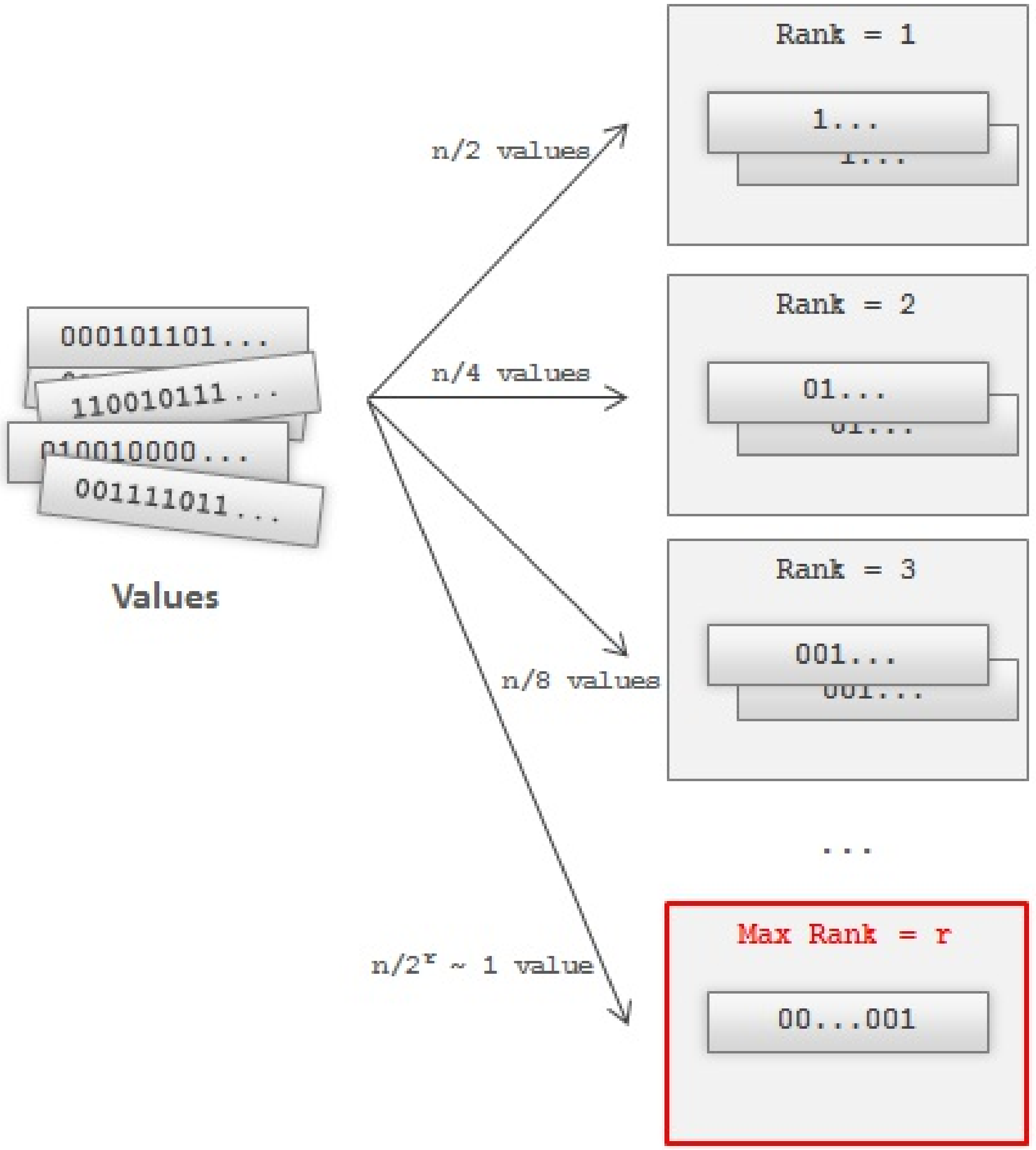
LogLog uses regular mean while HyperLogLog (HLL) uses harmonic mean to average the estimate cardinality calculated by different m buckets

LogLog & HyperLogLog

LogLog uses regular mean while HyperLogLog (HLL) uses harmonic mean to average the estimate cardinality calculated by different m buckets

HLL is able to estimate cardinalities of $> 10^9$ with a typical accuracy of 2%, using 1.5kB of memory

LOGLOG & HYPERLOGLOG

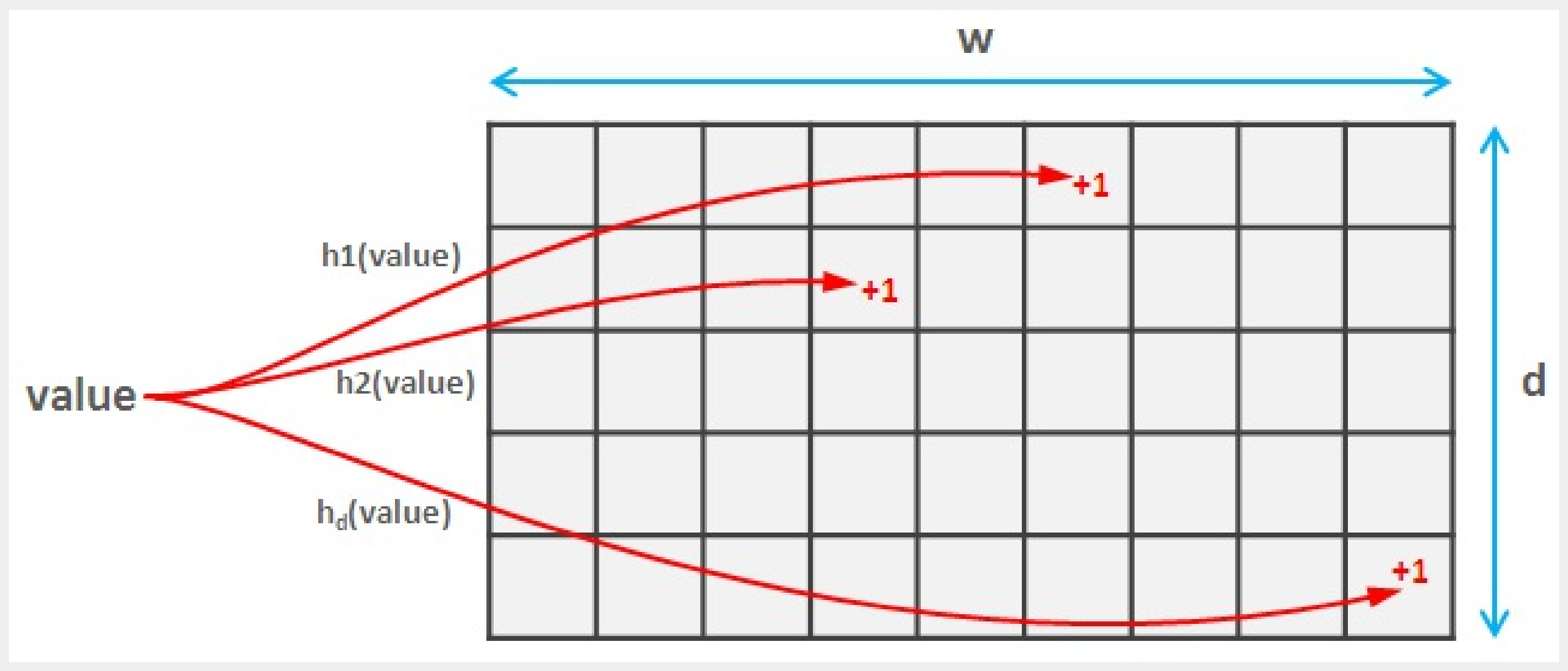


LogLogCounter
stores multiple
independent
measurements of r

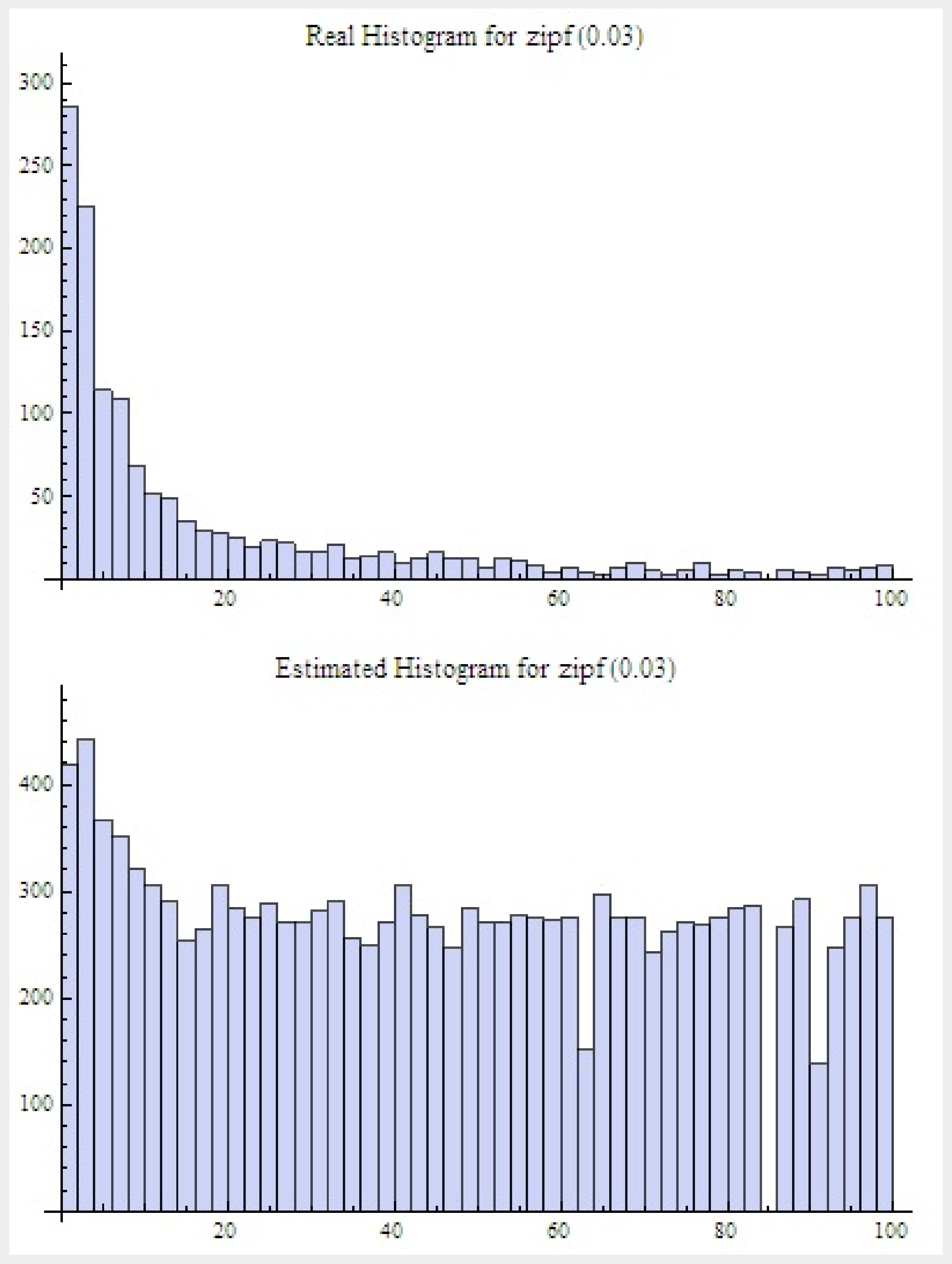
COUNT MIN SKETCH

$$\epsilon \leq \frac{2n}{w}$$

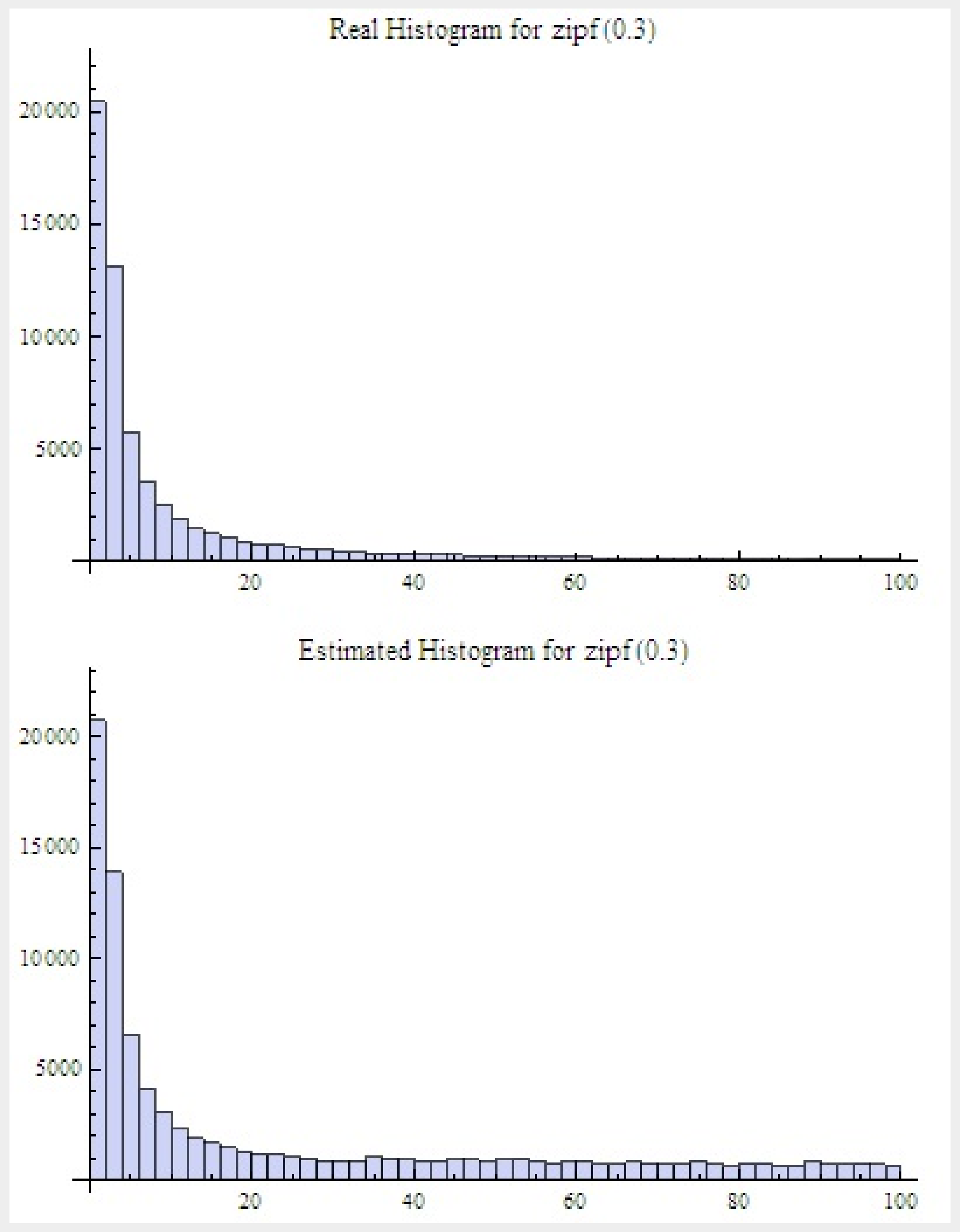
$$\delta = 1 - \left(\frac{1}{2}\right)^{d}$$



COUNT MIN SKETCH

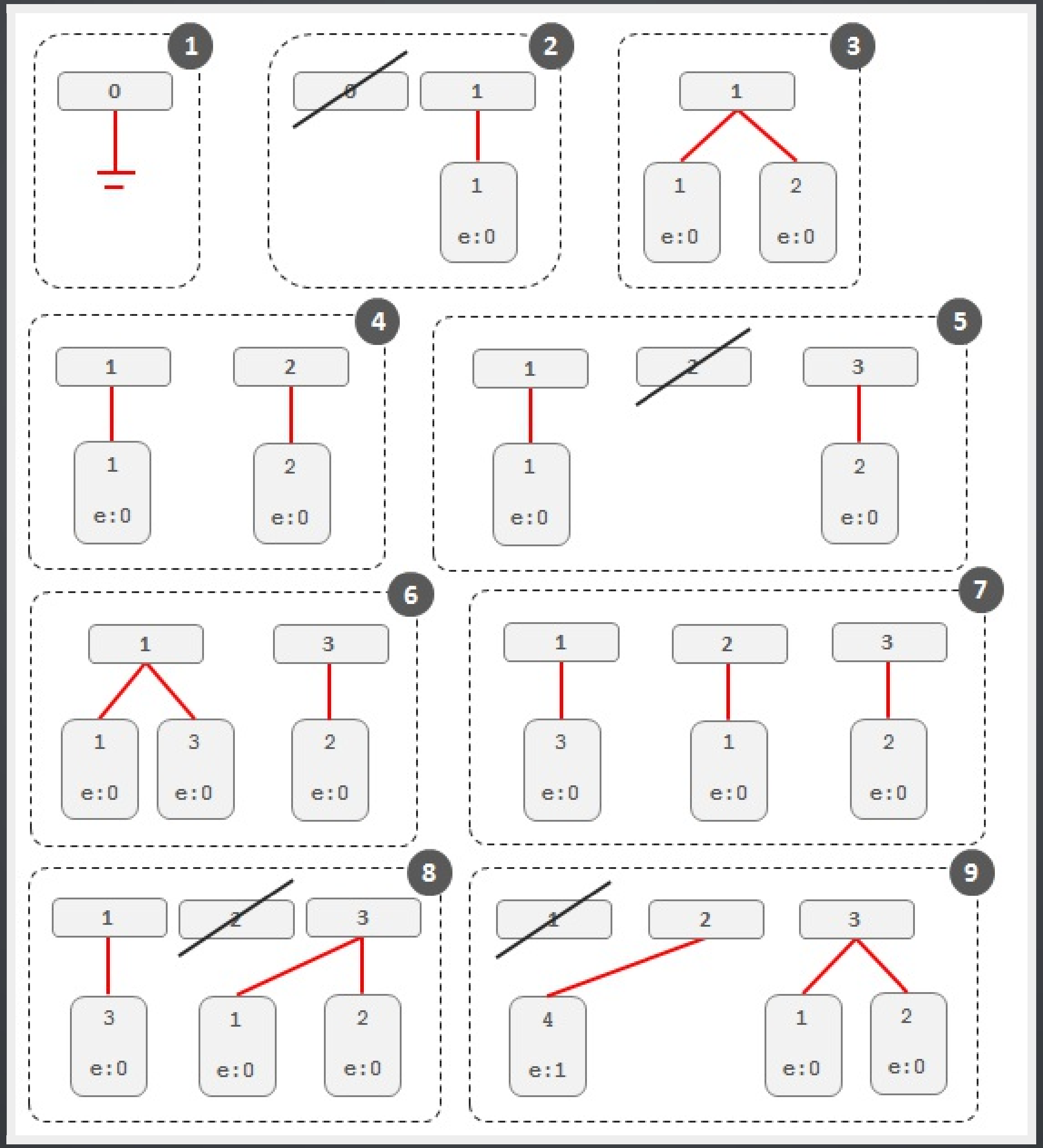


COUNT MIN SKETCH

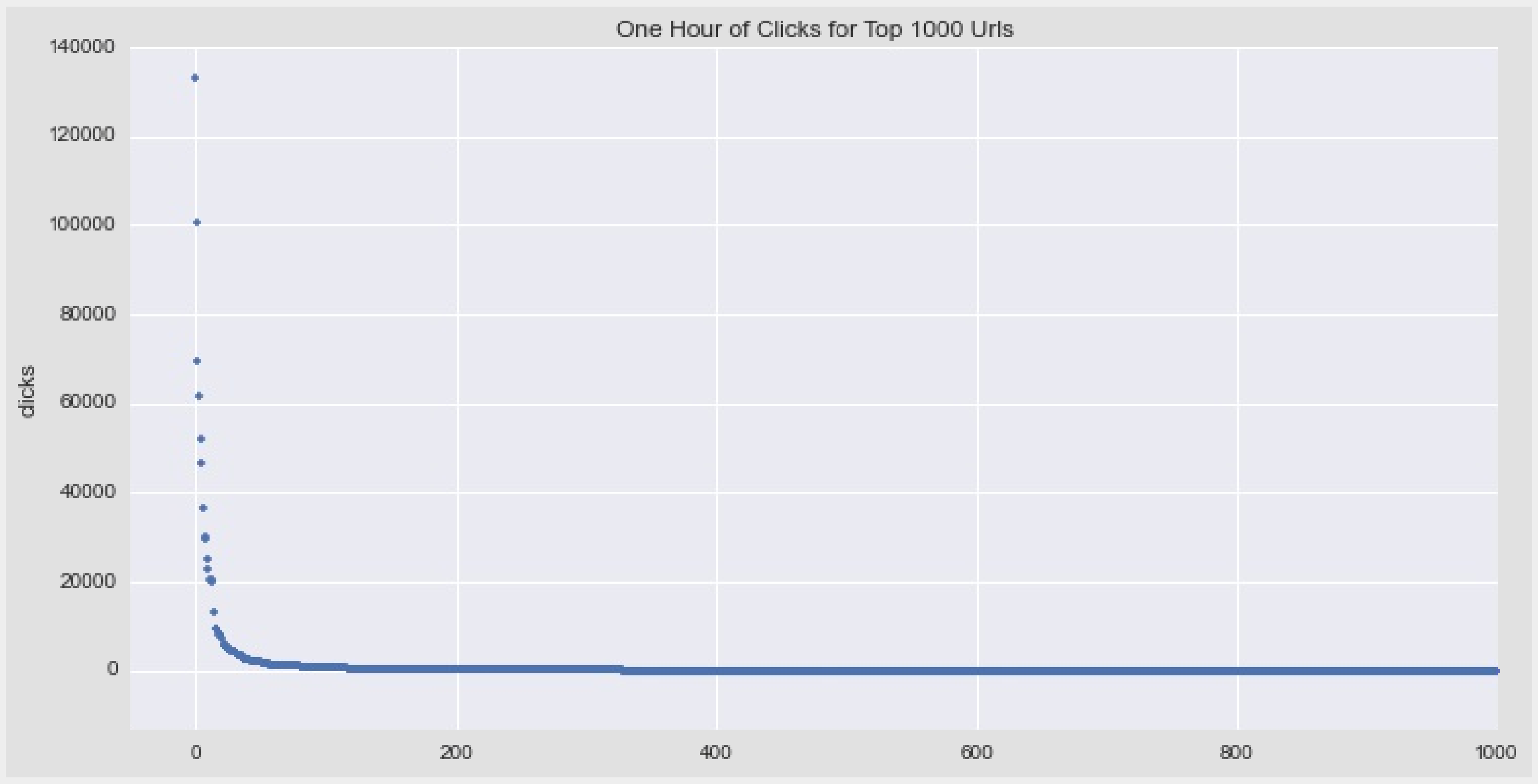


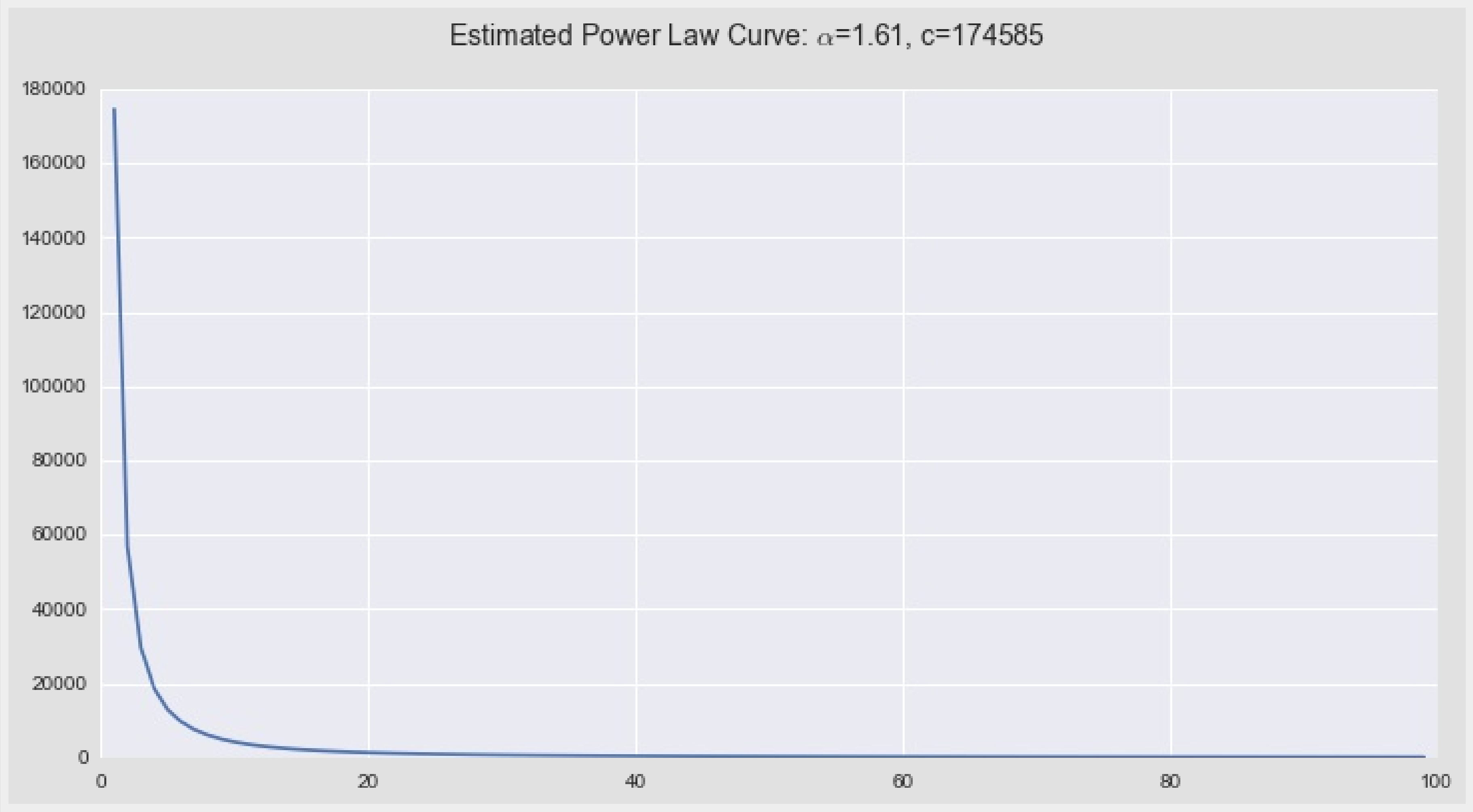
STREAM SUMMARY

```
input stream: {1,2,2,2,3,1,1,4}
```

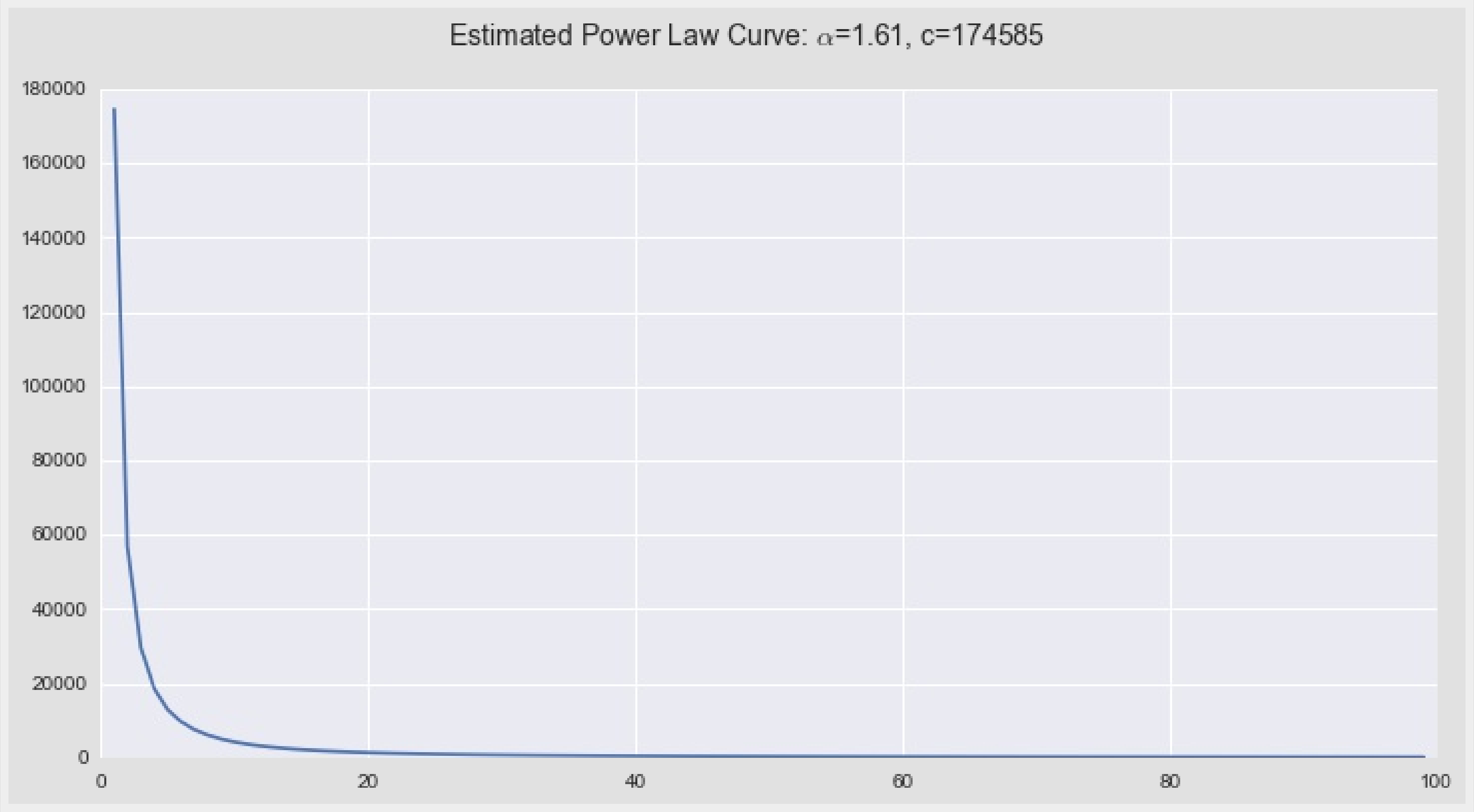


IMPLEMENTATION CONSIDERATIONS

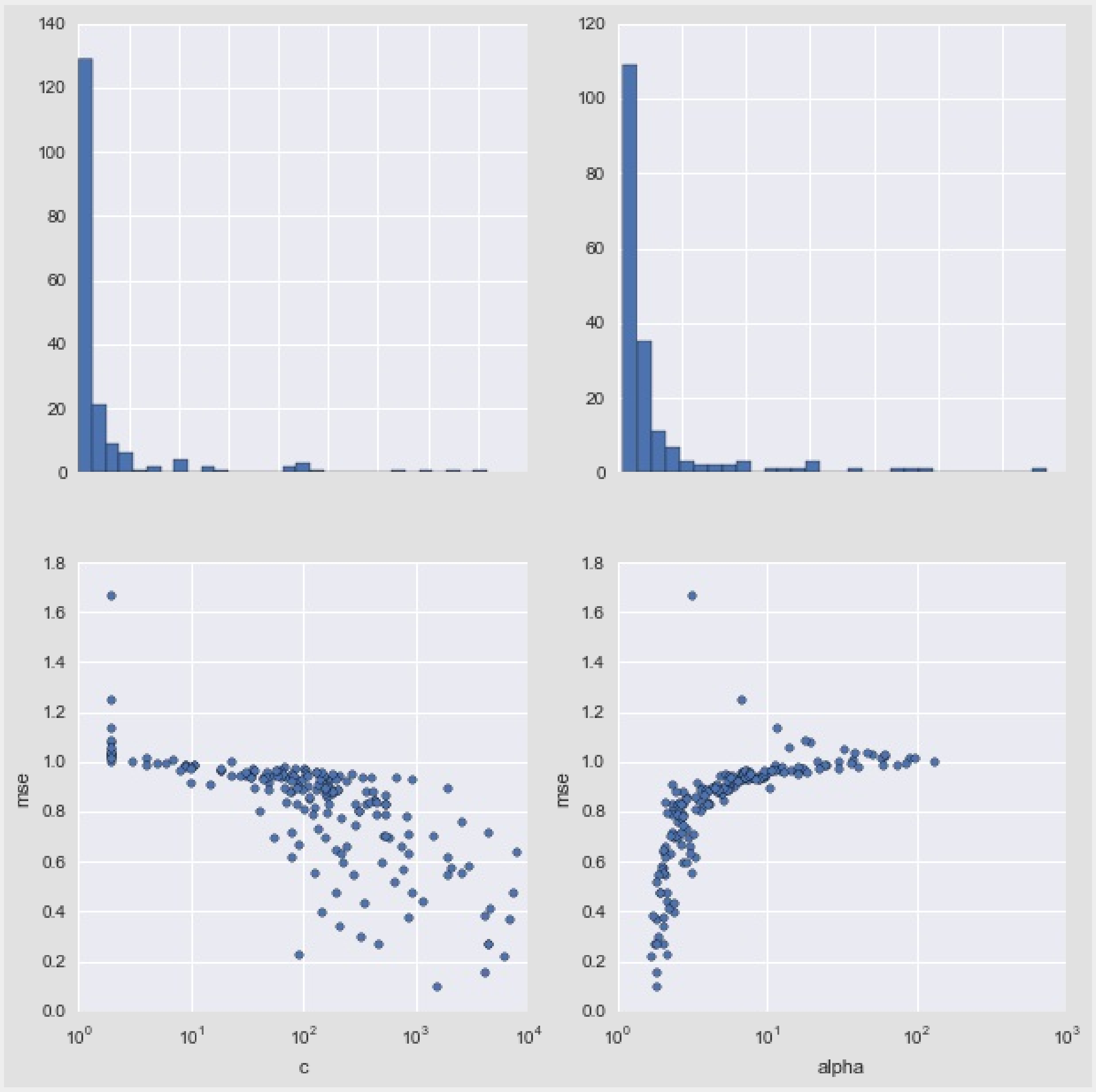
One Hour of Clicks for Top 1000 Urls
clicks
140000
120000
100000
80000
60000
40000
20000
0
0
200
400
600
800
1000



Estimated Power Law Curve: α=1.61, c=174585



Important that $\alpha > 1$


140
120
100
80
60
40
20
0
120
100
80
60
40
20
0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
mse
10^0
10^1
10^2
10^3
10^4
c
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
mse
10^0
10^1
10^2
10^3
alpha

FURTHER READING

- Probabilistic Data Structures for Web Analytics and Data Mining, Ilya Katsov
- Efficient Computation of Frequent and Top-K Elements in Data Streams, A. Metwally, D. Agrawal, A.E. Abbadi.
- HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm, P. Flayjolet, E.Fusy, O. Gandouet, F. Meunier.
- An Improved Data Stream Summary: The Count-Min Sketch and its Applications, . Cormode, S. Muthukrishnan.
- A Statistical Analysis of Probabilistic Counting Algorithms, P. Clifford, I. Cosma.
- Mining Massive Data Sets, Chapter 4Leskovec, Rajaraman, Ullman
- Stream-lib: Java library with implementations of many of