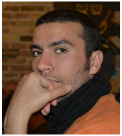
# Data Streams as Random Permutations: the Distinct Element Problem

Dedicated to the memory of Philippe Flajolet (1948-2011)

Conrado Martínez,
Univ. Politècnica de Catalunya, Barcelona, Spain

AofA, Montréal, June 2012

Joint work with:



A. Helmi

J. Lumbroso


A. Viola

# Introduction

- A data stream is a (very long) sequence

$$\mathcal{S} = s_1, s_2, s_3, \ldots, s_N$$

  of items $s_i$ drawn from some (large) domain $\mathcal{U}$, $s_i \in \mathcal{U}$
- The goal: to compute $y = y(\mathcal{S})$, but …

# Introduction

- A data stream is a (very long) sequence

$$\mathcal{S} = s_1, s_2, s_3, \ldots, s_N$$

  of items $s_i$ drawn from some (large) domain $\mathcal{U}$, $s_i \in \mathcal{U}$
- The goal: to compute $y = y(\mathcal{S})$, but ...

…there are limitations to our computational power:

- a single pass over the sequence
- very short time for computation on each item
- very small auxiliary memory: $M \ll N$; ideally $M = \Theta(1)$ or $M = \mathcal{O}(\log N)$
- no statistical hypothesis on the data

...there are limitations to our computational power:

- a single pass over the sequence
- very short time for computation on each item
- very small auxiliary memory: $M \ll N$; ideally $M = \Theta(1)$ or $M = \mathcal{O}(\log N)$
- no statistical hypothesis on the data

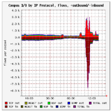
…there are limitations to our computational power:

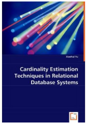
- a single pass over the sequence
- very short time for computation on each item
- very small auxiliary memory: $M \ll N$; ideally $M = \Theta(1)$ or $M = \mathcal{O}(\log N)$
- no statistical hypothesis on the data

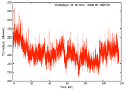
…there are limitations to our computational power:

- a single pass over the sequence
- very short time for computation on each item
- very small auxiliary memory: $M \ll N$; ideally $M = \Theta(1)$ or $M = \mathcal{O}(\log N)$
- no statistical hypothesis on the data

# Introduction





There are lots of applications for this data strem model:

- Network traffic analysis $\Rightarrow$ DoS/DDoS attacks, worms, ...
- Database query optimization
- Information retrieval $\Rightarrow$ similarity index
- Data mining
- And many more ...

We will often see $\mathcal{S}$ as a multiset

$$\{w_1 \circ f_1, \ldots, w_n \circ f_n\},$$

with

$$f_i = \text{frequency of the } i\text{th distinct element } w_i$$

# Introduction

Some typical problems:

- The cardinality of $\mathcal{S}$: card$(\mathcal{S}) = n \leqslant N \Leftarrow$ This paper
- Frequency moments $F_p = \sum_{1 \leqslant i \leqslant n} f_i^p$
  (N.B. $n = F_0, N = F_1$)
- The elements $w_i$ such that $f_i \geqslant k$ ($k$-elephants)
- The elements $w_i$ such that $f_i < k$ ($k$-mice)
- The elements $w_i$ such that $f_i \geqslant cN$, $0 < c < 1$ ($c$-icebergs)
- The $k$ most frequent elements
- …

Small auxiliary memory $\Rightarrow$

Exact solution too costly (or impossible) $\Rightarrow$

Randomized algorithms $\Rightarrow$

Estimation $\hat{\mathfrak{y}}$ of the quantity $\mathfrak{y}$

- The estimator $\hat{\mathfrak{y}}$ must be unbiased

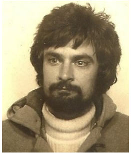
$$\mathsf{E}\left[\hat{\mathfrak{y}}\right] = \mathfrak{y}$$

- The estimator must be accurate (small standard error)

$$\mathsf{SE}\left[\hat{\mathfrak{y}}\right] := \frac{\sqrt{\mathsf{Var}\left[\hat{\mathfrak{y}}\right]}}{\mathsf{E}\left[\hat{\mathfrak{y}}\right]} < \epsilon,$$

e.g., $\epsilon = 0.01$ (1%)

# Probabilistic Counting



G.N. Martin

- Late in the 70s, G. Nigel N. Martin invents probabilistic counting, for database query optimization
- He detects systematic bias in his estimator, he tweaks the algorithm to correct the bias

# Probabilistic Counting

As I said over the phone, I started working on your algorithm when Kyu-Young Whang considered implementing it and wanted explanations/estimations. I find it simple, eleg
and amazingly powerful.

Ph. Flajolet

- When Flajolet learns about the algorithm, he contacts Martin and they team up to carry out a very detailed analysis giving the correcting factor and upper bounds for the standard error
- Their pioneering work (Flajolet & Martin, JCSS, 1985) introduces many of the ideas behind the most practical and successful cardinality estimators

# Estimating the cardinality

The first ingredient:

- Map each item $s_i$ to a value in $(0, 1)$ using a hash function* $h : \mathcal{U} \to (0, 1) \Rightarrow$ reproducible randomness
- The multiset $\mathcal{S}$ is mapped to a multiset

$$\mathcal{S}' = h(\mathcal{S}) = \{x_1 \circ f_1, \ldots, x_n \circ f_n\},$$

  with $x_i$ = hash$(w_i)$, $f_i$ = # of $x_i$'s
- The set of distinct elements $X = \{x_1, \ldots, x_n\}$ is a set of $n$ independent and uniformly distributed real numbers in $(0, 1)$

## Estimating the cardinality

The first ingredient:

- Map each item $s_i$ to a value in $(0,1)$ using a hash function* $h : \mathcal{U} \to (0,1) \Rightarrow$ reproducible randomness
- The multiset $\mathcal{S}$ is mapped to a multiset

$$\mathcal{S}' = h(\mathcal{S}) = \{x_1 \circ f_1, \ldots, x_n \circ f_n\},$$

  with $x_i = \text{hash}(w_i)$, $f_i =$ # of $x_i$'s
- The set of distinct elements $X = \{x_1, \ldots, x_n\}$ is a set of $n$ independent and uniformly distributed real numbers in $(0,1)$

*We disregard here *collisions*: if the hash values have enough bits the probability of collision can be neglected

# Probabilistic Counting

The second ingredient:

- Define some easily computable observable $R$ which is insensitive to repetitions, that is, it only depends on the underlying set of distinct elements:

$$R = R(\mathcal{S}) = R(X)$$

- Perform the probabilistic analysis of $R$ for a set $X$ of $n$ random real numbers. If

$$\mathrm{E}_n\left[R\right] = \varphi(n)$$

then it is reasonable to assume that the expected value of $\varphi^{-1}(R)$ will be close to $n$; we will need some correcting factor $\kappa$ to get an (asymptotically) unbiased estimator

$$\mathrm{E}_n\left[\kappa\varphi^{-1}(R)\right] = n + \text{l.o.t.}$$

# Probabilistic Counting

- For instance, in Flajolet & Martin's Probabilistic Counting the observable $R$ is the length of the longest prefix $0.0^{R-1}1$ such that all prefixes $0.0^{k}1$ appear among the hashed values, for $0 \leqslant k \leqslant R-1$
- $R$ is easy to compute and it does not depend on repetitions

$$E_n\left[R\right] \approx \log_2 n$$

and

$$E_n\left[\kappa 2^R\right] = n + o(n)$$

for

$$\kappa^{-1} = \frac{e^{\gamma}\sqrt{2}}{3} \prod_{k \geqslant 1} \left( \frac{(4k+1)(2k+1)}{2k(4k+3)} \right)^{(-1)^{\nu(k)}} \approx 0.77351\ldots$$

# Other estimators

- LogLog (Durand, Flajolet, 2003) and HyperLogLog (Flajolet, Fusy, Gandouet, Meunier, 2007) use bit patterns in the hash values to estime, like in Probabilistic Counting
- Order statistics (e.g., the $k$th smallest in the set of distinct hash values) have also been used to estimate cardinality: Bar-Yossef, Kumar & Sivakumar (2002); Bar-Yossef, Jayram, Kumar, Sivakumar & Trevisan (2002); Giroire (2005, 2009); Chassaing & Gérin (2006); Lumbroso (2010)

# Recordinality

- RECORDINALITY counts the number of records (more generally, $k$-records) in the sequence
- It depends in the underlying permutation of the first occurrences of distinct values, very different from the other estimators
- If we assume that the first occurrences of distinct values form a random permutation then no need for hash values!

## Recordinality

- $\sigma(i)$ is a record of the permutation $\sigma$ if $\sigma(i) > \sigma(j)$ for all $j < i$
- This notion is generalized to $k$-records: $\sigma(i)$ is a $k$-record if there are at most $k - 1$ elements $\sigma(j)$ larger than $\sigma(i)$ for $j < i$; in other words, $\sigma(i)$ is among the $k$ largest elements in $\sigma(1), \ldots, \sigma(i)$

# Recordinality

**procedure** RECORDINALITY($\mathcal{S}$)
    fill $T$ with the first $k$ distinct elements (hash values) of the stream $\mathcal{S}$
    $R \leftarrow k$
    **for all** $y \in \mathcal{S}$ **do**
        $x \leftarrow h(y)$
        **if** $x > \min(T) \wedge x \notin T$ **then**
            $R \leftarrow R + 1$; $T \leftarrow T \cup \{x\} \setminus \min(T)$
        **end if**
    **end for**
**end procedure**

Memory: $k$ hash values ($k \log n$ bits) + 1 counter ($\log\log n$ bits)

# Recordinality

## Theorem (Helmi, Martínez and Panholzer)

*Let $r_k$ denote the number of $k$-records in a permutation of size $n$. The exact distribution of $r_k$ is*

$$\textit{Prob}_n\{r_k = j\} = \begin{cases} [\![n = j]\!] & \textit{if } k > n, \\ k^{j-k}\dfrac{k!}{n!}\begin{bmatrix} n-k+1 \\ j-k+1 \end{bmatrix} & \textit{if } k \leqslant j \leqslant n \end{cases}$$

$\begin{bmatrix} n \\ j \end{bmatrix}$ *= signless Stirling numbers of the first kind;* $[\![P]\!] = 1$ *if* $P$ *true,* $= 0$ *otherwise*

# Recordinality

- The expected value of $r_k$ is $k \log(n/k) + \text{l.o.t.}$; it is reasonable then to assume that for

$$Z := k \exp(\phi \cdot r_k)$$

  we should have $\mathsf{E}_n[Z] \sim n$ for some suitable correcting factor $\phi$
- We can use the formula for $\mathsf{Prob}_n\{r_k = j\}$ to explictly compute $\mathsf{E}_n[Z]$ and to determine $\phi$, and then compute the standard error

# Recordinality

**Theorem**

*The* RECORDINALITY *estimator*

$$Z := k\left(1 + \frac{1}{k}\right)^{r_k - k + 1} - 1$$

*is an* *unbiased* *estimator of* $n$*:* $E_n[Z] = n$.

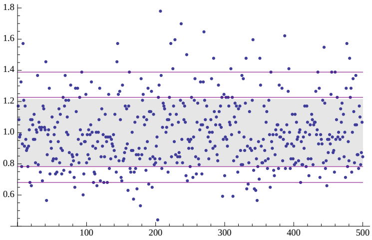
### Theorem

*The accuracy of* RECORDINALITY, *expressed in terms of standard error, asymptotically satisfies*

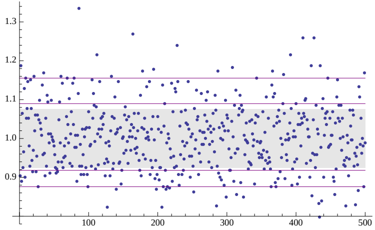
$$SE_n[Z] \sim \sqrt{\left(\frac{n}{ke}\right)^{\frac{1}{k}} - 1}$$

# Recordinality

For practical values of $n$, even for small $k$, the estimates may be significantly concentrated.
For instance, for $k = 10$, the estimates are within $\sigma$, $2\sigma$, $3\sigma$ of the exact count in respectively 91%, 96% and 99% of all cases.



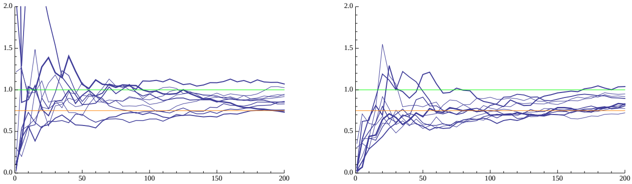

$k = 64$




$k = 256$

500 estimates of cardinality in Shakespare's *A Midsummer Night's Dream*; top and bottom lines (5%), centermost lines (70%); gray area (1 standard deviation)

# Other issues




Original texts    Randomly permuted texts

- RECORDINALITY does not depend on the hash values, only the relative ordering $\Rightarrow$ we can avoid using the hash function, provided the distinct elements appear (for the first time) in random order
- We can combine RECORDINALITY with any of the other $k$th order statistic estimators since they are independent; we can get **both** estimators with a single pass of the "scanning" algorithm

# Other issues

- The table of $k$th largest hash values gives us a random sample of $k$ distinct elements out of the $n \Rightarrow$ distinct sampling for free
- If we keep all distinct $k$-records, not just the $k$ largest distinct values, we have a random sample of expected size $k \log(n/k) \Rightarrow$ variable-size sampling!

# Concluding remarks

- First (?) application of combinatorics of random permutations to data stream algorithms
- Simple and elegant algorithms
- Nice combinatorics and mathematical analysis
- Many extensions to explore: sampling, sliding windows, similarity index, . . . . . .

Thanks a lot
for your attention!