Data Stream Analysis: a (new) triumph for Analytic Combinatorics

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

Conrado Martínez
Universitat Politècnica de Catalunya

ALEA in Europe Workshop, Vienna (Austria)
October 2017
Outline of the Course

Part 1: An Overview of Data Stream Analysis
Part 2: Intermezzo: A Crash Course on Analytic Combinatorics
Part 3: Case Study: Analysis of Recordinality
Part I

An Overview of Data Stream Analysis
A data stream is a (very long) sequence

\[ S = s_1, s_2, s_3, \ldots, s_N \]

of elements drawn from a (very large) domain \( \mathcal{U} (s_i \in \mathcal{U}) \)

The goal: to find \( y = y(S) \), but \ldots
Introduction

- A **data stream** is a (very long) sequence

  \[ S = s_1, s_2, s_3, \ldots, s_N \]

  of elements drawn from a (very large) domain \( \mathcal{U} \) (\( s_i \in \mathcal{U} \))

- The **goal**: to find \( y = y(S) \), but . . .
under rather stringent constraints (data stream model)

- a single pass over the data stream
- extremely short time spent on each single data item
- a limited amount $M$ of auxiliary memory, $M \ll N$; ideally $M = \Theta(1)$ or $M = \Theta(\log N)$
- no statistical hypothesis about the data
... under rather stringent constraints (data stream model)

- a single pass over the data stream
- extremely short time spent on each single data item
- a limited amount \( M \) of auxiliary memory, \( M \ll N \); ideally \( M = \Theta(1) \) or \( M = \Theta(\log N) \)
- no statistical hypothesis about the data
...under rather stringent constraints (data stream model)

- a single pass over the data stream
- extremely short time spent on each single data item
- a limited amount $M$ of auxiliary memory, $M \ll N$; ideally $M = \Theta(1)$ or $M = \Theta(\log N)$
- no statistical hypothesis about the data
Introduction

... under rather stringent constraints (data stream model)

- a single pass over the data stream
- extremely short time spent on each single data item
- a limited amount $M$ of auxiliary memory, $M \ll N$; ideally $M = \Theta(1)$ or $M = \Theta(\log N)$
- no statistical hypothesis about the data
There are a wide range of applications for the data stream model

- Network traffic analysis $\Rightarrow$ DoS/DDoS attacks, *worms*, ... 
- Database query optimization
- Information retrieval $\Rightarrow$ similarity index
- Data mining
- Recommendation systems
- and many more ...
There are a wide range of applications for the data stream model

- Network traffic analysis $\Rightarrow$ DoS/DDoS attacks, worms, ... 
- Database query optimization 
- Information retrieval $\Rightarrow$ similarity index 
- Data mining 
- Recommendation systems 
- and many more ...
There are a wide range of applications for the data stream model

- Network traffic analysis ⇒ DoS/DDoS attacks, worms, …
- Database query optimization
- Information retrieval ⇒ similarity index
- Data mining
- Recommendation systems
- and many more …
There are a wide range of applications for the data stream model

- Network traffic analysis ⇒ DoS/DDoS attacks, *worms*, …
- Database query optimization
- Information retrieval ⇒ similarity index
- Data mining
  - Recommendation systems
  - and many more …
There are a wide range of applications for the data stream model:

- Network traffic analysis $\Rightarrow$ DoS/DDoS attacks, *worms*, ... 
- Database query optimization
- Information retrieval $\Rightarrow$ similarity index
- Data mining
- Recommendation systems
- and many more ...
There are a wide range of applications for the data stream model:

- Network traffic analysis ⇒ DoS/DDoS attacks, worms, ... 
- Database query optimization 
- Information retrieval ⇒ similarity index 
- Data mining 
- Recommendation systems 
- and many more ...
We’ll look at $S$ as a multiset $\{z_1 \circ f_1, \ldots, z_n \circ f_n\}$, where

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

Some problems in data stream analysis:

- **Number of distinct elements**: $\text{card}(S) = n \leq N$
- Frequency moments $F_p = \sum_{1 \leq i \leq n} f_i^p$
  (N.B. $n = F_0, N = F_1$)
- (Number of) Elements $z_i$ such that $f_i \geq k$ (*k-elephants*)
- (Number of) Elements $z_i$ such that $f_i < k$ (*k-mice*)
- (Number of) Elements $z_i$ such that $f_i \geq cN$, $0 < c < 1$ (*c-icebergs*)
- The $k$ most frequent elements (top-$k$ elements)
- ...
Introduction

We’ll look at $S$ as a multiset $\{z_1 \circ f_1, \ldots, z_n \circ f_n\}$, where

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

Some problems in data stream analysis:

- **Number of distinct elements**: $\text{card}(S) = n \leq N$
- **Frequency moments** $F_p = \sum_{1 \leq i \leq n} f_i^p$
  
  (N.B. $n = F_0$, $N = F_1$)
- (Number of) Elements $z_i$ such that $f_i \geq k$ (k-elephants)
- (Number of) Elements $z_i$ such that $f_i < k$ (k-mice)
- (Number of) Elements $z_i$ such that $f_i \geq cN$, $0 < c < 1$ (c-icebergs)
- The $k$ most frequent elements (top-$k$ elements)

...
Introduction

We’ll look at $S$ as a multiset $\{z_1 \circ f_1, \ldots, z_n \circ f_n\}$, where

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

Some problems in data stream analysis:

- **Number of distinct elements**: $\text{card}(S) = n \leq N$
- Frequency moments $F_p = \sum_{1 \leq i \leq n} f_i^p$
  (N.B. $n = F_0, N = F_1$)
- (Number of) Elements $z_i$ such that $f_i \geq k$ (**k-elephants**)
- (Number of) Elements $z_i$ such that $f_i < k$ (**k-mice**)
- (Number of) Elements $z_i$ such that $f_i \geq cN, 0 < c < 1$ (**c-icebergs**)
- The $k$ most frequent elements (top-$k$ elements)
- …
We’ll look at $S$ as a multiset $\{z_1 \circ f_1, \ldots, z_n \circ f_n\}$, where

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

Some problems in data stream analysis:

- **Number of distinct elements**: $\text{card}(S) = n \leq N$
- Frequency moments $F_p = \sum_{1 \leq i \leq n} f_i^p$
  
  (N.B. $n = F_0$, $N = F_1$)
- (Number of) Elements $z_i$ such that $f_i \geq k$ ($k$-elephants)
- (Number of) Elements $z_i$ such that $f_i < k$ ($k$-mice)
- (Number of) Elements $z_i$ such that $f_i \geq cN$, $0 < c < 1$ ($c$-icebergs)
- The $k$ most frequent elements (top-$k$ elements)
- ...
Introduction

We’ll look at $S$ as a multiset $\{z_1 \circ f_1, \ldots, z_n \circ f_n\}$, where

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

Some problems in data stream analysis:

- **Number of distinct elements**: $\text{card}(S) = n \leq N$
- Frequency moments $F_p = \sum_{1 \leq i \leq n} f_i^p$
  
  (N.B. $n = F_0, N = F_1$)
- (Number of) Elements $z_i$ such that $f_i \geq k$ (**k-elephants**)
- (Number of) Elements $z_i$ such that $f_i < k$ (**k-mice**)
- (Number of) Elements $z_i$ such that $f_i \geq cN$, $0 < c < 1$ (**c-icebergs**)
- The $k$ most frequent elements (top-$k$ elements)
- ...
Introduction

We’ll look at $S$ as a multiset $\{z_1 \circ f_1, \ldots, z_n \circ f_n\}$, where

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

Some problems in data stream analysis:

- **Number of distinct elements**: $\text{card}(S) = n \leq N$
- Frequency moments $F_p = \sum_{1 \leq i \leq n} f_i^p$
  (N.B. $n = F_0$, $N = F_1$)
- (Number of) Elements $z_i$ such that $f_i \geq k$ (k-elephants)
- (Number of) Elements $z_i$ such that $f_i < k$ (k-mice)
- (Number of) Elements $z_i$ such that $f_i \geq cN$, $0 < c < 1$ (c-icebergs)
- The $k$ most frequent elements (top-$k$ elements)

...
We’ll look at $S$ as a multiset $\{z_1 \circ f_1, \ldots, z_n \circ f_n\}$, where

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

Some problems in data stream analysis:

- **Number of distinct elements**: $\text{card}(S) = n \leq N$
- Frequency moments $F_p = \sum_{1 \leq i \leq n} f_i^p$
  
  (N.B. $n = F_0$, $N = F_1$)
- (Number of) Elements $z_i$ such that $f_i \geq k$ (*k-elephants*)
- (Number of) Elements $z_i$ such that $f_i < k$ (*k-mice*)
- (Number of) Elements $z_i$ such that $f_i \geq cN$, $0 < c < 1$ (*c-icebergs*)
- The $k$ most frequent elements (top-$k$ elements)
- ...
Very limited available memory $\Rightarrow$ exact solution too costly or unfeasible
$\Rightarrow$ Randomized algorithms $\Rightarrow$ estimation $\hat{y}$ of the quantity of interest $y$

- $\hat{y}$ must be an unbiased estimator

$$\mathbb{E} [\hat{y}] = y$$

- The estimator must have a small standard error

$$\text{SE} [\hat{y}] := \frac{\sqrt{\text{Var} [\hat{y}]} }{\mathbb{E} [\hat{y}]} < \epsilon,$$

  e.g., $\epsilon = 0.01$ (1%)
Very limited available memory ⇒ exact solution too costly or unfeasible
⇒ Randomized algorithms ⇒ estimation \( \hat{y} \) of the quantity of interest \( y \)

- \( \hat{y} \) must be an unbiased estimator

\[
\mathbb{E}[\hat{y}] = y
\]

- The estimator must have a small standard error

\[
\text{SE}[\hat{y}] := \frac{\sqrt{\text{Var}[\hat{y}]} }{ \mathbb{E}[\hat{y}] } < \epsilon,
\]

E.g., \( \epsilon = 0.01 \) (1%)
In late 70s G. Nigel N. Martin invents probabilistic counting to optimize database query performance.

To correct the bias that he systematically found in his experiments, he introduced a “fudge” factor in the estimator.
Probabilistic Counting

When Flajolet learnt about the algorithm, he put it on a solid scientific ground, with a detailed mathematical analysis which delivered the exact value of the correction factor and a tight upper bound on the standard error.

As I said over the phone, I started working on your algorithm when Kye-Young Whang considered implementing it and wanted explanations estimations. I find it simple, elegant and amazing powerful.
Probabilistic Counting

- **First idea:** every element is hashed to a real value in $(0, 1)$
  ⇒ reproducible randomness

- The multiset $S$ is mapped by the hash function
  $h : \mathcal{U} \rightarrow (0, 1)$ to a multiset

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

  with $x_i = \text{hash}(z_i)$, $f_i = \# \text{ de } z_i$’s

- The set of distinct elements $X = \{x_1, \ldots, x_n\}$ is a set of $n$
  random numbers, independent and uniformly drawn from $(0, 1)$
Probabilistic Counting

- **First idea:** every element is hashed to a real value in $(0, 1) \Rightarrow$ reproducibile randomness

- The multiset $S$ is mapped by the hash function $h : \mathcal{U} \rightarrow (0, 1)$ to a multiset

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

  with $x_i = \text{hash}(z_i)$, $f_i = \# \text{ de } z_i$'s

- The set of distinct elements $X = \{ x_1, \ldots, x_n \}$ is a set of $n$ random numbers, independent and uniformly drawn from $(0, 1)$

*We’ll neglect the probability of collisions, i.e., $h(x_i) = h(x_j)$ for some $x_i \neq x_j$; this is reasonable if $h(x)$ has enough bits*
Probabilistic Counting

- **First idea**: every element is hashed to a real value in (0, 1) ⇒ reproducible randomness

- The multiset $S$ is mapped by the hash function* $h: \mathcal{U} \to (0, 1)$ to a multiset

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

  with $x_i = \text{hash}(z_i)$, $f_i = \# \text{ de } z_i$’s

- The set of distinct elements $X = \{x_1, \ldots, x_n\}$ is a set of $n$ random numbers, independent and uniformly drawn from (0, 1)

*We’ll neglect the probability of collisions, i.e., $h(x_i) = h(x_j)$ for some $x_i \neq x_j$; this is reasonable if $h(x)$ has enough bits
Flajolet & Martin (JCSS, 1985) proposed to find, among the set of hash values, the length of the largest prefix (in binary) $0.0^{R-1}1\ldots$ such that all shorter prefixes with the same pattern $0.0^{p-1}1\ldots$, $p \leq R$, also appear.

The value $R$ is an observable which can be easily be computed using a small auxiliary memory and it is insensitive to repetitions $←$ the observable is a function of $X$, not of the $f_i$’s.
Probabilistic Counting

- For a set of $n$ random numbers in $(0, 1) \rightarrow$
  \[ E[R] \approx \log_2 n \]
- However $E[2^R] \not\approx n$, there is a significant bias
Probabilistic Counting

For a set of \( n \) random numbers in \((0, 1)\) →

\[ \mathbb{E}[R] \approx \log_2 n \]

However \( \mathbb{E}[2^R] \not\sim n \), there is a significant bias
Probabilistic Counting

procedure \textsc{probabilisticCounting}(S)
    bmap ← ⟨0, 0, . . . , 0⟩
    for $s \in S$ do
        $y$ ← hash($s$)
        $p$ ← length of the largest prefix $0.0^{p-1}1 \ldots$ in $y$
        bmap[$p$] ← 1
    end for
    $R$ ← largest $p$ such that bmap[i] = 1 for all $0 \leq i \leq p$
    $\triangleright \phi$ is the correction factor
    return $Z := \phi \cdot 2^R$
end procedure

A very precise mathematical analysis gives:

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

$\Rightarrow \mathbb{E} [\phi \cdot 2^R] = n$
The standard error of $Z := \phi \cdot 2^R$, despite constant, is too large: $\text{SE}[Z] > 1$

- Second idea: repeat several times to reduce variance and improve precision
- Problem: using $m$ hash functions to generate $m$ streams is too costly and it’s very difficult to guarantee independence between the hash values
Stochastic averaging

- The standard error of $Z := \phi \cdot 2^R$, despite constant, is too large: $\text{SE}[Z] > 1$
- **Second** idea: repeat several times to reduce variance and improve precision
- Problem: using $m$ hash functions to generate $m$ streams is too costly and it’s very difficult to guarantee independence between the hash values
The standard error of $Z := \phi \cdot 2^R$, despite constant, is too large: $\text{SE}[Z] > 1$

Second idea: repeat several times to reduce variance and improve precision

Problem: using $m$ hash functions to generate $m$ streams is too costly and it’s very difficult to guarantee independence between the hash values
Stochastic averaging

- Use the first $\log_2 m$ bits of each hash value to “redirect” it (the remaining bits) to one of the $m$ substreams → stochastic averaging
- Obtain $m$ observables $R_1, R_2, \ldots, R_m$, one from each substream, and compute a mean value $\bar{R}$
- Each $R_i$ gives an estimation for the cardinality of the $i$-th substream, namely, $R_i$ estimates $n/m$
Stochastic averaging

- Use the first $\log_2 m$ bits of each hash value to “redirect” it (the remaining bits) to one of the $m$ substreams → stochastic averaging
- Obtain $m$ observables $R_1, R_2, \ldots, R_m$, one from each substream, and compute a mean value $\bar{R}$
- Each $R_i$ gives an estimation for the cardinality of the $i$-th substream, namely, $R_i$ estimates $n/m$
Use the first \( \log_2 m \) bits of each hash value to “redirect” it (the remaining bits) to one of the \( m \) substreams → stochastic averaging

Obtain \( m \) observables \( R_1, R_2, \ldots, R_m \), one from each substream, and compute a mean value \( \bar{R} \)

Each \( R_i \) gives an estimation for the cardinality of the \( i \)-th substream, namely, \( R_i \) estimates \( n/m \)
Stochastic averaging

There are many different options to compute an estimator from the $m$ observables

- **Sum of estimators:**
  \[ Z_1 := \phi_1 (2^{R_1} + \ldots + 2^{R_m}) \]

- **Arithmetic mean of observables** (as proposed by Flajolet & Martin):
  \[ Z_2 := m \cdot \phi_2 \cdot 2^{\frac{1}{m} \sum_{1 \leq i \leq m} R_i} \]
Stochastic averaging

- Harmonic mean (keep tuned):

\[ Z_3 := \phi_3 \cdot \frac{m^2}{2^{-R_1} + 2^{-R_2} + \ldots + 2^{-R_m}} \]

Since \(2^{-R_i} \approx m/n\), the second factor gives \(\approx m^2/(m^2/n) = n\).
**Stochastic averaging**

- All the strategies above yield a standard error of the form

\[
\frac{c}{\sqrt{m}} + \text{l.o.t.}
\]

Larger memory ⇒ improved precision!

- In *probabilistic counting* the authors used the arithmetic mean of observables

\[
\text{SE } [Z_{\text{ProbCount}}] \approx \frac{0.78}{\sqrt{m}}
\]
Stochastic averaging

- All the strategies above yield a standard error of the form
  \[ \frac{c}{\sqrt{m}} + \text{l.o.t.} \]

  \text{Larger memory } \Rightarrow \text{ improved precision!}

- In \textit{probabilistic counting} the authors used the arithmetic mean of observables

  \[ \text{SE} [Z_{\text{ProbCount}}] \approx \frac{0.78}{\sqrt{m}} \]
Durand & Flajolet (2003) realized that the bitmaps ($\Theta(\log n)$ bits) used by Probabilistic Counting can be avoided and propose as observable the largest $R$ such that the pattern $0.0^{R-1}1$ appears.

The new observable is similar to that of Probabilistic Counting but not equal: $R(\text{LogLog}) \geq R(\text{ProbCount})$.

Example

Observed patterns: $0.1101\ldots$, $0.010\ldots$, $0.0011\ldots$, $0.00001\ldots$

$R(\text{LogLog}) = 5$, $R(\text{ProbCount}) = 3$. 
Durand & Flajolet (2003) realized that the bitmaps ($\Theta(\log n)$ bits) used by \textit{Probabilistic Counting} can be avoided and propose as observable the largest $R$ such that the pattern $0.0^{R-1}1$ appears.

The new observable is similar to that of \textit{Probabilistic Counting} but not equal: $R(\text{LogLog}) \geq R(\text{ProbCount})$.

\textbf{Example}

Observed patterns: 0.1101\ldots, 0.010\ldots, 0.0011\ldots, 0.00001\ldots

$R(\text{LogLog}) = 5$, $R(\text{ProbCount}) = 3$
Durand & Flajolet (2003) realized that the bitmaps ($\Theta(\log n)$ bits) used by Probabilistic Counting can be avoided and propose as observable the largest $R$ such that the pattern $0.0^{R-1}1$ appears.

The new observable is similar to that of Probabilistic Counting but not equal: $R(\text{LogLog}) \geq R(\text{ProbCount})$.

Example

Observed patterns: 0.1101..., 0.010..., 0.0011..., 0.00001...

$R(\text{LogLog}) = 5, \quad R(\text{ProbCount}) = 3$
The new observable is simpler to obtain: keep updated the largest $R$ seen so far: $R := \max\{R, p\} \Rightarrow$ only $\Theta(\log \log n)$ bits needed, since $E[R] = \Theta(\log n)$!

We have $E[R] \sim \log_2 n$, but $E[2^R] = +\infty$, stochastic averaging comes to rescue!

For LogLog, Durand & Flajolet propose

$$Z_{\text{LogLog}} := \alpha_m \cdot m \cdot 2^{\frac{1}{m}} \sum_{1 \leq i \leq m} R_i$$
LogLog & HyperLogLog

- The new observable is simpler to obtain: keep updated the largest $R$ seen so far: $R := \max\{R, p\} \Rightarrow$ only $\Theta(\log \log n)$ bits needed, since $E[R] = \Theta(\log n)$!
- We have $E[R] \sim \log_2 n$, but $E[2^R] = +\infty$, stochastic averaging comes to rescue!
- For LogLog, Durand & Flajolet propose

$$Z_{LogLog} := \alpha_m \cdot m \cdot 2^\frac{1}{m} \sum_{1 \leq i \leq m} R_i$$
LogLog & HyperLogLog

- The new observable is simpler to obtain: keep updated the largest \( R \) seen so far: \( R := \max\{R, p\} \Rightarrow \text{only } \Theta(\log \log n) \) bits needed, since \( E[R] = \Theta(\log n)! \)
- We have \( E[R] \sim \log_2 n \), but \( E[2^R] = +\infty \), stochastic averaging comes to rescue!
- For LogLog, Durand & Flajolet propose

\[
Z_{\text{LogLog}} := \alpha_m \cdot m \cdot 2^{\frac{1}{m}} \sum_{1 \leq i \leq m} R_i
\]
The mathematical analysis gives for the correcting factor

\[ \alpha_m = \left( \Gamma\left(-1/m\right) \frac{1 - 2^{1/m}}{\ln 2} \right)^{-m} \]

that guarantees that \( E[Z] = n + l.o.t. \) (asymptotically unbiased) and the standard error is

\[ \text{SE}[Z_{\text{LogLog}}] \approx \frac{1.30}{\sqrt{m}} \]

Only \( m \) counters of size \( \log_2 \log_2(n/m) \) bits needed:

Ex.: \( m = 2048 = 2^{11} \) counters, 5 bits each (about 1 Kbyte in total), are enough to give precise cardinality estimations for \( n \) up to \( 2^{27} \approx 10^8 \), with an standard error less than 4%
The mathematical analysis gives for the correcting factor 

\[ \alpha_m = \left( \Gamma\left(-\frac{1}{m}\right) \frac{1 - 2^{1/m}}{\ln 2} \right)^{-m} \]

that guarantees that \( E[Z] = n + l.o.t. \) (asymptotically unbiased) and the standard error is 

\[ \text{SE}\left[Z_{\text{LogLog}}\right] \approx \frac{1.30}{\sqrt{m}} \]

Only \( m \) counters of size \( \log_2 \log_2(n/m) \) bits needed: 
Ex.: \( m = 2048 = 2^{11} \) counters, 5 bits each (about 1 Kbyte in total), are enough to give precise cardinality estimations for \( n \) up to \( 2^{27} \approx 10^8 \), with an standard error less than 4%
Flajolet, Fusy, Gandouet & Meunier conceived in 2007 the best algorithm known (cif. PF’s keynote speech in ITC Paris 2009)

Briefly: HyperLogLog combine the LogLog observables $R_i$ using the harmonic mean instead of the arithmetic mean

$$SE \left[ Z_{\text{HyperLogLog}} \right] \approx \frac{1.03}{\sqrt{m}}$$
LogLog & HyperLogLog

Flajolet, Fusy, Gandouet & Meunier conceived in 2007 the best algorithm known (cif. PF’s keynote speech in ITC Paris 2009)

Briefly: HyperLogLog combine the LogLog observables \( R_i \) using the harmonic mean instead of the arithmetic mean

\[
SE \left[ Z_{\text{HyperLogLog}} \right] \approx \frac{1.03}{\sqrt{m}}
\]
The idea of HyperLogLog stems from the analytical study of Chassaing & Gérin (2006) to show the optimal way to combine observables, but in their study the observables were the $k$-th order statistics of each substream. They proved that the optimal way to combine them is to use the harmonic mean.
The idea of HyperLogLog stems from the analytical study of Chassaing & Gérin (2006) to show the **optimal** way to combine observables, but in their study the observables were the $k$-th order statistics of each substream.

They proved that the optimal way to combine them is to use the harmonic mean.
Bar-Yossef, Kumar & Sivakumar (2002); Bar-Yossef, Jayram, Kumar, Sivakumar & Trevisan (2002) have proposed to use the $k$-th order statistic $X_{(k)}$ to estimate cardinality (KMV algorithm); for a set of $n$ random numbers, independent and uniformly distributed in $(0, 1)$

$$E[X_k] = \frac{k}{n + 1}$$

Giroire (2005, 2009) also proposes several estimators combining order statistics via stochastic averaging
Bar-Yossef, Kumar & Sivakumar (2002); Bar-Yossef, Jayram, Kumar, Sivakumar & Trevisan (2002) have proposed to use the $k$-th order statistic $X_{(k)}$ to estimate cardinality (KMV algorithm); for a set of $n$ random numbers, independent and uniformly distributed in $(0, 1)$

$$E[X_k] = \frac{k}{n + 1}$$

Giroire (2005, 2009) also proposes several estimators combining order statistics via *stochastic averaging*
The minimum of the set \( k = 1 \) does not allow a feasible estimator, but again stochastic averaging comes to rescue.

Lumbroso uses the mean of \( m \) minima, one for each substream,

\[
Z_{\text{MinCount}} := \frac{m(m - 1)}{M_1 + \ldots + M_m},
\]

where \( M_i \) is the minimum of the \( i \)-th substream.
Order Statistics

J. Lumbroso

- The minimum of the set \( k = 1 \) does not allow a feasible estimator, but again *stochastic averaging* comes to rescue.

- Lumbroso uses the mean of \( m \) minima, one for each substream.

\[
Z_{\text{MinCount}} := \frac{m(m - 1)}{M_1 + \ldots + M_m},
\]

where \( M_i \) is the minimum of the \( i \)-th substream.
• MinCount is an unbiased estimator with standard error 
  \( \frac{1}{\sqrt{m}} - 2 \)

• Lumbroso also succeeds to compute the probability distribution of \( Z_{\text{MinCount}} \) and the small corrections needed to estimate small cardinalities (to few elements hashing to one particular substream)
MinCount is an unbiased estimator with standard error $1/\sqrt{m-2}$.

Lumbroso also succeeds to compute the probability distribution of $Z_{\text{MinCount}}$ and the small corrections needed to estimate small cardinalities (to few elements hashing to one particular substream).
**Recordinality**

Recordinality (Helmi, Lumbroso, M., Viola, 2012) is a relatively novel estimator, vaguely related to order statistics, but based in completely different principles and it exhibits several unique features.

A more detailed study of Recordinality will be the subject of the second part of this course.
**Recordinality**

- **RECORDINALITY** (Helmi, Lumbroso, M., Viola, 2012) is a relatively novel estimator, vaguely related to order statistics, but based in completely different principles and it exhibits several unique features.

- A more detailed study of Recordinality will be the subject of the second part of this course.
How-to in Twelve Steps

1 Define some observable $R$ that depends only on the set of distinct elements (hash values) $X$ or the subsequence of their first occurrences in the data stream.

2 The observable must be:
   - insensitive to repetitions
   - very fast to compute, using a small amount of memory
How-to in Twelve Steps

1. Define some observable $R$ that depends only on the set of distinct elements (hash values) $X$ or the subsequence of their first occurrences in the data stream.

2. The observable must be:
   - insensitive to repetitions
   - very fast to compute, using a small amount of memory
Define some observable $R$ that depends only on the set of distinct elements (hash values) $X$ or the subsequence of their first occurrences in the data stream.

The observable must be:

- insensitive to repetitions
- very fast to compute, using a small amount of memory
1. Define some observable $R$ that depends only on the set of distinct elements (hash values) $X$ or the subsequence of their first occurrences in the data stream.

2. The observable must be:
   - insensitive to repetitions
   - very fast to compute, using a small amount of memory
3 Compute the probability distribution $\text{Prob}\{R = k\}$ or the density $f(x)\, dx = \text{Prob}\{x \leq R \leq x + dx\}$

4 Compute the expected value for a set of $|X| = n$ random i.i.d. uniform values in $(0, 1)$ or a random permutation of $n$ such values

$$E[R] = \sum_{k} k\text{Prob}\{R = k\} = f(n)$$

5 Under reasonable conditions, $E[f^{(-1)}(R)]$ should be similar to $n$, but a correcting factor will be necessary to obtain the estimator $Z$

$$Z := \phi \cdot f^{(-1)}(R) \Rightarrow E[Z] \sim n$$
3. Compute the probability distribution \( \text{Prob}\{R = k\} \) or the density \( f(x)\,dx = \text{Prob}\{x \leq R \leq x + dx\} \)

4. Compute the expected value for a set of \( |X| = n \) random i.i.d. uniform values in \( (0, 1) \) or a random permutation of \( n \) such values

\[
E[R] = \sum_{k} k \cdot \text{Prob}\{R = k\} = f(n)
\]

5. Under reasonable conditions, \( E\left[f^{-1}(R)\right] \) should be similar to \( n \), but a correcting factor will be necessary to obtain the estimator \( Z \)

\[
Z := \phi \cdot f^{-1}(R) \Rightarrow E[Z] \sim n
\]
How-to in Twelve Steps

3. Compute the probability distribution $\text{Prob}\{R = k\}$ or the density $f(x)\,dx = \text{Prob}\{x \leq R \leq x + dx\}$

4. Compute the expected value for a set of $|X| = n$ random i.i.d. uniform values in $(0, 1)$ or a random permutation of $n$ such values

$$E[R] = \sum_{k} k\text{Prob}\{R = k\} = f(n)$$

5. Under reasonable conditions, $E[f(-1)(R)]$ should be similar to $n$, but a correcting factor will be necessary to obtain the estimator $Z$

$$Z := \phi \cdot f(-1)(R) \Rightarrow E[Z] \sim n$$
Sometimes $E[Z] = +\infty$ or $\text{Var}[Z] = +\infty$ and stochastic averaging helps avoid this pitfall; in any case, it can be useful to use stochastic averaging

$$Z_m := F(R_1, \ldots, R_m)$$

Let $N_i$ denote the r.v. number of distinct elements going to the $i$th substream. Compute $E[Z]$:

$$E[Z_m] = \sum_{(n_1, \ldots, n_m): n_1 + \ldots + n_m = n} \frac{n!}{m^n} \sum_{j_1, \ldots, j_m} F(j_1, \ldots, j_m) \cdot \prod_{1 \leq i \leq m} \text{Prob}\{R_i = j_i \mid N_i = n_i\}$$
Sometimes $E[Z] = +\infty$ or $\text{Var}[Z] = +\infty$ and stochastic averaging helps avoid this pitfall; in any case, it can be useful to use stochastic averaging

$$Z_m := F(R_1, \ldots, R_m)$$

Let $N_i$ denote the r.v. number of distinct elements going to the $i$th substream. Compute $E[Z]$:

$$E[Z_m] = \sum_{(n_1, \ldots, n_m): n_1 + \ldots + n_m = n} \frac{n!}{m^n} \sum_{j_1, \ldots, j_m} F(j_1, \ldots, j_m) \prod_{1 \leq i \leq m} \text{Prob}\{R_i = j_i | N_i = n_i\}$$
The computation of $E[Z_m]$ should yield the correcting factor $\phi = \phi_m$ to compensate for the bias; a similar computation should allow us to compute $SE[Z_m]$.

Under quite general hypothesis $\text{Var}[Z_m] = \Theta(n^2/m)$ and $SE[Z_m] \approx c/\sqrt{m}$.

A finer analysis should provide the lower order terms $o(1)$ of the bias $E[Z_m]/n = 1 + o(1)$.
The computation of $E[Z_m]$ should yield the correcting factor $\phi = \phi_m$ to compensate the bias; a similar computation should allow us to compute $SE[Z_m]$.

Under quite general hypothesis $\text{Var}[Z_m] = \Theta(n^2/m)$ and $SE[Z_m] \approx c/\sqrt{m}$.

A finer analysis should provide the lower order terms $o(1)$ of the bias $E[Z_m]/n = 1 + o(1)$. 

How-to in Twelve Steps
The computation of $E[Z_m]$ should yield the correcting factor $\phi = \phi_m$ to compensate the bias; a similar computation should allow us to compute $SE[Z_m]$

Under quite general hypothesis $\text{Var}[Z_m] = \Theta(n^2/m)$ and $SE[Z_m] \approx c/\sqrt{m}$

A finer analysis should provide the lower order terms $o(1)$ of the bias $E[Z_m]/n = 1 + o(1)$
Careful characterization of the probability distribution of $Z_m$ is also important and useful \(\Rightarrow\) additional corrections or alternative ways to estimate the cardinality when it is small or medium \(\rightarrow\) very few distinct elements on each substream.

Experiment! Without experimentation your results will not draw attention from the practitioners; show them your estimator is practical in a real-life setting, support your theoretical analysis with experiments.
11 Careful characterization of the probability distribution of $Z_m$ is also important and useful $\Rightarrow$ additional corrections or alternative ways to estimate the cardinality when it is small or medium $\rightarrow$ very few distinct elements on each substream

12 Experiment! Without experimentation your results will not draw attention from the practitioners; show them your estimator is practical in a real-life setting, support your theoretical analysis with experiments
To estimate the number of \( k \)-elephants or \( k \)-mice in the stream we can draw a random sample of \( T \) distinct elements, together with their frequency counts. Let \( T_k \) be the number of \( k \)-mice (\( k \)-elephants) in the sample, and \( n_k \) the number of \( k \)-mice in the data stream. Then

\[
E \left[ \frac{T_k}{T} \right] = \frac{n_k}{n},
\]

with a decreasing standard error as \( T \) grows.
To estimate the number of $k$-elephants or $k$-mice in the stream we can draw a random sample of $T$ distinct elements, together with their frequency counts.

Let $T_k$ be the number of $k$-mice ($k$-elephants) in the sample, and $n_k$ the number of $k$-mice in the data stream. Then

$$E \left[ \frac{T_k}{T} \right] = \frac{n_k}{n},$$

with a decreasing standard error as $T$ grows.
The **distinct sampling** problem is to draw a random sample of distinct elements and it has many applications in data stream analysis.

In a random sample from the data stream (e.g., using the reservoir method) each distinct element $z_j$ appears with relative frequency in the sample equal to its relative frequency $f_j/N$ in the data stream $\Rightarrow$ needle-on-a-haystack.
The distinct sampling problem is to draw a random sample of distinct elements and it has many applications in data stream analysis.

In a random sample from the data stream (e.g., using the reservoir method) each distinct element \( z_j \) appears with relative frequency in the sample equal to its relative frequency \( f_j / N \) in the data stream \( \Rightarrow \) needle-on-a-haystack.
We need samples of distinct elements ⇒ distinct sampling

Adaptive sampling (Wegman, 1980; Flajolet, 1990; Louchard, 1997) is just such an algorithm (which also gives an estimation of the cardinality, as the size of the returned sample is itself a random variable)
Adaptive Sampling

We need samples of distinct elements ⇒ distinct sampling

Adaptive sampling (Wegman, 1980; Flajolet, 1990; Louchard, 1997) is just such an algorithm (which also gives an estimation of the cardinality, as the size of the returned sample is itself a random variable)
Adaptive Sampling

procedure ADAPTIVESAMPLING(S, maxC)
    C ← ∅; p ← 0
    for x ∈ S do
        if hash(x) = 0^p ... then
            C ← C ∪ {x}
            if |C| > maxC then
                p ← p + 1; filter C
            end if
        end if
    end for
    return C
end procedure

At the end of the algorithm, |C| is the number of distinct elements with hash value starting .0^p 1. \( n \) the number of strings in the subtree rooted at 0^p in a binary trie for \( n \) random binary string.
Adaptive Sampling

There are $2^p$ subtrees rooted at depth $p$

$$|C| \approx n/2^p \Rightarrow \mathbb{E} [2^p \cdot |C|] \approx n$$
Distinct Sampling in Recordinality and Order Statistics

- Recordinality and KMV collect the elements with the $k$ largest (smallest) hash values (often only the hash values).
- Such $k$ elements constitute a random sample of $k$ distinct elements.
- Recordinality can be easily adapted to collect random samples of expected size $\Theta(\log n)$ or $\Theta(n^\alpha)$, with $0 < \alpha < 1$ and without prior knowledge of $n!$ → variable-size distinct sampling → better precision in inferences about the full data stream.
Distinct Sampling in Recordinality and Order Statistics

- Recordinality and KMV collect the elements with the $k$ largest (smallest) hash values (often only the hash values).
- Such $k$ elements constitute a random sample of $k$ distinct elements.
- Recordinality can be easily adapted to collect random samples of expected size $\Theta(\log n)$ or $\Theta(n^\alpha)$, with $0 < \alpha < 1$ and without prior knowledge of $n$! ⇒ variable-size distinct sampling ⇒ better precision in inferences about the full data stream.
Distinct Sampling in Recordinality and Order Statistics

- Recordinality and KMV collect the elements with the $k$ largest (smallest) hash values (often only the hash values).
- Such $k$ elements constitute a random sample of $k$ distinct elements.
- Recordinality can be easily adapted to collect random samples of expected size $\Theta(\log n)$ or $\Theta(n^{\alpha})$, with $0 < \alpha < 1$ and without prior knowledge of $n!$.
  \[\Rightarrow\] variable-size distinct sampling \[\Rightarrow\] better precision in inferences about the full data stream.
Part II

Intermezzo: A Crash Course on Analytic Combinatorics
Two basic counting principles

Let $A$ and $B$ be two finite sets.

**The Addition Principle**
If $A$ and $B$ are disjoint then

$$|A \cup B| = |A| + |B|$$

**The Multiplication Principle**

$$|A \times B| = |A| \times |B|$$
Definition
A combinatorial class is a pair \((\mathcal{A}, |\cdot|)\), where \(\mathcal{A}\) is a finite or denumerable set of values (combinatorial objects, combinatorial structures), \(|\cdot| : \mathcal{A} \to \mathbb{N}\) is the size function and for all \(n \geq 0\)
\[
\mathcal{A}_n = \{x \in \mathcal{A} \mid |x| = n\} \text{ is finite}
\]
Combinatorial classes

Example

- $\mathcal{A}$ = all finite strings from a binary alphabet; $|s| =$ the length of string $s$
- $\mathcal{B}$ = the set of all permutations; $|\sigma| =$ the order of the permutation $\sigma$
- $\mathcal{C}_n =$ the partitions of the integer $n; |p| = n$ if $p \in \mathcal{C}_n$
Labelled and unlabelled classes

- In **unlabelled** classes, objects are made up of indistinguishable atoms; an atom is an object of size 1.
- In **labelled** classes, objects are made up of distinguishable atoms; in an object of size \(n\), each of its \(n\) atoms bears a distinct label from \(\{1, \ldots, n\}\).
Counting generating functions

Definition

Let $a_n = \#A_n$ be the number of objects of size $n$ in $A$. Then the formal power series

$$A(z) = \sum_{n \geq 0} a_n z^n = \sum_{\alpha \in A} z^{\vert \alpha \vert}$$

is the (ordinary) generating function of the class $A$. The coefficient of $z^n$ in $A(z)$ is denoted $[z^n]A(z)$:

$$[z^n]A(z) = [z^n] \sum_{n \geq 0} a_n z^n = a_n$$
Counting generating functions

Ordinary generating functions (OGFs) are mostly used to enumerate unlabelled classes.

Example

\[ \mathcal{L} = \{ w \in (0 + 1)^* | w \text{ does not contain two consecutive 0's} \} \]
\[ = \{ \epsilon, 0, 1, 01, 10, 11, 010, 011, 101, 110, 111, \ldots \} \]
\[ L(z) = z^{|\epsilon|} + z^{|0|} + z^{|1|} + z^{|01|} + z^{|10|} + z^{|11|} + \ldots \]
\[ = 1 + 2z + 3z^2 + 5z^3 + 8z^4 + \ldots \]

Exercise: Can you guess the value of \( L_n = [z^n]L(z) \)?
Counting generating functions

Definition

Let $a_n = \#A_n$ = the number of objects of size $n$ in $A$. Then the formal power series

$$\hat{A}(z) = \sum_{n \geq 0} a_n \frac{z^n}{n!} = \sum_{\alpha \in A} \frac{z^{|\alpha|}}{|\alpha|!}$$

is the exponential generating function of the class $A$. 
Counting generating functions

Exponential generating functions (EGFs) are used to enumerate labelled classes.

Example

\[ \mathcal{C} = \text{circular permutations} \]
\[ = \{ \epsilon, 1, 12, 123, 132, 1234, 1243, 1324, 1342, 1423, 1432, 12345, \ldots \} \]

\[ \hat{\mathcal{C}}(z) = \frac{1}{0!} + \frac{z}{1!} + \frac{z^2}{2!} + \frac{2z^3}{3!} + \frac{6z^4}{4!} + \cdots \]

\[ c_n = n! \cdot [z^n] \hat{\mathcal{C}}(z) = (n - 1)!, \quad n > 0 \]
Let $C = A + B$, the disjoint union of the unlabelled classes $\mathcal{A}$ and $\mathcal{B}$ ($\mathcal{A} \cap \mathcal{B} = \emptyset$). Then

$$C(z) = A(z) + B(z)$$

And

$$c_n = [z^n] C(z) = [z^n] A(z) + [z^n] B(z) = a_n + b_n$$
Let \( \mathcal{C} = \mathcal{A} \times \mathcal{B} \), the Cartesian product of the unlabelled classes \( \mathcal{A} \) and \( \mathcal{B} \). The size of \( (\alpha, \beta) \in \mathcal{C} \), where \( \alpha \in \mathcal{A} \) and \( \beta \in \mathcal{B} \), is the sum of sizes: \( |(\alpha, \beta)| = |\alpha| + |\beta| \).

Then

\[
C(z) = A(z) \cdot B(z)
\]

Proof.

\[
C(z) = \sum_{\gamma \in \mathcal{C}} z^{\gamma} = \sum_{(\alpha, \beta) \in \mathcal{A} \times \mathcal{B}} z^{\alpha} z^{\beta} = \sum_{\alpha \in \mathcal{A}} \sum_{\beta \in \mathcal{B}} z^{\alpha} \cdot z^{\beta} = \left( \sum_{\alpha \in \mathcal{A}} z^{\alpha} \right) \cdot \left( \sum_{\beta \in \mathcal{B}} z^{\beta} \right) = A(z) \cdot B(z)
\]
The $n$th coefficient of the OGF for a Cartesian product is the *convolution* of the coefficients $\{a_n\}$ and $\{b_n\}$:

$$c_n = [z^n] C(z) = [z^n] A(z) \cdot B(z) = \sum_{k=0}^{n} a_k b_{n-k}$$
Sequences

Let $A$ be a class without any empty object ($A_0 = \emptyset$). The class $C = \text{SEQ}(A)$ denotes the class of sequences of $A$’s.

$$C = \{(\alpha_1, \ldots, \alpha_k) \mid k \geq 0, \alpha_i \in A\}$$

$$= \{\epsilon\} + A + (A \times A) + (A \times A \times A) + \cdots = \{\epsilon\} + A \times C$$

Then

$$C(z) = \frac{1}{1 - A(z)}$$

Proof.

$$C(z) = 1 + A(z) + A^2(z) + A^3(z) + \cdots = 1 + A(z) \cdot C(z)$$
Labelled objects

Disjoint unions of labelled classes are defined as for unlabelled classes and \( \hat{C}(z) = \hat{A}(z) + \hat{B}(z) \), for \( C = A + B \). Also, \( c_n = a_n + b_n \).

To define labelled products, we must take into account that for each pair \((\alpha, \beta)\) where \( |\alpha| = k \) and \( |\alpha| + |\beta| = n \), we construct \( \binom{n}{k} \) distinct pairs by consistently relabelling the atoms of \( \alpha \) and \( \beta \):

\[
\alpha = (2, 1, 4, 3), \quad \beta = (1, 3, 2)
\]

\[
\alpha \times \beta = \{(2, 1, 4, 3, 5, 7, 6), (2, 1, 5, 3, 4, 7, 6), \ldots , (5, 4, 7, 6, 1, 3, 2)\}
\]

\[
\#(\alpha \times \beta) = \binom{7}{4} = 35
\]

The size of an element in \( \alpha \times \beta \) is \( |\alpha| + |\beta| \).
Labelled products

For a class $C$ that is labelled product of two labelled classes $A$ and $B$

$$C = A \times B = \bigcup_{\alpha \in A, \beta \in B} \alpha \times \beta$$

the following relation holds for the corresponding EGFs

$$\hat{C}(z) = \sum_{\gamma \in C} \frac{z^{|\gamma|!}}{|\gamma|!} = \sum_{\alpha \in A} \sum_{\beta \in B} \binom{|\alpha| + |\beta|}{|\alpha|} \frac{z^{|\alpha| + |\beta|}}{(|\alpha| + |\beta|)!}$$

$$= \sum_{\alpha \in A} \sum_{\beta \in B} \frac{1}{|\alpha|!|\beta|!} z^{|\alpha| + |\beta|} = \left( \sum_{\alpha \in A} \frac{z^{|\alpha|}}{|\alpha|!} \right) \cdot \left( \sum_{\beta \in B} \frac{z^{|\beta|}}{|\beta|!} \right)$$

$$= \hat{A}(z) \cdot \hat{B}(z)$$
Labelled products

The $n$th coefficient of $\hat{C}(z) = \hat{A}(z) \cdot \hat{B}(z)$ is also a convolution

$$c_n = [z^n] \hat{C}(z) = \sum_{k=0}^{n} \binom{n}{k} a_k b_{n-k}$$
Sequences

Sequences of labelled object are defined as in the case of unlabelled objects. The construction $C = \text{SEQ}(A)$ is well defined if $A_0 = \emptyset$.

If $C = \text{SEQ}(A) = \{\epsilon\} + A \times C$ then

$$
\hat{C}(z) = \frac{1}{1 - \hat{A}(z)}
$$

Example

Permutations are labelled sequences of atoms, $P = \text{SEQ}(Z)$. Hence,

$$
\hat{P}(z) = \frac{1}{1 - z} = \sum_{n \geq 0} z^n
$$

$$
n! \cdot [z^n]\hat{P}(z) = n!
$$
A dictionary of admissible unlabelled operators

<table>
<thead>
<tr>
<th>Class</th>
<th>OGF</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$</td>
<td>$1$</td>
<td>Epsilon</td>
</tr>
<tr>
<td>$Z$</td>
<td>$z$</td>
<td>Atomic</td>
</tr>
<tr>
<td>$A + B$</td>
<td>$A(z) + B(z)$</td>
<td>Disjoint union</td>
</tr>
<tr>
<td>$A \times B$</td>
<td>$A(z) \cdot B(z)$</td>
<td>Product</td>
</tr>
<tr>
<td>$\text{SEQ}(A)$</td>
<td>$\frac{1}{1-A(z)}$</td>
<td>Sequence</td>
</tr>
<tr>
<td>$\Theta A$</td>
<td>$\theta A(z) = z A'(z)$</td>
<td>Marking</td>
</tr>
<tr>
<td>$\text{MSET}(A)$</td>
<td>$\exp \left( \sum_{k&gt;0} A(z^k)/k \right)$</td>
<td>Multiset</td>
</tr>
<tr>
<td>$\text{PSET}(A)$</td>
<td>$\exp \left( \sum_{k&gt;0} (-1)^k A(z^k)/k \right)$</td>
<td>Powerset</td>
</tr>
<tr>
<td>$\text{CYCLE}(A)$</td>
<td>$\sum_{k&gt;0} \frac{\phi(k)}{k} \ln \frac{1}{1-A(z^k)}$</td>
<td>Cycle</td>
</tr>
</tbody>
</table>
A dictionary of admissible labelled operators

<table>
<thead>
<tr>
<th>Class</th>
<th>EGF</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$</td>
<td>1</td>
<td>Epsilon</td>
</tr>
<tr>
<td>$Z$</td>
<td>$z$</td>
<td>Atomic</td>
</tr>
<tr>
<td>$\mathcal{A} + \mathcal{B}$</td>
<td>$\hat{\mathcal{A}}(z) + \hat{\mathcal{B}}(z)$</td>
<td>Disjoint union</td>
</tr>
<tr>
<td>$\mathcal{A} \times \mathcal{B}$</td>
<td>$\hat{\mathcal{A}}(z) \cdot \hat{\mathcal{B}}(z)$</td>
<td>Product</td>
</tr>
<tr>
<td>$\text{SEQ}(\mathcal{A})$</td>
<td>$\frac{1}{1-\hat{\mathcal{A}}(z)}$</td>
<td>Sequence</td>
</tr>
<tr>
<td>$\Theta \mathcal{A}$</td>
<td>$\Theta \hat{\mathcal{A}}(z) = z\hat{\mathcal{A}}'(z)$</td>
<td>Marking</td>
</tr>
<tr>
<td>$\text{SET}(\mathcal{A})$</td>
<td>$\exp(\hat{\mathcal{A}}(z))$</td>
<td>Set</td>
</tr>
<tr>
<td>$\text{CYCLE}(\mathcal{A})$</td>
<td>$\ln \left( \frac{1}{1-\hat{\mathcal{A}}(z)} \right)$</td>
<td>Cycle</td>
</tr>
</tbody>
</table>
Bivariate generating functions

We need often to study some characteristic of combinatorial structures, e.g., the number of left-to-right maxima in a permutation, the height of a rooted tree, the number of complex components in a graph, etc.

Suppose $X : \mathcal{A}_n \rightarrow \mathbb{N}$ is a characteristic under study. Let

$$a_{n,k} = \#\{\alpha \in \mathcal{A} | |\alpha| = n, X(\alpha) = k\}$$

We can view the restriction $X_n : \mathcal{A}_n \rightarrow \mathbb{N}$ as a random variable. Then under the usual uniform model

$$\text{Prob}\{X_n = k\} = \frac{a_{n,k}}{a_n}$$
Bivariate generating functions

Define

\[ A(z, u) = \sum_{n, k \geq 0} a_{n,k} z^n u^k \]

\[ = \sum_{\alpha \in A} z^{|\alpha|} u^{X(\alpha)} \]

Then \( a_{n,k} = [z^n u^k] A(z, u) \) and

\[ \text{Prob}\{X_n = k\} = \frac{[z^n u^k] A(z, u)}{[z^n] A(z, 1)} \]
We can also define

\[
B(z, u) = \sum_{n,k \geq 0} \text{Prob}\{X_n = k\} \ z^n u^k = \sum_{\alpha \in A} \text{Prob}\{\alpha\} z^{|\alpha|} u^{X(\alpha)}
\]

and thus \(B(z, u)\) is a generating function whose coefficient of \(z^n\) is the probability generating function of the r.v. \(X_n\)

\[
B(z, u) = \sum_{n \geq 0} P_n(u) z^n \\
P_n(u) = [z^n]B(z, u) = E [u^{X_n}] = \sum_{k \geq 0} \text{Prob}\{X_n = k\} u^k
\]
Bivariate generating functions

Proposition

If $P(u)$ is the probability generating function of a random variable $X$ then

$$
P(1) = 1,$$

$$P'(1) = E[X],$$

$$P''(1) = E[X^2] = E[X(X - 1)],$$

$$Var[X] = P''(1) + P'(1) - (P'(1))^2$$
Bivariate generating functions

We can study the moments of $X_n$ by successive differentiation of $B(z, u)$ (or $A(z, u)$). For instance,

$$\bar{B}(z) = \sum_{n \geq 0} E[X_n] z^n = \frac{\partial B}{\partial u} \bigg|_{u=1}$$

For the $r$th factorial moments of $X_n$

$$B^{(r)}(z) = \sum_{n \geq 0} E[X_n^r] z^n = \frac{\partial^r B}{\partial u^r} \bigg|_{u=1}$$

$$X_n^r = X_n (X_n - 1) \cdots (X_n - r + 1)$$
Hwang’s Quasi-Powers Theorem

Let $B(z, u)$ be the BGF for a sequence $X_n$ of random variables such that

$$P_n(u) = E\left[u^{X_n}\right] = [z^n]B(z, u) = a(u) \cdot b(u)^{\lambda_n} \cdot (1 + o(1))$$

in a complex neighborhood of $u = 1$, with $\lambda_n \to \infty$, and $a(u)$ and $b(u)$ analytic functions in a neighborhood of $u = 1$ with $a(1) = b(1) = 1$. Then a proper normalization of $X_n$ satisfies a CLT:

$$\frac{X_n - E[X_n]}{\sqrt{\text{Var}[X_n]}} \xrightarrow{(d)} N(0, 1),$$

provided that $\text{Var}[X_n] \to \infty$. 
Consider the following specification for permutations

\[ \mathcal{P} = \{\emptyset\} + \mathcal{P} \times \mathbb{Z} \]

The BGF for the probability that a random permutation of size \( n \) has \( k \) left-to-right maxima is

\[
M(z, u) = \sum_{\sigma \in \mathcal{P}} \frac{z^{|\sigma|}}{|\sigma|!} u^{X(\sigma)},
\]

where \( X(\sigma) = \# \text{ of left-to-right maxima in } \sigma \)
The number of left-to-right maxima in a permutation

With the recursive decomposition of permutations and since the last element of a permutation of size $n$ is a left-to-right maxima iff its label is $n$

$$M(z, u) = \sum_{\sigma \in \mathcal{P}} \sum_{1 \leq j \leq |\sigma| + 1} \frac{z^{|\sigma| + 1}}{(|\sigma| + 1)!} u^{\chi(\sigma) + [j = |\sigma| + 1]}$$

$[P] = 1$ if $P$ is true, $[P] = 0$ otherwise.
The number of left-to-right maxima in a permutation

\[ M(z, u) = \sum_{\sigma \in \mathcal{P}} \frac{z^{\vert \sigma \vert + 1}}{(\vert \sigma \vert + 1)!} u^{X(\sigma)} \sum_{1 \leq j \leq \vert \sigma \vert + 1} u^{[j=\vert \sigma \vert + 1]} \]

\[ = \sum_{\sigma \in \mathcal{P}} \frac{z^{\vert \sigma \vert + 1}}{(\vert \sigma \vert + 1)!} u^{X\sigma}(\vert \sigma \vert + u) \]

Taking derivatives w.r.t. \( z \)

\[ \frac{\partial}{\partial z} M = \sum_{\sigma \in \mathcal{P}} \frac{z^{\vert \sigma \vert}}{\vert \sigma \vert !} u^{X\sigma}(\vert \sigma \vert + u) = z \frac{\partial}{\partial z} M + uM \]

Hence,

\[ (1 - z) \frac{\partial}{\partial z} M(z, u) - uM(z, u) = 0 \]
The number of left-to-right maxima in a permutation

Solving, since $M(0, u) = 1$

$$M(z, u) = \left(\frac{1}{1 - z}\right)^u = \sum_{n,k \geq 0} \left[ n \atop k \right] \frac{z^n}{n!} u^k$$

where $\left[ n \atop k \right]$ denote the (signless) Stirling numbers of the first kind, also called Stirling cycle numbers.

Hence

$$\text{Prob} \{X_n = k\} = \frac{\left[ n \atop k \right]}{n!}$$
The number of left-to-right maxima in a permutation

Taking the derivative w.r.t. $u$ and setting $u = 1$

$$m(z) = \frac{\partial}{\partial z} M(z, u) \bigg|_{u=1} = \frac{1}{1-z} \ln \frac{1}{1-z}$$

Thus the average number of left-to-right maxima in a random permutation of size $n$ is

$$[z^n] m(z) = E[X_n] = H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \ln n + \gamma + O(1/n)$$

$$\frac{1}{1-z} \ln \frac{1}{1-z} = \sum_{\ell} z^\ell \sum_{m>0} \frac{z^m}{m} = \sum_{n\geq 0} z^n \sum_{k=1}^{n} \frac{1}{k}$$
The number of left-to-right maxima in a permutation

Similarly, taking the second derivative w.r.t. \( u \) of \( M(z, u) \) and setting \( u = 1 \) we get the GF of the second factorial moment

\[
m_2(z) = \frac{\partial^2}{\partial z^2} M(z, u) \bigg|_{u=1} = \frac{1}{1 - z} \ln^2 \frac{1}{1 - z}
\]

Then

\[
[z^n]m_2(z) = \mathbb{E} \left[ X_n^2 \right] = 2 \sum_{0<j\leq n} \frac{H_{j-1}}{j} = H_n^2 - H_n^{(2)},
\]

\[
H_n^{(2)} = \sum_{1\leq j\leq n} \frac{1}{j^2}
\]

\[
\text{Var} \left[ X_n \right] = [z^n]m_2(z) + [z^n]m(z) - ([z^n]m(z))^2
\]

\[
= H_n^2 - H_n^{(2)} + H_n - H_n^2 = H_n - H_n^{(2)} = \ln n + O(1)
\]
The number of left-to-right maxima in a permutation

Since \( M(z, u) = (1 - z)^{-u} \) we have

\[
[z^n]M(z, u) = [z^n]\left(\frac{1}{1 - z}\right)^u = n!(\binom{n + u - 1}{n})(\equiv \frac{\Gamma(n + u)}{\Gamma(u)}
\]

Thus in a neighborhood of \( u = 1 \),

\[
E\left[u^{X_n}\right] = [z^n]M(z, u) = n^{u-1}(1 + o(1))
\]

and applying Hwang’s quasi-powers theorem with \( a(u) = 1 \), \( b(u) = \exp(u - 1) \) and \( \lambda_n = \ln n \) it follows that

\[
\frac{X_n - \ln n}{\sqrt{\ln n}} \xrightarrow{(d)} \mathcal{N}(0, 1)
\]
Part III

Case Study: Analysis of Recordinality
Introduction

Given the data stream $S = s_1, \ldots, s_N$, consider the substream

$$S_u = z_1, \ldots, z_n$$

with $z_i$ the $i$-th distinct element in $S$ in order of appearance.

Example

$$S = 3, 14, 1, 593, 26, 53, 5, 8979, 3, 23, 8, 46, 26, 433, 8, 3, 2, 8$$
$$S_u = 3, 14, 1, 593, 26, 53, 5, 8979, 23, 8, 46, 433, 2$$
Applying a hash function $h$ on $S_u$ allows us to see the data stream as a permutation $P_u$:

Example

$S_u = 3, 14, 1, 593, 26, 53, 5, 8979, 23, 8, 46, 433, 2$

$P_u = 3, 6, 1, 12, 8, 10, 4, 13, 7, 5, 9, 11, 2$

$S = 3, 14, 1, 593, 26, 53, 5, 8979, 3, 23, 8, 46, 26, 433, 8, 3, 2, 8$

$P = 3, 6, 1, 12, 8, 10, 4, 13, 3, 7, 5, 9, 8, 11, 5, 3, 2, 5$

To simplify this example take $h(x) = x$
**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**

- **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

- **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!
\( \sigma(i) \) 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 \( \sigma(1), \ldots, \sigma(i) \)
σ(i) is a record of the permutation σ if σ(i) > σ(j) for all j < i

This notion is generalized to k-records: σ(i) is a k-record if there are at most k − 1 elements σ(j) larger than σ(i) for j < i; in other words, σ(i) is among the k largest elements in σ(1), . . . , σ(i)
procedure RECORDINALITY(S)
    fill $T$ with the first $k$ distinct elements (hash values) of the stream $S$
    $R \leftarrow k$
    for all $s \in S$ do
        $x \leftarrow h(s)$
        if $x > \min(T) \land x \notin T$ then
            $R \leftarrow R + 1; \ T \leftarrow T \cup \{x\} \setminus \min(T)$
        end if
    end for
    return $Z = \varphi(R)$
end procedure

Memory: $k$ hash values ($k \log n$ bits) + 1 counter ($\log \log n$ bits)
To find the estimator $Z$, we need to fully understand the probabilistic behavior of $R$, the number of $k$-records in a random permutation of size $n$.

The recursive decomposition of permutations

$$\mathcal{P} = \epsilon + \mathcal{P} \times Z$$

is the natural choice for the analysis of $k$-records, with $\times$ denoting the labelled product.
Analysis of $k$-Records

For each $\sigma$ in $\mathcal{P}$, $\{\sigma\} \times \mathbb{Z}$ is the set of $|\sigma| + 1$ permutations

$$\{\sigma \star 1, \sigma \star 2, \ldots, \sigma \star (n + 1)\}, \quad n = |\sigma|$$

$\sigma \star j$ denotes the permutation one gets after relabelling $j$, $j + 1$, $\ldots$, $n = |\sigma|$ in $\sigma$ to $j + 1$, $j + 2$, $\ldots$, $n + 1$ and appending $j$ at the end

**Example**

$$32451 \star 3 = 425613$$
$$32451 \star 2 = 435612$$
Analysis of $\kappa$-Records

For each $\sigma$ in $\mathcal{P}$, $\{\sigma\} \times Z$ is the set of $|\sigma| + 1$ permutations

$$\{\sigma \star 1, \sigma \star 2, \ldots, \sigma \star (n + 1)\}, \quad n = |\sigma|$$

$\sigma \star j$ denotes the permutation one gets after relabelling $j$, $j + 1$, $\ldots$, $n = |\sigma|$ in $\sigma$ to $j + 1$, $j + 2$, $\ldots$, $n + 1$ and appending $j$ at the end

Example

$$32451 \star 3 = 425613$$
$$32451 \star 2 = 435612$$
Analysis of $k$-Records

- $\mathcal{R}(\sigma) =$ the set of $k$-records in permutation $\sigma$
- $r(\sigma) = \#\mathcal{R}(\sigma)$
- Let $X_j(\sigma) = 1$ if $n - k + 1 < j \leq n + 1$, $n = |\sigma|$; $X_j(\sigma) = 0$ otherwise.
- $r(\sigma \star j) = r(\sigma) + X_j(\sigma)$
Analysis of $k$-Records

- $\mathcal{R}(\sigma) =$ the set of $k$-records in permutation $\sigma$
- $r(\sigma) = \#\mathcal{R}(\sigma)$
- Let $X_j(\sigma) = 1$ if $n - k + 1 < j \leq n + 1$, $n = |\sigma|$; $X_j(\sigma) = 0$ otherwise.
- $r(\sigma \ast j) = r(\sigma) + X_j(\sigma)$
Analysis of $k$-Records

- $\mathcal{R}(\sigma) = \text{the set of } k\text{-records in permutation } \sigma$
- $r(\sigma) = \#\mathcal{R}(\sigma)$
- Let $X_j(\sigma) = 1$ if $n - k + 1 < j \leq n + 1$, $n = |\sigma|$; $X_j(\sigma) = 0$ otherwise.
- $r(\sigma \star j) = r(\sigma) + X_j(\sigma)$
Analysis of $k$-Records

- $\mathcal{R}(\sigma) =$ the set of $k$-records in permutation $\sigma$
- $r(\sigma) = \#\mathcal{R}(\sigma)$
- Let $X_j(\sigma) = 1$ if $n - k + 1 < j \leq n + 1$, $n = |\sigma|$; $X_j(\sigma) = 0$ otherwise.
- $r(\sigma * j) = r(\sigma) + X_j(\sigma)$
Analysis of $k$-Records

**Theorem**

Let $R(z, u) = \sum_{\sigma \in \mathcal{P} : |\sigma| \geq k} \frac{z^{|\sigma|}}{|\sigma|!} u^r(\sigma)$.

Then

$$\frac{\partial}{\partial z} ((1 - z) R(z, u)) = k(u - 1) R(z, u) + k \frac{u^k z^{k-1}}{k!}.$$
Analysis of $\kappa$-Records

\[ R(z, u) = \sum_{\sigma \in \mathcal{P} : |\sigma| \geq k} \frac{z^{|\sigma|}}{|\sigma|!} u^{r(\sigma)} = \frac{z^k u^k}{k!} + \sum_{n > k} \sum_{\sigma \in \mathcal{P}_n} \frac{z^{|\sigma|}}{|\sigma|!} u^{r(\sigma)} \]

\[ = \frac{z^k u^k}{k!} + \sum_{n > k} \sum_{1 \leq j \leq n} \sum_{\sigma \in \mathcal{P}_{n-1}} \frac{z^{|\sigma \star j|}}{|\sigma \star j|!} u^{r(\sigma \star j)} \]

\[ = \frac{z^k u^k}{k!} + \sum_{n > k} \sum_{1 \leq j \leq n} \sum_{\sigma \in \mathcal{P}_{n-1}} \frac{z^{|\sigma| + 1}}{(|\sigma| + 1)!} u^{r(\sigma) + X_j(\sigma)} \]

\[ = \frac{z^k u^k}{k!} + \sum_{n > k} \sum_{\sigma \in \mathcal{P}_{n-1}} \frac{z^{|\sigma| + 1}}{(|\sigma| + 1)!} u^{r(\sigma)} \sum_{1 \leq j \leq n} u^{X_j(\sigma)}. \]
Analysis of $k$-Records

Since $X_j(\sigma)$ is 1 if and only if $j > |\sigma| + 1 - k$ and 0 otherwise

$$\sum_{1 \leq j \leq n} u^{X_j(\sigma)} = (|\sigma| + 1 - k) + ku.$$

$$R(z, u) = \frac{z^k u^k}{k!} + \sum_{n>k} \sum_{\sigma \in \mathcal{P}_{n-1}} \frac{z^{|\sigma| + 1}}{(|\sigma| + 1)!} u^{r(\sigma)} \left( (|\sigma| + 1 - k) + ku \right).$$

The theorem follows after differentiation w.r.t. $z$ and a few additional algebraic manipulations.
To solve the PDE for $R(z, zu)$ we introduce

$$\Phi(z, u) := \frac{z^k \partial^k R(z, u)}{k!} \frac{\partial}{\partial z^k}$$

so that

$$[z^n] \Phi(z, u) = \binom{n}{k} [z^n] R(z, u)$$

and

$$(1 - z) \frac{\partial \Phi}{\partial z} - (k + 1) \Phi = k(u - 1) \Phi$$
Analysis of $k$-Records

The explicit solution for $\Phi(z, u)$ is easier, once we plug in the initial conditions, we get

$$\Phi(z, u) = \frac{(zu)^k}{1-z} \left( \frac{1}{1-z} \right)^ku$$

We can get easily average and variance for the number $R_n$ of $k$-records:

$$E[R_n] = \frac{1}{\binom{n}{k}} [z^n] \frac{\partial \Phi}{\partial u} \bigg|_{u=1}$$

$$= k(H_n - H_k + 1) = k \ln(n/k) + O(1)$$

Likewise

$$Var[R_n] = k(H_n - H_k) - k^2(H_n^{(2)} - H_k^{(2)}) = k \ln(n/k) + O(1)$$
Analysis of $k$-Records

From the explicit form of $\Phi(z, u)$

Theorem (Helmi, M., Panholzer, 2012)

$$\text{Prob}\{R_n = j\} = \begin{cases} \lfloor n = j \rfloor, & \text{if } n < k, \\ \lfloor n-k+1 \rfloor \frac{k^{j-k} \cdot k!}{n!}, & \text{if } k \leq j \leq n. \end{cases}$$
Let us assume for the moment that $k \leq R \leq n$. If $R < k$ then we are sure that $n = R$.

Since $E[R_n] = k \ln(n/k) + O(1)$ let us take

$$W = \exp(\phi \cdot R)$$

for some correcting factor $\phi$ to be determined and such that $E[W]$ is close (proportional?) to $n$. 

The Estimator for Recordionality
The Estimator for Recordinality

\[
E[\exp(\phi \cdot R)] = \sum_{j \geq k} \exp(\phi \cdot j) \Pr\{R = j\}
\]

\[
= \sum_{j \geq k} \exp(\phi \cdot j) \left[ \begin{array}{c} n - k + 1 \\ j - k + 1 \end{array} \right] \frac{k^{j-k} \cdot k!}{n!}
\]

\[
= \frac{k!}{n!k} \exp(\phi \cdot (k - 1)) \sum_{j \geq 1} \left[ \begin{array}{c} n - k + 1 \\ j \end{array} \right] (k \exp(\phi))^j
\]

Since

\[
\sum_{1 \leq j \leq m} \left[ \begin{array}{c} m \\ j \end{array} \right] z^j = z(z + 1) \cdots (z + m - 1) =: z^m
\]

\[
E[\exp(\phi \cdot R)] = \frac{k!}{n!k} \exp(\phi \cdot (k - 1)) (k \exp(\phi))^{n-k+1}
\]
The Estimator for Recordinality

If \( k \exp(\phi) = k + 1 \) then

\[
(k \exp(\phi))^{n-k+1} = (k + 1)^{n-k+1} = \frac{(n + 1)!}{k!}
\]

\[
\exp(\phi) = \left(1 + \frac{1}{k}\right)
\]

Hence

\[
E[\exp(\phi \cdot R)] = \frac{k!}{n!k} \exp(\phi \cdot (k - 1))(k \exp(\phi))^{n-k+1}
\]

\[
= \frac{n + 1}{k} \left(1 + \frac{1}{k}\right)^{k-1}
\]
The Estimator for Recordinality

Therefore if we set

\[ Z = k \left( 1 + \frac{1}{k} \right)^{-k+1} \exp(\phi \cdot R) - 1 \]

\[ = k \left( 1 + \frac{1}{k} \right)^{-k+1} \left( 1 + \frac{1}{k} \right)^R - 1 \]

\[ = k \left( 1 + \frac{1}{k} \right)^{R-k+1} - 1, \]

\[ E[Z] = n, \text{ exactly!!} \]
Two plots showing the accuracy of 500 estimates of the number of distinct elements contained in Shakespeare’s *A Midsummer Night’s Dream*. Left: $k = 64$. Right: $k = 256$. Above the top and below the bottom line: 5% of the estimates. Area within centermost lines: 70% estimates. Gray rectangle: area within one standard deviation from the mean.
### Recordinality in Practice

<table>
<thead>
<tr>
<th>$\kappa$</th>
<th>Recordinality Avg.</th>
<th>Error</th>
<th>Adaptive Sampling Avg.</th>
<th>Error</th>
<th>$\kappa$-th Order Statistic Avg.</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2737</td>
<td>1.04</td>
<td>3047</td>
<td>0.70</td>
<td>4050</td>
<td>0.89</td>
</tr>
<tr>
<td>8</td>
<td>2811</td>
<td>0.73</td>
<td>3014</td>
<td>0.41</td>
<td>3495</td>
<td>0.44</td>
</tr>
<tr>
<td>16</td>
<td>3040</td>
<td>0.54</td>
<td>3012</td>
<td>0.31</td>
<td>3219</td>
<td>0.28</td>
</tr>
<tr>
<td>32</td>
<td>3010</td>
<td>0.34</td>
<td>3078</td>
<td>0.20</td>
<td>3159</td>
<td>0.18</td>
</tr>
<tr>
<td>64</td>
<td>3020</td>
<td>0.34</td>
<td>3020</td>
<td>0.15</td>
<td>3071</td>
<td>0.12</td>
</tr>
<tr>
<td>128</td>
<td>3042</td>
<td>0.14</td>
<td>3032</td>
<td>0.11</td>
<td>3070</td>
<td>0.10</td>
</tr>
<tr>
<td>256</td>
<td>3044</td>
<td>0.08</td>
<td>3027</td>
<td>0.07</td>
<td>3037</td>
<td>0.06</td>
</tr>
<tr>
<td>512</td>
<td>3043</td>
<td>0.04</td>
<td>3043</td>
<td>0.05</td>
<td>3046</td>
<td>0.04</td>
</tr>
</tbody>
</table>

Table: Estimating the number of distinct elements in Shakespeare's *A Midsummer Night's Dream* ($n = 3031$). Normalized average and the empirical standard deviation divided by $n$. 10 000 simulations.
### Recordinality in Practice

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>43658</td>
<td>1.19</td>
<td>59474</td>
<td>0.94</td>
<td>81724</td>
<td>1.30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>35230</td>
<td>0.52</td>
<td>47432</td>
<td>0.38</td>
<td>57028</td>
<td>0.41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>57723</td>
<td>0.98</td>
<td>49889</td>
<td>0.29</td>
<td>52990</td>
<td>0.23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>48686</td>
<td>0.45</td>
<td>49480</td>
<td>0.23</td>
<td>50556</td>
<td>0.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>47617</td>
<td>0.34</td>
<td>50524</td>
<td>0.14</td>
<td>51146</td>
<td>0.13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>50097</td>
<td>0.17</td>
<td>50452</td>
<td>0.09</td>
<td>50947</td>
<td>0.08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>51742</td>
<td>0.11</td>
<td>50857</td>
<td>0.06</td>
<td>50348</td>
<td>0.06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>49496</td>
<td>0.09</td>
<td>49920</td>
<td>0.06</td>
<td>50084</td>
<td>0.04</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table:** Experiments for a random stream containing \( n = 50\,000 \) distinct elements—here 25\,000 simulations were run.
To Know More: General References

Philippe Flajolet and Robert Sedgewick. 
*Analytic Combinatorics.*

Ronald L. Graham, Donald E. Knuth, and Oren Patashnik.
*Concrete Mathematics.*

S. Muthu Muthukrishnan.
Data streams: Algorithms and applications.
*Foundations and Trends in Theoretical Computer Science,*
To Know More: Research Papers

Counting Distinct Elements in a Data Stream.  
*Randomization and Approximation Techniques (RANDOM)*,  

Marianne Durand and Philippe Flajolet.
LogLog Counting of Large Cardinalities.  

Philippe Flajolet.
On adaptive sampling.  
To Know More: Research Papers


To Know More: Research Papers

A. Helmi, J. Lumbroso, C. Martínez, and A. Viola.
Counting distinct elements in data streams: the random permutation viewpoint.

Jérémie Lumbroso.
An optimal cardinality estimation algorithm based on order statistics and its full analysis.