Relative Error Streaming Quantiles

Pavel Veselý

University of Warwick

WOLA 2020 (recorded 1 July 2020)

Joint work in progress with Graham Cormode (Warwick), Zohar Karnin (Amazon), Edo Liberty (HyperCube), and Justin Thaler (Georgetown)
Selection Problem & Streaming

- Input: $N$ numbers
- Goal: find the $k$-th smallest
  - e.g.: the median, 99th percentile
- $O(N)$ time offline algorithm [Blum et al. ’73]

Streaming restrictions:
- just one pass over the data
- limited memory: $o(N)$
- provide worst-case guarantees

Main objective: space

No streaming algorithm for exact selection
$\Omega(N)$ space needed to find the median [Munro & Paterson ’80, Guha & McGregor ’07]

What about finding an approximate median?
Selection Problem & Streaming

- **Input**: $N$ numbers
- **Goal**: find the $k$-th smallest
  - e.g.: the median, 99th percentile
- $O(N)$ time offline algorithm [Blum et al. '73]
- **Streaming restrictions**:
  - just one pass over the data
  - limited memory: $o(N)$
  - provide worst-case guarantees

**Main objective**: space
Selection Problem & Streaming

- Input: $N$ numbers
- Goal: find the $k$-th smallest
  - e.g.: the median, 99th percentile
- $O(N)$ time offline algorithm [Blum et al. '73]
- Streaming restrictions:
  - just one pass over the data
  - limited memory: $o(N)$
  - provide worst-case guarantees

Main objective: space
Selection Problem & Streaming

- Input: $N$ numbers
- Goal: find the $k$-th smallest
  - e.g.: the median, 99th percentile
- $\mathcal{O}(N)$ time offline algorithm [Blum et al. '73]
- Streaming restrictions:
  - just one pass over the data
  - limited memory: $o(N)$
  - provide worst-case guarantees

Main objective: space

No streaming algorithm for exact selection

$\Omega(N)$ space needed to find the median [Munro & Paterson '80, Guha & McGregor '07]
Selection Problem & Streaming

• Input: \( N \) numbers
• Goal: find the \( k \)-th smallest
  • e.g.: the median, 99th percentile
• \( \mathcal{O}(N) \) time offline algorithm [Blum et al. ’73]
• Streaming restrictions:
  • just one pass over the data
  • limited memory: \( o(N) \)
  • provide worst-case guarantees

Main objective: **space**

No streaming algorithm for exact selection
\( \Omega(N) \) space needed to find the median [Munro & Paterson ’80, Guha & McGregor ’07]

What about finding an approximate median?
Approximate Median & Quantiles with Uniform Error

How to define an approximate median?
Approximate Median & Quantiles with Uniform Error

How to define an approximate median?

φ-quantile = ⌈φ · N⌉-th smallest element (φ ∈ [0, 1])

- Median = .5-quantile

Offline summary: sort data & select ∼ 1/2ε items

Very well-solved both in theory & practice:

- Deterministic algs.: space Θ(1/ε · log ε N) optimal [Greenwald & Khanna '01, Cormode, V. '20]
- Randomized algs.: space Θ(1/ε) optimal (w/ const. probability of too high error) [Karnin et al. '16]
Approximate Median & Quantiles with Uniform Error

How to define an approximate median?

\[ \phi\text{-quantile} = \lceil \phi \cdot N \rceil\text{-th smallest element} \quad (\phi \in [0, 1]) \]

- Median = .5-quantile
- Quartiles = .25, .5, and .75-quantiles
- Percentiles = .01, .02, …, .99-quantiles
Approximate Median & Quantiles with Uniform Error

How to define an approximate median?

\( \phi \)-quantile = \( \lceil \phi \cdot N \rceil \)-th smallest element (\( \phi \in [0, 1] \))

- Median = .5-quantile
- Quartiles = .25, .5, and .75-quantiles
- Percentiles = .01, .02, \ldots, .99-quantiles

\( \varepsilon \)-approximate \( \phi \)-quantile = any \( \phi' \)-quantile for \( \phi' = [\phi - \varepsilon, \phi + \varepsilon] \)

- .01-approximate medians are .49- and .51-quantiles (and items in between)
Approximate Median & Quantiles with Uniform Error

How to define an approximate median?

\( \phi \)-quantile = \( \lceil \phi \cdot N \rceil \)-th smallest element (\( \phi \in [0, 1] \))

- Median = .5-quantile
- Quartiles = .25, .5, and .75-quantiles
- Percentiles = .01, .02, \ldots, .99-quantiles

\( \varepsilon \)-approximate \( \phi \)-quantile = any \( \phi' \)-quantile for \( \phi' = [\phi - \varepsilon, \phi + \varepsilon] \)

- .01-approximate medians are .49- and .51-quantiles (and items in between)

Offline summary: sort data & select \( \sim \frac{1}{2\varepsilon} \) items
Approximate Median & Quantiles with Uniform Error

How to define an approximate median?

φ-quantile = ⌈φ · N⌉-th smallest element (φ ∈ [0, 1])

- Median = .5-quantile
- Quartiles = .25, .5, and .75-quantiles
- Percentiles = .01, .02, . . . , .99-quantiles

ε-approximate φ-quantile = any φ′-quantile for φ′ = [φ − ε, φ + ε]

- .01-approximate medians are .49- and .51-quantiles (and items in between)

Offline summary: sort data & select ∼ \( \frac{1}{2\epsilon} \) items

Very well-solved both in theory & practice:

- Deterministic algs.: space \( \Theta\left(\frac{1}{\epsilon} \cdot \log \epsilon N\right) \) optimal [Greenwald & Khanna '01, Cormode, V. '20]
- Randomized algs.: space \( \Theta\left(\frac{1}{\epsilon}\right) \) optimal (w/ const. probability of too high error) [Karnin et al. '16]
Motivation for Relative Error

Often need to track percentiles 50, 70, 90, 95, 99, 99.5, ...
Motivation for Relative Error

Often need to track percentiles 50, 70, 90, 95, 99, 99.5, ... 

- e.g.: network latencies are long-tailed [Masson et al. ’19]

- 98.5th percentile = 2s
- 99.5th percentile = 20s
- maximum \( \geq 1 \) minute

Motivation for Relative Error

Often need to track percentiles 50, 70, 90, 95, 99, 99.5, \ldots

- e.g.: network latencies are long-tailed [Masson et al. '19]

\begin{itemize}
  \item 98.5th percentile = 2s
  \item maximum $\geq$ 1 minute
  \item 99.5th percentile = 20s
  \item uniform error $\varepsilon = 0.01$ not useful for 99.5th percentile
\end{itemize}

Motivation for Relative Error

Often need to track percentiles 50, 70, 90, 95, 99, 99.5, ... 

- e.g.: network latencies are long-tailed [Masson et al. ’19]

![Histogram of network latencies](image)


- 98.5th percentile = 2s
- 99.5th percentile = 20s
- maximum $\geq$ 1 minute
- uniform error $\varepsilon = 0.01$ not useful for 99.5th percentile

Can we have a stronger error guarantee?
Can we understand the tails of the distribution better?
Quantiles with Relative Error

Query $\phi$-quantile for $\phi \in [0,1] \rightarrow$ return $\phi'$-quantile for $\phi' = \phi \pm \epsilon \phi$

uniform error: $\phi' = \phi \pm \epsilon$

• Essentially the same for $\phi = \Omega(1)$, say $\phi = 0$.

• Much stronger for extreme values of $\phi$ such as $\phi = 1/\sqrt{N}$.

Cumulative distribution function:

Selection: query $k$-th smallest $\rightarrow$ return $(k \pm \epsilon k)$-th smallest in the stream

Offline summary: sort data & select $O(\epsilon \cdot \log \epsilon N)$ items

• example for $\epsilon = 0.25$: 2 items, 2 $\epsilon$ items, 4 $\epsilon$ items, 8 $\epsilon$ items
Quantiles with Relative Error

Query $\phi$-quantile for $\phi \in [0, 1] \rightarrow$ return $\phi'$-quantile for $\phi' = \phi \pm \varepsilon \phi$

uniform error: $\phi' = \phi \pm \varepsilon$
Quantiles with Relative Error

Query $\phi$-quantile for $\phi \in [0, 1] \rightarrow$ return $\phi'$-quantile for $\phi' = \phi \pm \varepsilon \phi$

uniform error: $\phi' = \phi \pm \varepsilon$

- Essentially the same for $\phi = \Omega(1)$, say $\phi = 0.5$
- Much stronger for extreme values of $\phi$ such as $\phi = 1/\sqrt{N}$
Quantiles with Relative Error

Query $\phi$-quantile for $\phi \in [0, 1] \rightarrow$ return $\phi'$-quantile for $\phi' = \phi \pm \varepsilon \phi$

uniform error: $\phi' = \phi \pm \varepsilon$

- Essentially the same for $\phi = \Omega(1)$, say $\phi = 0.5$
- Much stronger for extreme values of $\phi$ such as $\phi = 1/\sqrt{N}$

Cumulative distribution function:
Quantiles with Relative Error

Query $\phi$-quantile for $\phi \in [0, 1] \rightarrow$ return $\phi'$-quantile for $\phi' = \phi \pm \varepsilon \phi$

uniform error: $\phi' = \phi \pm \varepsilon$

- Essentially the same for $\phi = \Omega(1)$, say $\phi = 0.5$
- Much stronger for extreme values of $\phi$ such as $\phi = 1/\sqrt{N}$

Cumulative distribution function:

Selection: query $k$-th smallest $\rightarrow$ return $(k \pm \varepsilon k)$-th smallest in the stream
Quantiles with Relative Error

Query $\phi$-quantile for $\phi \in [0, 1] \rightarrow$ return $\phi'$-quantile for $\phi' = \phi \pm \varepsilon \phi$

uniform error: $\phi' = \phi \pm \varepsilon$

- Essentially the same for $\phi = \Omega(1)$, say $\phi = 0.5$
- Much stronger for extreme values of $\phi$ such as $\phi = 1/\sqrt{N}$

Cumulative distribution function:

![Graph showing cumulative distribution function with uniform and relative error](image)

Selection: query $k$-th smallest $\rightarrow$ return $(k \pm \varepsilon k)$-th smallest in the stream

Offline summary: sort data & select $O \left( \frac{1}{\varepsilon} \cdot \log \varepsilon N \right)$ items

- example for $\varepsilon = 0.25$: $\frac{2}{\varepsilon}$ items, $\frac{2}{\varepsilon}$ items, $\frac{4}{\varepsilon}$ items, $\frac{8}{\varepsilon}$ items
Streaming Algorithms for Relative Error $\varepsilon$
Streaming Algorithms for Relative Error $\varepsilon$

State of the art: space $\sim$ # of stored items

- Deterministic: $\mathcal{O}\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N \cdot \log M\right)$ for integers $\{1, \ldots, M\}$

[Cormode et al. '06]
Streaming Algorithms for Relative Error $\varepsilon$

State of the art: space $\sim$ # of stored items

- Deterministic: $O\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N \cdot \log M\right)$ for integers $\{1, \ldots, M\}$

  $O\left(\frac{1}{\varepsilon} \cdot \log^3 \varepsilon N\right)$ [Zhang & Wang '07]

  $\leq$

- Randomized: $O\left(\frac{1}{\varepsilon^2} \cdot \log \varepsilon N\right)$ (by sampling) [Gupta & Zane '03, Zhang et al. '06]

  $O\left(\frac{1}{\varepsilon} \cdot \log 1.5 \varepsilon N\right)$ [Cormode, Karnin, Liberty, Thaler, V. ’20+]

  $\Omega\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N\right)$ [Cormode et al. ’05]
Streaming Algorithms for Relative Error $\varepsilon$

State of the art: space $\sim \#$ of stored items

- Deterministic: $O\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N \cdot \log M\right)$ for integers $\{1, \ldots, M\}$

  $O\left(\frac{1}{\varepsilon} \cdot \log^3 \varepsilon N\right)$ [Zhang & Wang '07]

  $\Omega\left(\frac{1}{\varepsilon} \cdot \log^2 \varepsilon N\right)$ [Cormode & V. '20]

  $\leq$

[Cormode et al. '06]

Pavel Veselý  Relative Error Streaming Quantiles 5 / 10
Streaming Algorithms for Relative Error \( \varepsilon \)

State of the art: space \( \sim \# \) of stored items

- Deterministic: \( \mathcal{O} \left( \frac{1}{\varepsilon} \cdot \log \varepsilon N \cdot \log M \right) \) for integers \( \{1, \ldots, M\} \)
  - \( \mathcal{O} \left( \frac{1}{\varepsilon} \cdot \log^3 \varepsilon N \right) \) [Zhang & Wang '07]
  - \( \Omega \left( \frac{1}{\varepsilon} \cdot \log^2 \varepsilon N \right) \) [Cormode & V. '20]

- Randomized: \( \mathcal{O} \left( \frac{1}{\varepsilon^2} \cdot \log \varepsilon N \right) \) (by sampling) [Gupta & Zane '03, Zhang et al. '06]
Streaming Algorithms for Relative Error $\varepsilon$

State of the art: space $\sim \#$ of stored items

- Deterministic: $O\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N \cdot \log M\right)$ for integers $\{1, \ldots, M\}$
  - $O\left(\frac{1}{\varepsilon} \cdot \log^3 \varepsilon N\right)$ [Zhang & Wang '07]
  - $\Omega\left(\frac{1}{\varepsilon} \cdot \log^2 \varepsilon N\right)$ [Cormode & V. '20]

- Randomized: $O\left(\frac{1}{\varepsilon^2} \cdot \log \varepsilon N\right)$ (by sampling) [Gupta & Zane '03, Zhang et al. '06]
  - $O\left(\frac{1}{\varepsilon} \cdot \log^{1.5} \varepsilon N\right)$ [Cormode, Karnin, Liberty, Thaler, V. '20+]
Streaming Algorithms for Relative Error $\varepsilon$

State of the art: space $\sim$ # of stored items

- Deterministic: $O\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N \cdot \log M\right)$ for integers $\{1, \ldots, M\}$
  
  $O\left(\frac{1}{\varepsilon} \cdot \log^3 \varepsilon N\right)$ [Zhang & Wang '07]

  $\Omega\left(\frac{1}{\varepsilon} \cdot \log^2 \varepsilon N\right)$ [Cormode & V. '20]

- Randomized: $O\left(\frac{1}{\varepsilon^2} \cdot \log \varepsilon N\right)$ (by sampling) [Gupta & Zane '03, Zhang et al. '06]

  $O\left(\frac{1}{\varepsilon} \cdot \log^{1.5} \varepsilon N\right)$ [Cormode, Karnin, Liberty, Thaler, V. '20+]

  $\Omega\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N\right)$ [Cormode et al. '05]
Technique for Designing Randomized Algorithms


- Buffers of size $B$ arranged at $O(\log N)$ levels
Technique for Designing Randomized Algorithms

Used in: [Manku et al. '99, Agarwal et al. '12, Wang et al. '13, Karnin et al. '16]

• Buffers of size $B$ arranged at $O(\log N)$ levels

Input stream $\rightarrow$
Technique for Designing Randomized Algorithms


- Buffers of size $B$ arranged at $O(\log N)$ levels

![Diagram of input stream and buffers](image-url)
Technique for Designing Randomized Algorithms


- Buffers of size $B$ arranged at $O(\log N)$ levels

Buffer full $\Rightarrow$ compact:

1. Sort items
Technique for Designing Randomized Algorithms

Used in: [Manku et al. '99, Agarwal et al. '12, Wang et al. '13, Karnin et al. '16]

- Buffers of size $B$ arranged at $O(\log N)$ levels

Buffer full $\Rightarrow$ compact:

1. Sort items
2. Select even or odd indexes with equal probability
   - Promote items at selected indexes
   - Discard the rest
Technique for Designing Randomized Algorithms

Used in: [Manku et al. '99, Agarwal et al. '12, Wang et al. '13, Karnin et al. '16]

- Buffers of size $B$ arranged at $O(\log N)$ levels

Buffer full $\Rightarrow$ compact:

1. Sort items

2. Select even or odd indexes with equal probability
   - Promote items at selected indexes
   - Discard the rest
Technique for Designing Randomized Algorithms


- Buffers of size $B$ arranged at $\mathcal{O}(\log N)$ levels

Buffer full $\Rightarrow$ compact:  
1. Sort items
2. Select even or odd indexes with equal probability
   - Promote items at selected indexes
   - Discard the rest
Technique for Designing Randomized Algorithms

Used in: [Manku et al. '99, Agarwal et al. '12, Wang et al. '13, Karnin et al. '16]

- Buffers of size $B$ arranged at $O(\log N)$ levels

Buffer full $\Rightarrow$ compact:

1. Sort items

2. Select even or odd indexes with equal probability
   - Promote items at selected indexes
   - Discard the rest
Technique for Designing Randomized Algorithms

Used in: [Manku et al. '99, Agarwal et al. '12, Wang et al. '13, Karnin et al. '16]

- Buffers of size $B$ arranged at $\mathcal{O}(\log N)$ levels

Buffer full $\Rightarrow$ compact: 1. Sort items

2. Select even or odd indexes with equal probability
   - Promote items at selected indexes
   - Discard the rest

- A compaction at level $h$ adds $\pm 2^h$ to the error of an item $y$
  iff odd number of items $x \leq y$ compacted
Technique for Designing Randomized Algorithms

Used in: [Manku et al. '99, Agarwal et al. '12, Wang et al. '13, Karnin et al. '16]

- Buffers of size $B$ arranged at $O(\log N)$ levels

Buffer full $\Rightarrow$ compact:

1. Sort items
2. Select even or odd indexes with equal probability
   - Promote items at selected indexes
   - Discard the rest

- A compaction at level $h$ adds $\pm 2^h$ to the error of an item $y$
  iff odd number of items $x \leq y$ compacted
Analysis with a Simple Compactor

Fix item $y$: 

- $R(y) = \text{rank of } y \text{ in the input stream } = \# \text{ of items } x \leq y$
- $\hat{R}(y) = \text{estimated rank of } y$
- $\text{Err}(y) = |R(y) - \hat{R}(y)|$ is the error

Goal: show that $\text{Err}(y) \leq \varepsilon R(y)$ with constant probability
**Analysis with a Simple Compactor**

Fix item $y$:  
- $R(y) = \text{rank of } y \text{ in the input stream} = \# \text{ of items } x \leq y$
- $\hat{R}(y) = \text{estimated rank of } y$
- $\text{Err}(y) = |R(y) - \hat{R}(y)|$ is the error

Goal: show that $\text{Err}(y) \leq \varepsilon R(y)$ with constant probability

- A compaction at level $h$ adds $\pm 2^h$ to the error
  
  *iff* odd number of items $x \leq y$ compacted
Analysis with a Simple Compactor

Fix item $y$:  
- $R(y) = \text{rank of } y \text{ in the input stream} = \# \text{ of items } x \leq y$
- $\hat{R}(y) = \text{estimated rank of } y$
- $\text{Err}(y) = |R(y) - \hat{R}(y)|$ is the error

Goal: show that $\text{Err}(y) \leq \varepsilon R(y)$ with constant probability

- A compaction at level $h$ adds $\pm 2^h$ to the error
  - iff odd number of items $x \leq y$ compacted
- At most $\frac{R(y)}{2^h}$ compactions involving items $x \leq y$
  - rank of $y$ decreases by factor of $\sim 2$ at every level
**Analysis with a Simple Compactor**

Fix item $y$:
- $R(y) = \text{rank of } y \text{ in the input stream} = \# \text{ of items } x \leq y$
- $\hat{R}(y) = \text{estimated rank of } y$
- $\text{Err}(y) = |R(y) - \hat{R}(y)|$ is the error

Goal: show that $\text{Err}(y) \leq \varepsilon R(y)$ with constant probability

- A compaction at level $h$ adds $\pm 2^h$ to the error iff odd number of items $x \leq y$ compacted
- At most $\frac{R(y)}{2^h}$ compactions involving items $x \leq y$
  - rank of $y$ decreases by factor of $\sim 2$ at every level
- Highest level $H(y)$ affecting the error for $y$ satisfies $2^{H(y)} \leq \mathcal{O}(R(y)/B)$
Analysis with a Simple Compactor

Fix item \( y \):

- \( R(y) \) = rank of \( y \) in the input stream = \# of items \( x \leq y \)
- \( \hat{R}(y) \) = estimated rank of \( y \)
- \( \text{Err}(y) = |R(y) - \hat{R}(y)| \) is the error

Goal: show that \( \text{Err}(y) \leq \varepsilon R(y) \) with constant probability

- A compaction at level \( h \) adds \( \pm 2^h \) to the error
  
  \( \text{iff} \) odd number of items \( x \leq y \) compacted

- At most \( \frac{R(y)}{2^h} \) compactions involving items \( x \leq y \)
  
  - rank of \( y \) decreases by factor of \( \sim 2 \) at every level

- Highest level \( H(y) \) affecting the error for \( y \) satisfies \( 2^{H(y)} \leq \mathcal{O}(R(y)/B) \)

Variance of \( \text{Err}(y) \) ≤ \( \sum_{h=0}^{H(y)} 2^{2h} \frac{R(y)}{2^h} \leq 2^{H(y)} R(y) \leq \frac{R(y)^2}{B} \) (up to constant factors)
Analysis with a Simple Compactor

Fix item $y$:

- $R(y) =$ rank of $y$ in the input stream = \# of items $x \leq y$
- $\hat{R}(y) =$ estimated rank of $y$
- $\text{Err}(y) = |R(y) - \hat{R}(y)|$ is the error

Goal: show that $\text{Err}(y) \leq \varepsilon R(y)$ with constant probability

- A compaction at level $h$ adds $\pm 2^h$ to the error
  - iff odd number of items $x \leq y$ compacted
- At most $\frac{R(y)}{2^h}$ compactions involving items $x \leq y$
  - rank of $y$ decreases by factor of $\sim 2$ at every level
- Highest level $H(y)$ affecting the error for $y$ satisfies $2^{H(y)} \leq \mathcal{O}(R(y)/B)$

Variance of $\text{Err}(y) \leq \sum_{h=0}^{H(y)} 2^{2h} \frac{R(y)}{2^h} \leq 2^{H(y)} R(y) \leq \frac{R(y)^2}{B}$ (up to constant factors)

For $\text{Err}(y) \leq \varepsilon \cdot R(y)$ w/ const. probability, we need $\text{Var}[\text{Err}(y)] \leq \varepsilon^2 R(y)^2$

$\Rightarrow$ need to choose $B \sim \frac{1}{\varepsilon^2}$ 😞
Relative compactor

Compaction affecting the error should remove $k$ items $x \leq y$ on average

- $\Rightarrow$ at most $\frac{R(y)}{2^h \cdot k}$ compactions involving items $x \leq y$ at level $h$
Relative compactor

Compaction affecting the error should remove $k$ items $x \leq y$ on average

- $\Rightarrow$ at most $\frac{R(y)}{2^h \cdot k}$ compactions involving items $x \leq y$ at level $h$

- Split buffer into sections

$$B/2 \text{ slots (never compacted)} \quad [\log_2(n/k)] \text{ sections with } k \text{ slots each}$$

- Section $j$ compacted in every $2^j$-th time
Relative compactor

Compaction affecting the error should remove \( k \) items \( x \leq y \) on average

- \( \Rightarrow \) at most \( \frac{R(y)}{2^h \cdot k} \) compactions involving items \( x \leq y \) at level \( h \)

- Split buffer into sections

<table>
<thead>
<tr>
<th></th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( B/2 ) slots (never compacted)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \lceil \log_2(n/k) \rceil ) sections with ( k ) slots each</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Section \( j \) compacted in every \( 2^j \)-th time
- \( B = 2 \cdot k \cdot \lceil \log_2(n/k) \rceil \)
Analysis with Relative Compactor

Fix item $y$:  
- $R(y) = \text{rank of } y \text{ in the input stream} = \# \text{ of items } x \leq y$
- $\hat{R}(y) = \text{estimated rank of } y$
- $\text{Err}(y) = |R(y) - \hat{R}(y)|$ is the error

Goal: show that $\text{Err}(y) \leq \varepsilon R(y)$ with constant probability
Analysis with Relative Compactor

Fix item $y$:  
- $R(y) = \text{rank of } y \text{ in the input stream } = \# \text{ of items } x \leq y$
- $\hat{R}(y) = \text{estimated rank of } y$
- $\text{Err}(y) = |R(y) - \hat{R}(y)|$ is the error
  
  Goal: show that $\text{Err}(y) \leq \varepsilon R(y)$ with constant probability

- A compaction at level $h$ adds $\pm 2^h$ to the error
  
  iff odd number of items $x \leq y$ compacted
Analysis with Relative Compactor

Fix item $y$: • $R(y) =$ rank of $y$ in the input stream $=$ \# of items $x \leq y$
  - $\hat{R}(y) =$ estimated rank of $y$
  - $\text{Err}(y) = |R(y) - \hat{R}(y)|$ is the error

Goal: show that $\text{Err}(y) \leq \varepsilon R(y)$ with constant probability

- A compaction at level $h$ adds $\pm 2^h$ to the error
  - iff odd number of items $x \leq y$ compacted
- At most $\frac{R(y)}{2^h \cdot k}$ compactions involving items $x \leq y$
Analysis with Relative Compactor

Fix item $y$:  
- $R(y) =$ rank of $y$ in the input stream = $\#$ of items $x \leq y$
- $\hat{R}(y) =$ estimated rank of $y$
- Err($y$) = $|R(y) - \hat{R}(y)|$ is the error

Goal: show that Err($y$) $\leq \varepsilon R(y)$ with constant probability

- A compaction at level $h$ adds $\pm 2^h$ to the error  
  iff odd number of items $x \leq y$ compacted
- At most $\frac{R(y)}{2^h \cdot k}$ compactions involving items $x \leq y$
- Highest level $H(y)$ affecting the error for $y$ satisfies $2^{H(y)} \leq O(R(y)/B)$
Analysis with Relative Compactor

Fix item $y$:  
- $R(y) =$ rank of $y$ in the input stream $= \# \text{ of items } x \leq y$
- $\hat{R}(y) =$ estimated rank of $y$
- $\text{Err}(y) = |R(y) - \hat{R}(y)|$ is the error

Goal: show that $\text{Err}(y) \leq \varepsilon R(y)$ with constant probability

- A compaction at level $h$ adds $\pm 2^h$ to the error 
  *iff* odd number of items $x \leq y$ compacted
- At most $\frac{R(y)}{2^h \cdot k}$ compactions involving items $x \leq y$
- Highest level $H(y)$ affecting the error for $y$ satisfies $2^{H(y)} \leq \mathcal{O}(R(y)/B)$

Variance of $\text{Err}(y)$ at most (up to constants)

$$
\sum_{h=0}^{H(y)} 2^{2h} \frac{R(y)}{2^h \cdot k} \leq 2^{H(y)} \frac{R(y)}{k} \leq \frac{R(y)^2}{kB} \leq \frac{R(y)^2}{k^2 \cdot \log(\varepsilon N)}
$$
Analysis with Relative Compactor

Fix item $y$:

- $R(y) = \text{rank of } y \text{ in the input stream} = \# \text{ of items } x \leq y$
- $\hat{R}(y) = \text{estimated rank of } y$
- $\text{Err}(y) = |R(y) - \hat{R}(y)|$ is the error

Goal: show that $\text{Err}(y) \leq \varepsilon R(y)$ with constant probability

- A compaction at level $h$ adds $\pm 2^h$ to the error
  
  iff odd number of items $x \leq y$ compacted

- At most $\frac{R(y)}{2^h \cdot k}$ compactions involving items $x \leq y$

- Highest level $H(y)$ affecting the error for $y$ satisfies $2^{H(y)} \leq O(R(y)/B)$

Variance of $\text{Err}(y)$ at most (up to constants)

$$\sum_{h=0}^{H(y)} 2^{2h} \frac{R(y)}{2^h \cdot k} \leq 2^{H(y)} \frac{R(y)}{k} \leq \frac{R(y)^2}{kB} \leq \frac{R(y)^2}{k^2 \cdot \log(\varepsilon N)}$$

- Choose $k = \frac{1}{\varepsilon \cdot \sqrt{\log(\varepsilon N)}}$, so that $\text{Var}(\text{Err}(y)) \leq \varepsilon^2 R(y)^2$

- Then $B = O\left(\frac{1}{\varepsilon} \cdot \sqrt{\log(\varepsilon N)}\right)$ and $O(\log(\varepsilon N))$ levels $\Rightarrow$ space $O\left(\frac{1}{\varepsilon} \cdot \log^{1.5} \varepsilon N\right)$
Relative Error: Conclusions

Randomized sketch of size $O\left(\frac{1}{\varepsilon} \cdot \log^{1.5} \varepsilon N\right)$ (const. probability of error)

- $\sqrt{\log(1/\delta)}$ dependence on failure probability $\delta$

Lower bound $\Omega\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N\right) \Rightarrow$ gap $\sqrt{\log(\varepsilon N)}$
Relative Error: Conclusions

Randomized sketch of size $O\left(\frac{1}{\varepsilon} \cdot \log^{1.5} \varepsilon N\right)$ (const. probability of error)

- $\sqrt{\log(1/\delta)}$ dependence on failure probability $\delta$

Lower bound $\Omega\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N\right) \Rightarrow$ gap $\sqrt{\log(\varepsilon N)}$

Extensions:
- Handling unknown stream lengths
- Mergeability, and more
- Python code at GitHub

More: paper Relative Error Streaming Quantiles at arXiv (to be updated till WOLA)
Relative Error: Conclusions

Randomized sketch of size $O\left(\frac{1}{\varepsilon} \cdot \log^{1.5} \varepsilon N\right)$ (const. probability of error)

- $\sqrt{\log(1/\delta)}$ dependence on failure probability $\delta$

Lower bound $\Omega\left(\frac{1}{\varepsilon} \cdot \log \varepsilon N\right)$ $\Rightarrow$ gap $\sqrt{\log(\varepsilon N)}$

Extensions:
- Handling unknown stream lengths
- Mergeability, and more
- Python code at GitHub

More: paper *Relative Error Streaming Quantiles* at arXiv (to be updated till WOLA)

Questions welcomed at Slack or pavel.vesely@warwick.ac.uk