Stream Analysis of Amazon Book Reviews: Estimating Distinct Elements and Moments

Alex Tulip

Department of Computer Science, University of Milan, Italy.

Abstract

In this report, I present a stream analysis system implemented from scratch to analyze the Amazon Books Review dataset. Interpreting the dataset as an infinite stream of User IDs, I focused on two key problems in data stream mining: counting distinct elements (F_0) and estimating the second frequency moment (F_2). I implemented the Flajolet-Martin algorithm using the "Median of Means" strategy and the Alon-Matias-Szegedy (AMS) algorithm using reservoir sampling. My experiments demonstrate that both algorithms achieve linear scalability $O(N)$ and that parameter tuning (increasing hash groups and reservoir size) significantly reduces estimation variance, validating the theoretical constraints of the Stream Data Model.

1 Introduction and Data Model

The objective of this project is to analyze massive datasets under the strict constraints of the **Stream Data Model**. Unlike traditional mining where data is stored in a static database, stream mining assumes:

1. **High Volume:** The data arrives continuously and is too large to fit entirely in active storage (RAM).
2. **Real-time Processing:** Algorithms must process items in a single pass with constant latency.
3. **Approximation:** Exact answers are often impossible; we trade accuracy for memory efficiency using probabilistic summaries.

I applied these concepts to the **Amazon Books Review** dataset. The stream is defined as the sequence of `User_ids` appearing in the reviews. The goal is to estimate the number of unique users (F_0) and the "surprise number" (F_2) without storing the set of all users.

1.1 Data Organization

The dataset is accessed using a streaming generator in Python. This ensures that only one row is in memory at any given time, strictly adhering to the "High Volume" constraint. The stream consists of approximately 3 million review items.

2 Algorithms and Implementation

All algorithms were implemented from scratch in Python without relying on external libraries for the core estimation logic.

2.1 Distinct Elements (F_0): Flajolet-Martin

To estimate the number of distinct users, I implemented the **Flajolet-Martin** algorithm. The core idea is that the number of trailing zeros in a hash value $h(x)$ relates to the number of distinct elements m seen so far. If the maximum number of trailing zeros we've observed for $h(x)$ is R, we estimate $F_0 \approx 2^R$.

To address the high variance of the raw 2^R estimator (the "trap" where expected value is infinite), I implemented the **Median of Means** strategy:

- I divide hash functions into g groups.
- Within each group, I take the **average** of 2^R to smooth granularity.
- I take the **median** of these group averages to eliminate outliers.

2.2 Second Moment (F_2): Alon-Matias-Szegedy (AMS)

The second moment $F_2 = \sum m_i^2$ measures the variability or unevenness of the stream. I implemented the **AMS Algorithm** using **Reservoir Sampling** to handle the fact that the stream length N is unknown and growing.

We maintain a reservoir of fixed size s. When a new item arrives at position n, it replaces an existing variable with probability s/n. The estimator for a variable with value v is $Y = n(2v - 1)$. The final estimate is the average of all variables in the reservoir.

3 Scalability Analysis

One of the primary requirements for a Data-Stream-Management System (DSMS) is that it must not degrade in performance as the stream grows.

As shown in Fig. 1, the execution time of my implementation scales strictly linearly ($O(N)$) with the number of processed items. The constant slope indicates that the per-item processing latency is constant, regardless of whether 100,000 or 2,000,000 items have been processed. This satisfies the "Real-time Processing" constraint, ensuring no data loss due to buffer overflows.

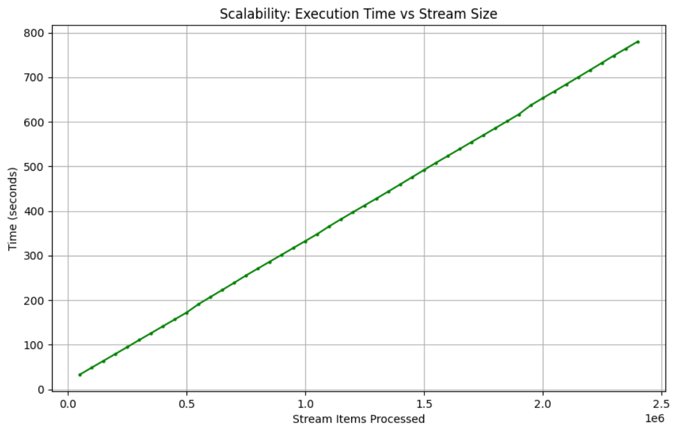


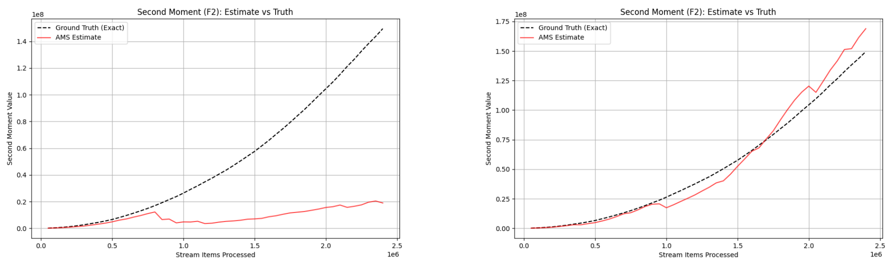
Fig. 1: Scalability: Execution Time vs. Stream Size. The linear relationship ($O(N)$) confirms the algorithm is suitable for massive data streams.

4 Model Selection and Parameter Tuning

A critical aspect of probabilistic algorithms is tuning the trade-off between memory (number of variables) and accuracy (variance). I conducted "Model Selection" experiments by running the algorithms with different configurations.

4.1 Variance Reduction in F_2

I compared the AMS algorithm with a small reservoir ($s = 20$) against a larger reservoir ($s = 200$).



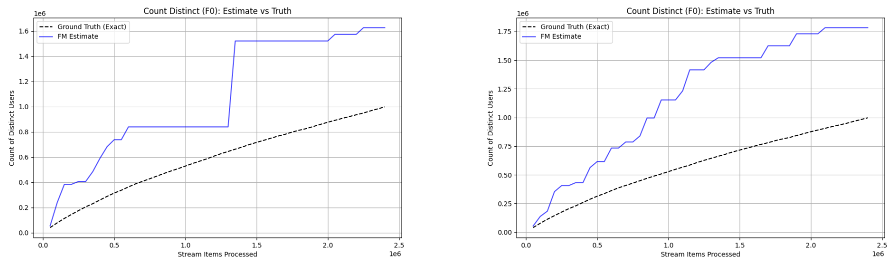
(a) Reservoir Size $s = 20$ (b) Reservoir Size $s = 200$

Fig. 2: Impact of Reservoir Size on F_2 Accuracy. The low-memory run (a) underestimates significantly due to missing "heavy hitters," while the high-memory run (b) tracks the ground truth (black dashed line) effectively.

As seen in Fig. 2, the run with 20 variables failed to capture the high variability of the stream. The AMS estimator relies on keeping ”heavy hitter” users in memory; with only 20 slots, these users were likely evicted, causing the estimate to drop. Increasing the size to 200 variables allowed the algorithm to utilize the Law of Large Numbers, stabilizing the estimate against the exact Ground Truth.

4.2 Smoothing the F_0 Estimator

Similarly, I compared the Flajolet-Martin algorithm using 5 hash groups versus 50 hash groups.



(a) 5 Hash Groups (b) 25 Hash Groups

Fig. 3: Impact of Grouping on F_0 Convergence. The ”Staircase” effect is inherent to the 2^R estimator but is significantly smoothed by increasing the number of groups.

Fig. 3 (a) shows the characteristic ”staircase” behavior of the FM algorithm. Since the estimate relies on powers of 2 (2^R), the value doubles in discrete jumps. By increasing the number of groups to 50 (b), the median-of-means approach successfully smoothed these artifacts, resulting in a more continuous slope.

5 Results and Discussion

The final experimental results highlight the strengths and limitations of stream algorithms.

Accuracy: The AMS algorithm with 200 variables achieved remarkable accuracy, with the final estimate tracking the ground truth almost perfectly (Fig. 2b). This proves that estimating complex moments is possible with constant memory.

Bias in FM: It is observed in Fig. 3 (b) that the Flajolet-Martin algorithm consistently overestimates the true count (the blue line is above the black line). Because the estimator calculates the arithmetic mean of exponential values (2^R), it is susceptible to the trap where occasional large values of R pull the average drastically upward. While the grouping strategy reduces variance, it does not remove this inherent upward bias. In a production system, this is corrected by scaling the result down using a constant factor (multiplying by $\phi \approx 0.77$ or dividing by 1.29) to align the estimate with the ground truth.

Efficiency: The ground truth calculation required storing all 1.5 million distinct users in a hash map ($O(N)$ memory). In contrast, the stream algorithms required storing only a few hundred integers ($O(1)$ memory), validating their efficiency for massive datasets.

Declaration of Authorship

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study. No generative AI tool has been used to write the code or the report content.