

# Reservoir-based Random Sampling with Replacement from Data Stream

Byung-Hoon Park<sup>†</sup>, George Ostrouchov<sup>‡</sup>, Nagiza F. Samatova and Al Geist  
Computer Science and Mathematics Division,  
Oak Ridge National Laboratory<sup>‡</sup>  
P.O. Box 2008, Oak Ridge, TN 37831-6367

## Abstract

Random sampling is a widely accepted basis for estimation from large data sets that outstrip available computer memory. When the data comes as a stream, its total size is potentially infinite and usually only one pass through the data is possible. Reservoir sampling is a method of maintaining a fixed size random sample from streaming data. All reservoir schemes that have been introduced in the past are random sampling without replacement; no duplicates are allowed in a sample. This paper introduces a new method for reservoir sampling with replacement. We first prove that the proposed method indeed maintains a random sample with replacement at any given time. Then we introduce a refined version that significantly speeds up the overall sampling procedure.

**keywords:** data stream mining, random sampling, reservoir sampling, probabilistic method

## 1 Introduction

Sampling is a process of choosing a suitable representative subset from a population of interest so that it can be used for estimating population characteristics. A process produces a *random* sample when all possible samples of the same size have the same probability.

A data stream is a sequence of elements that are ordered by time stamps. More formally, a data stream is a sequence  $\mathcal{S} = \{e_1, e_2, \dots, e_n, \dots\}$ , where the subscripts indicate the order in which the elements are observed. Examples of data streams include scientific simulation data, satellite images, financial data, web-server logs, feedback from sensor networks, etc. Processing a data stream is intrinsically different from processing conventional statically stored databases in several aspects. First, the volume of data is ever increasing and typically unknown *a priori* when a query is posted. Second, immediate or near real-time response is often required. Third, an element is usually processed once,

thus any algorithm which requires multiple scans of the data is not feasible. Inevitably, this necessitates the maintenance of a small footprint summary (For a nice overview of data stream processing, refer to [1]).

Random sampling is a basic, but probably the most widely accepted approach to extracting a representative synopsis from a big database. It particularly serves as a standard technique in building histograms for query optimization [2, 5]. Usage of random sampling for fast incremental maintenance of histograms based on the updates to the databases was discussed in [6], which can be also understood within a context of data streams. Maintaining a random sample of a fixed size, when the population size is unknown, first appears in [3] and better versions are later discussed in [8, 10]. Subsequently more elaborate sampling techniques are introduced [12, 5, 9]. All of these techniques address sampling without replacement; each element from data stream can be chosen only once. Although the difference between sampling with and without replacement is negligible with large population size, there are still situations when sampling with replacement is preferred. For example, consider a situation when a number of queries are posted and each query requires bootstrapped samples from a different time interval.

This paper introduces Reservoir Sampling With Replacement (RSWR) that dynamically maintains a sample of fixed size  $k$  from a data stream. It guarantees that each element from a data stream is selected for any position in the reservoir with equal probability no matter when it is received. The paper is organized as follows. In section 2, we first describe reservoir sampling without replacement. Then we introduce our new sampling scheme RSWR in its simplest form and prove that it indeed produces a random sample. Later sections address how RSWR is improved to speed up the overall sampling process. Finally, section 4 concludes the paper.

---

<sup>†</sup>Corresponding authors [parkbh@ornl.gov](mailto:parkbh@ornl.gov),  
[ostrouchovg@ornl.gov](mailto:ostrouchovg@ornl.gov)

<sup>‡</sup>Oak Ridge National Laboratory is managed by UT-Battelle for the LLC U.S. D.O.E. under Contract No. DE-AC05-00OR22725.

## 2 Reservoir Sampling from a Data Stream

There are situations when prior information about population size is not readily available. Reservoir sampling was introduced to maintain a random sample in such cases. The sample (or reservoir) is updated in such a way that all possible samples have equal probability of being in the reservoir at any time. Before we describe reservoir sampling in depth, let us define our notations first.  $\mathcal{S}_n = \{e_1, e_2, \dots, e_n\}$  is a partial data stream that contains the first  $n$  stream elements.  $\mathcal{R}_n$  denotes the reservoir of size  $m$  after  $e_n$  is received. In fact,  $\mathcal{R}_n$  is a sample from  $\mathcal{S}_n = \{e_1, e_2, \dots, e_n\}$ . Finally,  $\mathcal{J}_n = \{i_1, i_2, \dots, i_m\}$  are the indices of the elements in  $\mathcal{R}_n$ . Now we begin by describing an algorithm that maintains  $\mathcal{J}_n$  as a random sample without replacement from  $\mathcal{S}_n$  for all  $n$ .

### 2.1 Reservoir Sampling Without Replacement

A sampling scheme is said to produce a random sample without replacement of size  $m$  when every possible combination of  $m$  distinct elements has the same probability,  $1/\binom{n}{m}$ , of being selected, where  $n$  is the population size [4]. Reservoir Sampling With-Out replacement (RSWO) exactly guarantees this without prior knowledge of  $n$  (population size). The origins of RSWO can be traced to apparently independent descriptions in [8] (with attribution to Alan Waterman) and in [10] (who provide the first rigorous proof that it works). Later, Vitter [12] provides faster implementations of the algorithm by generating skipping intervals.

Because sampling is without replacement, all items in the sample are distinct and we can keep track of them by the ordered set  $i_1 < i_2 < \dots < i_m$ . RSWO first fills the reservoir, so that  $\mathcal{J}_m = \{1, 2, \dots, m\}$ . For  $n > m$ ,  $e_n$  is selected with probability  $\frac{m}{n}$  replacing a uniformly selected unit in the reservoir. Mcleod and Bellhouse [10] show that this procedure results in equal probabilities, namely  $1/\binom{n}{m}$ , for all possible samples of size  $m$  at any point  $n$  in the stream. Details of RSWO are described in Figure 1.

### 2.2 Reservoir Sampling with Replacement

Sampling with replacement should guarantee that every element in the population has a uniform probability of being chosen for any position in the sample regardless of what is in other positions of the sample. Within the context of sampling from a data stream, this should be understood as: an element  $e_j \in \mathcal{S}_n$  can be placed at any given position of  $\mathcal{R}_n$  with the probability  $\frac{1}{n}$ . This includes the possibility of an item appearing more than once in the reservoir. Formally,

$$(2.1) \quad P(\mathcal{J}_n = \{i_1, i_2, \dots, i_m\}) = \frac{1}{n^m},$$

---

```
RSWO( $\mathcal{R}, m, \mathcal{S}$ )
```

```
begin
```

```
1. for each  $e_n \in \{e_1, e_2, \dots\}$  from  $\mathcal{S}$ 
```

```
2.   if  $n \leq m$ 
```

```
3.     insert  $e_n$  into  $\mathcal{R}_{n-1}$ 
```

```
4.   else
```

```
5.     insert  $e_n$  into  $\mathcal{R}_{n-1}$  with probability of  $\frac{m}{n}$ 
       and at the same time evict an element from
        $\mathcal{R}_{n-1}$  with uniform probability.
```

```
6.   end if
```

```
7.    $\mathcal{R}_n \leftarrow \mathcal{R}_{n-1}$ 
```

```
8. end for
```

```
end
```

---

Figure 1: Reservoir Sampling With-Out Replacement (RSWO).  $\mathcal{R}$  and  $\mathcal{S}$  denote the reservoir of size  $m$  and data stream source respectively.

where  $1 \leq i_j \leq n$  for  $j = 1, \dots, m$ , because there are  $n^m$  different possible samples of size  $m$ . Note that because items are selected with replacement, the reservoir positions are independent and

$$(2.2) \quad P(i_j = k) = \frac{1}{n} \text{ for all } 1 \leq j \leq m, 1 \leq k \leq n$$

is equivalent to the joint statement (2.1). This is in contrast to sampling without replacement, where the positions are dependent and once an element is chosen for a position it cannot be chosen for another position.

RSWR produces a random sample from a population when its size is unknown. It is particularly devised to extract a random sample from a data stream. It resembles RSWO in many aspects; it uses a reservoir  $\mathcal{R}$  to maintain a sample, and performs a probability test to insert a new element into the reservoir. However, it allows that each  $e_i$  can be selected  $k$  times ( $0 \leq k \leq m$ ) following the binomial probability distribution. In fact, for a newly observed element  $e_n$  from  $\mathcal{S}_n$ , it performs  $m$  independent Bernoulli trials, and evicts  $k$  elements from  $\mathcal{R}_{n-1}$  with uniform probabilities, where  $k$  denotes the number of successes in  $m$  trials. Then  $k$  copies of  $e_n$  are into  $\mathcal{R}_{n-1}$ , which becomes  $\mathcal{R}_n$ . Lemmas 2.1 and 2.2 prove that each of  $m$  copies of  $e_n$  is inserted into the  $j$ th position of  $\mathcal{R}_n$  with the probability of  $1/n$ , and any element  $e_r \in \mathcal{S}_n$  stays in the  $j$ th position of  $\mathcal{R}_n$  with the probability of  $1/n$ , which shows that RSWR satisfies (2.1) and (2.2), and therefore provides a random sample with replacement. Details of RSWR are described in Figure 2.

LEMMA 2.1. Let  $e_r$  be the  $j$ th element in  $\mathcal{R}_{n-1}$  (i.e.,

$i_j \in \mathcal{I}_{n-1}$  is  $r$ ). Then after observing the  $n$ th element  $e_n$  from  $\mathcal{S}_n$ , the probability that  $e_r$  will stay in  $\mathcal{R}_n$  is  $\frac{n-1}{n}$ .

**Proof:** If we let  $u$  be the random variable that denotes the number of successes from  $m$  Bernoulli trials for  $e_n$ , the probability that  $e_r$  will stay at the  $j$ th position of  $\mathcal{R}_n$  is

$$\begin{aligned}
& P(e_r \in \mathcal{R}_n, i_j = r | e_r \in \mathcal{R}_{n-1}, i_j = r) \\
&= \sum_{k=0}^m P(u = k) P(e_r \text{ is not evicted} | u = k) \\
&= \sum_{k=0}^m \binom{m}{k} \left(\frac{1}{n}\right)^k \left(\frac{n-1}{n}\right)^{m-k} \left(\frac{m-k}{m}\right) \\
&= \sum_{k=0}^{m-1} \binom{m}{k} \left(\frac{1}{n}\right)^k \left(\frac{n-1}{n}\right)^{m-k} \left(\frac{m-k}{m}\right) \\
&= \sum_{k=0}^{m-1} \frac{m!}{(m-k)!k!} \left(\frac{m-k}{m}\right) \left(\frac{1}{n}\right)^k \left(\frac{n-1}{n}\right)^{m-k} \\
&= \sum_{k=0}^{m-1} \frac{(m-1)!}{(m-k-1)!k!} \left(\frac{1}{n}\right)^k \left(\frac{n-1}{n}\right)^{m-k} \\
&= \left(\frac{n-1}{n}\right) \sum_{k=0}^{m-1} \binom{m-1}{k} \left(\frac{1}{n}\right)^k \left(\frac{n-1}{n}\right)^{m-1-k} \\
&= \left(\frac{n-1}{n}\right) \blacksquare
\end{aligned}$$

LEMMA 2.2. After observing  $e_n \in \mathcal{S}_n$ , the probability that  $e_k$  ( $1 \leq k \leq n$ ) stays in the  $j$ th position of  $\mathcal{R}_n$  is  $\frac{1}{n}$  for any  $j$ . Therefore RSWR is a random sampling with replacement.

**Proof:** By the definition of RSWR,  $e_n$  is inserted into  $j$ th position of  $\mathcal{R}_n$ , if the  $j$ th Bernoulli trial is a success, and its probability is  $\frac{1}{n}$ . For any earlier element  $e_r$  ( $1 \leq r < n$ ), the probability that  $e_r$  is in the  $j$ th position of  $\mathcal{R}_n$  is,

$$\begin{aligned}
& P(e_r \in \mathcal{R}_n, i_j = r) \\
&= P(e_r \in \mathcal{R}_r, i_j = r) P(e_r \in \mathcal{R}_{r+1}, i_j = r | e_r \in \mathcal{R}_r, \\
&\quad i_j = r) \cdots P(e_r \in \mathcal{R}_n, i_j = r | e_r \in \mathcal{R}_{n-1}, i_j = r) \\
&= \frac{1}{r} \times \frac{r}{r+1} \times \frac{r+1}{r+2} \times \cdots \times \frac{n-1}{n} \\
&= \frac{1}{n} \blacksquare
\end{aligned}$$

---

RSWR( $\mathcal{R}, m, \mathcal{S}$ )

begin

1. for each  $e_n \in \{e_1, e_2, \dots\}$  from  $\mathcal{S}$
  2.  $k \leftarrow$  the number of successes from  $m$  independent Bernoulli trials
  3. Evict  $k$  elements from  $\mathcal{R}_{n-1}$  with uniform probability.
  4. Insert  $k$  copies of  $e_n$  into  $\mathcal{R}_{n-1}$
  5.  $\mathcal{R}_n \leftarrow \mathcal{R}_{n-1}$
  6. end
- 

Figure 2: Reservoir Sampling With Replacement (RSWR).  $\mathcal{R}$  and  $\mathcal{S}$  denote the reservoir of size  $m$  and data stream source respectively. In Step 2, the success probability of each Bernoulli trial is  $1/n$ .

### 3 Faster Sampling By Skipping Elements

Given the  $n$ th element  $e_n$ , RSWR inserts on average  $m/n$  copies of it into the reservoir  $\mathcal{R}_n$ . As  $n$  becomes large,  $e_n$  will be dropped (i.e. inserted zero times) with high probability. This observation suggests that we can further refine RSWR by deriving a probability distribution of skipping intervals, i.e. the number of consecutive elements to be dropped. This section introduces RSWR\_SKIP, a variation of RSWR that significantly speeds up the sampling process by skipping elements in a data stream by generating such skipping intervals.

#### 3.1 Generating Skipping Interval Distribution

By  $P(s = k)$ , let us denote the probability of skipping (rejecting) the next  $k$  elements,  $e_{n+1}, \dots, e_{n+k}$  after processing  $e_n$ . In other words, it is the probability that from  $m$  Bernoulli trials for  $e_{n+1}, \dots, e_{n+k}$ , no success is observed.

$$\begin{aligned}
& P(s = k) \\
&= [\prod_{j=1}^k P(\text{Reject } e_{n+j})] P(\text{Accept } e_{n+k+1}) \\
&= \left[ \prod_{j=1}^k \left(1 - \frac{1}{n+j}\right)^m \right] \left[ 1 - \left(1 - \frac{1}{n+k+1}\right)^m \right] \\
&= \left[ \prod_{j=1}^k \left(\frac{n+j-1}{n+j}\right)^m \right] \left[ 1 - \left(1 - \frac{1}{n+k+1}\right)^m \right] \\
&= \left(\frac{n}{n+1}\right)^m \left(\frac{n+1}{n+2}\right)^m \left(\frac{n+2}{n+3}\right)^m \cdots \\
&\quad \left(\frac{n+k-1}{n+k}\right)^m \left[ 1 - \left(1 - \frac{1}{n+k+1}\right)^m \right]
\end{aligned}$$

$$\begin{aligned}
&= \left(\frac{n}{n+k}\right)^m \left[1 - \left(1 - \frac{1}{n+k+1}\right)^m\right] \\
&= \left(\frac{n}{n+k}\right)^m \left[1 - \left(\frac{n+k}{n+k+1}\right)^m\right] \\
&= \left(\frac{n}{n+k}\right)^m - \left(\frac{n}{n+k+1}\right)^m \\
&= n^m \left[\left(\frac{1}{n+k}\right)^m - \left(\frac{1}{n+k+1}\right)^m\right].
\end{aligned}$$

It is not difficult to see that  $P(s = k)$  is indeed a probability distribution

$$\begin{aligned}
&\sum_{k=0}^{\infty} P(s = k) \\
&= \sum_{k=0}^{\infty} n^m \left[\left(\frac{1}{n+k}\right)^m - \left(\frac{1}{n+k+1}\right)^m\right] \\
&= n^m \left[\left(\frac{1}{n}\right)^m - \left(\frac{1}{n+1}\right)^m + \left(\frac{1}{n+1}\right)^m - \dots\right] \\
&= n^m \left(\frac{1}{n}\right)^m \\
&= 1.
\end{aligned}$$

Now consider how to select the number of elements to skip with a simple random experiment. The cumulative distribution function method [7] can be used. The cumulative distribution  $P(s \leq t)$  is written as,

$$\begin{aligned}
&P(s \leq t) \\
&= \sum_{k=0}^t P(s = k) \\
&= \sum_{k=0}^t n^m \left[\left(\frac{1}{n+k}\right)^m - \left(\frac{1}{n+k+1}\right)^m\right] \\
&= n^m \sum_{k=0}^t \left[\left(\frac{1}{n+k}\right)^m - \left(\frac{1}{n+k+1}\right)^m\right] \\
&= n^m \left[\left(\frac{1}{n}\right)^m - \left(\frac{1}{n+1}\right)^m + \left(\frac{1}{n+1}\right)^m - \dots\right] \\
&= n^m \left[\left(\frac{1}{n}\right)^m - \left(\frac{1}{n+t+1}\right)^m\right] \\
&= 1 - \left(\frac{n}{n+t+1}\right)^m
\end{aligned}$$

Given  $q \in (0, 1)$ , a random number from the uniform distribution, the number of elements to skip,  $t$  is given by solving

$$q = 1 - \left(\frac{n}{n+t+1}\right)^m$$

$$\begin{aligned}
\left(\frac{n}{n+t+1}\right)^m &= 1 - q \\
\frac{n}{n+t+1} &= \sqrt[m]{1-q} \\
n+t+1 &= \frac{n}{\sqrt[m]{1-q}} \\
t &= \frac{n}{\sqrt[m]{1-q}} - n - 1,
\end{aligned}$$

and taking the next greater integer. That is, we can skip  $\lceil \frac{n}{\sqrt[m]{1-q}} - n - 1 \rceil$  elements given a random number  $q$ , where  $\lceil \cdot \rceil$  is the *ceiling* operator that returns the next greater integer.

---

SKIP( $n, m$ )

begin

1.  $q \leftarrow$  generate a random number in  $(0,1)$
  2.  $t \leftarrow \lceil \frac{n}{\sqrt[m]{1-q}} - n - 1 \rceil$
  3. return  $t$
- end

RSWR\_SKIP( $\mathcal{R}, m, \mathcal{S}$ )

begin

1.  $n \leftarrow 0$
  2. while the stream flows in
  3.    $t \leftarrow$  SKIP( $n, m$ )
  4.   if  $t > 0$
  5.     skip the next  $t$  elements  $(e_{n+1}, e_{n+2}, \dots, e_{n+t})$
  6.   end if
  7.    $q \leftarrow$  generate a random number in  $(\frac{1}{n+t+1}, 1)$
  8.    $k \leftarrow$  CHOOSE( $\frac{1}{n+t+1}, q$ )
  9.   evict  $k$  elements from  $\mathcal{R}$  with uniform probability
  10.   insert  $k$  copies of  $e_{n+t+1}$  into  $\mathcal{R}$
  11.    $n \leftarrow n + t + 1$
  12. end while
- end
- 

Figure 3: Efficient Reservoir Sampling With Replacement (Skipping version).  $\mathcal{R}$  and  $\mathcal{S}$  denote the reservoir of size  $m$  and data stream source respectively. CHOOSE( $p, q$ ) is a function that implements either binary or sequential search method described in Section 3.2, where  $p$  denotes a success probability of a Bernoulli trial and  $q$  is a random number that determines the number of successes from the  $m$  trials. Note that  $q$  is generated in a way that the chance of zero success is excluded (See step 7).

### 3.2 Search With Cumulative Binomial Probability Distribution

Although RSWR\_SKIP signif-

icantly speeds up sampling process, performing  $m$  Bernoulli trials for every accepted element can still be improved. If  $u$  denotes the number of successes from  $m$  independent Bernoulli trials, then  $P(u > k)$  can be expressed as:

$$(3.3) \quad P(u > k) = \sum_{i=k+1}^m \binom{m}{i} p^i (1-p)^{m-i},$$

where  $p$  is the success probability of each trial.

Now with a random number  $q \in [0,1]$ , we can choose  $k$  (the number of successes) if  $P(u > k+1) < q \leq P(u > k)$ . Using binary search, we can find  $k$  in  $O(\log m)$  steps. However, note that the expected value of  $k$  for  $e_n$  is  $\frac{m}{n}$ . Therefore, in practice, as  $n$  (the number of elements observed) becomes sufficiently large, even for a relatively small  $k$ ,  $P(u > k)$  will be close to zero. Thus, it will be sufficient to check  $P(u > k)$  for the first few  $k = 0, 1, \dots$ . Note that probability (3.3) can be computed much faster using the incomplete beta function [11] if  $m$  is larger than a dozen.

#### 4 Conclusion

A novel method, called RSWR, that dynamically maintains a random sample from an ever-growing data stream is proposed in this paper. RSWR particularly addresses sampling with replacement. We provide a formal proof that RSWR indeed maintains a random sample with replacement. Subsequently, we propose an extended version RSWR\_SKIP that speeds up the sampling process significantly by skipping a number of consecutive elements. We also show that such an interval is easily generated by a simple random experiment. To the best of our knowledge, RSWR is the first sampling scheme that addresses sampling with replacement from a population of unknown size.

RSWR will find immediate application in query processing from data streams. Consider, for example, the case when multiple random samples drawn from different time intervals are desired. This corresponds to the case when multiple queries are handled simultaneously. It will be particularly useful if RSWR can be used under a moving window strategy, where a sample is selected from the last  $k$  elements. In such a case, the removal of expired elements from the reservoir needs to be resolved. We are currently investigating an extension of RSWR in this direction.

#### Acknowledgments

Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL). This work was partially funded by the SciDAC program in the DOE Office of

Advanced Scientific Computing Research. This research used resources of the Center for Computational Sciences at Oak Ridge National Laboratory.

#### References

- [1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. of the 2002 ACM Symp. on Principles of Database Systems (PODS 2002)*, June 2002.
- [2] Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. Random sampling for histogram construction: how much is enough? In *ACM SIGMOD*, pages 436–447, 1998.
- [3] C.T. Fan, M.E. Muller, and I. Rezucha. Development of sampling plans by using sequential (item by item) selection techniques and digital computers. *Journal of the American Statistical Association*, 57:387–402, June 1962.
- [4] W. Feller. *An Introduction to Probability Theory and Its Application*, volume 1. Wiley, New York, 3 edition, 1968.
- [5] Phillip B. Gibbons and Yossi Matias. New sampling-based summary statistics for improving approximate query answers. In *ACM SIGMOD*, pages 331–342, Seattle, WA, June 1998.
- [6] Phillip B. Gibbons, Yossi Matias, and Viswanath Poosala. Fast incremental maintenance of approximate histograms. In Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, and Manfred A. Jeusfeld, editors, *Proc. 23rd Int. Conf. Very Large Data Bases, VLDB*, pages 466–475. Morgan Kaufmann, 25–27 1997.
- [7] William J. Kennedy, Jr. and James E. Gentle. *Statistical Computing*. Marcel Dekker, New York, 1980.
- [8] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, 1981.
- [9] G. Manku and R. Motwani. Approximate frequency counts over data streams. In *The 28th International Conference on Very Large Data Bases*, Hong Kong, China, August 2002.
- [10] A.I. McLeod and D.R. Bellhouse. A convenient algorithm for drawing a simple random sample. *Applied Statistics*, 32(2):182–184, 1983.
- [11] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, January 1993.
- [12] J.S. Vitter. Random sampling with reservoir. *ACM Transactions on Mathematical Software*, 11:37–57, March 1985.