Affirmative Sampling: Theory and Applications

Jérémie Lumbroso
Department of Computer Science, Princeton University, NJ, USA

Conrado Martínez
Department of Computer Science, Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract

Affirmative Sampling is a practical and efficient novel algorithm to obtain random samples of distinct elements from a data stream. Its most salient feature is that the size S of the sample will, on expectation, grow with the (unknown) number n of distinct elements in the data stream. As any distinct element has the same probability to be sampled, and the sample size is greater when the "diversity" (the number of distinct elements) is greater, the samples that Affirmative Sampling delivers are more representative than those produced by any scheme where the sample size is fixed *a priori* – hence its name. Our algorithm is straightforward to implement, and several implementations already exist.

2012 ACM Subject Classification Theory of computation → Data structures design and analysis; Theory of computation → Design and analysis of algorithms; Theory of computation → Sketching and sampling

Keywords and phrases Data streams, Distinct sampling, Random sampling, Cardinality estimation, Analysis of algorithms

Digital Object Identifier 10.4230/LIPIcs.AofA.2022.12

Funding This work has been supported by funds from the MOTION Project (Project PID2020-112581GB-C21) of the Spanish Ministry of Science & Innovation MCIN/AEI/10.13039/501100011033, and by Princeton University, and its Department of Computer Science.

Acknowledgements We want to thank the anonymous reviewers who carefully read the submitted article and made very useful remarks and suggested changes which have allowed us to improve the paper and correct a couple of flawed arguments.

1 Introduction

Drawing random samples from a population is the starting point of any statistical inference, and also in closely related tasks arising in machine learning, information retrieval, data stream analysis, randomized algorithms and many more.

Algorithms to sample elements without replacement from a set of N have a long history [9, 10, 24, 25, 26, 33], but a notable turning point was Vitter's Reservoir Sampling algorithm [32][1] as it was one of the first algorithms not requiring N to be known in advance. After the *reservoir* of size k is filled with the first k elements, it contains at all moments a random sample of size k of the N elements seen so far. This is a useful property that we would also like to have in our sampling algorithms: namely, the ability to return a random sample at any moment.

Despite this, in the context of (large) data streaming, where we deal with streams in which an element can appear many times, Reservoir Sampling may not be the most appropriate tool, as it samples all items with identical probability k/N. If there are a few elements that appear very frequently in the data stream, the reservoir will likely only contain instances of

[1] Refer to Table 1 for a comprehensive comparison of Reservoir Sampling and other existing algorithms, to Affirmative Sampling and its variant both presented in this paper.

these few very frequent elements, and the sample won't be representative enough to make useful statistical inferences (e.g., to infer how many distinct elements there are in the data stream or to make predictions about elements that are not very frequent), or to make these inferences without too much error.

1.1 Novelty of Affirmative Sampling

Our sampling algorithm is novel, as it combines three properties. First, it is an *online algorithm*, that can produce a guaranteed random sample at any time while processing a data stream[2]. Second, it is a *distinct sampling algorithm*, which means it only keeps one copy of a given element, no matter how frequently it may occur in the stream, and therefore avoids being flooded by frequent elements. Third, it is able to *grow the sample size when the number of distinct elements grows*, ensuring we can have a fixed error when inferring population sizes (this is, as far as we know, a completely novel property).

More formally, the data stream $\mathcal{Z} = z_1, \ldots, z_N$ is a (typically very long) sequence of items z_i from some domain or universe $\mathcal{U}$, where there are $n \leq N$ distinct elements (we call n the *cardinality* of $\mathcal{Z}$). Thus underlying $\mathcal{Z}$ we have, on the one hand, the multiset

$$M = M(\mathcal{Z}) = \{x_1^{f_1}, x_2^{f_2}, \ldots, x_n^{f_n}\},$$

where $x_i^{f_i}$ denotes that the i-th distinct element x_i occurs $f_i > 0$ times in $\mathcal{Z}$, and on the other hand, the set X of distinct elements

$$X = X(\mathcal{Z}) = \{x_1, x_2, \ldots, x_n\}.$$

Our goal is to obtain a sample $\mathcal{S} \subset X$ such that any distinct element x_i has the same probability to be sampled as any other. In other words, we want our sampling algorithms to produce *random samples* from the underlying set of distinct elements: any sample $\mathcal{S}$ of size $S = |\mathcal{S}|$ must have probability $1/\binom{n}{S}$ of being returned by the algorithm. Unlike Reservoir Sampling, our algorithm therefore avoids having a sample that could be saturated by frequent elements, by only storing one occurrence of an element in the sample – this is generally called *distinct sampling* (which is the name of a family of algorithms, as well as a specific algorithm [11]).

The problem is challenging because the cardinality n of $\mathcal{Z}$ is unknown, and we would like our algorithm to avoid multiple passes, to spend very little time to process each item in $\mathcal{Z}$ and to use very little memory, e.g., $\Theta(1)$ or $\Theta(\log n)$ (besides the amount of memory for the samples, which will be usually small compared to n, say $\mathcal{O}(\log n)$)[3]. Moreover, the algorithm should not make any statistical assumptions about $\mathcal{Z}$ and it should work online, that is, be able to return a random sample at any given moment of the execution.

Obtaining a random sample of k distinct elements for some fixed value k can be thought as *folklore* (see for instance [6, 8]): using a carefully chosen hash function, it is enough to keep the k distinct elements in $\mathcal{Z}$ with the largest (smallest) hash values seen so far. This method, called bottom-k by several authors has many applications beyond random sampling (see for example [8]). For example, it can be found at the heart of cardinality estimators like in [4, 5]

[2] Nevertheless, the ideas and results in this paper apply in many other contexts.

[3] These requirements, for instance, exclude computing the cardinality n with an initial pass iterating over the stream, and uses some data structure to keep track of elements that have been seen: this would introduce an additional initial pass, which makes the algorithm less time efficient and may not even be an option in online applications; it would require memory proportional to the cardinality, e.g., $\Theta(n)$.

or in the estimation of the similarity of sets [6]. We can assume that the probability that the hash value of some x_i is among the k largest hash values is k/n and thus the corresponding sample will be random. As different occurrences of x_i all will have the same hash value, we will either sample x_i when processing its first occurrence or not all; moreover if some sampled element is evicted from the sample (because some element with a larger hash value is seen and sampled) then that element will not be sampled ever again. The sampling algorithm can thus maintain frequency counters and other useful data about the sample elements because we have them from their very first occurrences; every time we see a new occurrence of an element already in the sample we can update that information accordingly.

▶ **Definition 1.** *We will call* dependable *a sampling algorithm with this property, e.g., that the algorithm can produce exact frequency counts for the elements in its sample.*

Besides this *folklore* algorithm for distinct sampling mentioned above, the first algorithm designed with that purpose in mind is the famed *Distinct Sampling* proposed by Wegman in 1984 and analyzed in depth by Flajolet [11], and popularized by Gibbons[4] [17]. The algorithm was further analyzed by Louchard [28].

The size S of the sample returned by Distinct Sampling is a random variable, and it is used (together with another parameter called the *depth*) to estimate the cardinality n of the data stream. Of course, the samples are random, and they can be used to make statistical inferences about the data stream; Distinct Sampling is also dependable. But the size of the sample is bounded by a fixed value B (the so-called *cache size*), $S \leq B$; because of the way the algorithm works, we have $B/2 \leq \mathbb{E}\{S\} \leq B$ (actually, $B/2 \leq S \leq B$ with high probability).

In this paper we propose a new distinct sampling algorithm called *Affirmative Sampling* (AS, for short). As we have mentioned, the main difference with existing distinct sampling algorithms is that Affirmative Sampling returns random samples of variable size S and $\mathbb{E}\{S\}$ grows with n (despite n is unknown!). This is a very useful property, as larger samples will give more accurate inferences if the size n of the "target" population gets large. As the algorithms previously discussed, AS is dependable (in the meaning introduced by Def. 1) and very easy to understand and implement, with good performance in practice. AS draws upon the ideas behind the cardinality estimator *Recordinality* [20] (briefly reviewed in Subsection 3.3 here) as well as the replacement mechanisms in hiring strategies [2, 20, 19, 21].

1.2 Plan of This Paper

In the first part of the paper (Section 2), we describe AS in detail, discuss its main properties, and then analyze its expected performance (Subsection 2.1). In the second part (Section 3), we discuss and analyze unbiased estimators for the proportion and the absolute number of distinct elements satisfying a certain property (Subsections 3.2 and 3.3, respectively). For example, we might be interested in the proportion or number of elements with relative frequency below 1%, or the number of distinct elements occurring among the last W items seen in the data stream. We also discuss how can we use the algorithm to draw an element close to the median or some other α quantile – assuming some total order to compare elements in the data stream, we want an element that is larger than $\alpha \cdot n$ of the n distinct elements in the data stream (Subsection 3.4). The samples can also be used to accurately estimate, as a by-product, the cardinality n of the data stream (Subsection 3.3).

[4] We will stick to the name *Distinct Sampling* used by Gibbons, instead of *Adaptive Sampling* used by Flajolet and later by other authors.

After that, in Section 4 we discuss variants of the standard AS algorithm; these are simple modifications using slightly different rules to decide when an inspected element becomes part of the sample or not, and in the first case, if some element is evicted from the sample or the sample grows. To investigate these variants we profit from the generality of some of our results (which are not specific to Affirmative Sampling, but to any algorithm drawing random samples of distinct elements) and from the existing literature about the so-called *hiring problem*, a.k.a. *select sets* (see for example [2, 16, 19, 21, 22, 23, 27] and references therein).

We end in Section 5 with some final conclusions and final remarks.

2 The Algorithm

Affirmative Sampling receives a single parameter $k \geq 1$, which we fix in advance, and the data stream $\mathcal{Z}$. If the number n of distinct elements in $\mathcal{Z}$ is equal or smaller than k then the algorithm will read the entire data stream and end with a sample that represents the whole population $X(\mathcal{Z})$. This is rather uninteresting and obviously seldom happening; we can therefore safely assume that $n > k$ (actually, $n \gg k$).

Algorithm 1 Affirmative Sampling: the basic variant.

```
procedure AffirmativeSampling(k, Z)
    fill S with the first k distinct elements (and hash values)
      in the stream Z
    for all remaining z ∈ Z do
        y := hash(z)
        if y < y* ≡ min hash value in S ≡ hash(z*) then
            discard z;
        else if z ∈ S then
            freq[z] := freq[z] + 1; update other statistics of z;
        else if y > k-th largest hash value in S then
            S := S ∪ {z}; freq[z] := 1; ...
        else      ▷ z replaces the element z* with min. hash value
            S := S ∪ {z} \ {z*}; freq[z] := 1; ...
        end if
    end for
    return S
end procedure
```

The initial phase of the algorithm collects in the sample $\mathcal{S}$ the first k distinct elements in the data stream $\mathcal{Z}$, together with their hash values and frequency counts – this entails scanning a subsequence of length $\geq k$ until the $(k+1)$-th distinct element occurs in the data stream. Then we enter the main loop in which we process one by one the remaining items. Let z be the current item. If z is already in $\mathcal{S}$, we just need to increment z's frequency count (and possibly some other associated statistics). Otherwise, we compute a hash value y for z and compare y with the smallest hash value of any element in $\mathcal{S}$. We shall assume here that hash values have enough bits to make the probability of collisions negligible[5] and thus assume

[5] While we cannot rule out collisions on a theoretical basis, we can safely make this assumption on practical grounds.

that every distinct element has a distinct hash value; moreover we will adopt the pragmatic assumption that hash values behave as uniform random variates (see for instance [13]), that is, for any element z and any real value $x \in [0,1]$ we have $\text{hash}(z) \in [0,1]$ and

$$\Pr\{\text{hash}(z) \leq x\} = x.$$

Our first result links the size of the random sample $\mathcal{S}$ returned by AS with the number of records in a random permutation of size n. More specifically with the number of *k-records*; given a permutation σ of size n, we say that i is a (left-to-right) k-record in σ if $\sigma(i) < \sigma(j)$ for at most $k-1$ values of j, $1 \leq j < i$. This is a natural generalization of the well-known notion of records in permutations. Both standard records, $k = 1$, and k-records have been discussed and studied in the literature, see [3] and references therein. Results about the number of k-records in a random permutation were derived in [2] and later largely extended in [21]; although computing the expected number of k-records is a mere exercise, other moments or the probability distribution were not considered in other previous works, as far as we know.

▶ **Lemma 2.** *Let $\mathcal{Y}$ be the permutation of size n induced by the hash values of the corresponding first occurrences of the n distinct elements in a data stream $\mathcal{Z}$. Then the size of the sample $\mathcal{S}$ returned by AS is the number of k-records in $\mathcal{Y}$.*

Proof. Under our assumptions of uniqueness and uniformity of hash values, the sample size increases exactly by one each time that a new distinct element has a hash value that is among the k largest hash values seen so far, that is, when the item is a k-record in the permutation $\mathcal{Y}$. ◀

▶ **Lemma 3.** *Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be the set of distinct elements in the data stream $\mathcal{Z}$. The sample $\mathcal{S}$ returned by AS is random: any subset of size $|\mathcal{S}|$ has exactly the same probability of being returned. In symbols, for any value of s and any subset $\mathcal{A} \subseteq \mathcal{X}$ of size s*

$$\Pr\{\mathcal{S} = \mathcal{A} \,|\, |\mathcal{S}| = s\} = \frac{1}{\binom{n}{s}}.$$

Equivalently, any element $x \in \mathcal{X}$ has the same probability of being sampled.

Proof. Every time we find a new item z whose hash value is larger than the minimum hash value in $\mathcal{S}$ we either add z to $\mathcal{S}$ or replace the element z^* with minimum hash value with z. Thus, if the sample has size S at any given moment, $\mathcal{S}$ will contain the S distinct elements in $\mathcal{Z}$ with largest hash values. By the way, this also proves that AS is dependable (see Def. 1) – the reasoning is exactly the same as in the case of fixed size samples, returning the k elements with largest hash values seen so far.

Since the hash values are random, any subset $\mathcal{A}$ of s distinct elements has the same probability of being the subset of elements with the s largest hash values. ◀

The size $S := S_{n,k}$ of $\mathcal{S}$ is given by the number of k-records in a random permutation of size n. This random variable is very well understood, including its exact probability distribution (see, for instance [21]). Its expected value and variance are

$$\begin{aligned}
\mathbb{E}\{S\} &= k(H_n - H_k + 1) = k\ln(n/k) + k + \mathcal{O}(1) \\
\mathbb{V}\{S\} &= k(H_n - H_k) - k^2(H_n^{(2)} - H_k^{(2)}) = k\ln(n/k) - k + k^2/n + \mathcal{O}(1),
\end{aligned} \tag{1}$$

where $H_n = \sum_{1 \le j \le n} 1/j = \ln n + \mathcal{O}(1)$ denotes the n-th harmonic number and $H_n^{(2)} = \sum_{1 \le j \ln n} (1/j^2) = \pi^2/6 - 1/n + \mathcal{O}(n^{-2})$ denotes the n-th harmonic number of order 2. The probability generating function of S is [21]

$$\sigma_{n,k}(u) = \sum_{\ell \ge 0} \Pr\{S_{n,k} = \ell\} u^\ell = u^k \frac{\binom{n+k(u-1)}{n}}{\binom{ku}{k}}, \tag{2}$$

from which the expected value and the variance above can be easily derived. Likewise, it's easy to show, albeit cumbersome, that for any $r \ge 0$, $\mathbb{E}\{S^{\underline{r}}\} = \Theta((\log n)^r)$ (we give a proof in Appendix A). From the explicit expression for the probability distribution of S one can also show (see Subsection 3.2)

$$\mathbb{E}\left\{\frac{1}{S}\right\} \sim \frac{1}{\mathbb{E}\{S\}} \sim \frac{1}{k \ln(n/k)}.$$

This last result will be fundamental in our analysis of accuracy of inferences made from the samples drawn using AS (in particular, see Subsection 3.2).

2.1 Complexity

In order to make sure that hash values collide with negligible probability we need that hash values are $\Theta(\log n)$ bits long. A logarithmic number of bits is also necessary for each of the elements in the sample (as they are distinct!), therefore the sample needs total memory $\Theta(|\mathcal{S}| \log n)$; in expectation this is $\Theta(k \log^2 n)$. To support efficient querying and removal of the element with minimum hash value and of the element with k-th largest hash value, one easy and efficient solution is to store the hash values of the sample elements in a min-heap $\mathcal{H}_1$ for the k elements with largest values and another min-heap $\mathcal{H}_2$ for the remaining $|\mathcal{S}| - k$ elements. These two heaps need $\mathcal{O}(|\mathcal{S}| \log n)$ bits. We also need an efficient way to check if a given element belongs or not to the sample; for that we can store the elements of the sample in a hash table (using a hash function which is completely independent and different from the hash function used by the AS algorithm!). That hash table will need $\Theta(|\mathcal{S}| \log n)$ bits. In summary, the memory consumption is a constant factor from the memory needed for the sought result (the size of the sample in bits).

As for the time complexity of the algorithm, the first phase requires time $\mathcal{O}(N_k)$ where $N_k \ge k$ is the number of elements N_k until the first occurrence of the $(k+1)$-th distinct element. At the end of this phase, the min-heap $\mathcal{H}_2$ is empty and the min-heap $\mathcal{H}_1$ contains the first k distinct hash values; it can be constructed in $\Theta(k)$ right at the end of the first phase. Then, during the second phase we process the remaining $N - N_k$ items. Most of them are immediately discarded. Either their hash value is below the minimum hash value in the sample – just check the top of $\mathcal{H}_2$ in time $\Theta(1)$ – or they are already in the sample – this query also needs time $\Theta(1)$ on average using the hash table. Hence these elements can be processed in constant expected time per item. On the other hand, for each of the $S = |\mathcal{S}|$ elements in the table we will have incurred $\mathcal{O}(\log S)$ time to add them; moreover there will be F_n items that were added to the sample and later kicked out from the sample (never to be sampled again), and we also incur time $\mathcal{O}(\log S)$ to add them to the sample and to remove them later. Indeed, when an element has a hash value among the largest k hash values it will be inserted in $\mathcal{H}_1$ (time $\mathcal{O}(\log k)$), but before that the minimum of $\mathcal{H}_1$ must be removed (time $\mathcal{O}(\log k)$) and then added to $\mathcal{H}_2$ (in time $\mathcal{O}(\log(|\mathcal{S}| - k)) = \mathcal{O}(\log |\mathcal{S}|)$). For elements with hash value smaller that the k-th largest hash value but larger than the minimum hash value in the sample (check the top of $\mathcal{H}_1$ and $\mathcal{H}_2$ in constant time), we must remove the minimum in $\mathcal{H}_2$ and add then the new element to $\mathcal{H}_2$ in time $\mathcal{O}(\log |\mathcal{S}|)$.

Let $C(n, N; k)$ denote the complexity of AS with parameter k to process a data stream of length N that contains n distinct elements. Then we can write

$$C(n, N; k) = \Theta(N + k) + \sum_{k<i\leq n} C_i(n, N; k) = \Theta(N) + \sum_{k<i\leq n} C_i(n, N; k)$$

where $C_i(n, N; k)$ is the cost of processing the first occurrence of the i-th distinct element; it is either 0 if the element is discarded (because its hash value is below the minimum hash value in the current sample) or $\mathcal{O}(\log(S_i + 1))$ if it is added to the sample, where S_i denotes the size of the sample just before processing the i-th distinct element. This cost $\mathcal{O}(\log S_i)$ accounts for the addition of the i-th distinct element to the sample (either to $\mathcal{H}_1$ or $\mathcal{H}_2$, which ever corresponds) and the eventual transfer of one element from $\mathcal{H}_1$ to $\mathcal{H}_2$ or the removal of one element from $\mathcal{H}_2$ (and hence from $\mathcal{S}$). We say $C_i = 0$ if the element is discarded because the cost of processing that element is already accounted for in the $\Theta(N)$ term. Likewise the sum starts at $i = k + 1$ because the cost of processing the first k distinct elements is also accounted for within the $\Theta(N + k)$ term. Call Y_i the indicator random for the event "the i-th distinct element is added to the sample". Then

$$\begin{aligned}\mathbb{E}\{C(n, N; k)\} &= \Theta(N) \\ &\quad + \sum_{k<i\leq n} \Big\{ \mathbb{E}\{C_i \,|\, Y_i = 1\} \Pr\{Y_i = 1\} + \mathbb{E}\{C_i \,|\, Y_i = 0\} \Pr\{Y_i = 0\} \Big\} \\ &= \Theta(N) + \mathcal{O}\left(\sum_{k<i\leq n} \mathbb{E}\{\log S_i\} \Pr\{Y_i = 1\} \right)\end{aligned}$$

The sample size never decreases, hence $S_1 \leq S_2 \leq \cdots S_n \equiv S$; on the other hand $\Pr\{Y_i = 1\} = \mathbb{E}\{Y_i\} = S_i/i$ (it is the probability that the element is among the S_i largest elements of the i elements seen so far) so we can write

$$\begin{aligned}\mathbb{E}\{C(n, N; k)\} &= \Theta(N) + \mathcal{O}\left(\mathbb{E}\{\log S\} \sum_{k<i\leq n} \mathbb{E}\{Y_i\} \right) \\ &= \Theta(N) + \mathcal{O}\left(\mathbb{E}\left\{ \mathbb{E}\{\log S\} \sum_{k<i\leq n} \frac{S_i}{i} \right\} \right) \\ &= \Theta(N) + \mathcal{O}\left(\mathbb{E}\{\log S\} \sum_{k<i\leq n} \mathbb{E}\left\{ \frac{S_i}{i} \right\} \right) \\ &= \Theta(N) + \mathcal{O}\left(\mathbb{E}\{\log S\} \mathbb{E}\{S\} \sum_{k<i\leq n} \frac{1}{i} \right) \\ &= \Theta(N) + \mathcal{O}\left(k \log^2(n/k) \mathbb{E}\{\log S\} \right).\end{aligned}$$

As $\log x$ is concave, Jensen's inequality gives us $\mathbb{E}\{\log S\} \leq \log \mathbb{E}\{S\}$ and putting everything together we find

$$\mathbb{E}\{C(n, N; k)\} = \mathcal{O}\left(N + k \log^2(n/k) \log\log(n/k) \right). \tag{3}$$

3 Making Inferences about the Data Stream

3.1 Frequent Items vs. Frequent Properties

It is important to distinguish between the frequency of an item x and frequent properties P. Consider some property P and the subset $X_P = \{x \in \mathcal{Z} \mid x \text{ satisfies } P\}$ of distinct elements in $\mathcal{Z}$ that satisfy P; we say P is frequent if $n_P = |X_P| = \Theta(n)$. Notice that some properties, like being a very frequent element, can't be a frequent property; for instance, at most $\lfloor 1/c \rfloor$ distinct elements might have relative frequency $\geq c$. As an important consequence, estimating with fixed size samples the proportion of elements in $\mathcal{Z}$ that satisfy P, that is, n_P/n, can be accurately done only if P is frequent.

Distinct Sampling, with its fixed sample size, is thus not a good choice if we want to make inferences about frequent elements (in that case we should sample with probability proportional to the frequency) or the property of interest is not frequent. If not enough elements in the data stream satisfy the property, the samples of variable size drawn by AS can come to the rescue.

3.2 Estimating Proportions

Consider some property $P(x)$ depending only on the occurrences of x in the data stream $\mathcal{Z}$, and let n_P be the number of distinct elements in the data stream satisfying P. We make this restriction as we want to ensure that determining whether x satisfies P or not can be efficiently done and requires little memory.

Let

$$\vartheta_P := \frac{n_P}{n}$$

denote the fraction of elements that satisfy P. If we take a random sample of $S = |\mathcal{S}|$ distinct elements, now with $S > 0$ a random variable, then the probability that there are S_P elements in the sample that satisfy P is given by the hypergeometric distribution

$$\frac{\binom{n_P}{S_P}\binom{n-n_P}{S-S_P}}{\binom{n}{S}}. \tag{4}$$

Let us assume in the computations below and for the remaining of the paper that $n \geq S \geq k \geq 2$, that is, that the sampling algorithm will return at least $k \geq 2$ distinct elements. Otherwise, if the data stream contains less than k distinct element, the sample contains **all** distinct elements in the data stream and their relevant statistics and we can answer queries exactly.

The following theorem is well-known and can be found in many textbook on statistics, see for instance [7].

▶ **Theorem 4.** *Let $\hat{\vartheta}_P := S_P/S$ and assume that $S > 0$. Then*

$$\mathbb{E}\left\{\hat{\vartheta}_P\right\} = \vartheta_P.$$

That is, $\hat{\vartheta}_P$ is an unbiased estimator for ϑ_P, no matter what the probability distribution of $S = |\mathcal{S}|$ is; in particular, it is also true if $S = M$ for some fixed constant $M > 0$.

Proof. We expand $\mathbb{E}\left\{\hat{\vartheta}_P\right\}$ using the formula of total probability conditioning on the events $S=\ell$, and plug-in the hypergeometric distribution for the probability that S_P of the $S=\ell$ elements drawn at random without replacement satisfy the property P. Then

$$\begin{aligned}
\mathbb{E}\left\{\hat{\vartheta}_P\right\} &= \sum_{\ell>0}\mathbb{E}\left\{\frac{S_P}{S}\,\middle|\, S=\ell\right\}\Pr\{S=\ell\} = \sum_{\ell>0}\Pr\{S=\ell\}\sum_{j=0}^{\ell}\frac{j}{\ell}\frac{\binom{n_p}{j}\binom{n-n_p}{\ell-j}}{\binom{n}{\ell}}\\
&= \sum_{\ell>0}\Pr\{S=\ell\}\sum_{j=1}^{\ell}\frac{n_p}{n}\frac{\binom{n_p-1}{j-1}\binom{n-n_p}{\ell-j}}{\binom{n-1}{\ell-1}}\\
&= \frac{n_p}{n}\sum_{\ell>0}\Pr\{S=\ell\}\sum_{j=0}^{\ell-1}\frac{\binom{n_p-1}{j}\binom{n-n_p}{\ell-j-1}}{\binom{n-1}{\ell-1}} = \frac{n_p}{n}\sum_{\ell>0}\Pr\{S=\ell\}\\
&= \frac{n_p}{n} = \vartheta_P,
\end{aligned}$$

where we have used Vandermonde's convolution in the final step to simplify the summation on j, yielding the statement of the theorem. ◀

Quite intuitively, the accuracy of the estimator $\hat{\vartheta}_P$ will depend of the size of the sample. Indeed, for $\mathbb{V}\left\{\hat{\vartheta}_P\right\}$ and now assuming that $n \geq S \geq 2$ we get the following result, which can also be found, for instance, in [31, 7] for the case where the size of the sample S is fixed and not a random variable itself.

▶ **Theorem 5.** *Let $\hat{\vartheta}_P := S_P/S$ and assume that $S \geq 2$. Then*

$$\mathbb{V}\left\{\hat{\vartheta}_P\right\} = \frac{n_p(n-n_p)}{n(n-1)}\cdot\left(\mathbb{E}\left\{\frac{1}{S}\right\}-\frac{1}{n}\right).$$

Proof. We start computing $\mathbb{E}\left\{(\hat{\vartheta}_P)^2\right\}$ following the same steps as in the proof of Theorem 4.

$$\begin{aligned}
\mathbb{E}\left\{\hat{\vartheta}_P^2\right\} &= \sum_{\ell>0}\mathbb{E}\left\{\frac{{S_P}^2}{S}\,\middle|\, S=\ell\right\}\Pr\{S=\ell\} = \sum_{\ell>0}\Pr\{S=\ell\}\sum_{j=0}^{\ell}\frac{j^2}{\ell^2}\frac{\binom{n_p}{j}\binom{n-n_p}{\ell-j}}{\binom{n}{\ell}}\\
&= \sum_{\ell>0}\Pr\{S=\ell\}\sum_{j=1}^{\ell}\frac{j}{\ell}\frac{n_p}{n}\frac{\binom{n_p-1}{j-1}\binom{n-n_p}{\ell-j}}{\binom{n-1}{\ell-1}}\\
&= \frac{n_p}{n}\sum_{\ell>0}\frac{1}{\ell}\Pr\{S=\ell\}\sum_{j=0}^{\ell-1}j\frac{\binom{n_p-1}{j}\binom{n-n_p}{\ell-j-1}}{\binom{n-1}{\ell-1}}\\
&= \frac{n_p}{n}\sum_{\ell>0}\frac{1}{\ell}\Pr\{S=\ell\}\left\{\sum_{j=0}^{\ell-1}(j-1)\frac{\binom{n_p-1}{j}\binom{n-n_p}{\ell-j-1}}{\binom{n-1}{\ell-1}}+\sum_{j=0}^{\ell-1}\frac{\binom{n_p-1}{j}\binom{n-n_p}{\ell-j-1}}{\binom{n-1}{\ell-1}}\right\}\\
&= \frac{n_p}{n}\sum_{\ell>0}\frac{1}{\ell}\Pr\{S=\ell\}\frac{1}{\binom{n-1}{\ell-1}}\left\{(n_P-1)\binom{n-2}{\ell-2}+\binom{n-1}{\ell-1}\right\}\\
&= \frac{n_p}{n}\sum_{\ell>0}\frac{1}{\ell}\Pr\{S=\ell\}\left\{\frac{(n_P-1)(\ell-1)}{n-1}+1\right\}\\
&= \frac{n_P(n_P-1)}{n(n-1)}+\frac{n_P(n-n_P)}{n(n-1)}\mathbb{E}\left\{\frac{1}{S}\right\}
\end{aligned}$$

Hence, assuming that $n \geq S \geq 2$, we have

$$\mathbb{V}\left\{\hat{\vartheta}_P\right\} = \frac{n_p(n-n_p)}{n(n-1)} \cdot \left(\mathbb{E}\left\{\frac{1}{S}\right\} - \frac{1}{n}\right). \qquad \blacktriangleleft$$

If the behavior of the random variable S is smooth enough[6] and $\mathbb{E}\{S\} \to \infty$ when $n \to \infty$ then the accuracy of the estimator $\hat{\vartheta}_P$ will improve, as the variance will decrease and tend to 0 as $n \to \infty$. Indeed, the standard error of $\hat{\vartheta}_P$ satisfies

$$\mathrm{SE}\{\hat{\vartheta}_P\} := \frac{\sqrt{\mathbb{V}\left\{\hat{\vartheta}_P\right\}}}{\vartheta_P} \sim \sqrt{\frac{(1-\vartheta_P)}{\vartheta_P}\,\mathbb{E}\left\{\frac{1}{S}\right\}}. \tag{5}$$

For the samples drawn by AS we have $\mathbb{E}\{1/S\} \sim 1/\mathbb{E}\{S\}$. Indeed, we can start from the bivariate generating function [21] for $S = S_{n,k}$

$$S_k(z,u) = \sum_{\ell\geq 0}\sum_{n\geq 0} \Pr\{S_{n,k} = \ell\}\binom{n}{\ell}u^\ell z^n = \frac{(zu)^k}{(1-z)^{ku+1}}. \tag{6}$$

Integrating $S_k(z,u)/u$ we can recover $\mathbb{E}\{1/S\}$; there is an explicit primitive in terms of the confluent hypergeometric functions, namely Whittaker's $M_{\mu,\nu}(z)$ function [1], and it can be shown that around the singularity $z = 1$ we have

$$\int_0^1 \frac{1}{u}S_k(z,u)\,du \sim_{z\to 1} \frac{z^k}{k(1-z)^{k+1}\ln\left(\frac{1}{1-z}\right)} + \mathcal{O}\left((1-z)^{-k-1}\left(\log\frac{1}{1-z}\right)^{-2}\right),$$

hence, using standard singularity analysis results (see, for instance [14, 15]) we finally get

$$\mathbb{E}\left\{\frac{1}{S}\right\} \sim \frac{1}{\binom{n}{k}}[z^{n-k}]\frac{1}{k(1-z)^{k+1}\ln\left(\frac{1}{1-z}\right)} \sim \frac{1}{k\ln(n/k)}.$$

Then, roughly speaking, the standard error approaches 0 whenever $n/(n_P\,\mathbb{E}\{S\}) \to 0$ and n goes to ∞, which is indeed the case for AS if $n_P = \Omega\left(\frac{n}{\log n}\right)$. In many cases we will be intested in properties such that $n_P = \Theta(n)$, hence the standard error of $\hat{\vartheta}_P$ will decrease and go to 0 as $1/\sqrt{\mathbb{E}\{S\}}$, a rate of decay which we find in many estimators in data stream analysis (e.g., Probabilistic Counting [13], HyperLogLog [12], to name a few) where instead of $\mathbb{E}\{S\}$ we have M, the fixed size of the memory used by those estimators.

3.3 Estimating Cardinalities

We can also estimate the number n_P of elements that satisfy P, not just the proportion n_P/n. For that we can use

$$C_1 := \frac{S_P}{S}\left(k\,(1+1/k)^{S-k+1} - 1\right)$$

or

$$C_2 := \frac{S_P}{S}\frac{S-1}{1-Y_{(S)}},$$

where S_P and S are as in the previous section and $Y_{(S)}$ is the smallest hash value in the sample (the S-th largest hash value in the data stream). Indeed $\mathbb{E}\{C_1\} = \mathbb{E}\{C_2\} = n_p$.

[6] In the sense that $\mathbb{E}\{1/S\} = \mathcal{O}(1/\mathbb{E}\{S\})$; by Jensen's inequality we have only $1/\mathbb{E}\{S\} \leq \mathbb{E}\{1/S\}$.

For the first estimator C_1, recall that the size of the sample $S = |\mathcal{S}|$ returned by AS coincides with the number of k-records in the random permutation of size n induced by the first occurrences of the hash values of the n distinct elements in the data stream. Then the estimator

$$R := k\left(1+\frac{1}{k}\right)^{|\mathcal{S}|-k+1} - 1,$$

called RECORDINALITY, is an unbiased estimator of n (see [20]), that is, $\mathbb{E}\{R\} = n$.

To compute R we would not need to collect the sample $\mathcal{S}$ collected by AS: tracking the k distinct elements in the data stream with largest hash value and how many times we have updated that table would suffice – that is, we would need significantly less memory to achieve the same accuracy to estimate the cardinality. However, if our main goal is to draw a random sample of size growing with n – as drawn by AS– we can get, as a by-product, estimates of the cardinality of the data stream "for free," as well as estimates of the absolute number n_P of elements that satisfy a property P. Indeed,

$$\begin{aligned}
\mathbb{E}\{C_1\} &= \mathbb{E}\left\{\frac{S_P}{S}R\right\} = \sum_{\ell>0}\Pr\{S=\ell\}\,\mathbb{E}\left\{\frac{S_P}{S}R \,\middle|\, S=\ell\right\} \\
&= \sum_{\ell>0}\Pr\{S=\ell\}\sum_{j=0}^{\ell}\frac{j}{\ell}\left(k\left(1+\frac{1}{k}\right)^{\ell-k+1}-1\right)\frac{\binom{n_P}{j}\binom{n-n_P}{\ell-j}}{\binom{n}{\ell}} \\
&= \sum_{\ell>0}\Pr\{S=\ell\}\left(k\left(1+\frac{1}{k}\right)^{\ell-k+1}-1\right)\sum_{j=0}^{\ell}\frac{j}{\ell}\frac{\binom{n_P}{j}\binom{n-n_P}{\ell-j}}{\binom{n}{\ell}} \\
&= \sum_{\ell>0}\Pr\{S=\ell\}\left(k\left(1+\frac{1}{k}\right)^{\ell-k+1}-1\right)\frac{n_P}{n} = \frac{n_P}{n}\,\mathbb{E}\{R\} = n_p.
\end{aligned}$$

Alternatively we can extend the well-know KMV estimator [4, 5, 29] to samples of varying-size. KMV has a parameter $k \geq 2$, fixed in advance, and uses the bottom-k algorithm, that is, collects a table with the k distinct elements with smallest hash values seen so far. Then the largest hash value $Y_{(k)}$ in the table is the k-th smallest hash value in the data stream, that is, the k-th smallest number in a set of n random numbers independently and uniformly drawn in $[0,1]$. Hence

$$Z_k := \frac{k-1}{Y_{(k)}}$$

is an unbiased estimator of n and its standard error is $\mathcal{O}(1/\sqrt{k-2})$.

But once you use AS you have a sample of size $S = |\mathcal{S}|$, with $\mathbb{E}\{S\} = k\ln(n/k)$ and therefore using the smallest hash value in the sample (the S-th largest hash value in the data stream) we can get sharper cardinality estimations. Indeed, if we denote $Y_{(S)}$ the smallest hash value in the sample[7] then $Z := (S-1)/(1-Y_{(S)})$ is an unbiased estimator of

[7] AS is formulated in terms of elements with largest hash values, whereas KMV is formulated in terms of the k-th smallest hash value in the data stream; we have adapted the estimator to work with the k-th largest hash value in the data stream instead. We can therefore use the smallest in the sample $Y_{(S)}$ plugging the value $1 - Y_{(S)}$ in the KMV estimator. Alternatively, AS could be reformulated in terms of the elements with smallest hash values; all its properties would remain unaltered or could easily be adapted, but we have preferred to keep the original formulation in terms of largest hash values, also to honor the inspiration drawn from the RECORDINALITY algorithm.

n, while the standard error goes to 0 as $n \to \infty$ (admittedly, at a very slow rate, i.e., it is $\mathcal{O}(\log^{-1/2} n)$). We do not use here the subscript k in the estimator, as the sample is now of size S, and to distinguish it from the estimator Z_k based in the k-th order statistics of the set of hash values in the sample, for a fixed k.

In what follows we assume that $n \geq k \geq 2$, hence $S \geq k$ (or $\Pr\{S < k\} = 0$). For the expectation we have

$$\begin{aligned}\mathbb{E}\{Z\} &= \sum_{\ell \geq k} \mathbb{E}\{Z \mid S = \ell\} \Pr\{S = \ell\} = \sum_{\ell \geq k} \Pr\{S = \ell\} \mathbb{E}\left\{\frac{(\ell-1)}{1 - Y_{(\ell)}}\right\} \\ &= n \sum_{\ell \geq k} \Pr\{S = \ell\} = n.\end{aligned}$$

Likewise

$$\begin{aligned}\mathbb{E}\left\{Z^2\right\} &= \sum_{\ell \geq k} \mathbb{E}\left\{Z^2 \mid S = \ell\right\} \Pr\{S = \ell\} = \sum_{\ell \geq k} \Pr\{S = \ell\} \mathbb{E}\left\{\frac{(\ell-1)^2}{(1 - Y_{(\ell)})^2}\right\} \\ &= \sum_{\ell \geq k} \Pr\{S = \ell\} \frac{n^{\underline{2}}}{(\ell-2)} = n(n-1)\,\mathbb{E}\left\{\frac{1}{S-2}\right\},\end{aligned}$$

and

$$\mathbb{V}\{Z\} = n(n-1)\,\mathbb{E}\left\{\frac{1}{S-2}\right\} - n^2.$$

It turns out that $\mathbb{E}\{1/(S-2)\} \sim (k \ln(n/k))^{-1}$, hence

$$\mathrm{SE}\{Z\} = \sqrt{\mathbb{E}\{1/(S-2)\}\,(1-1/n) - 1} \sim \frac{1}{\sqrt{k \ln(n/k)}}.$$

Using Z we can also obtain accurate estimates of the number n_P of elements that satisfy P. Following the same steps as in the computation of $\mathbb{E}\{C_1\}$ we obtain

$$\mathbb{E}\{C_2\} = \mathbb{E}\left\{\frac{S_P}{S} Z\right\} = n_P,$$

as we stated at the beginning of this subsection.

3.4 Finding Approximate α-Quantiles

We can use AS to estimate the median and other α-quantiles in the data stream – the rank of an element x being the number of **distinct** elements smaller or equal to x.

Let x^* the α-quantile in the random sample $\mathcal{S}$ returned by AS. That is, $\lceil \alpha \cdot S \rceil$ elements in $\mathcal{S}$ are smaller or equal than x^*. These elements have the property of being $\leq x^*$, and the same proportion of elements in $\mathcal{Z}$ will have that property. That is, the expected value of $\frac{\lceil \alpha \cdot S \rceil}{S} = n_P/n$ is $\alpha + \xi(\alpha, S)$, with $\xi(\alpha, S) \to 0$ as $S \to \infty$. Therefore, the expected rank of x^* in the data stream is $\alpha \cdot n + \xi(\alpha, S) \cdot n \approx \lceil \alpha \cdot n \rceil$.

4 α-Affirmative Sampling: Producing Samples of Size n^α

Instead of using the k-th largest hash value in the sample to decide if a new item is added or not to the sample we can use a different criterion, for instance add it if the hash value is above $(100 \cdot \alpha)\%$ of the hash values. These strategies have also been widely studied (see [16, 22, 19, 23, 27] and the references therein) in the context of the so-called *hiring problem*.

The expected size of the sample when we add elements (not replacing any other, but making the sample grow) if their hash value is above $(100 \cdot \alpha)\%$ of the hash values will be $\Theta(n^\alpha)$. All collected elements with hash value above or equal to the one that defines the "threshold" constitute a random sample of size $\Theta(n^\alpha)$, and there would be no need to use the replacement mechanism, as long as we already get a sample of growing size. However, the algorithm needs to keep not only the elements above the "threshold", but all the elements which were above the "threshold" at some point along the execution of the algorithms. If we apply the replacements mechanism then we guarantee that all collected items, not just those above the threshold, constitute a random sample. This means only a little additional effort, as the expected number of replacements will only be $\Theta(n^\alpha)$ (see [22, 19]). We shall call the resulting algorithm α-*Affirmative Sampling* (α-AS, for short).

The average complexity of processing a data stream of N elements (of which n are distinct) with α-AS will be $\mathcal{O}(N + n^\alpha \log n)$, the analysis follows similar steps to those in Subsection 2.1. The expectation of both $S = |\mathcal{S}|$ and F_n (the number of replacements) is $\Theta(n^\alpha)$. Precise estimates of the constant factor in $\mathbb{E}\{S\}$ and $\mathbb{E}\{F_n\}$ exist, but there are some subtleties that depend on the exact definition of what is the element defining the threshold. For instance, if we add elements whenever their hash value is above the median, it is clear what to do if the size of the sample is odd; but if the size of the sample is even, which is the element that sets the threshold? We refer the reader to [22] or [23] for a more detailed discussion of this issue.

Most results in Section 3 apply, in particular, estimating the proportion of distinct elements that satisfy some property or finding elements close to the median and other quantiles of the data stream. Estimating the absolute number of distinct elements that satisfy P or the cardinality of the data stream is also possible using the generalization of KMV to samples of variable size (that is, the estimators C_2 and Z, respectively, using the smallest hash value in the sample).

To compute the variance and standard error in the estimation of the proportion of distinct elements that satisfy P we would need to know $\mathbb{E}\{1/S\}$; we conjecture that $\mathbb{E}\{1/S\} = \Theta(1/\mathbb{E}\{S\}) = \Theta(n^{-\alpha})$, but because of the lack of information about the distribution of S, we haven't proved it. In the case of the rule "above the median" [22] we have the exact expressions for $\Pr\{S = j\}$. Using the results and techniques there, it is easy to show that

$$\begin{aligned}\mathbb{E}\{1/S\} = \sum_{j\geq 0} j^{-1} \Pr\{S = j\} &\sim \sum_{\ell=1}^{n^{1/2+\epsilon}} \frac{1}{n} e^{-\ell^2/n}(1 + \mathcal{O}(1/\ell)) \\ &\sim \frac{1}{\sqrt{n}} \int_0^\infty e^{-x^2}\, dx = \frac{1}{2}\sqrt{\frac{\pi}{n}},\end{aligned}$$

which is proportional to $1/\mathbb{E}\{S\}$; indeed, $\mathbb{E}\{S\} = \sqrt{\pi n} + \mathcal{O}(1)$ [22].

Likewise, an estimator of the cardinality of the data stream based upon the size S of the sample should be possible if we had a complete characterization of the probability distribution of S; but as far as we know this is only known for the rule "above the median" [22]. In general, we anticipate that $R := \phi \cdot S^{1/\alpha}$ should be an estimator of n, but a detailed knowledge of the probability distribution of S is needed to find the correcting factor ϕ that guarantees that $\mathbb{E}\{R\} \sim n$. In the particular case of the rule "above the median" (the one studied in [22], not the rule in [16] and [27] with $p = 1/2$, there are subtle differences), a suitable unbiased cardinality estimator is $R := S^2/4$. The exact and asymptotic estimation of $\Pr\{S = j\}$ in [22] can be used to show that

Table 1 Sampling Algorithms – a comparative. AS = Affirmative Sampling (this paper). α-AS (this paper, section 4). RS = Reservoir Sampling. BOT = Bottom-k. DS = Distinct Sampling. * = dependable means the exact frequency counts of each element in the sample can be obtained, see Def. 1. ** = conjectured. *** = $S^2/4$ is an unbiased estimator of n, for $\alpha = 1/2$.

Algorithm	**AS**	**α-AS**	**RS**	**BOT**	**DS**
Parameter	k	$\alpha \in (0,1)$	k	k	B
Distinct elements	Yes	Yes	No	Yes	Yes
Dependable*	Yes	Yes	No	Yes	Yes
Exp. sample size S	$k\ln(n/k)$	$\Theta(n^\alpha)$	k	k	$\frac{B}{2\ln 2} \approx 0.72\,B$
Std. deviation σ_S	$\sqrt{k\ln(n/k)}$	$\Theta(n^\alpha)$	0	0	$\frac{\sqrt{2}\sqrt{3\ln 2 - 2)B}}{4\ln 2} + \mathcal{O}(1)$
Std. inference err.	$\frac{1}{\sqrt{k\ln(n/k)}}$	$\Theta(n^{-\alpha})^{**}$	No	$1/\sqrt{k}$	$1/\sqrt{B\ln 2}$
Runtime $\mathcal{O}(N+\ldots)$	$k\log^2(n/k)\log\log(n/k)$	$n^\alpha \log n$	$k\log k\log(N/k)$	$k\log k\log(n/k)$	$B\log n$
Memory (in bits)	$\Theta(k\log(n/k)\log n)$	$\Theta(n^\alpha \log n)$	$\Theta(k\log n)$	$\Theta(k\log n)$	$\Theta(B\log n)$
Cardinality estim.	Yes [20]	$S^2/4^{***}$	No	Yes [4, 20, 29]	Yes [11]

$$\mathbb{E}\left\{S^2\right\} = \sum_{j\geq 0} j^2 \Pr\{S=j\} \sim \sum_{\ell=1}^{n^{1/2+\epsilon}} \frac{8\ell^3}{n} e^{-\ell^2/n}(1+\mathcal{O}(1/\ell))$$
$$\sim 8n\int_0^\infty x^3 e^{-x^2}\,dx = 4n,$$

and thus $\mathbb{E}\{R\} = n + o(n)$. The estimator R can also be used to estimate the absolute number n_P of distinct elements satisfying P, namely,

$$C_1 := (S_P \cdot S)/4 \implies \mathbb{E}\{C_1\} \sim n_P.$$

5 Conclusions and Final Remarks

Affirmative Sampling is a simple, elegant and practical algorithm to produce random samples of distinct elements from a "population" of n elements. Contrary to other existing algorithms (which produce random samples of fixed or expected constant size), it is the first distinct sampling algorithm that produces random samples with (expected) size growing with n. In this sense, AS gracefully adapts to the size n of the population and gives us probabilistic guarantees of increasing accuracy of the inferences made, just the opposite of what would happen if the size of the samples is fixed or bounded – this is the case, for instance, of the well known bottom-k and the Distinct Sampling algorithms mentioned in the introduction. Table 1 summarizes the features of the several sampling algorithms introduced in this paper as well as others discussed here.

As we mentioned in our abstract, our algorithm uses primitives that are straightforward to access in most programming languages, and is therefore readily implementable. Several implementations exist, for instance our reference code in Python, see [30].

The idea of replacements can be also used in combination to sampling algorithms such as the famous Reservoir Sampling [32], which returns a random sample of fixed sized from the data stream. As we mentioned in the introduction and in Table 1, Reservoir Sampling samples occurrences z_j in the data stream, not distinct elements. A given element x might appear several times in the sample. Actually, the relative frequency of x within the sample will be, on average, the relative frequency of x in the data stream. To combine replacements with Reservoir Sampling, we just need to change AS so that instead of using the hash value

of each item, we use a random number associated with every element of the data stream. That way, we would be able to collect a random sample of expected size $\sim k \ln(N/k)$ without prior knowledge of the length N of the data stream.

References

1. M. Abramowitz and I.A. Stegun, editors. *Handbook of Mathematical Functions*. Dover Publ., New York, 1964.
2. M. Archibald and C. Martínez. The hiring problem and permutations. In *Proc. of the 21Int. Col. on Formal Power Series and Algebraic Combinatorics (FPSAC)*, volume AK of *Discrete Mathematics & Theoretical Computer Science (Proceedings)*, pages 63–76, 2009.
3. B. C. Arnold, N. Balakrishnan, and H. N. Nagaraja. *Records*. Wiley series in probability and mathematical statistics. John Wiley & Sons, Inc., New York, 1998.
4. Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In J. D. P. Rolim and S. P. Vadhan, editors, *Randomization and Approximation Techniques, 6th International Workshop, RANDOM 2002, Cambridge, MA, USA, September 13-15, 2002, Proceedings*, volume 2483 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2002. doi:10.1007/3-540-45726-7_1.
5. K. S. Beyer, R. Gemulla, P. J. Haas, B. Reinwald, and Y. Sismanis. Distinct-value synopses for multiset operations. *Commun. ACM*, 52(10):87–95, 2009. doi:10.1145/1562764.1562787.
6. A. Z. Broder. On the resemblance and containment of documents. In B. Carpentieri, A. De Santis, U. Vaccaro, and J.A. Storer, editors, *Proc. of the Compression and Complexity of SEQUENCES 1997*, pages 21–29. IEEE Computer Society, 1997. doi:10.1109/SEQUEN.1997.666900.
7. W. G. Cochran. *Sampling Techniques*. John Wiley & Sons, Inc., New York, 3rd edition, 1977.
8. E. Cohen and H. Kaplan. Summarizing data using bottom-k sketches. In Indranil Gupta and Roger Wattenhofer, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing (PODC 2007)*, pages 225–234. ACM, 2007. doi:10.1145/1281100.1281133.
9. J. Ernvall and O. Nevalainen. An algorithm for unbiased random sampling. *The Computer Journal*, 25(1):45–47, 1982. doi:10.1093/comjnl/25.1.45.
10. 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(298):387–402, 1962. doi:10.2307/2281647.
11. Ph. Flajolet. On adaptive sampling. *Computing*, 43(4):391–400, 1990. doi:10.1007/BF02241657.
12. Ph. Flajolet, É. Fusy, O. Gandouet, and F. Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In Ph. Jacquet, editor, *Proc. of the 2007 Conference on Analysis of Algorithms (AofA 07)*, volume AH of *Discrete Mathematics & Theoretical Computer Science (Proceedings)*, pages 127–146, 2007.
13. Ph. Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985. doi:10.1016/0022-0000(85)90041-8.
14. Ph. Flajolet and A. Odlyzko. Singularity analysis of generating functions. *SIAM Journal on Discrete Mathematics*, 3(1):216–240, 1990. doi:10.1137/0403019.
15. Ph. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009. doi:10.1017/CBO9780511801655.
16. J. Gaither and M. D. Ward. Analytic methods for select sets. *Probability in the Engineering and Informational Sciences*, 26:561–568, 2012. doi:10.1017/S0269964812000186.
17. P. B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In P. M. G. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. T. Snodgrass, editors, *Proceedings of 27th International Conference on Very Large Data Bases (VLDB 2001)*, pages 541–550. Morgan Kaufmann, 2001.

18 R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison Wesley, 2nd edition, 1994.

19 A. Helmi. *The Hiring Problem and its Algorithmic Applications*. PhD thesis, Dept. Computer Science, Universitat Politècnica de Catalunya, 2013.

20 A. Helmi, J. Lumbroso, C. Martínez, and A. Viola. Counting distinct elements in data streams: the random permutation viewpoint. In N. Broutin and L. Devroye, editors, *Proc. of the 23rd Int. Meeting on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms (AofA)*, volume AQ of *Discrete Mathematics & Theoretical Computer Science (Proceedings)*, pages 323–338, 2012. doi:10.46298/dmtcs.3002.

21 A. Helmi, C. Martínez, and A. Panholzer. Analysis of the strategy "hiring above the m-th best candidate". *Algorithmica*, 70(2):267–300, 2014. doi:10.1007/s00453-014-9895-3.

22 A. Helmi and A. Panholzer. Analysis of the "hiring above the median" selection strategy for the hiring problem. *Algorithmica*, 66(4):762–803, 2013. doi:10.1007/s00453-012-9727-2.

23 S. Janson. The hiring problem with rank-based strategies. *Electronic Journal of Probability*, 24:1–35, 2019. doi:10.1214/19-EJP382.

24 T.G. Jones. A note on sampling a tape file. *Comm. ACM*, 5(6):343, 1962. doi:10.1145/367766.368159.

25 J. Kawarasaki and M. Bbuya. Random numbers for simple random sampling without replacement. Technical Report 7, Keio University, Dept. Mathematics, 1982.

26 D.E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison-Wesley, 3 edition, 1997.

27 S. Langowski and M. D. Ward. Moments of select sets. In M. Mishna and J. I. Munro, editors, *Proceedings of the 16th Workshop on Analytic Algorithmics and Combinatorics, ANALCO 2019, San Diego, CA, USA, January 6, 2019*, pages 67–73. SIAM, 2019. doi:10.1137/1.9781611975505.7.

28 G. Louchard. Probabilistic analysis of adaptative sampling. *Random Structures & Algorithms*, 10(1–2):157–168, 1997.

29 J. Lumbroso. An optimal cardinality estimation algorithm based on order statistics and its full analysis. *Discrete Mathematics & Theoretical Computer Science*, 2010.

30 J. Lumbroso and C. Martínez. Affirmative Sampling: Reference Python Implementation, March 2022. doi:10.5281/zenodo.6601690.

31 J. W. Tukey. Some sampling simplified. *Journal of the American Statistical Association*, 45(252):501–519, December 1950. doi:10.2307/2280719.

32 J. S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985. doi:10.1145/3147.3165.

33 J.S. Vitter. Faster methods for random sampling. *Comm. ACM*, 27(7):703–718, 1984. doi:10.1145/358105.893.

A Moments of the sample size S

In this appendix, we will write $S_{n,k}$ for the size of the sample $\mathcal{S}$ generated by AS, when given a data stream that contains n distinct elements and the initial parameter k – we have been not making this dependence on n and k explicit in most of the paper, and have written simply S. As we have discussed $S_{n,k}$ coincides with the number of k-records in a random permutation. Recall the bivariate generating function for $S_{n,k}$ given in (6)

$$S_k(z,u) = \sum_{n\geq k} \binom{n}{k} \sum_{\ell\geq 0} \Pr\left\{S_{n,k}=\ell\right\} z^n u^\ell = \frac{(zu)^k}{(1-z)^{ku+1}}.$$

Differentiating $S_k(z,u)$ r times with respect to u and setting $u=1$, we obtain the generating function

$$S_k^{(r)}(z) = \left.\frac{\partial^r S_k(z,u)}{\partial u^r}\right|_{u=1} = \sum_{n\geq k} \binom{n}{k} \mathbb{E}\left\{S_{n,k}^{\underline{r}}\right\} z^n,$$

where $x^{\underline{r}} := x\cdot(x-1)\cdots(x-r+1)$ denotes the r-th falling factorial of x [18].

The dominant singularity of $S_k^{(r)}(z)$ is located at $z=1$; therefore we have

$$S_k^{(r)}(z) \sim_{z\to 1} \frac{k^r z^k}{(1-z)^{k+1}} \ln\left(\frac{1}{1-z}\right)^r.$$

Applying standard singularity analysis results [14, 15] we obtain an asymptotic estimate for the n-th coefficient, and from there the sought asymptotic estimate for $\mathbb{E}\left\{S_{n,k}^{\underline{r}}\right\}$:

$$[z^n]S_k^{(r)}(z) \sim \frac{k^r n^k}{k!}(\ln n)^r + \text{l.o.t.}$$
$$\mathbb{E}\left\{S_{n,k}^{\underline{r}}\right\} \sim \frac{1}{\binom{n}{k}}[z^n]S_k^{(r)}(z) = k^r(\ln n)^r + \text{l.o.t.}$$

Notice also that $\mathbb{E}\left\{S_{n,k}^{r}\right\} = \Theta(\mathbb{E}\left\{S_{n,k}^{\underline{r}}\right\})$; it is enough to express $S_{n,k}^{r}$ in terms of the falling factorials $S_{n,k}^{\underline{i}}$ $(0\leq i\leq r)$ and the Stirling numbers of the second kind (see, for instance, [18]): for all x and $n\geq 0$

$$x^n = \sum_k \left\{{n \atop k}\right\} x^{\underline{k}}.$$