Sampling without Replacement in Linear Time

MATTI JAKOBSSON
University of Vaasa, Raastuvankatu 33 SF-65100 Vaasa 10, Finland

One method for selecting a sample of m different elements from a file of n records is to repeatedly select a random element until we have m different ones. We show that the number of selections is on average smaller than 2 ln (2) m and that the algorithm has a linear running time if we use a hash table for the elements already selected in the sample.

1. INTRODUCTION

Let us consider the problem of selecting a sample of m distinct elements in a population of n elements. Sampling without replacement is trivial in textbook examples with balls in an urn. In computer files it is not feasible to actually remove the records while sampling. Some solutions are given by Goodman & Hedetniemi and Ernvall & Nevalainen.[1,2] They both reject the simple TRYAGAIN procedure of sampling (Algorithm 1),

Algorithm 1

While the size of sample $< m$
begin
select a random integer X, $1 \leqslant X \leqslant n$ **if** X is not in the sample insert it in the sample
end

because more than m selections must be made. They both use and analyse an alternative which changes the locations of the records in the file so that an element is never chosen more than once. The procedure is as follows.

Algorithm 2

For $i = 1$ to m
begin
select a random integer $X, 1 \leqslant X \leqslant i+1$; exchange the locations of the Xth and $n-i+1$th element
end

At the end of the procedure the sample consists of the last m elements of the file. The locations of the elements in the file are not actually exchanged but indirect addressing is employed. Goodman and Hedetniemi use a vector of the length n. In the ith step of the algorithm the Xth element of the vector gets the value $n-i+1$ showing that the xth element is replaced by the last one. Ernvall and Nevalainen implement the vector as a hash table of m elements. The table contains the elements in the sample and the addresses of the records that replace the ones already in the sample.

The running time of algorithm 2 is linear with respect to m according to Ref. 2. Goodman and Hedetniemi analysed algorithm 1 and showed that the expected number of selections in order to select a sample of m elements is $0(n \log n)$.

We shall take a closer look at algorithm 1 and show that its running time is linear with respect to m, the size of the sample.

2. ANALYSIS OF ALGORITHM 1

Goodman and Hedetniemi[2] give a formula for the average number of times an element must be selected until we have m different elements in the sample as

$$E(m,n) = n \sum_{k=n-m+1}^{n} 1/k,\ 0 < m \leqslant n \text{ and } E(0,n) = 0. \tag{1}$$

Let us now consider formula (1). For a fixed n we notice that $E(m,n)$ is a concave function of m; each increment

$$E(m,n) - E(m-1,n) = 1/(n-m+1)$$

is positive and $E(m,n)-E(m-1,n) > E(m-1,n)-E(m-2,n)$.

Also Ernvall and Nevalainen noted that if m is larger than $\lfloor n/2 \rfloor$ we can choose the $n-m$ elements *not to be selected in the sample*. Therefore $E(\lfloor n/2 \rfloor, n)$ is the highest value of $E(m,n)$. Because $E(m,n)$ is concave, all the values of $E(m,n)$ are below the line between the points $(0,0)$ and $(E(\lfloor n/2 \rfloor, n), n/2)$. Therefore

$$E(m,n) < \frac{E(\lfloor n/2 \rfloor, n)}{\lfloor n/2 \rfloor} m = G(n)\, m.$$

Let us now consider the value of $G(n)$. We assume first that n is even.

$$\begin{aligned} G(n) &= \frac{E(\lfloor n/2 \rfloor, n)}{\lfloor n/2 \rfloor} = \frac{E(n/2,n)}{n/2} = 2 \sum_{k=n/2+1}^{n} 1/k \qquad (2) \\ &= 2\Big(\sum_{k=1}^{n} 1/k - \sum_{k=1}^{n/2} 1/k\Big) \\ &= 2((\ln(n) + C + \mathrm{Rem}(n)) - (\ln(n/2) + C + \mathrm{Rem}(n/2)) \\ &< 2 \ln (2). \end{aligned}$$

The sum of n first terms of the harmonic series equals

$$\ln(n) + C + \mathrm{Rem}(n),$$

where C is the Euler's constant $C = 0.577\ldots$ The sum approaches $\ln(n)+C$ smoothly and is always smaller than $\ln(n)+C$. Therefore $\mathrm{Rem}(n/2) > \mathrm{Rem}(n) > 0$ holds for all positive n.

It is easy to show that the inequality (2) holds for odd values, too. In fact, if n is even, $G(n+1) < G(n)$. Now we can find for $E(m,n)$ a linear upper bound of the form

$$E(m,n) < 2 \ln (2)\, m,$$

which is independent of n.

We have now shown that the while-loop in algorithm 1 is performed on average less than $2 \ln(2) m$ times.

Because we want to show that the running time of the algorithm is linear we must show that the actions within the while-loop can be performed in constant time. The

table for the order numbers (X) of the elements already selected in the sample is implemented as a hash table of m elements. The approach was also used by Ernvall and Nevalainen for algorithm 2. We use hashing with separate chaining.[3] As the hash function we use $\lfloor m(X-1)/n \rfloor$. An array of m elements is used for the roots of the collision chains. The order numbers of the elements selected in the sample can be placed in another array of m elements. The average number of elements in a chain after the hth element is selected in the sample is clearly h/m. For each selected random number X we must access the root node and on average h/m elements in the chain ($h/m < 1$) and insert X in the chain if it is not found there. This can be performed in constant time for each X.

When m is greater than $n/2$, we selected the $n-m$ elements not to be in the sample. Thereafter we must scan the file and pick the m elements from the file of n records. For this we need to have the negated sample elements in the order of the file. When we use $\lfloor m(X-1)/n \rfloor$ as the has function we can sort the negated sample (or the sample) in linear time (see address calculation sorting in Ref. 3, p. 99). Then the file is scanned and the sample is selected. This requires n operations. Note, however, that this phase is needed when $m > n/2$ holds. For each of the m elements in the sample this means less than two operations. The algorithm is still linear in respect of m when $m > n/2$.

Note that more complex sampling arrangements like the one in this paper are useful for relatively small sample sizes. If m is large, say greater than $n/2$, there are more straightforward sampling methods requiring n operations, which may be smaller than $0(m)$ for a large m. A simple example is selection sampling in Goodman and Hedetniemi[2]. The algorithm scans the file of n records and selects the ith record in the sample with the probability $(m-2)/(n-1+1)$, where s is the number of elements already in the sample.

3. CONCLUSIONS

We can conclude that when selecting a sample of m elements in a set of n without replacement using TRYAGAIN sampling we must select on average less than $2\ln(2)\,m \approx 1.4\,m$ elements in order to find a sample of m different elements. If the list of elements in the sample is implemented as a hash table algorithm 1 has a linear running time on average. The worst-case running time of the algorithm is infinite, because the same element may be selected repeatedly any number of times.

REFERENCES

1. J. Ernvall & O. Nevalainen, An algorithm for unbiased random sampling, *The Computer Journal* **25** (1), 45–47 (1982).
2. S. Goodman & S. Hedetniemi, *Introduction to the Design and Analysis of Algorithms*. McGraw-Hill, New York (1977).
3. D. Knuth, *The Art of Computer Programming*, vol. 3. Addison-Wesley, Reading, Massachusetts (1973).