1 Random Sampling

- Input:
  - List$[1 \ldots N]$
  - Length of database N (if known)
  - Length of sample n
- Output:
  - Sample$[1 \ldots n]$

1.1 Sampling Without Replacement

- W.l.o.g., we can always assume that $n \leq N/2$.

1.1.1 Straightforward Solutions, Non-Sequential Algorithms

Generate Random Indices: Generate random integers R uniformly between 1 and N, until n *different* values have been found.

- **Theorem:** In expectation, we will need to generate $N \cdot (H_N - H_{N-n+1})$ random numbers [Balls into bins, Coupon Collector's problem].
- Let T_i denote the time to "hit" the i-th distinct row after $(i-1)$ rows have been hit. The T_i's have geometric distribution with success probability $(N-i+1)/N$. We have $\mathrm{E}[T_i] = N/(N-i+1)$. Then, we need an expected number $\sum_{i=1}^{n} \mathrm{E}[T_i] = N \cdot (H_N - H_{N-n+1}) \approx N \cdot \ln \frac{N}{N-n+1}$ of random variates.
- Problems: Need to store which rows have been hit, i.e., at least $\Omega(n)$ memory, most practical approach: Hash table. Fastest approach: Bit array of length $\Theta(N)$

Generate Random Remaining Indices:

for $m \leftarrow 1, \ldots, n$ **do**
    $R \leftarrow$ random$(\{1 \ldots N - m + 1\})$
    $j \leftarrow$ index of R'th non-null element in List
    Sample$[m] \leftarrow$ List$[j]$
    List$[j] \leftarrow$ null

- Naive solution has running time $\Theta(nN)$.
- Problems: Running time is $\Omega(N)$, modifies List.

Fisher-Yates Shuffle [1]: (Improvement of "Generate Random Remaining Indices")

for $m \leftarrow 1, \ldots, n$ **do**
    $R \leftarrow$ random$(\{m \ldots N\})$
    Swap List$[m]$ and List$[R]$
Sample$[1 \ldots n] \leftarrow$ List$[1 \ldots n]$

- The minor modification improves the running time to $O(n)$.

1.1.2 Sequential Algorithms

Probabilistic Sampling: Select each row with probability n/N.

- Avoids modifying the database. Size of sample size has distribution $B(N, n/N)$. While *expected* sample size is n, standard deviation is $\sqrt{n(1-n/N)}$, i.e., approx $\sqrt{n}$ if $n \ll N$.

Selection Sampling: Modify the previous idea: If m records have already been selected, the $(t+1)$-th record should be added to the sample with probability $\frac{n-m}{N-t}$. See [4, p. 142]. First discovered by [2] and [3].

- **Theorem**: The algorithm will not run off the end. The sample is completely unbiased.
- At some record $t \in [N]$, we will either have $n = m$, in which case no further record with be selected. Or we will have $n - m = N - t$, in which case every record after t will be selected.

  Let $1 \leq x_1 \leq \cdots \leq x_n \leq N$ be the sample indices. This choice is obtained with probability $p := \prod_{i=1}^{N} p_t$ where

  $$p_{t+1} = \begin{cases} \frac{(N-t)-(n-m)}{N-t} & \text{if } x_m < t+1 < x_{m+1} \\ \frac{n-m}{N-t} & \text{if } t+1 = x_{m+1} \end{cases}$$

  The denominator of the product is $N!$. The numerator contains the terms $n, n-1, \ldots, 1$ for the t's which are x's, and $N-n, N-n-1, \ldots, 1$ for the t's that are not x's. Hence, probability $p = \frac{(N-n)!n!}{N!} = 1/\binom{N}{n}$.
- Problems: Running time is $\Theta(N)$
- Solution: (a) Do not generate a random variable for every row, but generate random variables that tell how many rows should be skipped. (b) Use clever way to compute the probability distribution for that.
- Let's temporarily redefine notation. Let N be the number of remaining record ($N - t$ previously) and n be the number of remaining samples to take ($n - m$ previously). Let S be a random variable for the number of records to skip. For $s \in \{0 \ldots N - n\}$,

  $$\Pr[S = s] = \left(\prod_{i=0}^{s-1} \frac{(N-i)-n}{N-i} \right) \cdot \frac{n}{N-s} = \frac{n \cdot (N-n)^{\underline{s}}}{N^{\underline{s+1}}},$$

  otherwise $\Pr[S = s] = 0$. The cumulative distribution function is

  $$F(s) = \Pr[S \leq s] = 1 - \frac{(N-n)^{\underline{s+1}}}{N^{\underline{s+1}}}.$$

1.1.3 Digression: Generating Random Variates

- Generate a random variate S according to this distribution F: Generate a random variate U uniformly at random in $[0, 1]$. Then, return the minimal S such that $U \leq F(S)$. [The probability that $S \leq s$ is the probability that $U \leq F(s)$, namely $F(s)$.]

  In this case: Set S as the minimal $s \in \{0 \ldots N - n\}$ such that

  $$1 - U \geq \frac{(N-n)^{\underline{s}}}{N^{\underline{s+1}}}, \quad \text{or,} \quad U \cdot N^{\underline{s+1}} \geq (N-n)^{\underline{s}} \quad \text{[substituting } U \text{ by } 1-U\text{]}$$
- Problem: Running time still $\Theta(N)$ because to skip s rows we need $\Theta(s)$ multiplications and comparisons, i.e., $\Theta(N)$ in total.

The Squeeze Method

- A better way to sample from distribution F is using von Neumann's rejection-acceptance framework together with the *squeeze method*.

- We need a random variable with distribution G that approximates F well. Let h be a function, $f(x) := \Pr[S = x]$, and g be a probability mass function such that $h(s) \leq f(s) \leq c \cdot g(s)$. Generate S as follows:

  1: Generate X according to the mass function g
  2: Generate U uniformly at random between 0 and 1
  3: **if** $U < h(X)/cg(X)$ **then** accept ▷ saves time if h is easier to compute than f
  4: **if** $U < f(X)/cg(X)$ **then** accept
  5: Go back to step 1

- **Theorem:** The posterior distribution of X (given acceptance) is F.

- We are interested in $\Pr[X = x \mid \text{Accept}]$.

$$\Pr[\text{Accept} \mid X = x] = \frac{f(x)}{cg(X)}$$

$$\Pr[X = x] = g(x)$$

$$\Pr[\text{Accept}] = \sum_x \Pr[\text{Accept} \mid X = x] \cdot \Pr[X = x] = \frac{1}{c} \qquad \text{[law of total probability]}$$

$$\Pr[X = x \mid \text{Accept}] = \frac{\Pr[\text{Accept} \mid X = x] \cdot \Pr[X = x]}{\Pr[\text{Accept}]} = f(x) \qquad \text{[Bayes' theorem]}$$

- With additional tricks, we can generate the random variate S in expected time $O(1)$. Hence, we can achieve expected running time $O(n)$ for an optimized version of the Selection Sampling algorithm. [8]

1.1.4 Unknown Population Size

Reservoir Sampling

```
Sample[1...n] ← List[1...n]
for t ← n+1, ..., N do
    with probability n/t do
        R ← random({1...n})
        Sample[R] ← List[t]
```

- **Theorem:** The algorithm is completely unbiased. [5]

- Let p_t be the probability that any particular sample *set* is chosen after t rows have been read. Clearly, when $t = n$ (base case), $p_t = 1/\binom{t}{n} = 1$. Now assume $p_t = 1/\binom{t}{n}$ and consider the point when $t + 1$ rows have been read (inductive step). If row $t + 1$ was not included, the probability of the sample is

$$p_{t+1} = p_t \cdot (1 - n/(t+1)) = \frac{n!}{t^{\underline{n}}} \cdot \frac{t+1-n}{t+1} = \frac{n!}{(t+1)^{\underline{n}}} = 1/\binom{t+1}{n}.$$

If row $t+1$ was included, the sample can arise when the previous sample contained the $(n-1)$ other sample items and one of $(t+1-n)$ items not in the sample.

$$p_{t+1} = (t+1-n) \cdot p_t \cdot n/(t+1) \cdot 1/n = p_t \cdot (1 - n/(t+1))$$

- One can again use similar techniques as for sampling with known population size [7]. Running time of this optimized version is $O(n(1 + \log \frac{N}{n}))$.

1.2 Sampling with Replacement

Reservoir Sampling:

for $t \leftarrow 1, \dots, N$ **do**
    for $i \leftarrow 1, \dots, n$ **do**
        with probability $\frac{1}{t}$ **do**
            Sample$[i] \leftarrow$ List$[t]$

- **Theorem:** The algorithm is completely unbiased. (First established in [6], but the proof is unnecessarily complicated.)

- After round $t \in \{1 \dots N\}$, the probability that row $s \in \{1 \dots t\}$ occurs at sample position $i \in \{1 \dots n\}$ is Pr[row s chosen for position i] $\cdot$ Pr[position i unchanged for rows $s+1, \dots, t$], i.e.,

$$\frac{1}{s} \cdot \frac{s}{s+1} \cdot \frac{s+1}{s+2} \cdots \frac{t-1}{t} = \frac{1}{t}\,.$$

Since the rows at each sample position are stochastically independent, the sample is completely unbiased.

References

[1] Richard Durstenfeld. Algorithm 235: Random permutation. *Communications of the ACM*, 7(7):420, 1964. DOI: `10.1145/364520.364540`.

[2] C. T. Fan, Mervin E. Muller, and Ivan 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. URL `http://www.jstor.org/stable/2281647`.

[3] T. G. Jones. A note on sampling a tape-file. *Communications of the ACM*, 5(6):343, 1962. DOI: `10.1145/367766.368159`.

[4] Donald Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison-Wesley, 3rd edition, 1997.

[5] A. I. McLeod and D. R. Bellhouse. A convenient algorithm for drawing a simple random sample. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 32(2):182–184, 1983. URL `http://www.jstor.org/stable/2347297`.

[6] Byung-Hoon Park, George Ostrouchov, Nagiza F. Samatova, and Al Geist. Reservoir-based random sampling with replacement from data stream. In *Proceedings of the Fourth SIAM International Conference on Data Mining (SDM'04)*, pages 492–496, 2004. URL `http://www.siam.org/meetings/sdm04/proceedings/sdm04_053.pdf`.

[7] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11 (1):37–57, 1985. DOI: `10.1145/3147.3165`.

[8] Jeffrey Scott Vitter. Faster methods for random sampling. *Communications of the ACM*, 27(7):703–718, 1984. DOI: `10.1145/358105.893`.