Weining Bao weiningb@usc.edu 8287723653

Project 10: Reservoir Sampling

In the previous classes, we have discussed about how to generate continuous random samples through the rejection method by
1) knowing the distribution function of what you are going to generate and by
2) having an instrumental distribution.

Now, we are curious about how to generate uniform samples without replacement from a set of elements. Namely, assume that the number of elements in the set is N and that the number of random samples (called reservoir samples) we want to get is K. Typically, N is assumed to be much larger than K. Our purpose is to generate the K random samples uniformly without replacement. This is called the reservoir sampling and is very important in computer science. Notice that the order of the random samples is not taken into account here. Answer the following questions.

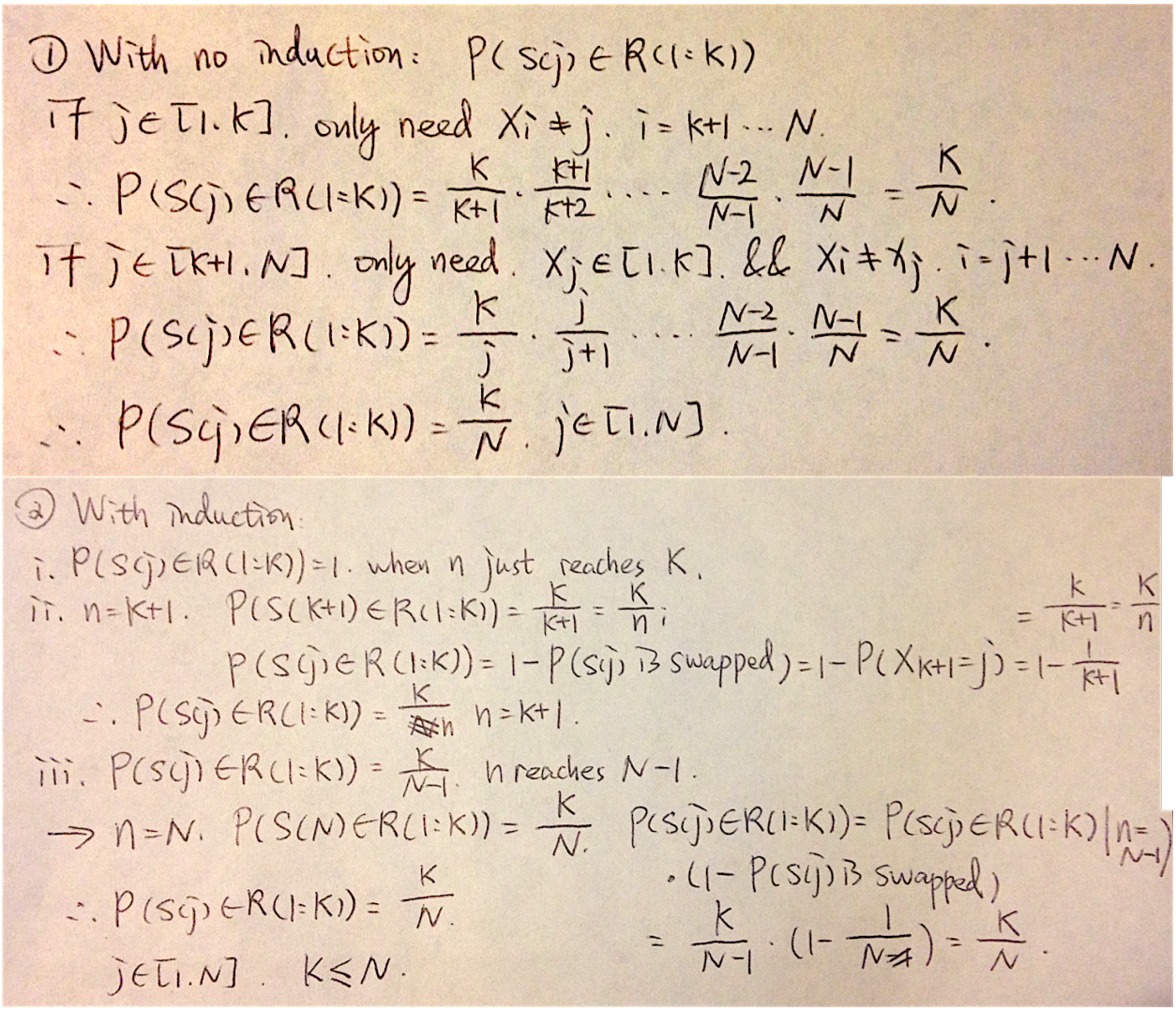
a) Algorithm R: A famous method to generate the random samples by a single pass is the following. Let S(1 : N) denote the set array and R(1 : K) denote the random sample array. The goal is to fill R with some values from S. We first put R(i) := S(i) for i = 1,⋯ ,K. Then for each j = K + 1,⋯ ,N, we generate a discrete uniform random number Xj in [1,j]. If Xj ≤ K, then we set R(Xj) := S(j); otherwise, we do nothing. Write out the pseudo-code[1] or the matlab code for this algorithm.

```
/*
  S has items to sample, R will contain the result
*/
ReservoirSample(S[1..n], R[1..k])
  // fill the reservoir array
  for i = 1 to k
      R[i] := S[i]

  // replace elements with gradually decreasing probability
  for i = k+1 to n
    j := random(1, i)   // important: inclusive range
    if j <= k
        R[j] := S[i]
```

b) For Algorithm R, show that for each S(i) for i = 1,⋯ ,K, P(S(j) ∈ R(1 : K)) = $\frac{K}{N}$ for j ∈ {1,⋯ ,N}. (Hint: use induction.)

[1] https://en.wikipedia.org/wiki/Reservoir_sampling



c) Use the Algorithm R to simulate the 6-from-49 lottery (choose 6 numbers from 49 numbers without repetition). Run the experiment for several times and verify that the appearance of the 49 numbers is uniform.

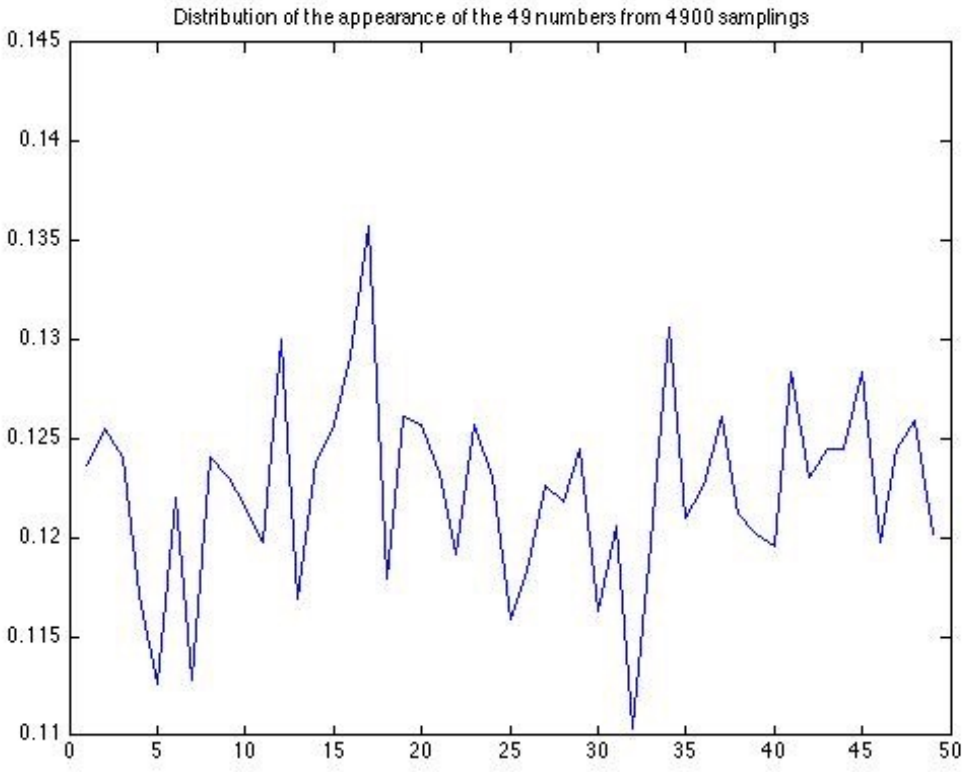
6-from-49 lottery is done multiply times t. The results are stored in a t * 6 matrix. The numbers of appearance of different numbers from 1 to 49 are stored in a 1 * 49 vector. The vector is used to plot the distribution and calculate the mean. Here are the codes in Matlab and results:

```
fprintf('\n 6-from-49 lottery with Algorithm R: \n');
k = 6; % samples wanted
n = 49; % size of sample space
times = 4900; % times of sampling
r = zeros(times, k); % storage of result
dist = zeros(1, n); % distribution of number sampled
for i = 1:times % run times simulations
   for j = 1:k % initialize samples with 1...k
      r(i,j) = j;
   end
```

```
    for j = k+1:n % generate random integer and update samples
        rd = randi(j);
        if rd <= k
            r(i, rd) = j;
        end
    end
    for j = 1:k % update distribution of sampled numbers
        dist(r(i, j)) = dist(r(i, j)) + 1;
    end
end
figure;
plot(dist./times);
title(['Distribution of the appearance of the ', num2str(n), ' numbers from '...
      , num2str(times), ' samplings']);
fprintf('\nThe mean of distribution is %f with theoretical value %f.\n', ...
        mean(dist)/times, k / n);
```

6-from-49 lottery with Algorithm R:

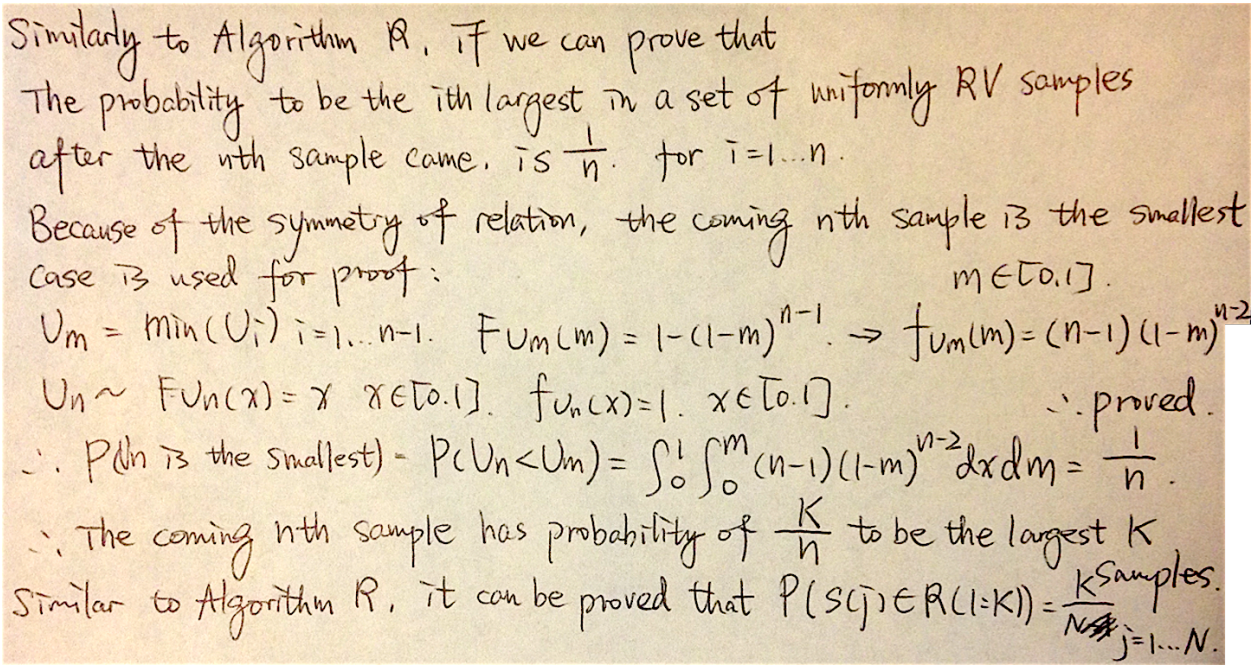
The mean of distribution is 0.122449 with theoretical value 0.122449.



From the results it can be found that the appearance of the 49 numbers is quite uniform and the mean is the same as theoretical value $\frac{K}{N} = \frac{6}{49} = 0.122449$.

d) The best part of the Algorithm R is that it uses constant additional memory. If we are allowed to use linear memory for data storage, the reservoir sampling can be implemented by random sort. The algorithm is as follows.

Algorithm Random Sort: For i = 1, · · · , N , generate Ui as continuous uniform random numbers in [0,1]. Then we pick K elements from S(1 : N) with the largest Ui. Show that by this algorithm, we can still yield P(S(j) ∈ R(1 : K)) = $\frac{K}{N}$.



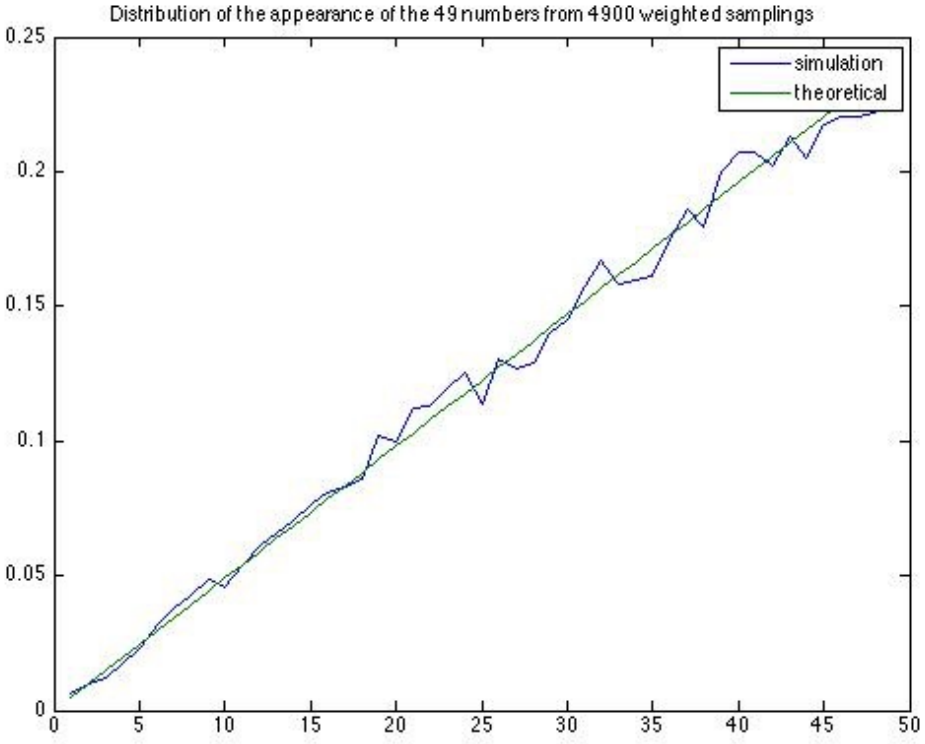
e) The advantage of the Algorithm Random Sort is that it can be implemented in parallel by MapReduce. Moreover, we can modify it for sampling with weights. Let us give each element in the set S a weight wi. The goal is to sample such that P(S(i) ∈ R(1 : K)) = $\frac{\omega_i}{\sum_{t=1}^{N} \omega_t}$.

Algorithm A: For i = 1, · · · , N , generate Vi := $U_i^{\frac{1}{\omega_i}}$ where Ui are continuous uniform random numbers in [0,1]. Pick K numbers from S(1 : N) with the largest Vi. Simulate the 6-from-49 lottery by letting wi := i for i = 1, · · · , 49 with Algorithm A. Run the experiment for several times and verify that the appearance of the 49 numbers is distributed according to the weights.

6-from-49 lottery is done multiply times t. The results are stored in a t * 6 matrix. The numbers of appearance of different numbers from 1 to 49 are stored in a 1 * 49 vector. The vector is used to plot the distribution and calculate the mean. Here are the codes in Matlab and results:

```
fprintf('\n 6-from-49 lottery with Algorithm A: \n');
k = 6; % samples wanted
n = 49; % size of sample space
times = 4900; % times of sampling
w = zeros(1, n); % weights of different number
```

```
for i = 1:n
    w(i) = i;
end
r = zeros(times, k); % storage of result
rd = zeros(1, n); % random weighted numbers
dist = zeros(1, n); % distribution of number sampled
for i = 1:times % run times simulations
    for j = 1:n % generate n random weighted numbers
        rd(j) = rand() ^ (1/w(j));
    end
    [st, idx] = sort(rd, 'descend'); % sort the random numbers in descend
    r(i,:) = idx(1:k);
    for j = 1:k % update distribution of sampled numbers
        dist(r(i, j)) = dist(r(i, j)) + 1;
    end
end
figure;
plot(dist./times); hold all;
s = sum(w);
plot(w.*k./s); hold off;
title(['Distribution of the appearance of the ', num2str(n), ' numbers from '...
      , num2str(times), ' weighted samplings']);
legend('simulation', 'theoretical');
fprintf('\nThe mean of distribution is %f with theoretical value %f.\n', ...
        mean(dist)/times, k/n);
```



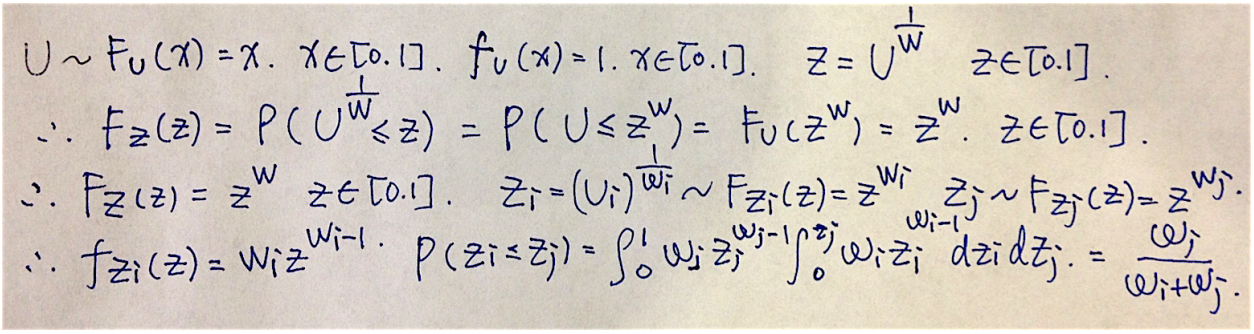
6-from-49 lottery with Algorithm A:

The mean of distribution is 0.122449 with theoretical value 0.122449.

From the results it can be found that the appearance of the 49 numbers is quite close to the theoretical weight ratios and the mean is the same as theoretical value $\frac{K}{N} = \frac{6}{49} = 0.122449$.

f) Prove that P(S(i) ∈ R(1 : K)) = $\frac{\omega_i}{\sum_{j=1}^{N} \omega_j}$ holds in Algorithm A. (Hint: first show that $P(U_i^{\frac{1}{\omega_i}} \le U_j^{\frac{1}{\omega_j}}) = \frac{\omega_j}{\omega_i + \omega_j}$.)

Following proves $P(U_i^{\frac{1}{\omega_i}} \le U_j^{\frac{1}{\omega_j}}) = \frac{\omega_j}{\omega_i + \omega_j}$:



Detailed proof of Algorithm A is provided in[2], the essence follows these steps:

1. Define Weighted Random Sampling WRS Algorithm D:
2. Define Algorithm A:

Algorithm A (*High level description*).

Input: A population V of n weighted items
Output: A WRS of size m
1: For each $v_i \in V$, $u_i = random(0, 1)$ and $k_i = u_i^{1/w_i}$
2: Select the m items with the largest keys k_i as a WRS

WRS Definition (*Algorithm D*).

Input: *A population V of n weighted items*
Output: A set S with a WRS of size m
1: Repeat Steps 2 and 3 for $k = 1, 2, \ldots, m$
2: The probability of v_i to be selected is: $p_i(k) = \frac{w_i}{\sum_{s_j \in V-S} w_j}$
3: Randomly select an item $v_k \in V - S$ and insert it into S

3. Prove that $For\ any\ permutation\ \Pi\colon P_A(\Pi) = P_D(\Pi).$

[2] Pavlos S. Efraimidis, Paul G. Spirakis, Weighted random sampling with a reservoir, Information Processing Letters, Volume 97, Issue 5, 16 March 2006, Pages 181–185

Remark 2. For Algorithm D, the probability that w_n is selected first is $w_n/(w_1 + w_2 + \cdots + w_n)$. Given that w_n is the first item, the probability that w_{n-1} is the second item is $w_{n-1}/(w_1 + w_2 + \cdots + w_{n-1})$, etc. Hence

$$P_D(\Pi) = \prod_{i=1}^{n} \frac{w_i}{w_1 + w_2 + \cdots + w_i}.$$

So when $\alpha = 1$,

Proposition 3. *Let U_i, for $i = 1, 2, \ldots, n$, be independent random variables with uniform distribution in $(0, 1)$. For $i = 1, 2, \ldots, n$ and positive reals w_i, let X_i be the random variables $X_i = (U_i)^{1/w_i}$. Then for any real α in $[0, 1]$:*

$$P[X_1 \leqslant \cdots \leqslant X_n \leqslant \alpha] = \alpha^{w_1 + \cdots + w_n} \cdot \prod_{i=1}^{n} \frac{w_i}{w_1 + \cdots + w_i}.$$

For every item v_i of the population, the probability that v_i is selected in a sample of size m is equal to the sum of the probabilities of all permutations (of all items) where v_i is in one of the first m positions. This sum of probabilities is the same for Algorithms D and A (by Lemma 4). Hence:

This proves that Algorithm D and A are equivalent and the condition of Algorithm D,

2: The probability of v_i to be selected is: $p_i(k) = \frac{w_i}{\sum_{s_j \in V - S} w_j}$

Proposition 5. *Algorithm A generates a WRS.*

is exactly what we need to prove.

Besides, the paper also gives a generalized proof for the proof made in part d.

Remark 6. Let $X_1, X_2, \ldots, X_k$ be independent random variables in the range $[0, 1]$ with the same continuous distribution function $F_X(x)$ such that $F_X(0) = 0$ and $F_X(1) = 1$. Then $P[X_k$ is in the largest m out of k items$] = m/k$.