prisoner problem

George Papadopoulos

Wording

You are part of an exploration group of $2n$ people to a new planet. Unfortunately when you reach the new found planet, an alien civilization places you under arrest and imprisons you. They decide to put you on a strange trial and if you win they will set you free. Firstly they assign to each one of you a different number from 1 to $2n$. The prison contains a room in which there are placed $2n$ boxes and on each one of these boxes you can see a painted number from 1 to $2n$. In each of those boxes the alien guards place $2n$ enumerated cards from 1 to $2n$ shuffled. Each one of the prisoners enters the room with the boxes and must find the box with the number that is assigned to him/her by opening n boxes at maximum. When one enters and finishes the trial, the cards are placed again in each box exactly as they were shuffled in the beginning of the trial, meaning that for each prisoner the placement of the cards in the boxes is the same. If each one of the prisoners finds the number assigned to him/her according to the procedure, all are free to go and unfortunately during the trial you cannot communicate with each other. Since you are clever enough and know well the probability theory, before you begin you decide to follow the following strategy:

Each one that enters the room firstly opens a box with the number that he/she is assigned to by the guards, if the number written on the card is the number assigned to him/her as a prisoner the prisoner stops and the next one enters the room to be put on the same test with the same placement of cards in the boxes as the previous one, if not he/she opens the box with the number that he/she reads on the card and so on until n boxes at maximum are opened, if he/she cannot find his/her number on any card of the sequence of numbers he/she gets from the boxes, all the prisoners are killed and the trial ends.

(a) Write a function *single_prisoner* which computes experimentally the probability that one of you will fail in the trial following the strategy described previously.

solution

```r
## inputs
## n --> the number of boxes,
## k --> the number of the prisoner,
## nreps --> the number of repetitions of the experiment

single_prisoner <- function(n = 10,k = 1,nreps = 1000) {
    if (k > 2*n) {
        stop(sprintf("prisoner id exceeds number of boxes %d",2*n))
    }
    cache <- vector("numeric",nreps)
    for (ii in 1:nreps) {
        boxes <- sample(2*n) # permute boxes once per trial
        q <- boxes[k] # for the first time choose the box
                      # with the number of the prisoner
        for (jj in 1:n) {
            if (q == k) { # if card number and prisoner id are the same,
                          # success, add 1 to the cache vector!
                cache[ii] <- 1
                break
            }
            else {
                q <- boxes[q]
            }
        }
    }
    1 - sum(cache)/nreps
}
```

```r
## near 1/2
single_prisoner(100,5,1000)
```

```
[1] 0.503
```

```r
single_prisoner(100,1,1000)
```

```
[1] 0.536
```

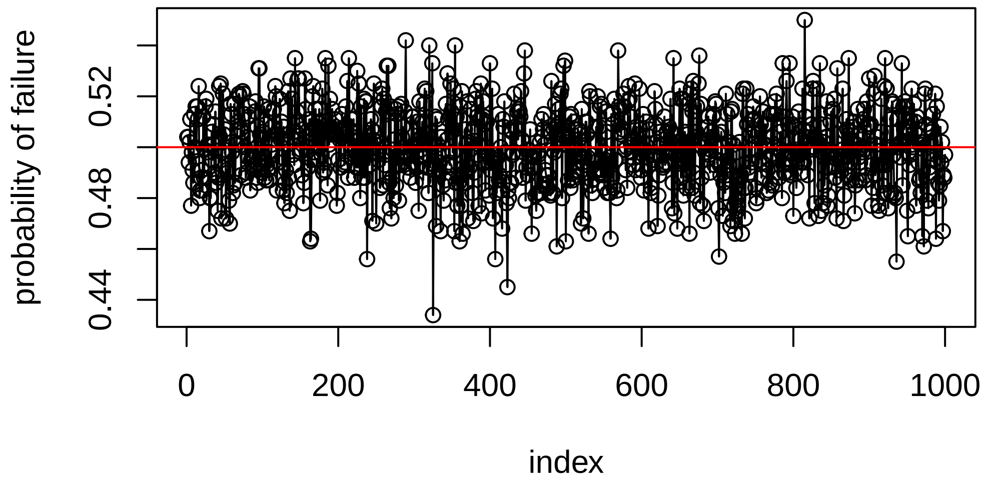
```r
single_prisoner(700,699,1000)
```

```
[1] 0.505
```

```
single_prisoner(500,5,1000)
```

```
[1] 0.519
```

```
plot(x = 1:1000,sapply(1:1000,function(i) single_prisoner()),
     type = "o",xlab = 'index',ylab = "probability of failure")
abline(h = 0.5,col = "red")
```



(b) Write a function named *all_prisoners* which computes experimentally the probability to leave you free, which means that all the prisoners have to succeed in the trial following the strategy described in the first two paragraphs. The function will take as inputs the same arguments as in the previous function omitting k, since all the prisoners will be put to the trial, remember that if one fails the guards kill everyone.

solution

```
all_prisoners <- function(n = 10,nreps = 1000) {
    cache <- vector("numeric",nreps)
    for (ii in 1:nreps) {
        boxes <- sample(2*n) # boxes for each simulation
        prisoners_index <- sample(2*n,n)
        for (jj in 1:n) {
            ind <- prisoners_index[jj]
            status <- FALSE
            for (kk in 1:n) {
                if (prisoners_index[jj] == boxes[ind]) {
                    status <- TRUE
                    break
                } else {
                    ind <- boxes[ind]
                }
            }
            if (!isTRUE(status)) {
                break
            }
            if (jj == n & isTRUE(status)) {
                    cache[ii] <- 1
            }
        }
    }
    sum(cache)/nreps
}
```

```
## should be over 0.3 some mistakes are possible little smaller than 0.3
all_prisoners(100,2000)
```

```
[1] 0.2935
```

```
all_prisoners(200)
```

```
[1] 0.305
```

```
all_prisoners(50,1000)
```

```
[1] 0.326
```

```
all_prisoners(500,1000)
```

```
[1] 0.299
```

```r
all_prisoners(20,1e4)
```

```
[1] 0.3127
```

(c) Suppose that you don't have an extended knowledge of mathematics and you don't follow any strategy. Before each prisoner gets into the room he/she chooses beforehand n numbers from 1 to $2n$ and then opens the corresponding boxes hoping that one of them will contain the card with his/her number. Compute the theoretical probability that he/she will be freed and write a program to confirm your theory. Finally compute the theoretical lower bound of the probability of success for the strategy followed in the first two problems.

solution

The probability is $\frac{n}{2n} = \frac{1}{2}$.

```r
single_prisoner_no_strategy <- function(n = 10,k = 1,nreps = 1000) {
    if (k > 2*n) stop(sprintf("prisoner number exceeds %d",2*n))
    cache <- vector("numeric",nreps)
    for (ii in 1:nreps) {
        boxes <- sample(2*n)
        choices <- sample(2*n,n)
        if (k %in% boxes[choices]) {
            cache[ii] <- 1
        }
    }
    sum(cache)/length(cache)
}
```

We expect that for the random strategy of pre-choosing n boxes this strategy will lead to sure death of all prisoners since $\frac{1}{2^{2n}} \to 0$ as $n \to \infty$.

```r
all_prisoners_no_strategy <- function(n = 10,nreps = 1000) {
    cache <- vector("numeric",nreps)
    for (ii in 1:nreps) {
        boxes <- sample(2*n)
        prisoners_index <- sample(2*n,n)
        for (jj in 1:n) {
            choices <- sample(2*n,n)
            status  <- FALSE
            if (prisoners_index[jj] %in% boxes[choices]) {
                    status <- TRUE
                    break
                }
                if (status == FALSE) {
                    break
                }
                if (status == TRUE & jj == n) {
                    cache[ii] == 1
                }
            }
        }
        sum(cache)/nreps
    }
```

```
## no strategy pre-choose near 1/2 possibility of success for single prisoner
single_prisoner_no_strategy()
```

```
[1] 0.521
```

```
single_prisoner_no_strategy(100,5)
```

```
[1] 0.509
```

```
single_prisoner_no_strategy(200,15)
```

```
[1] 0.494
```

```
## all_prisoners_no strategy 0 probability of success
all_prisoners_no_strategy()
```

```
[1] 0
```

```
all_prisoners_no_strategy(100,2000)
```

```
[1] 0
```

```r
all_prisoners_no_strategy(20,1000)
```

```
[1] 0
```

Theoretical Explanations For Questions (a) And (b).

Computation of minimum probability for success if the strategy explained in (a) is followed:

Let us compute the permutations that contain the circles with exactly length ℓ greater than n, these are the circles that lead to failure.

- $\sum_{k>n,k\leq 2n} \binom{2n}{k}(k-1)!(2n-k)! = \sum_{k>n} \frac{(2n)!}{k}$

Thus the probability of failure is

- $\frac{1}{(2n)!} \sum_{k>n} \frac{(2n)!}{k} = \sum_{k>n} \frac{1}{k}$

and so if the strategy is followed the maximum probability of failure is

- $\sum_{k>n} \frac{1}{k} \leq \int_{n}^{2n} \frac{1}{t}\, dt = \log(2n) - \log(n) = \log(2)$

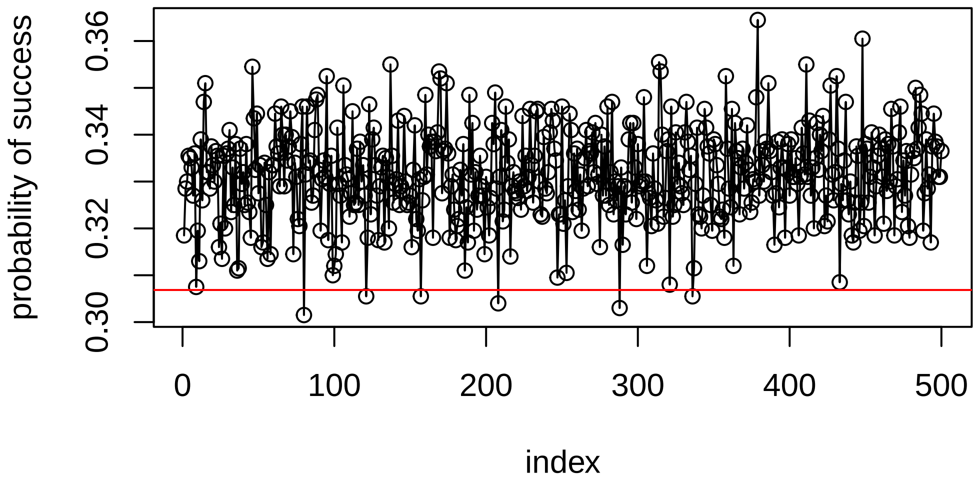
and finally the minimum probability of success is $1 - \log(2)$.

Let's test multiple times the all_prisoners function to see the percentage of the successes below $1 - \log(2)$. We expect a 5% Type I error rate since theoretically we know the minimum probability of success and also expect computation R errors due to rounding.

```r
w <- sapply(1:500, function(i) all_prisoners(nreps = 2000))
sum(w <= 1-log(2))/500
```

```
[1] 0.012
```

```r
plot(x = 1:500,w,type = "o",xlab = 'index',ylab = "probability of success")
abline(h = 1-log(2),col = "red")
```


probability of success
index