Stone Picking Problem

Ngai Chun Tsui

12/4/2021

Contents

---

1 Overview

1.1 Problem Overview

There is a video on YouTube which is about a story of stone picking.

https://www.youtube.com/watch?v=GXjlYWw7ZPI

In the story, a person walks along a path. The path has stones of different sizes. He is required to pick up the largest stone, but he only has one chance. He is not allowed to turn back. If he thinks a previous stone is the largest, he cannot go back to pick it up. He can only pick up the stone he sees at the moment or bet to see if he can pick up a larger one in the road ahead.

The story may sound familar to you. When I first read the story, I thought that it was a story to teach people not to be too greedy. Some people would never get satisfied and they would keep looking for a larger stone as they walked along the path. They would end up getting nothing. In fact, we might find ourselves in similar situations in life. For example, we keep looking for new jobs until we find the one that we think is the best.

The YouTuber solves the problem using mathematics beautifully. Here I will offer another approach using Monte Carlo Simulation. I make some assumptions, so my case is different from the case in the video.

1.2 Methodology

As I will use a mathamatical approach, I will make assumptions to make the case easier for analysis.

1. Our goal is not to pick the largest stone, but a stone that is as large as possible.
2. A person, whom we identify as tester, will walk along a path. The path has 1000 stones.
3. The stone size is represented by a value ranged between 1 and 100. The value is uniformly distributed. Our goal is to pick up the stone whose value is as large as possible.
4. The tester will see stone one by one. Each time he will see only one stone and he will decide whether to pick it up or not. If he decides not to pick up, he moves along and sees another stone and makes decision.
5. The tester knows the number of stone in the path. If he does not pick up any stone until the last stone, he will pick up the last stone as it is better than nothing.
6. I will use different strategies and use Monte Carlo simulation to test. I will use the mean of the simulated results to evaluate which strategy is the best.

---

2 Analysis

2.1 Initialize the Variables (Step 1 in R file)

```r
# The number of Monte Carlo replication
B <- 10000

# The number of drawing
n <- 1000

# The maximum integer value of the range
max_value <- 100
```

2.2 Strategy 1 - Random Drawing (Step 2 in R file)

In this strategy, the tester will just pick up a stone randomly.

```r
set.seed(1)
S <- replicate(B, {
  stones <- sample(1:max_value, n, replace=TRUE)
  draw <- sample(stones, 1)
})

mean(S)
```

```
## [1] 50.4028
```

```r
acc_results <- tibble("Method" = "Random Drawing",
                      "Comparator Size" = "N.A.",
                      Mean = mean(S))
acc_results %>% knitr::kable()
```

Method	Comparator Size	Mean
Random Drawing	N.A.	50.4028

In the above table, the value is the mean of the values of stones in simulations. The value is close to the mean of the stone value distribution = 50.

2.3 Strategy 2 - Sample with the First 10/20/.../90% of Stones and Compare to Average (Step 3 in R file)

In this strategy, the tester will first record the sizes of the first 10% of stones (i.e. the first 100 stones) and calculate the mean. The tester will not pick up any stone in the first 100 stones. He will only record the sizes. Then as he walks along the path, he will pick up immediately the stone with size which is larger than or equal to the computed mean. If he does not see any stone with size larger than or equal to the computed mean, he will pick up the last stone. We repeat this approach for 10/20/.../90% sample to compare the differences.

```r
# Array of sample size to be collected for comparison = 10,20,...,90%
# If number of stones = 1000, then the array will be [100,200,...,900]
size_array <- seq(from = (10*n/100), to = (90*n/100), by = n/10)

for (my_size in size_array) {
  set.seed(1)
  S <- replicate(B, {
    stones <- sample(1:max_value, n, replace=TRUE)
    last_stone <- stones[n]

    # Take the first x stones and get the mean
    # x = the currency my_size
    my_sample <- stones[1:my_size]
    comparator <- mean(my_sample)

    # From x+1 to n, compare to the comparator, if it is larger, pick.
```

```r
    pick <- -1;
    s_index <- my_size + 1
    for (i in s_index:n) {
      if (stones[i] >= comparator) {
        pick <- stones[i]
        break;
      }
    }

    if (pick < 0) {
      pick <- last_stone
    }

    pick
  })

  mean(S)

  acc_results <- bind_rows(acc_results, tibble("Method" = "Compare to Average",
                                               "Comparator Size" = as.character(my_size),
                                               Mean = mean(S)))
}

acc_results %>% knitr::kable()
```

Method	Comparator Size	Mean
Random Drawing	N.A.	50.4028
Compare to Average	100	75.6301
Compare to Average	200	75.7050
Compare to Average	300	75.1404
Compare to Average	400	75.3791
Compare to Average	500	75.6764
Compare to Average	600	75.5834
Compare to Average	700	75.3444
Compare to Average	800	75.4345
Compare to Average	900	75.4605

In the table above, the column 'Comparator Size' represents the size of the sample used to compute the comparator. For example, if we pick 10% of the total stones to compute the mean as the comparator, the 'Comparator Size' will be 100. We can see that the sample percentage does not make much difference. We expect to pick a stone with value around 75 using this strategy.

2.4 Strategy 3 - Sample with the First 10/20/.../90% of Stones and Compare to the 75th Percentile (Step 4 in R file)

This is similar to strategy 2. The difference is that now the tester will use the 75th percentile instead of the mean as the benchmark. If he sees stone with size larger than or equal to the computed 75th percentile, he will pick it up.

```r
# Array of sample size to be collected for comparison = 10,20,...,90%
size_array <- seq(from = (10*n/100), to = (90*n/100), by = n/10)

for (my_size in size_array) {
  set.seed(1)
  S <- replicate(B, {
    stones <- sample(1:max_value, n, replace=TRUE)
    last_stone <- stones[n]

    # Take the first x stones and get the mean
    # x = the currency my_size
    my_sample <- stones[1:my_size]
    comparator <- quantile(my_sample, probs = 0.75)

    # From x+1 to n, compare to the comparator, if it is larger, pick.
    pick <- -1;
    s_index <- my_size + 1
    for (i in s_index:n) {
      if (stones[i] >= comparator) {
        pick <- stones[i]
        break;
      }
    }

    if (pick < 0) {
      pick <- last_stone
    }

    pick
  })

  mean(S)

  acc_results <- bind_rows(acc_results, tibble("Method" = "Compare to 75th percentile",
                                               "Comparator Size" = as.character(my_size),
                                               Mean = mean(S)))
}

acc_results %>% knitr::kable()
```

Method	Comparator Size	Mean
Random Drawing	N.A.	50.4028
Compare to Average	100	75.6301
Compare to Average	200	75.7050
Compare to Average	300	75.1404
Compare to Average	400	75.3791
Compare to Average	500	75.6764
Compare to Average	600	75.5834
Compare to Average	700	75.3444
Compare to Average	800	75.4345

Method	Comparator Size	Mean
Compare to Average	900	75.4605
Compare to 75th percentile	100	87.6102
Compare to 75th percentile	200	87.7695
Compare to 75th percentile	300	87.5993
Compare to 75th percentile	400	87.7731
Compare to 75th percentile	500	87.7812
Compare to 75th percentile	600	87.8198
Compare to 75th percentile	700	87.6524
Compare to 75th percentile	800	87.8272
Compare to 75th percentile	900	87.6181

2.5 Strategy 4 - Sample with the First 10/20/.../90% of Stones and Compare to the 90th Percentile (Step 5 in R file)

This is similar to strategy 3. The difference is that now the tester will use the 90th percentile instead of the 75th percentle as the benchmark.

```r
# Array of sample size to be collected for comparison = 10,20,...,90%
size_array <- seq(from = (10*n/100), to = (90*n/100), by = n/10)

for (my_size in size_array) {
  set.seed(1)
  S <- replicate(B, {
    stones <- sample(1:max_value, n, replace=TRUE)
    last_stone <- stones[n]

    # Take the first x stones and get the mean
    # x = the currency my_size
    my_sample <- stones[1:my_size]
    comparator <- quantile(my_sample, probs = 0.9)

    # From x+1 to n, compare to the comparator, if it is larger, pick.
    pick <- -1;
    s_index <- my_size + 1
    for (i in s_index:n) {
      if (stones[i] >= comparator) {
        pick <- stones[i]
        break;
      }
    }

    if (pick < 0) {
      pick <- last_stone
    }

    pick
  })
```

```
  mean(S)

  acc_results <- bind_rows(acc_results, tibble("Method" = "Compare to 90th percentile",
                                               "Comparator Size" = as.character(my_size),
                                               Mean = mean(S)))
}

acc_results %>% knitr::kable()
```

Method	Comparator Size	Mean
Random Drawing	N.A.	50.4028
Compare to Average	100	75.6301
Compare to Average	200	75.7050
Compare to Average	300	75.1404
Compare to Average	400	75.3791
Compare to Average	500	75.6764
Compare to Average	600	75.5834
Compare to Average	700	75.3444
Compare to Average	800	75.4345
Compare to Average	900	75.4605
Compare to 75th percentile	100	87.6102
Compare to 75th percentile	200	87.7695
Compare to 75th percentile	300	87.5993
Compare to 75th percentile	400	87.7731
Compare to 75th percentile	500	87.7812
Compare to 75th percentile	600	87.8198
Compare to 75th percentile	700	87.6524
Compare to 75th percentile	800	87.8272
Compare to 75th percentile	900	87.6181
Compare to 90th percentile	100	95.1254
Compare to 90th percentile	200	95.1637
Compare to 90th percentile	300	95.2424
Compare to 90th percentile	400	95.2619
Compare to 90th percentile	500	95.2319
Compare to 90th percentile	600	95.2819
Compare to 90th percentile	700	95.2042
Compare to 90th percentile	800	95.2300
Compare to 90th percentile	900	95.1990

---

3 Results & Conclusion

```
acc_results %>% knitr::kable()
```

Method	Comparator Size	Mean
Random Drawing	N.A.	50.4028
Compare to Average	100	75.6301
Compare to Average	200	75.7050
Compare to Average	300	75.1404
Compare to Average	400	75.3791
Compare to Average	500	75.6764
Compare to Average	600	75.5834
Compare to Average	700	75.3444
Compare to Average	800	75.4345
Compare to Average	900	75.4605
Compare to 75th percentile	100	87.6102
Compare to 75th percentile	200	87.7695
Compare to 75th percentile	300	87.5993
Compare to 75th percentile	400	87.7731
Compare to 75th percentile	500	87.7812
Compare to 75th percentile	600	87.8198
Compare to 75th percentile	700	87.6524
Compare to 75th percentile	800	87.8272
Compare to 75th percentile	900	87.6181
Compare to 90th percentile	100	95.1254
Compare to 90th percentile	200	95.1637
Compare to 90th percentile	300	95.2424
Compare to 90th percentile	400	95.2619
Compare to 90th percentile	500	95.2319
Compare to 90th percentile	600	95.2819
Compare to 90th percentile	700	95.2042
Compare to 90th percentile	800	95.2300
Compare to 90th percentile	900	95.1990

From the table, we can see that strategy 4 is the best. For strategy 2,3,4, the size of the sample (10/20/.../90% of total) does not make much difference. Using strategy 4, we get a stone with size = 95 on average, much better than the random picking strategy with size = 50. We can see that with this strategy, we can perform much better than others who does not play any strategy (picking randomly).

There are some limitations in this study. For example, I have not studied the distribution of the data in each strategy. I only use the mean and draw a conclusion. From the YouTube video which I mention at the beginning, it seems that using the largest stone in the sample as the benchmark is the best. I do not have time to test it in this project due to limited time. These are the possible areas in which I can do further study and improve. I used to think of this stone picking story as a parable to teach philosophy. It is fun to analyze this bedtime story in a mathematical approach.