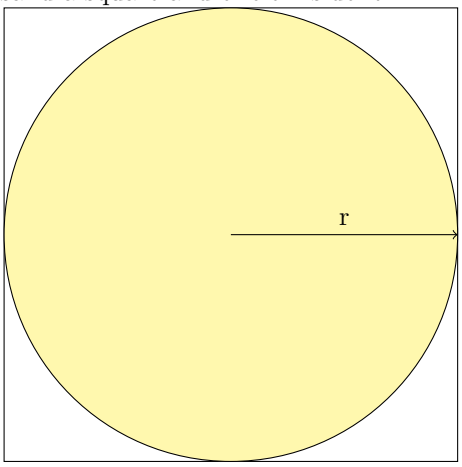
# Monte Carlo method

dr Szymon Murawski

January 14, 2020

## 1 Calculating $\pi$ Ancient Greeks way

Suppose we want to calculate the value of $\pi$ back in the Ancient Greeks time, that is without the help of very precise instruments. We can start by drawing on sand a square and circle inside it:




We know, that the area of square is $P_s = (2r)^2$, while are of circle is $P_c = \pi r^2$. We can combine those two equations to obtain:

$$\pi = \frac{4P_c}{P_s} \tag{1}$$

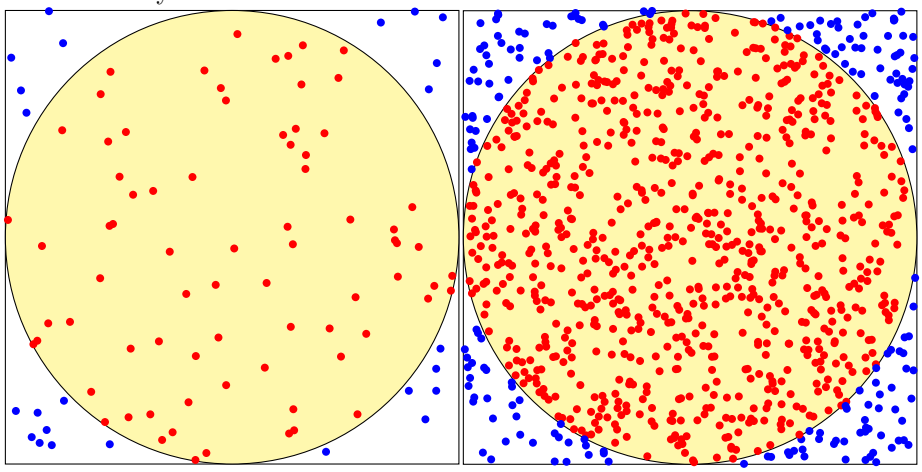
Now we have expressed $\pi$ as a ratio of the two areas. This might not look terribly helpful, until we observe that instead of ratio of areas we can take ratio of random points inside the circle and square. Ancient Greeks would throw rocks onto the picture, we can just print random numbers:

If by $N_c$ we define all the points that are within the circle and $N$ is the total number of points, then our equation becomes:

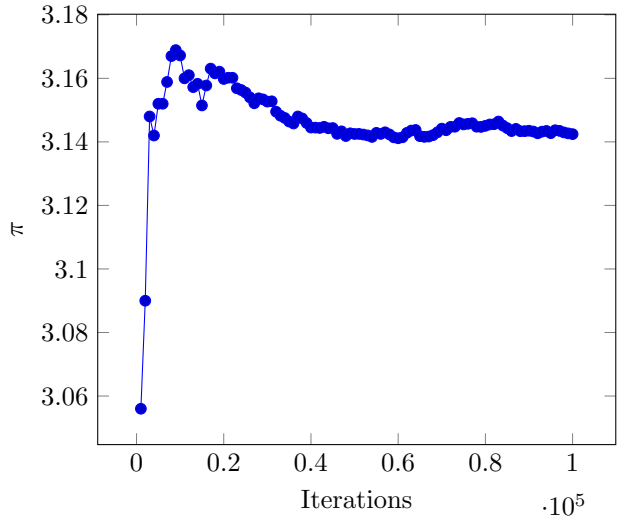
$$\pi \approx \frac{4N_c}{N} \tag{2}$$

The quality of the result depends on the number of points and in the limit of $N \to \infty$ we get exact solution.

Presto! Now you know what is **Monte Carlo Method**!



$\pi \approx \frac{4 \cdot 76}{100} \approx 3.04$ $\pi \approx \frac{4 \cdot 811}{1000} \approx 3.24399$




## 2 Monte Carlo Method

Monte Carlo method is a technique, that uses random sampling to produce a solution to a given problem. This method is suitable for very complex problems, where exact solution would be unfeasible (or impossible) to compute.

Name Monte Carlo comes from it's inventor, polish mathematician Stanislaw Ulam. While laying in bed in hospital out of boredom he tried to compute what is the probability of successfully laying out a solitaire. Pure combinatoric

approach proved unsuccessful, as the number of variables was too big, so he wonder whether just laying out one hundredth of them and checking how many came out successful would be better. He then presented the idea to other scientists working on Manhattan Project where the potential of this method was quickly realized. Since a code name was required, Monte Carlo was proposed, as in Monte Carlo Casino Ulam's uncle regularly lost money to gambling.

The core of Monte Carlo method is to probe the states of the system using some kind of distribution. In case of calculating $\pi$ the distribution was uniform - every point had the same chance of being drawn. There are however problems, where different distributions are used.

Monte Carlo method is used many different fields, some examples include:

- Integration - especially good for multidimensional or non-uniform functions
- Risk assessment
- Studying the growth of crystals and polymers
- Simulating lattice systems (used for study of superconductors)
- Folding proteins
- Calculating doses of radiation in radiotherapy treatments
- Predicting airport traffic

# 3 Monte Carlo Risk Assessment

Let's consider a plan for a project, that consists of three tasks. Tasks must be completed in sequence and our analysis gave the best, worst and average case scenarios.

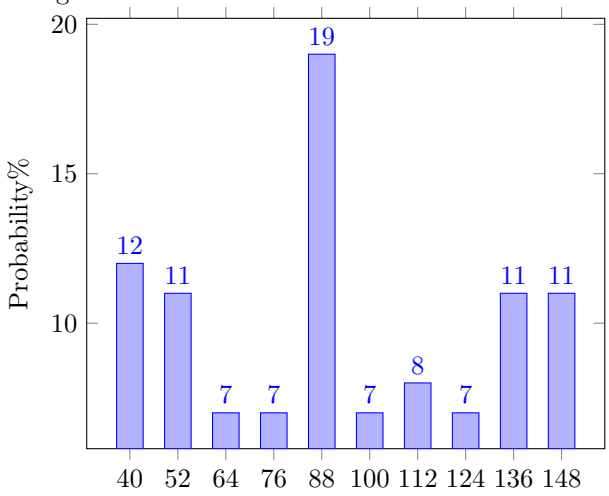
| **Task** | **Best case** | **Average case** | **Worst case** |
|---|---|---|---|
| Task 1 | 20 Days | 30 Days | 40 Days |
| Task 2 | 10 Days | 50 Days | 90 Days |
| Task 3 | 10 Days | 20 Days | 30 Days |

The question now is, how long will the project take? We can easily say what is the best case scenario for the whole project (40 days) or worst case (160 days), but in reality we can never expect best (or worst) scenarios for all the tasks. The way to solve this task with Monte Carlo is to simply select at random a large number of possible plans and calculate the averages:

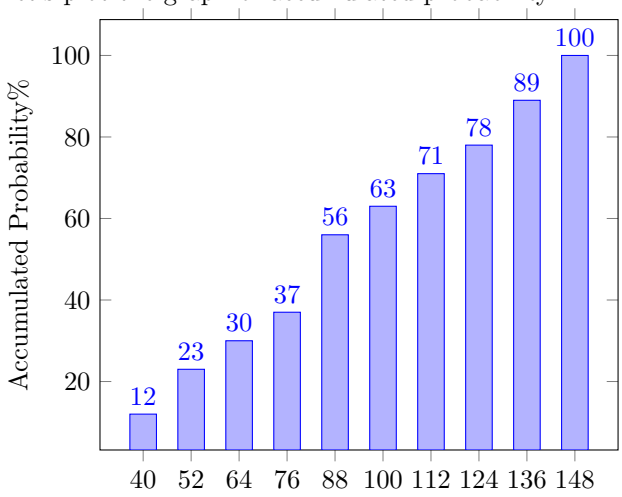
| **Random plan number** | **time** |
|---|---|
| 1 | $20 + 10 + 20 = 50$ days |
| 2 | $30 + 50 + 20 = 100$ days |
| 3 | $30 + 90 + 10 = 130$ days |
| ... | ... |
| 1000 | $20 + 10 + 10 = 40$ days |

The average length of the plan is just a simple average over above results. However, we can play a little with data like this and present them in much nicer way.

First of all let's group the results into buckets and see how many data points are in a given bucket:




From this graph we can see what is the distribution of plan estimates. Next one let's plot the graph of accumulated probability:




This graph informs us with what probability the project will be done in a given time. Based on this graph we can say, that although there is only 56% probability that the project will be done in 63 days, we can be 89% sure, that it will take only in 136 days.

## 3.1 Implementation details

The core of the Monte Carlo method is visible in the below code:

```
1 public int Simulate()
2 {
3   int totalCostOfRandomPlans = 0;
4   int iterations = 1000;
5   for( int i = 0; i < iterations; i++){
6     int randomPlanCost = 0;
7     foreach (var task in Plan.GetTasks){
8       randomPlanCost += task.GetRandomEstimate()
9   }
10  totalCostOfRandomPlans += randomPlanCost;
11  }
12  return totalCostOfRandomPlans/iterations
13 }
```

Variable `iterations` controls the precision of our simulation, the higher the value the better the result, at the cost of longer runnig time. In lines 7-9 for every task our plan is composed of a random estimate is taken from the possible options and added to current plan estimate. The result is the total cost of all simulated plans divided by number of iterations.

A good explanatory of "bucketing" the results can be found under the following link: `https://xaviergeerinck.com/creating-a-bucketing-function`

For a simple problem like this default random number generator offered by language should be sufficient. However, for more serious applications various generator engines should be tested. The topic of generating random numbers is very interesting, so i really recommend diving into it!