STAT 513/413: Lecture 10
How it all works

(the playbook of Monte Carlo)

Recall: a simple problem

Problem: what is the probability of obtaining 2 or less heads in 3 coin tosses.

Closed-form solution: because this is very easy problem, we can use the equiprobability argument

There are 8 possibilities


```
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    1
[3,]    0    1    0
[4,]    0    1    1
[5,]    1    0    0
[6,]    1    0    1
[7,]    1    1    0
[8,]    1    1    1
```


Each of them has the same probability - that is, 1/8. The event "2 or less" includes 7 of them; hence the probability is $7/8 = 0.875$.

Estimating the result instead

This was rather a simple problem - but believe me, there may be others that are not so simple. Some of them are really difficult - to the point that nobody knows the closed-form solution.

If the problem is easy, we may want to check the solution somehow. If the problem is difficult, we want to obtain at least some approximation.

So what we do? We *estimate* the solution instead, using random numbers. We repeatedly generate the same situation ("experiment"), in which it is fairly easy to check by computer whether the event ("2 or less") happens or not. We calculate the relative frequency of that - that will be our probability estimate

Remember, we are learning now: hence the problems may be deliberately easy. In assignments/exam, if you are asked to estimate *the solution, you are to use random numbers. If you by chance know the solution in closed-form, that it great: you can verify whether your estimate is close to it (and thus whether your program is not misguided). But it does not generate any credit: we are in the learning environment, remember that.*

An exploratory R interactive session

```
> try=replicate(10,rbinom(3,1,0.5))
> try
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    0    0    1    0    0    1    0    1    1     1
[2,]    1    1    1    0    1    0    0    1    1     0
[3,]    1    1    1    1    0    1    0    1    0     1
> apply(try,2,sum)
[1] 2 2 3 1 1 2 0 3 2 2
> apply(try,2,sum) <= 2
[1]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
> sum(apply(try,2,sum) <= 2)
[1] 8
> mean(apply(try,2,sum) <= 2)
[1] 0.8
```


The resulting "program" can actually be written in one line

```
> mean(apply(replicate(10,rbinom(3,1,0.5)),2,sum) <= 2)
[1] 1
```

Equal to 1??? That is quite off... Apparently, we need to generate more than 10 triples... (Why?)

All work and no play makes Jack a dull boy

(We should have kept the seed. Just in case.)

```
> mean(apply(replicate(1000,rbinom(3,1,0.5)),2,sum) <= 2)
[1] 0.858
> mean(apply(replicate(1000,rbinom(3,1,0.5)),2,sum) <= 2)
[1] 0.854
> mean(apply(replicate(1000,rbinom(3,1,0.5)),2,sum) <= 2)
[1] 0.888
> mean(apply(replicate(10000,rbinom(3,1,0.5)),2,sum) <= 2)
[1] 0.8806
> mean(apply(replicate(10000,rbinom(3,1,0.5)),2,sum) <= 2)
[1] 0.8761
> mean(apply(replicate(1000000,rbinom(3,1,0.5)),2,sum) <= 2)
[1] 0.875105
> mean(apply(replicate(1000000,rbinom(3,1,0.5)),2,sum) <= 2)
[1] 0.874815
> mean(apply(replicate(1000000,rbinom(3,1,0.5)),2,sum) <= 2)
[1] 0.875326
> mean(apply(replicate(10000000,rbinom(3,1,0.5)),2,sum) <= 2)
[1] 0.8748622
```

The moral so far?

Does the above confirm that

the *probability* of an event is something that pertains to the relative frequency of occurrence of an event, in a number of repetitions - where *we believe that increasing the number of repetitions gives us better hold on the probability*"

We are still somewhat at intuitive and at the same time learning level here. We know the solution; we can it couple of times at different replication sizes and note how the number gets closer to it (?) and fluctuates more closely to it (?)

Although we will also look at how to assess the precision using probabilistic tools for certain type of problems, this intuitive approach can still help us in quite (more) general domain of problems

The Playbook of Monte Carlo: Intuitive Approach

Monte Carlo, or simulation, methods estimate probabilistic quantities that are difficult to calculate exactly, by their estimates based on repetitive evaluation of computer random numbers related to the problem. (For instance, probabilities are estimated by relative frequencies.)

To get an idea about the quality of such an estimate, the *intuitive approach* starts with some small number of replications first (*very* small to debug the code) and then gradually increases the number of replications, trying first some moderate number of replications - when the result is obtained instantly - and repeating several times. If the results with growing number of replication exhibit less and less fluctuations, then a sort of empirical vindication is reached: it is likely that the path undertaken is the right one, and that when the result with desired precision is obtained not instantly, but in several minutes, hours, or days, it will not be a disappointment.

Classical Monte Carlo

Classical Monte Carlo deals with the estimation of expected values, with probabilities also falling under this case. This particular problem allows for more than merely intuitive approach: it is possible to assess the precision more rigorously via existing mathematical theorems.

However, intuitive approach can be used even in situations when such theorems are not available - but we still believe in a similar favorable type of behavior.

Notation

We are after quantity ϑ

We compute its estimate T_N

as a function of several random numbers

the subscript N indicating how many random numbers were used in the computation of T_N – the higher N, the more of those

- so we should perhaps write T_N as a function of those random numbers, but to simplify the notation, we rather do not

Performance guarantees I: consistency

The "performance guarantees" we are to discuss here are mathematical theorems, not empirical observations.

Consistency says mathematically that

$$T_N \quad \text{converges} \quad \text{to } \mu \quad (\text{in some sense}) \quad \text{when } N \to \infty$$

A formal expression of "in some sense" is habitually in terms of the convergence in probability:

$$P[|T_N - \vartheta| \geqslant \varepsilon] \to 0 \quad \text{when } n \to \infty \quad \text{for every } \varepsilon > 0$$

Consistency is, in a sense, a minimal requirement. It does not say anything about precision, it does not say how fast the probability approaches zero - it says only that after all, eventually, the objective will be reach.

Without consistency, the whole scheme may turn out rather ill-conceived and the estimates T_N obtained for various n rather misleading. Even when an appropriate theorem is not available, we should at least have some hope that consistency takes place anyway.

The median

Not everything in probabilistic/statistical context are means; for instance, the *median* of a probability distribution P is defined to be a number $\mathfrak{m}$ such that

$$\mathrm{P}((-\infty, \mathfrak{m}]) \geqslant 1/2 \quad \text{and} \quad \mathrm{P}[(\mathfrak{m}, +\infty)) \geqslant 1/2$$

A very common situation is that

$$\mathrm{P}((-\infty, \mathfrak{m}] = \mathrm{P}[(\mathfrak{m}, +\infty)) = 1/2$$

In such a case, the median is $\mathrm{F}^{-1}(1/2)$

where F is the cumulative distribution function of P

Median is not a mean and cannot be interpreted as such. Nonetheless, random numbers can be used also to get some knowledge about it. Suppose that the X_i's follow exponential distribution with $\lambda = 1$; we would like to have a look at what is the median is at this case.

Experiment with the median

This is an example of an intuitive approach to Monte Carlo

```
> median(rexp(10))
[1] 0.9457155
> median(rexp(1000))
[1] 0.6670174
> median(rexp(1000000))
[1] 0.694167
> median(rexp(10000000))
[1] 0.6931607
```


Again, this is still very easy problem: we actually know the result more or less in a closed form; an R function for its calculation is

```
> log(2)
[1] 0.6931472
> qexp(0.5)
[1] 0.6931472
```

We can generalize it to a more general case: an arbitrary quantile, $F^{-1}(p)$ for $p \in (0, 1)$

Consistency of the median

In general, median is not mean, and even cannot be expressed through it, so the rigorous Monte Carlo theory for means does not apply.

Nonetheless, we have still establish consistency here, as there is a convergence theory for the median (similar to that of the mean). It can be shown that if M_n is the median calculated from n independent random variables with the same distribution, all of them having the median m, then M_n converges to m in probability

$$P[|M_n - m| \geqslant \varepsilon] \to 0 \quad \text{as } n \to \infty \text{ for every } \varepsilon > 0,$$

So, we have some guarantee that with growing n, our estimates are more and more closer to the actual value of the median

Wrapping up the philosophy here

The notion of randomness adopted here would require that the outcomes of a computer random number generator should be "beyond reach"

> Well, are they not? We can hardly foresee what will come next, and even less control it. We have only a somewhat "godly" option of restarting everything again, or starting another round if we did not like the previous one

The notion of probability adopted here, "frequency in a long run" says that

> we should have well defined "chance situation: repeatable situations (we do have it here, right?) in well-defined sample spaces (we do have them, right?) There are no numbers going from $-\infty$ to ∞, everything is finite (although albeit sometimes large) and neither any numbers with infinite decimal expansion
>
> and that probability is then something that can be estimated as the relative frequency of occurrence of an event in a number of repetitions of the experiment; we believe that increasing the number of repetitions gives us better estimate of the probability

Is it?

```
> mean(apply(replicate(1000000,rbinom(3,1,0.5)),2,sum) <= 2)
[1] 0.875507
```

... about 5 seconds

```
> mean(apply(replicate(10000000,rbinom(3,1,0.5)),2,sum) <= 2)
[1] 0.8750205
```

... less than a minute (about 50")

```
> mean(apply(replicate(100000000,rbinom(3,1,0.5)),2,sum) <= 2)
[1] 0.8749999
```

... after a couple of minutes (about 8'38")

So: if you cannot wait, or computer cannot handle it...

And finally, what is "long run"?

Remember: the probability that a randomly drawn Canadian is female?

Simple: $$\frac{\text{number of Canadian females}}{\text{total number of Canadians}}$$

Here, "long run" is to go over all Canadians (and all Canadian females)

What is a "long run" in the computer?

The period of a random number generator

Recall: not only the computer environment is deterministic, but it is also finitary: there is only finite number of numbers (albeit it can be very large) representable in a computer. That is, if the state of a deterministic recursive function - what a computer random number generator is - is fully described a vector of numbers, then it must happen that after many iterations the state became the same as it was once before.

In other words, every random number generator has a **period**: the number of repetitions after which it starts to repeat the same random numbers. This is the "long run" for a random number generator. If we would be able to run estimation of a probability via random numbers up to this point, we would obtain its "true", "population" value.

The "Mersenne-Twister" random number generator, which is now a default in R, has period $2^{19937}-1$; for the period of other alternatives, see the output of `?RNG`.