# Contents

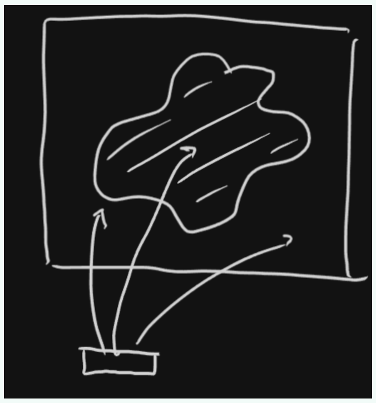
# Chapter 1

# Montecarlo Simulations

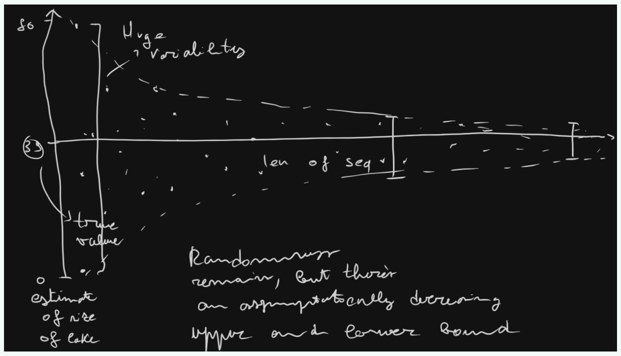
1. Simulation ([[Sampling]]) 2. Integration ([[Expectations]]) 3. Optimisation
Called "Montecarlo" (casino) because of its use of randomly generated numbers. These are used to solve something that isn't random.

**Example 1.0.1**

Given a field with a fixed area (eg. $20km^2$) and a lake inside of it with an irregular shape with its own area.

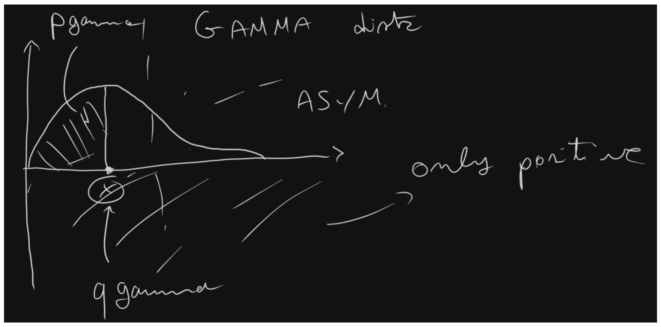


How do we estimate (compute/measure is impossible - Coastline Paradox) the area of the lake? Note: the area of the lake isn't random, but injecting randomness makes it much easier to solve.
We can shoot cannon balls into the field **at random**, recording if it hit water or not. By repeating this process, we get a long string of binary values which were **randomly generated** (in a weird way).
We can now define a random variable $X$ (in this case a Bernoulli) with $n$ recorded outcomes $X_1, ..., X_n$
Seen as the area of the lake and field don't change, the probability of hitting water remains the same. So for each random variable $P(X_1 = 1) = ... = P(X_n = 1)$, they are **identically distributed**.
Even if we keep hitting grass in one direction, we can't decide to move the cannon based on this information, because this wouldn't be random and it would cause dependence between the n random variables, which need to be iid (independent identically distributed).
Given $A$ ="hit water", $P(A)\alpha\frac{\text{Area Lake}}{\text{Area Field}}$. So if we want to estimate the area of the lake, we can multiply the field area by the probability of hitting water. The measurement has become probabilistic.
To find such probability, we can find the avg. value of $X$ (sample proportion/mean). The more samples we have, the more precise the measurement (thanks to CLT).


80
Huge variability
33
len of seq
true value
0
estimate of size of lake
Randomness remain, but there an asymptotically decreasing upper and lower bound

## 1.1 R

Suite of pre-implemented functions, use them!
Gamma distribution is a continuous distribution of positive random variables.



Use ‘rexp‘, ‘rgamma‘, ect. for preloaded distributions (you can search for the name in the help, selecting from a set of 4 different funcitons, d for density, p for probability, q to get value from probability, r for random selections).
How to sample from *any* distribution? For example, in Baeysian statistics we don’t know the shape of the posterior.
To view data, we can use the ‘hist‘ function for a histogram. If we want densities, we can use ‘freq = FALSE‘ (this is what we’ll use). This is actually an estimator of the distribution of the random variable. We can also use ‘curve‘, which is also an estimator, but a smooth one.
Be careful about the size of the plots (don’t be tricked). Height of the histogram estimates the density, but we can’t tell if they’re independent. The second plot is used for this case, plotting points in a 2d space with slightly shifted values. The acf (auto correlation funciton) plot measures correlation of a time series, by taking a sequence and shifting it many times, recording the correlation of a vector with itself. The x axis indicates the "lag", so at 0 it’s 1 (correlation between two identical vectors). If our values are independent, we should see no correlation (close to 0).
By setting the seed, we can fix the pseudo-random sequence.

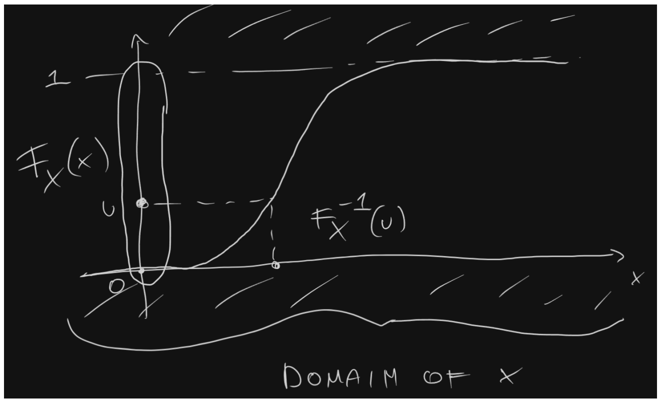
**Example 1.1.1** (Field example in R)
Matrix with ones (lake) and zeros (field). We can plot this now. We then randomly select a matrix element and record if it’s a 0 or 1.
Note that we’re using discrete values. We can sample using the ‘sample‘ function, with which we can define the set of tuples to sample from, the amount, with or without replacement (in our case the former with ‘replace=TRUE‘).
The accuracy depends on the amount of random variables whose outcome we can observe, at a cost to computational time.
To define a function, the correct syntax is: Name <- function(parameters)

## 1.2 Probability Integral Transform

Remember to search for other packages that might implement the funciton
We have a starting funciton of which we take the inverse. We are using the cumulative integral function, thus the "Integral" part.
The PIT can produce any random variable starting from a uniform. Lets say we want a random variable with density $f$ and cdf $F$:

$$F_X(x) = \int_{-\infty}^{x} f_X(t)dt$$

We set $U = F_X(X)$ and solve for $X$. Note that $F_X$ has to be invertible, although this is almost always the case.



The cdf always (with some exceptions) exists, but the density might not. So the first thing we find is the cumulative, and if it exists we can calc the integral and get the density.
We use the definition of the generalised inverse:

$$F^{-1}(u) = inf\{x; F(x) \geq u\}$$

If $U \sim Unif(0,1)$, then $F_X^{-1} = X$. This only works for the uniform distribution.
We have that $F_X(x) = P(X \leq x) = \int_{-\infty}^{x} f_X(t)dt$. In general, it's a **monotonic, non-decreasing** function. Outside of weird cases (greater than numerable amount of discontinuous points in $f_X$ i think), it's continuous.
We can't sample directly from X because we don't know its distribution. But we can select a random value from $F_X(x)$ then find its inverse, but does this truly give the correct distribution?
The uniform distribution has some cool props: - $U \sim Unif(a,b) \iff \forall u \in [a,b].P(U \leq u) = F_U(u) = u$
Thus:

$$P(F_X^{-1}(U) \leq x) = P(F_X(F_X^{-1}(U)) \leq F_X(x) = P(U \leq F_X(x)) = F_X(x)$$

So we can say

$$X = F_X^{-1}(U)$$