Contents

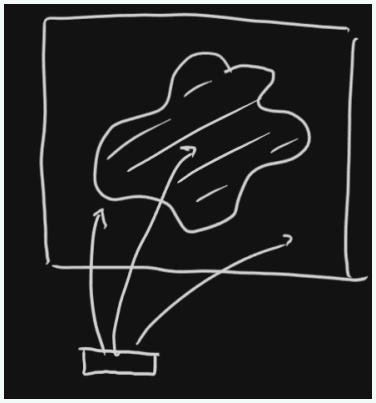
Chapter 1

Montecarlo Simulations

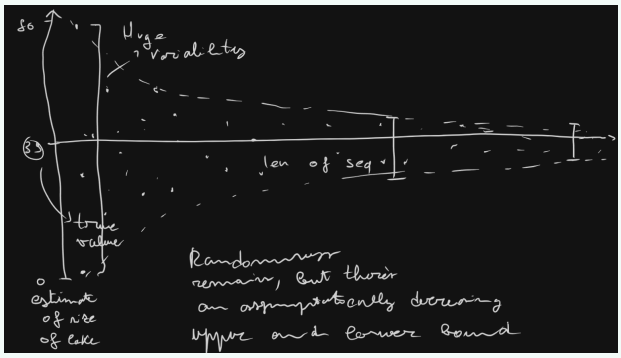
1. Simulation ([[Sampling]]) 2. Integration ([[Expectations]]) 3. Optimisation
Called "Montecarlo" (casino) because of its use of randomly generated numbers. These are used to solve something that isn't random.

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.

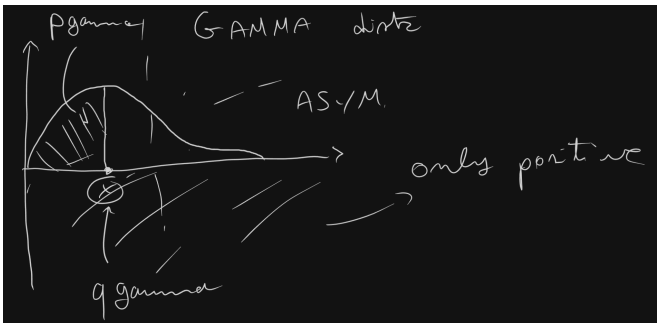


How do we estimate (compute/measure is impossible - Coastline Paradox) the area of the lake? Note: the area of the lake isn't random, but injecting randomness makes it much easier to solve.
We can shoot cannon balls into the field **at random**, recording if it hit water or not. By repeating this process, 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).



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)

Chapter 2

Sampling

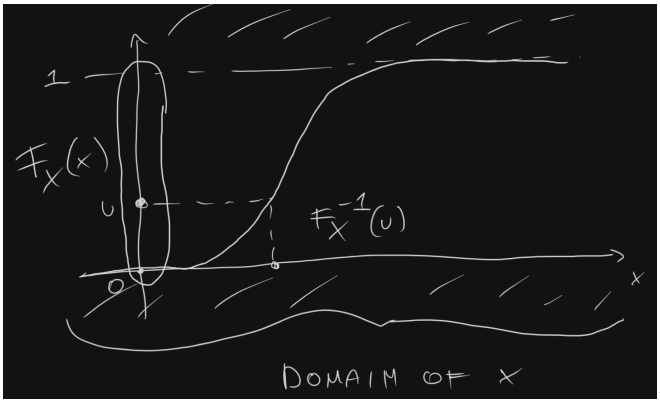
2.1 Direct Continuous

2.1.1 PIT

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

2.1.2 Known Distributions

Used when a distribution is easy to construct using other distributions which are easy to simulate. Given $X_i \sim Exp(1)$ i.i.d, three standard distributions can be derived as:

- $Y = 2\sum_{j=1}^{v} X_j \sim \chi^2_{2v}$ (Chi-squared distribution)
- $Y = \beta \sum_{j=1}^{a} X_j \sim G(a, \beta)$ (Gamma distribution)
- $Y = \frac{\sum_{j=1}^{a} X_j}{\sum_{j=1}^{a+b} X_j} \sim Be(a, b)$ (Beta distribution)

Where $v, a, b \in \mathbb{N}^* = \{1, 2, 3, ...\}$.

Gaussian

$F_X(x) = (x) = P(X \leftarrow x)$ $X \sim N(\mu, \sigma^2)$, has no closed form solution (no analytical solution), so we can't use PIT (we can't invert it).
Box-Muller algorithm is a direct transformation method which gives two normals from two uniforms. We don't have to remember the algorithm. It's **exact**, but not the most efficient.
Why is there π in a Gaussian distr? Looking at the algorithm, we use trigonometric functions, which is weird.
R uses the PIT by using a very close approximation of the cdf which is invertible.

Multivariate Gaussian

Multivariate Gaussian instead of having a single variable $X \sim N(\mu, \sigma^2)$, we have a vector:

$$() \sim N_p(\mu, \Sigma)$$

Where μ is a vector and Σ is a Matrix that indicates the variances (on the diag) and covariances between each pair of RV. It's called a variance and covariance matrix, and it's simmetric and positive definite (all variances are different to 0).
Each individual RV of a multivariate gaussian is a gaussian (gaussian family is closed under). Note, the opposite is not true.
If Σ is not diagonal, it means at least two RV are dependent.
This pops up for PCA.
Inverse standardisation for multivariate: How do we square a matrix? Cholesky! We take the Cholesky decomposition $\Sigma = AA^T$ (more or less the square root of Σ).

$$Y \sim N_p(0, 1) = AY \sim N_p(0, \Sigma)$$

There are R packages that replicate these steps, but it's faster and better to sample from a gaussian and then stretch and squeeze the vector the rebuild the covariance structure. 2

2.1.3 General Case (Using Derivative)

GIven $X \sim f_X(x)$ and $Z = g(X) : X = g^{-1}(Z)$ (g allows an inverse), then:

$$f_Z(z) = f_X(g^{-1}(z)) \cdot |\frac{\delta g^{-1}(z)}{\delta z}|$$

Note:
We don't always have a density.

Example 2.1.1 (Uniform R.V.)

$U \sim Unif(0, 1) \rightarrow f_U(u) = \begin{cases} 0 & u \notin [0, 1] \\ 1 & u \in [0, 1] \end{cases}$

$X = (b - a) \cdot U + a$ (linear transformation of a uniform). What is the distribution of X?

First we calc the inverse:

$$X - a = (b - a) \cdot U$$
$$U = \frac{X - a}{b - a} = g^{-1}(\cdot)$$

Now the derivative:

$$\frac{\partial g^{-1}(X)}{\partial x} = \frac{\partial[\frac{X-a}{b-a}]}{\partial x}$$
$$\frac{\partial[\frac{X-a}{b-a} - \frac{a}{b-a}]}{\partial x}$$
$$\frac{\partial[\frac{X}{b-a}]}{\partial x} = \frac{1}{b-a}$$

So, applying the general transform:

$$f_X(x) = f_U(\frac{x-a}{b-a}) \cdot \frac{1}{b-a} = \begin{cases} 0 & f_U = 0 \iff x \notin [a,b] \\ \frac{1}{b-a} \end{cases}$$

Thus $X \sim Unif(a,b)$

Example 2.1.2 (Gaussian)

$X \sim N(\mu, \sigma^2)$ and $Z = \frac{X-\mu}{\sigma} = g(X)$ (never seen before transformation)
$X = Z \cdot \sigma + \mu = g^{-1}(X)$
So the derivative is simply

$$\frac{\partial g^{-1}(Z)}{\partial z} = \sigma$$

$$f_Z(z) = f_X(g^{-1}(z)) \cdot \sigma$$
$$\frac{1}{\sqrt{2\pi}} \cdot \frac{1}{\sigma} \cdot e^{-\frac{(z\sigma+\mu-\mu)^2}{-2\sigma^2}} \cdot \sigma$$
$$\frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{z^2}{2}}$$

This is the density of the standard gaussian distribution $N(0,1)$, and this is why we standardize in this way.

We now look at direct distributions for discrete random variables. After, we'll look at indirect meethods with the accept-reject algorithm.

2.2 Direct Discrete

Non-continuous cdf.

2.2.1 General Algorithm

Based on inverse transform, but not the same as PIT. This is not used much, because most discrete distributions are known.
We have to store the values of the cdf (we have to be able to compute it). Then sample from a uniform and check where the value falls:

$$X = k \iff p_{k-1} < U < p_k$$

Note:

If we have a very large Poisson, we'll have to store many probability values for computation.

$$P_X(x) = P(X = x) = \binom{10}{x} p^x (1-p)^{10-x}$$

The problem is calculation (binomial can get very large), expecially for unlimited distributions like poisson.
Given two RV X, Y, then

$$X = Y \iff F_X(x) = F_Y(y)$$

The binomial distribution isn't very used, in contrast to Poisson which was very used in sports analysis seen as it **counts** "stuff" in a specific time window (highly connected to exponential). It's also known as the "rare distribution", seen as the only parameter λ tells you on avarege the number of events you expect to happen, but this is also the value of the variance (which doesn't happen in real life)

$$X \sim Poiss(\lambda) \to \begin{cases} E(X) = \lambda \\ Var(X) = \lambda \end{cases}$$

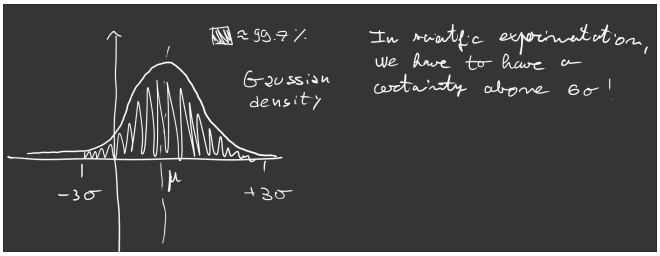
This is called the **overdispersion problem** ($Var(X) \gg E(X)$), and it's the main reason for moving on to more comlex models. So the computation is quite difficult, also because of factorial and exponential terms.

$$Poisson(\lambda) \to \lambda \to +\infty \to\sim\sim N(\lambda, \lambda)$$

So we can use a Gaussian to aproximate a Poisson distribution.
When $\lambda > 50$ we can say that most of the probability of the RV is

$$\lambda + = 3\sqrt{\lambda}$$

So this is the range of values for X that we should consider to calculate the distribution. This is because of its similarity to the gaussian.



So some of the values in the domain of X are useless to check, seen as they're highly unlykely

Example 2.2.1

$X \sim Poisson(100)$. If we consider the Gaussian aprox. most of the values will be in $(70, 130)$.
So if

2.3 Indirect

Accept-reject