Contents

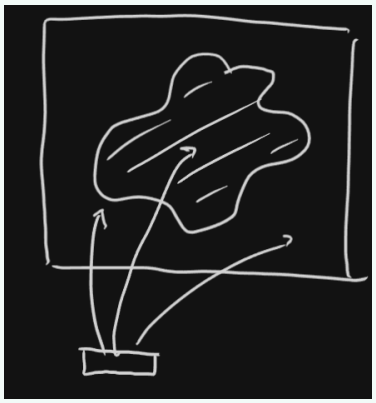
Chapter 1

Montecarlo Simulations

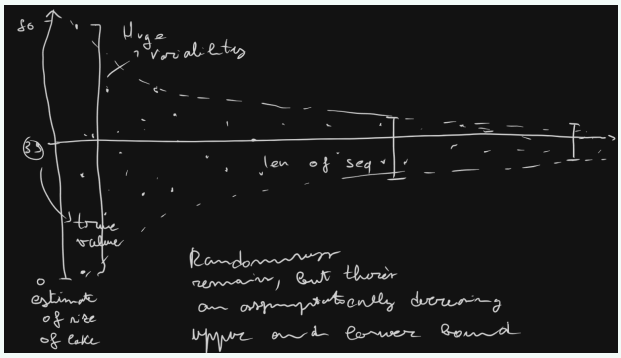
1. Simulation ([[Sampling]]) 2. Integration ([[Expectations]]) 3. Optimisation
Called "Montecarlo" (casino) because of its use of randomly generated numbers. These are used to solve something that isn't random.

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.

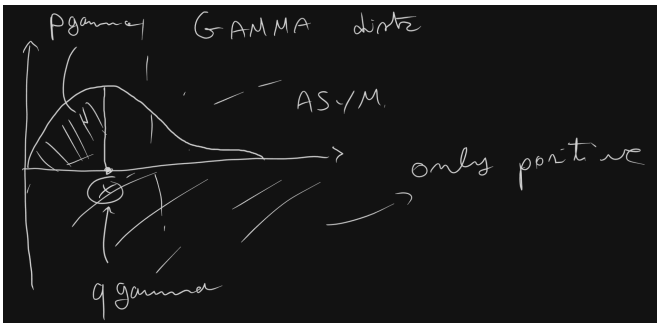


How do we estimate (compute/measure is impossible - Coastline Paradox) the area of the lake? Note: the area of the lake isn't random, but injecting randomness makes it much easier to solve.
We can shoot cannon balls into the field **at random**, recording if it hit water or not. By repeating this process,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

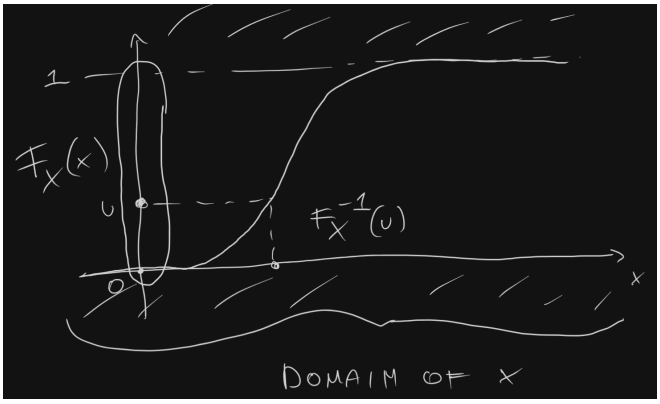
2.1 Direct Continuous

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)

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

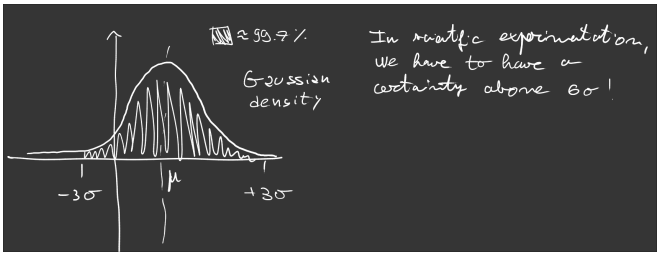
This is called the **overdispersion problem** ($Var(X) \gg E(X)$), and it's the main reason for moving on to more comlex models. So the computation is quite difficult, also because of factorial and exponential terms.

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

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

2.3.1 Bayes' Theorem

Giuro che questo non lo rifaccio

Continuous

Given two cont RV X, Y

$$f_{X|Y=y}(x)$$

is the distribution of X given some info (Y)

$$(X|Y = y) \text{ random variable}$$

we're just focusing on a specific cross section of the population (e.g. income at a specific age).
We want to find its density, we can use Bayes

$$f_{X|Y=y}(x) = \frac{f_{X,Y}}{f_Y(y)} = \frac{f_{Y|X=x}(y)f_X(x)}{f_Y(y)}$$

When you want to move a RV behind the conditioning line, you divide by the marginal.
We can use the marginal distribution for the denominator

$$\frac{f_{X,Y}(x,y)}{\int_Y f_{X,Y}(x,t)dt}$$

by marginalizing out Y (same idea as the discrete version).

2.3.2 Expectation

Given X cont. RV $X \sim f_X(x)$:

$$E_X(X) = \int_{D_X} x f_X(x) dx$$

Note:
Always indicate the distribution used as a subscript. This will become important.

$$E_Y(f_{X|Y}(X|Y)) = \int_{D_X} f_Y(y) f_{X|Y}(X|Y) dy$$

This is a product of a conditional times a marginal (moving Y from the back to the front). So we have the joint

$$= \int_{D_Y} f_{X,Y}(X,Y) dy = f_X(x)$$

So we can use expectation to marginalize out variables. Why is this important?
If we have values sampled from a conditional distribution, taking the expected values (average) we're reming the influence of Y. This is the building block for the Montecarlo-Markov Chain (most widely used Montecarlo method).

2.3.3 Accept-Reject Method

We need to know:

- When it works
- Main ingredients
- No proof!
- The actual algorithm

It's the first indirect method for sampling. It works with two distributions:

- A **candidate** dist.
- A **targe** dist.

Remember the lake? Let's say the area is $1 \times 1 = 1km^2$. We are shooting cannon balls at random, so

$$X \sim Unif(0, 1)$$

For every two random values we get a result

$$(u_1, u_2) = \text{HIT!}$$

So we generate a uniform for each side.
By changing the shape of the lake, we can get a more interesting result

We're now randomly testing a **density function** with two uniform RV.

Ingredients

- **Denisty** f of interest (only functional form) and be able to compute it.
- **Candidate** distribution g.

The candidate choice is up to us. Some are smarter than others, but as long as we don't pick functions that don't satisfy a certain condition (we'll see later), it's ok. In the exam we'll be given the candidate.
The candidate generates a value. We need to test

$$u \leqslant \frac{f_X(y)}{M g_Y(y)}$$

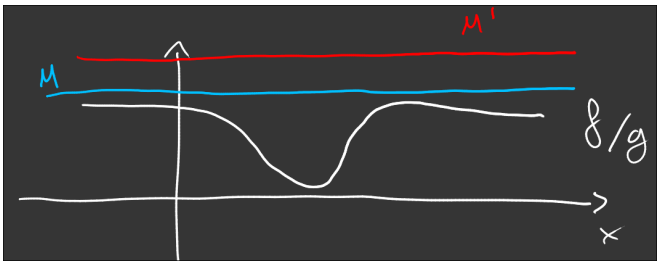
If it's satisied we accept (it comes from f_X). Otherwise, we reject it and try again (similar to the discrete case).
Proposal density constraints:

- Target and candidate have to have compatible supports ($S_X \subseteq S_Y$).
- There is a constant M with $f(x)/g(x) \leqslant M \ \forall x$.

The last requirement is less trivial. It means that the ratio between the distributions has to be bounded.
If these are satisfied, then

1. Generate $Y \sim g$ and **independently** generate from $U \sim Unif(0, 1)$.
2. Check if $U \leqslant \frac{1}{M} \frac{f(Y)}{g(Y)}$ then set $X = Y$ (accept)
3. If it's not satisfied, discard Y and U and start again.

Note that $M = sup_x \frac{f(x)}{g(x)}$. The crucial requirement that M exists means that M can have many different values.



The best value for M is the one that reduces the area where the points fall and don't belong to the density, reducing efficiency

$$P(\text{Accept}) = \frac{1}{M}$$

So in the case of the graph you should choose the blue line.

Proof

Does it work? We need to think from a statistical pov: Are the accepted values coming from $f_X(x)$?
What is the distr. of the accepted values? What is the CDF?
Let's define the event

$$A = \text{"Accept a value"}$$

Where A is basically

$$Y|U \leqslant \frac{1}{M}\frac{f(Y)}{g(Y)}$$

so a distribution conditional to a test where $U \sim Unif(0,1)$ and $Y \sim g(y)$. Lets look at the CDF

$$\begin{aligned}
P(Y \leqslant x | U \leqslant \frac{1}{M}\frac{f(Y)}{g(Y)}) &= \frac{P(C,D)}{P(D)} \\
&= \frac{P(Y \leqslant x, U \leqslant \frac{1}{M}\frac{f(Y)}{g(Y)})}{P(U \leqslant \frac{1}{M}\frac{f(Y)}{g(Y)})} \\
&= \frac{\int_{-\infty}^{x}\left(\int_{-\infty}^{\frac{1}{M}\frac{f(Y)}{g(Y)}} 1\,du\right)\cdot g_Y(y)\,dy}{\int_{-\infty}^{+\infty}\left(\int_{-\infty}^{\frac{1}{M}\frac{f(Y)}{g(Y)}} 1\,du\right)\cdot g_Y(y)\,dy} \\
&= \frac{\int_{-\infty}^{x}\frac{1}{M}\frac{f(Y)}{g(Y)}g_Y(y)\,dy}{\int_{-\infty}^{+\infty}\frac{1}{M}\frac{f(Y)}{g(Y)}g_Y(y)\,dy} \\
&= \frac{\int_{-\infty}^{x} f(Y)\,dy}{\int_{-\infty}^{+\infty} f(Y)\,dy} \\
&= \frac{P(X \leqslant x)}{1} = F_X(x)
\end{aligned}$$

So the accepted values have the same cdf as X!

Note:

g and M are arbitrary and don't matter for the correctness of the algorithm.

2.4 Accept - Reject algorithm

TODO

Example 2.4.1 (AC- algh)

We aim to simulate a random variable X from the Standard Normal distribution using the Accept-Reject method with a Laplace (Double Exponential) candidate distribution.

- Target density (f): The Standard Normal distribution ($\mathcal{N}(0,1)$):

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \quad x \in \mathbb{R}$$

- Candidate Density (g): The Laplace distribution ($Laplace(1)$):

$$g(y) = \frac{1}{2} \exp(-|y|), \quad y \in \mathbb{R}$$

We need to find a constant M such that $f(x) \leq Mg(x)$ for all x. This is equivalent to finding the maximum of the ratio $f(x)/g(x)$:

$$\frac{f(x)}{g(x)} = \frac{\frac{1}{\sqrt{2\pi}} e^{-x^2/2}}{\frac{1}{2} e^{-|x|}} = \sqrt{\frac{2}{\pi}} \exp\left(-\frac{x^2}{2} + |x|\right)$$

To maximize this ratio, we maximize the expression $h(x) = \sqrt{\frac{2}{\pi}} \exp\left(-\frac{x^2}{2} + |x|\right)$, for doing that we have to set the derivate to 0:

$$h'(x) = \sqrt{\frac{2}{\pi}} \exp\left(-\frac{x^2}{2} + |x|\right) \left(-x + \frac{x}{|x|}\right) = 0$$

Solving this, we find that the maximum occurs at $x = 1$ and $x = -1$. Evaluating $h(x)$ at these points gives:

$$M = h(1) = h(-1) = \sqrt{\frac{2}{\pi}} e^{-1/2+1} = \sqrt{\frac{2e}{\pi}}$$

So we have

$$U \leqslant \frac{1}{M} \frac{f(Y)}{g(Y)} = \frac{1}{\sqrt{\frac{2e}{\pi}}} \cdot \sqrt{\frac{2}{\pi}} \exp\left(-\frac{Y^2}{2} + |Y|\right) = e^{-1/2} \exp\left(-\frac{Y^2}{2} + |Y|\right)$$

```
dlaplace <- function(x) 0.5 * exp(-abs(x)) # this is the Laplace density
curve(dlaplace(x), 6,6) # BE CAREFUL, if you dont specify the range curve will assume [0,1]
plaplace <- function(x) ifelse(x < 0, 0.5 * exp(x), 1 - 0.5 * exp(-x)) # Laplace CDF
qlaplace <- function(u) ifelse(u < 0.5, log(2 * u), -log(2 * (1 - u))) # Laplace Quantile function -> I
rlaplace <- function(n) qlaplace(runif(n)) # n random variables from Laplace distribution, using PIT

# Accept-Reject algorithm to sample from N(0,1) using Laplace(1) as candidate
rnorm_ar <- function(n, g, rg, M, report = true){ # g is the density of candidate, rg is the random gen
  x<- rep(0,n) # vector to store accepted values
  ntry <- 0 # number of tries
  for (i in 1:n){
    done <- false
    while (!done){
      # FOUR STEPS OF AR ALGHORITHM
      ntry <- ntry + 1
      y <- rg(1) # sample from the Proposal
      u<- M * runif(1) # sample from Uniform
      if (u < (dnorm(y)/g(y))) done <- true # accept
```

```
        # PAY ATTENTION: dnorm is the density of the target distribution SO IT MUST BE CHANGED IF THE TARG
    }
    x[i] <- y # store accepted value
  }
    if (report){
      cat("Acceptance Rate:", n / ntry, "\n")
    }
    retrun (x)
}

Nsim <- 10^4
M <- sqrt(2*exp(1)/pi)
x<- rnorm_ar (Nsim, dlaplace, rlaplace, M)
```

An other algh in R:

```
  # GEnereting beta from uniform
  a = 2.7
  b = 6.3
  curve (dbeta(x,a,b), col="red")

  # f is already our ratio cause g(y)=1 (uniform density)
  f<-function(x) dbeta(x,a,b)
  M <- optimize (f, interval = c(0,1), maximum=T)$objective #

  rbeta <- function (n,a,b.M,report = TRUE){
    x<- rep(0,n)
    ntry <- 0
    for(i in 1:n){
      done = FALSE
      while(!done){
        ntry<- ntry +1
        y=runif(1)
        u=M*runif(1)
        if (u<dbeta(y,a,b)/dunif(y)) done = TRUE
      }
      x[i]<- y
    }
    if (report) cat("I needed ", ntry, "trials to get", n, retrun(x))
  }
  Nsim <- 10^4
  set.seed(10)
  x - rbeta2(Nsim, a, b, M)

  par(mfrow = c(1,2))
  hist(x, freq= FALSE)
  curve(dbeta(x,a,b), col = "red", add= TRUE)
```

Alternative for ar IF YOU HAVE REDUCED COMPUTATIONAL RESOURCE

```
### Generating betas from betas
# ------------------------------------------------------------
# OBIETTIVO: Simulare da una Beta(2.7, 6.3) usando una Beta(2, 6)
# come densità strumentale (Proposal g).
# ------------------------------------------------------------

# 1. Definizione dei parametri della distribuzione Target f(x)
a = 2.7
```

```
b = 6.3

# 2. Definizione della funzione rapporto r(x) = f(x) / g(x)
# f(x) = dbeta(x, a, b)  [Target]
# g(x) = dbeta(x, 2, 6)  [Proposal]
f <- function(x) dbeta(x, a, b) / dbeta(x, 2, 6)

# 3. Calcolo della costante M
# M deve essere tale che f(x) <= M * g(x) per ogni x.
# Troviamo il massimo del rapporto f(x)/g(x) usando 'optimize'.
# $objective estrae il valore numerico del massimo.
M <- optimize(f, maximum=T, interval=c(0,1))$objective

# 4. Definizione della funzione di generazione (Algoritmo AR)
rbeta3 <- function(n, a, b, M, report=TRUE){

    # Inizializzazione del vettore che conterrà i campioni accettati
    x <- rep(0, n)

    # Contatore per monitorare l'efficienza (totale tentativi fatti)
    ntry <- 0

    # Ciclo per generare n valori accettati
    for (i in 1:n){
        done <- FALSE

        # Ciclo while: continua a provare finché non accetta un candidato
        while (!done){
            ntry <- ntry + 1

            # A. Generazione del Candidato Y dalla Proposal g ~ Beta(2, 6)
            y <- rbeta(1, 2, 6)

            # B. Generazione della variabile ausiliaria U
            # Qui si usa u scala M: u = M * Uniform(0,1)
            u <- M * runif(1)

            # C. Test di Accettazione
            # La condizione teorica è: U_raw <= f(y) / (M * g(y))
            # Moltiplicando per M, diventa: u <= f(y) / g(y)
            if (u < dbeta(y, a, b) / dbeta(y, 2, 6)) done <- TRUE
        }
        # Se la condizione è vera, salviamo il candidato y nel vettore x
        x[i] <- y
    }

    # Report sull'efficienza dell'algoritmo (tasso di accettazione)
    if (report) cat("I needed ", ntry, " trials to get ", n, " samples.\n")

    return(x)
}

# 5. Esecuzione della simulazione
Nsim = 10^4
# Esempio di chiamata (nota: la riga è tagliata nell'immagine)
# campioni <- rbeta3(Nsim, a, b, M)
```

Example 2.4.2 (Penso esercizio)

So we have $f(x) \alpha \exp\left(-\frac{x^2}{2}\right)[\sin(6x)^2 + 3\cos(x)^2 \cdot \sin(4x)^2 + 1]$

1. Plot $f(x)$ to show $M \cdot g(x)$ bounds it where g_Y is the density of a Std. GAUSSIAN $g_Y(y) = \frac{1}{\sqrt{2\pi}} \cdot \exp\{-\frac{y^2}{2}\}$ and find M.
2. Generate 2500 numbers from f_X.
3. Deduce from the acceptance rate of the AR algorithm, an approximation of the normalizing constant.

```
#UNNORMALIZED (you have proportional to in the text of the exer)
fx <- function(x){
  exp(-x^2/2)*(sin(6*x)^2 + 3*(cos(x)^2)*(sin(4*x)^2)+1)
}

fx.over.g <- function(x){
  fx(x)/dnorm(x) # if we have a differente propoasal dnorm is gonna change
}

m <- optimize(fx.over.g, interval=c(-6,6), maximum = T)$objective
curve(fx, -9,9, ylim=c(-1,7)) # this is the density
curve(m*dnorm(x), -5,5, col="red", add=T)
curve(15*dnorm(x), -6,6, col="blue", add=T)

# B
AR.sampling <- function(n, f, g, rg, m, report=TRUE){
  x <- rep(0,n)
  tries <- 0
  for(i in 1:n){
    done <- FALSE
    while(!done){

      y <- rg(1)
      u <- runif(1)*m
      if (u <= f(y) / (g(y))) {
        done <- TRUE
      }
      tries <- tries + 1
    }
    x[i] <- y
  }
  if(report) cat("Total tries:", tries, "\n")
  return(list(x=x, tries = tries))
}
```

Chapter 3

Montecarlo Integration

3.1 Theory

3.1.1 Introduction

The goal is computing an integral of the from

$$I = \sin_{\mathcal{X}} h(x)f(x)dx$$

where:

- $\mathcal{X} \subseteq \mathbb{R}^n$ is the domain of integration
- $f(x)$ is a density function (can be unnormalized)
- $h(x)$ is the function we want to integrate

the core of montecarlo integration is in a interpretation of the integral as an expectation value of a RV. If we consider a RV X with density $f(x)$, then we can write

$$I = E_f[h(X)] = \sin_{\mathcal{X}} h(x)f(x)dx$$

this identity is crucial because it allows us to use the properties of RVs to compute integrals.

Note:

although an integral it is not usual to appear in this form (for instance $\int_a^b g(x)dx$), we can always rewrite it in this form by choosing a suitable density $f(x)$. For example, if we want to compute $\int_a^b g(x)dx$, we can choose $f(x)$ as the uniform density in $[a, b]$:

$$f(x) = \begin{cases} \frac{1}{b-a} & x \in [a, b] \\ 0 & otherwise \end{cases}$$

so that

$$\int_a^b g(x)dx = \int_a^b g(x) \cdot \frac{1}{b-a} \cdot (b-a)dx = (b-a) \int_a^b g(x)f(x)dx$$

once we have established this identity the method for approximating I is straightforward:

- Generation: generate n samples (i.i.d.) from the density $f(x)$: $X_1, X_2, \ldots, X_n \sim f(x)$
- Estimation: compute the sample mean of $h(X)$:

$$\hat{I}_n = \frac{1}{n} \sum_{i=1}^{n} h(X_i)$$

the value $\hat{I}_n$ is our estimate of the integral I.
Okay, gg for this introduction. Now let us to a concept that explain that:

3.1.2 The low of Large Numbers and low of Central limit

Consider an infinite sequence of independent and identically distributed (i.i.d.) random variables $X_1, X_2, \ldots$ defined on a probability space $(\Omega, \mathcal{F}, P)$. Let these variables have a finite expectation $\mathbb{E}[X_1] = \mu$.

Definition 3.1.1: Sample mean

it's defined the *sample mean* of the first n variables as:

$$\overline{X}_n = \frac{1}{n} S_n = \frac{1}{n} \sum_{i=1}^{n} X_i$$

The Law of Large Numbers (LLN) describes the convergence of $\overline{X}_n$ to the true parameter μ as the sample size n approaches infinity. There are two versions of this law, distinguished by the type of convergence.

Theorem 3.1.1 Weak Law of Large Numbers (WLLN) - Convergence in Probability

The sample mean $\overline{X}_n$ converges to μ in *probability*. That is, for any margin of error $\epsilon > 0$, the probability that the absolute difference between the sample mean and the true mean exceeds ϵ tends to zero as $n \to \infty$:

$$\lim_{n \to \infty} P\left(|\overline{X}_n - \mu| > \epsilon\right) = 0$$

Often denoted as: $\overline{X}_n \xrightarrow{P} \mu$.

Theorem 3.1.2 Strong Law of Large Numbers (SLLN) - Convergence Almost Surely

The sample mean $\overline{X}_n$ converges to μ *almost surely* (a.s.). That is, the event that the limit of the sequence $\overline{X}_n$ equals μ occurs with probability 1:

$$P\left(\lim_{n \to \infty} \overline{X}_n = \mu\right) = 1$$

Often denoted as: $\overline{X}_n \xrightarrow{a.s.} \mu$.

Note:

If the random variables also have a finite variance $Var(X_i) = \sigma^2 < \infty$, then the variance of the sample mean decreases linearly with n:

$$Var(\overline{X}_n) = Var\left(\frac{1}{n} \sum_{i=1}^{n} X_i\right) = \frac{1}{n^2} \sum_{i=1}^{n} Var(X_i) = \frac{n\sigma^2}{n^2} = \frac{\sigma^2}{n}$$

Example 3.1.1 (Code in R() for SLLN)

```
### LLN: Law of Large Numbers
    # ---- Sezione 1: Esempi preliminari ----
    n <- 1
    X <- rexp(n, rate=5) # generate one r.v., E(X)=1/lambda = 0.2
    n <- 20
    X.seq <- rexp(n, rate=5) # sequence of X_1, X_2, ..., X_n
    X.bar <- mean(X.seq) # sample mean

    ### ---- Sezione 2: Convergenza della Media (SLLN) ----
    n <- 5000
    X.seq <- rexp(n, rate=5)
```

```r
# Calcola la media cumulativa: (X_1+...+X_n) / n
X.bar <- cumsum(X.seq)/(1:n)  # sample mean sequence, for 1 we have X.bar[1]=X.seq[1],

plot(1:n, X.bar, type="l", col=1, lwd=1, ylim=c(0, 0.4))
abline(h=0.2, col=2, lwd=2)
# line for E(X) = 1/lambda = 0.2

# ------------------------------------------------------------
# what about the theoretical variance? (Var(X.bar) = sigma^2 / n)
# Var(X) = 1/25 = 0.04
plot(1:n, 0.04/(1:n), type="l", col=1, lwd=1, ylim=c(0, 0.04))
abline(h=0, col=2, lwd=2)
# ------------------------------------------------------------

# what if we actually compute the empirical variance of the process?
# Calcola la media cumulativa degli scarti quadratici
var.X.bar <- cumsum((X.seq-X.bar)^2)/(1:n)
# Scalando ancora per 1/n si ottiene una stima della Var(X.bar)
var.X.bar <- var.X.bar/1:n

plot(1:n, var.X.bar, type="l", col=1, lwd=1)
```

So we have convergence in:

- Probability (weak)
- A.S. (?) (strong)

TODO: understand this
While the Law of Large Numbers guarantees that the sample mean converges to the true expectation, the Central Limit Theorem (CLT) describes the *distribution* of the error for large sample sizes. This allows us to quantify the uncertainty of our Monte Carlo estimates.

Theorem 3.1.3 Central Limit Theorem (CLT)

Let $X_1, X_2, \dots$ be a sequence of independent and identically distributed (i.i.d.) random variables with:

- Finite expectation: $\mathbb{E}[X_i] = \mu$
- Finite variance: $Var(X_i) = \sigma^2 < \infty$

Then, the random variable $\sqrt{n}(\overline{X}_n - \mu)$ converges in *distribution* to a Normal random variable with mean 0 and variance σ^2 as $n \to \infty$:

$$\sqrt{n}(\overline{X}_n - \mu) \xrightarrow{d} \mathcal{N}(0, \sigma^2)$$

Equivalently, the standardized sample mean converges to a Standard Normal distribution:

$$\frac{\overline{X}_n - \mu}{\sigma/\sqrt{n}} \xrightarrow{d} \mathcal{N}(0, 1)$$

For a sufficiently large sample size n (simulation runs), the estimation error $(\overline{X}_n - \mu)$ is approximately Normally distributed:

$$(\overline{X}_n - \mu) \approx \mathcal{N}\left(0, \frac{\sigma^2}{n}\right)$$

This approximation justifies the construction of asymptotic confidence intervals. For instance, a 95% confidence

interval for μ is given by:

$$\left[\overline{X}_n - 1.96\frac{\sigma}{\sqrt{n}}, \quad \overline{X}_n + 1.96\frac{\sigma}{\sqrt{n}}\right]$$

In practice, since the true variance σ^2 is unknown, it is replaced by the sample variance estimator S_n^2.
We can now say something about the uncertainty of the result by using the gaussian.

3.2 Numerical Deterministic Aproach

Most of the time, this works and should be prefered. In R they are implemented as the functions *area* (can't use infty) and *integrate*. They don't work with multiple dimensions and we need some information about the funciton to integrate.

3.3 Montecarlo Method

Okay, let's formalize the montecarlo method for integration.
As said before, we want this identity:

Definition 3.3.1: Monte Carlo Integral Representation

Let X be a random variable with probability density function $f(x)$ defined on a domain $\mathcal{X} \subseteq \mathbb{R}^d$. We define the target integral I as the expected value of the function $h(X)$:

$$I = \mathbb{E}_f[h(X)] = \int_{\mathcal{X}} h(x)f(x)\,dx$$

assuming that the expectation exists and is finite, i.e., $\mathbb{E}_f[|h(X)|] < \infty$.

Then it's time for exstimator of montecarlo:

Definition 3.3.2: Monte Carlo Estimator for Integration

Given n independent and identically distributed (i.i.d.) samples $X_1, X_2, \ldots, X_n$ drawn from the probability density function $f(x)$, the Monte Carlo estimator $\overline{h}_n$ for the integral I is defined as:

$$\overline{h}_n = \frac{1}{n}\sum_{i=1}^{n} h(X_i)$$

where each X_i is a sample from the distribution defined by $f(x)$.

Example 3.3.1 (3.1)

use integrate() to compute

$$\Gamma(\lambda) = \int_0^{+\infty} x^{\lambda-1}exp(-x)dx \quad D = [0, +\infty[$$

which is the gamma distribution.

```
#Use of the integrate function
ch <- function(la){
  integrate(function(x){x^(la-1)*exp(-x)},0,Inf)$val
}

curve(lgamma(x), from=0, to=10)
plot(lgamma(seq(0.01,10,le=100)),
     log(apply(as.matrix(seq(0.01,10,le=100)),1,ch)),
```

```
      xlab="log(integrate(f))",
      ylab=expression(log(Gamma(lambda))),pch=19,cex=.6)
```

Here integrate works correctly.

Example 3.3.2 (3.2)

Consider a sample of $n = 10$ RVs from a Cauchy distribution with location parameter $\theta = 350$

$$X_1, X_2, ..., X_{10} \sim Cauchy(\theta = 350)$$

We want to calculate the marginal probability of the data we have conditional to the parameter theta

$$P(X_1, ..., X_n) = \int_\Theta P(X_1, ..., X_n|\theta)P(\theta)d\theta$$

We can reconcile the Bayesian representation with the usual frequentist approach by stating that $P(\theta)\alpha 1$ (flat prior) so that no value of θ is favoured, so we can say

$$= \int_{-\infty}^{+\infty} P(X_1, ..., X_n|\theta)d\theta$$

Assuming iid data, we can simplify as such

$$= \int_{-\infty}^{+\infty} \prod_{i=1}^{n} P(X_i|\theta)d\theta$$

But the Cauchy is a continuous distribution, so we can only use the distribution

$$= \int_{-\infty}^{+\infty} \prod_{i=1}^{n} \frac{1}{\pi(1 + (x_i - \theta)^2)}d\theta$$

Numerical Approach

- integrate()
- area()

```
#Comparison between integrate and area functions
cac <- rcauchy(10)+350
lik <- function(the){
  u <- dcauchy(cac[1]-the)
  for (i in 2:10) u <- u*dcauchy(cac[i]-the)
  return(u)
}
curve(dcauchy(x,location=350),from = 200, to=400)
integrate(lik,-Inf,Inf)
integrate(lik,200,400)

library(MASS)

cac <- rcauchy(10)
curve(dcauchy(x),from = -6, to=6)
nin <- function(a) integrate(lik,-a,a)$val
nan <- function(a) area(lik,-a,a)
x <- seq(1,10^3,le=10^4)
y <- log(apply(as.matrix(x),1,nin))
z <- log(apply(as.matrix(x),1,nan))
plot(x,y,type="l",ylim=range(cbind(y,z)),lwd=2)
lines(x,z,lty=2,col="red",lwd=2)
```

In this case, integrate doesn't work (as the mean of the distribution isn't defined). Area works, but we need to know where to integrate so it's kinda useless.