Simulation for Decision Making

Short Course

Jorge de la Vega Góngora

Statistics Department
Instituto Tecnológico Autónomo de México
email: jorge.delavega@itam.mx

May 14, 2024

Course Presentation

Course Outline

- Morning Session I 10:00 - 11:30
  - Introduction: Systems, Models & Simulation
  - Motivating examples of Stochastic Simulation
  - Types of simulation analysis
- Morning Session II 11:45 - 13:00
  - Random number generation
  - Random variable generation methods.
- Evening Session I 14:30 - 16:00
  - Input and Output analysis
  - Counterfactual analysis
- Evening Session II 16:15 - 17:30
  - Case Study: queues and inventories

Goals

- To study Monte Carlo simulation in the decision theory framework
- Practical applications of simulation to solve problems
- MC in a Business Model and counterfactual analysis
- Discrete event simulations
- Materials: https://github.com/jvega68/ISI

Morning Session I

Systems, models & Simulation I

- Many possible definitions of *system*. Working definition:

System and process (Schmidt & Taylor, 1970)

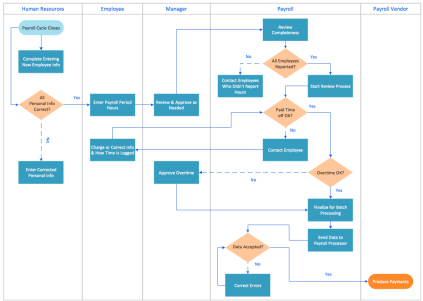
- A **system** is a collection of entities that act and interact together to accomplish some goal. It is composed by:
  - Entities: components of a system
  - Attributes: characteristics of the entities
  - Activities: actions that take time to execute
  - States: collection of variables necessary to describe the system in a perticular time
  - Events: instantaneous occurrence that change the state of the system

- Example: Payroll system:
  - Record of employees, hierarchy levels and salary scale
  - Record of extra time, abscences, new hires and quits.
  - Deductions, discounts, compensations, social security contributions
  - Bank accounts, deposits, withdrawals

Process

A **process** is the series of actions or operations in the components of a system that conducts to the end of the system.

- Example: Payroll process: describes the activities that connect the components of the payroll system, that conducts to the right amount deposited in the employee' account every time period.



Systems, models & Simulation III

Simulation

Simulation is the *imitation* of the behavior of a process, system or phenomena of the real world in time and/or space, *usually under controlled conditions*.

- Simulation is useful to analyze aspects of a system, typically to predict future states of that system.
- Support decisions of the system management: take strategic decisions, organize the components of a system to optimize a process, etc.
  - Should consider an extra teller in a Bank branch/retail store?
  - Should consider one queue or multiple queues?
  - What should be the configuration of the stages of a process?

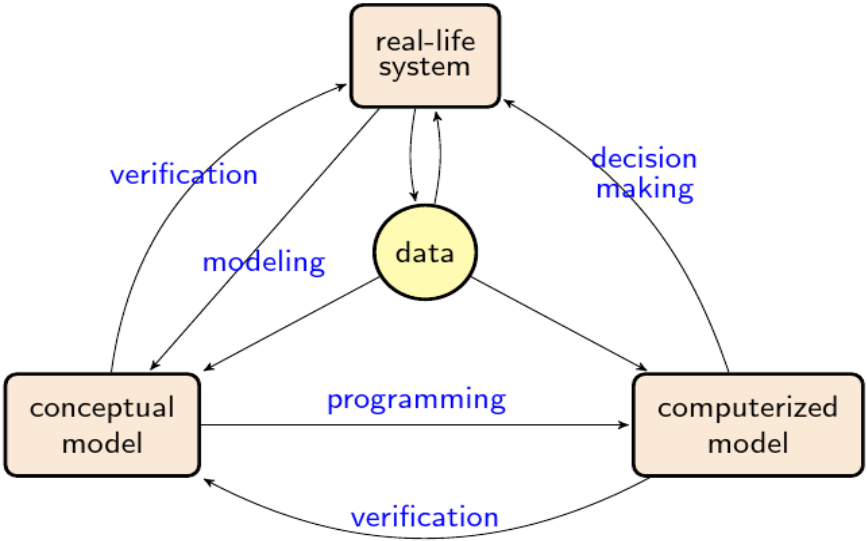
Systems, models & Simulation IV

- Design and building a system. Assess a particular hypothesis, model, or configuration of the system (Movie: *The founder*)

- Understand a concept, gain knowledge of certain domain. For example: Gain understanding of aggression.

Pros and cons of simulation

- Pros
  - Subtitute complex analytical models.
  - A simulation model can be detailed to several levels.
  - Allows to study the behavior of a system.
  - Simulation is cheaper than the savings that generates.
  - A process cannot be understood by stopping it. Understanding must move with the flow of the process, must join it and flow with it (Frank Herbert, *Dune*)
- Cons
  - Sometimes pretend to replace analytical thinking.
  - Applies: “garbish in, garbish out”.
  - Needs the application of the right statistical analysis.
  - Sometimes it is hard to define the adequate level of detail (sometimes too much, sometimes to little).

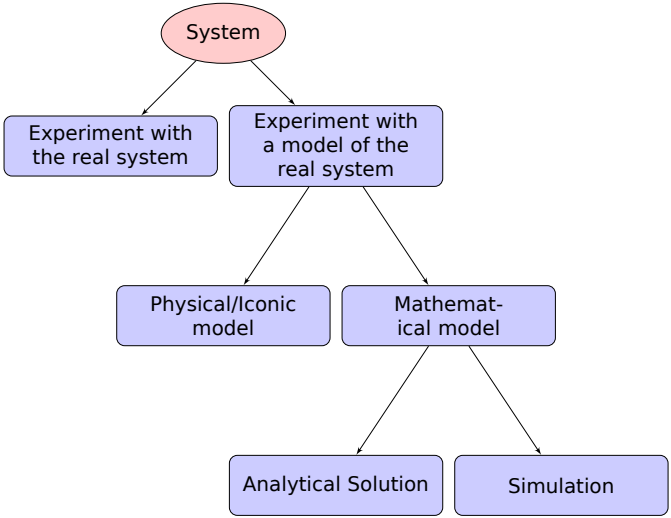

real-life system
verification
modeling
data
decision making
conceptual model
programming
computerized model
verification

Simulation and models

- A simulation model tries to reproduce reality emphasizing some elements of interest. But fidelity increase complexity, so a compromise is required.

> “All models are wrong, but some of them are useful”
>
> — George Box

Simulation and models

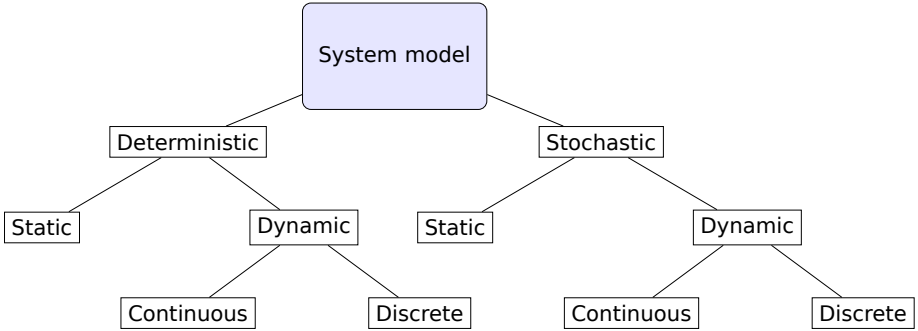


Characteristics of a simulation problem

Accordingly to the characteristics of the model, the simulation technique may be different

- Deterministic or stochastic
  - The model has random components?
  - Deterministic models: Mortgage flow of payments with fixed rate (under a fixed scenario).
  - Stochastic models: waiting line.
- ¿Static or dynamic?
  - Is the time a significative variable?
  - Static models: chance games, Monte Carlo integration.
  - Dynamic models: inventory.
- Discrete or continuous?
  - How the system state changes?
  - Discrete models: inventory, queues.
  - Continuous models: population growth, Prices in the Fx market.

Classification of simulation models



Types of simulation I

There are several taxonomies of simulation. We consider:

- Monte Carlo or stochastic simulation: allows to model uncertainty phenomena. The probabilistic model can be parameterized or adjusted using real data.
- Discrete event simulation: allows to model the operation of a system as a sequence of events on time. Each event occurs at a particular time and defines a change in the state of the system.

Monte Carlo Simulation

Example 1. Area estimation

We have a map with scale 1:N in a paper sheet with area A_{paper}. How we determine the area of the surface of territory?

$$A_T = R \times A_{paper} \times e_{Map} = R \times A_{paper} \times N$$

where

$$\begin{aligned} R &= \text{areas ratio} = \frac{A_T}{A_{paper}} \\ e_{Map} &= \text{map scale} \\ A_{paper} &= \text{paper area (known)} \end{aligned}$$

Two alternatives:

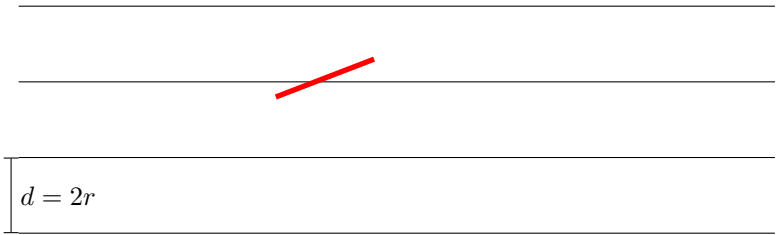
- Destructive solution
  - Cut the map in m small pieces. According to the dominant color of the piece, classify it in terrain or nothing.
  - Mix the pieces
  - Extract a random sample with replacemente and record the number of terrain pieces (m_T).
  - With probability 1, when $m \to \infty$

$$\frac{m_T}{m} \to R = \frac{A_T}{A_{paper}}$$

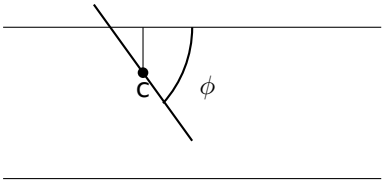
- Non destructive solution?

Example 2. Estimation of π: Buffon needle problem (1733) I

A needle of length r is thrown at random in a wood floor made of parallel stripes with the same width d ($r \leq d$). What is the probability that the needle will lie across a line between two stripes?



Example 2. Estimation of π: Buffon needle problem (1733) II



- Let $c =$ distance from middle of the needle to the edge of the closest stripe.
- For needle of length r, $0 \leq c \leq r/2 = d/4$ and $0 \leq \phi \leq \pi/2$. We can assume that both are uniform, independent random variables.
- $\sin\phi = \frac{c}{r/2} \Rightarrow c = \frac{r}{2}\sin\phi$
- The needle crosses if and only if $c \leq \frac{r}{2}\sin\phi$
- Then

$$\begin{aligned} P[\text{cross}] &= P[0 \leq c \leq r/2, 0 \leq \phi \leq \pi/2] \\ &= \int_0^{\pi/2}\int_0^{\frac{r}{2}\sin\phi} \frac{4}{d\pi} dc d\phi \\ &= \frac{2r}{\pi d} = \frac{1}{\pi} \end{aligned}$$

Example 3. Buffon

Buffon' needle problem can be used as a way to estimate the value of π. See other ways:

- Throw a needle "at random" N times.
- Let $X =$ be the number of crosses. Then $\hat{p}_N = \frac{X}{N}$, therefore $\hat{\pi} = \frac{1}{\hat{p}_N}$.

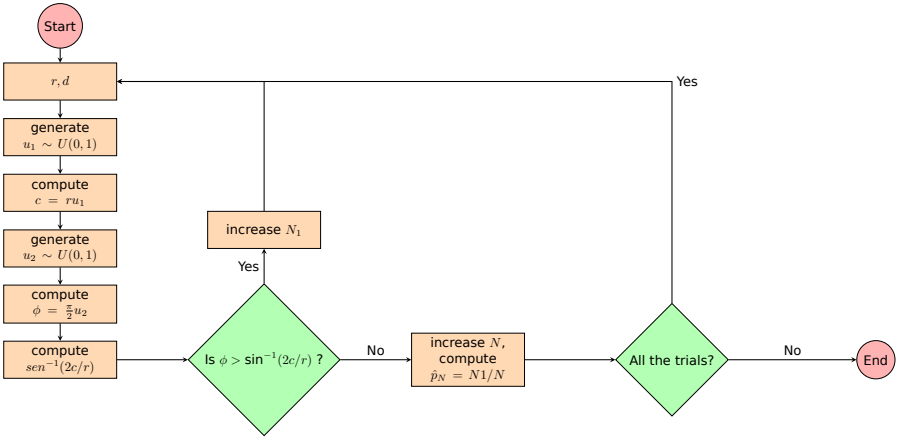
Further, we know that $X|N \sim Bin(N, p)$, so $\text{Var}(\hat{p}_N|N) = \frac{p(1-p)}{N}$. For N large, by the Central Limit Theorem:

$$\sqrt{N}(\hat{p}_N - p) \xrightarrow{N\to\infty} \mathcal{N}(0, p(1-p))$$

An estimate of the variance of $\hat{p}_N$ is $\frac{\hat{p}_N(1-\hat{p}_N)}{N}$ and a confidence interval for p is:

$$\hat{p}_N \pm z_{1-\alpha/2} \left(\frac{\hat{p}_N(1-\hat{p}_N)}{N} \right)^{1/2}$$

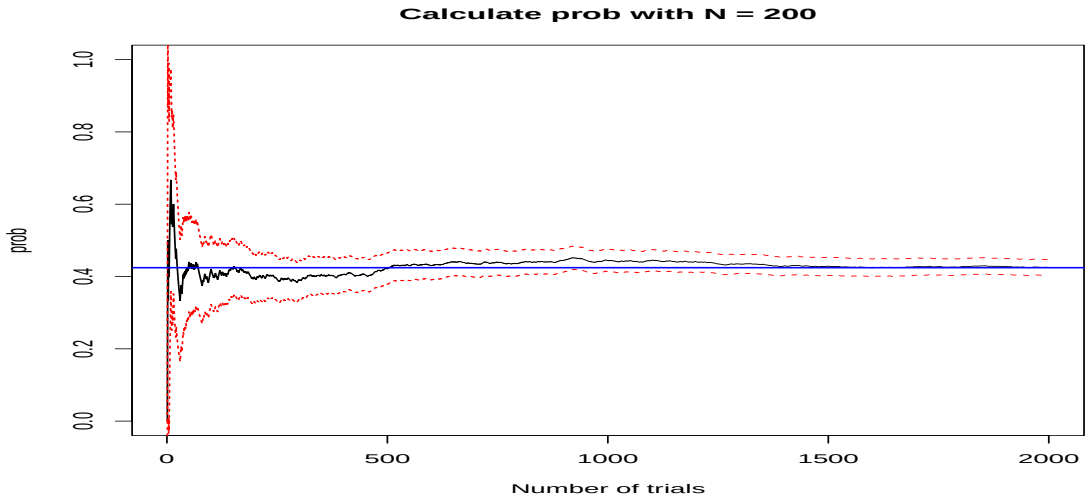
Example 3. Buffon: simulation



Example 3. Buffon: R code

```r
# Buffon's needle

# Parameter definition
N <- 2000             # Num of trials
alfa <- 0.05          # significance level
z <- qnorm(1-alfa/2) # quantile

NI <- rep(1,N)        # indicator for needles that cross
r <- 10               # length of needle
d <- 15                       # width of stripes

# Initiate simulation
u1 <- runif(N, min = 0, max = d/2)
u2 <- runif(N, min = 0, max = pi/ 2)
prob <- ifelse(r/2*sin(u2) >= u1, 1, 0)
prob <- cumsum(prob)/1:N
lim.inf <- prob - z*sqrt(prob*(1-prob)/1:N)
lim.sup <- prob + z*sqrt(prob*(1-prob)/1:N)

# Graph
plot(1:N, prob, type = "l", ylim = c(0,1), xlab = "Number of trials",
main = " Calculate prob with N = 200")
lines(1:N, lim.inf, col = "red", lty = 2)
lines(1:N, lim.sup, col = "red",lty = 2)
abline(h = 2*r/(pi*d), col = "blue", lwd = 2)
text(100,2*r/(pi*d)+1, as.character(2*r/(pi*d)))
```

Ejemplo 3. Aguja de Buffon: simulación



Discrete Event Simulation.

Discrete event simulation I

Discrete event simulation (DES)

A DES model has three attributes:

1. At least some of the system state variables are random
2. Time evolution of the states of the system is recorded
3. Changes in the system states are associated with events that occur at discrete time instances only.

Some Examples:

- Markov chains
  - Queues
  - Inventory models
- Flows in organizational process
  - Health center activity: patients,nurses, doctors
  - Products being manipulated in a supply chain

Discrete event simulation II

A DES model requires:

- A simulation clock
- A list of planned future events, in chronological order.
- A procedure or method to execute each type of event.

Example 1: Waiting line

- Consider one service unit with one server (teller, machine, table, etc.)
- The clients are waiting in line to receive service

We would like to estimate the average time expend in line, measure as the time that passes since arrival to the line to the moment the service starts.
The state variables of the system are:

- Server status (idle, bussy)
- The number of clients waiting in line $\{0, 1, 2, \ldots\}$
- The time of arrival of each client to the queue
- The service time of each client

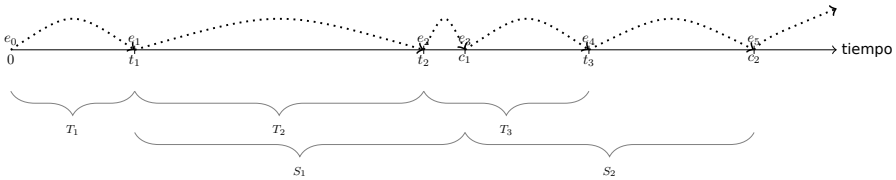
There are two types of relevant events in this system:

- Arrival of a client
- Departure of a client

Example 1: Waiting line

Let:

$t_i =$ time of arrival of the i-th client
$T_i = t_i - t_{i-1} =$ time between arrivals (interarrival) of clients $i-1$ and i
$S_i =$ time of service of client i
$D_i =$ wait in line of client$i$
$c_i = t_i + D_i + S_i =$ time at which client i completes service and departs
$e_i =$ time of occurrence of event i of any type



We will comeback to this model again later.

Using simmer in R I

- Package simmer is a process-oriented and trajectory-based DES package for R.
- simmer exploits the concept of *trajectory*: a common path in the simulation model for entities of the same type.
- It reduces considerably the programming complexity of some models and allows to define some complex models.

Simple example I

- single client who arrives at service unit at a fixed time, spends also a fixied time and leaves. The simulation runs 100 minutes

```
library(simmer)
client <-
        trajectory("Client") %>%
        log_("Hello") %>% # shows text at specific events
        timeout(10) %>%   # time spent in the unit
        log_("Bye")
shop <-
        simmer("shop") %>%
        add_generator("Client", client, at(5))
shop %>% run(until = 100)
```

```
5: Client0: Hello
15: Client0: Bye
simmer environment: shop | now: 15 | next:
{ Monitor: in memory }
{ Source: Client | monitored: 1 | n_generated: 1 }
```

- An overview of the events of the simulation:

```
        shop %>% get_mon_arrivals()
```

```
     name start_time end_time activity_time finished replication
1 Client0          5       15            10     TRUE           1
```

Add randomness to the process I

- Allow multiple arrivals at random and a server:

```
client <- trajectory("Client") %>%
log_("Hello") %>%
seize("counter") %>% # the customer to join the queue at the counter
timeout(function() rnorm(1,10,2)) %>%
release("counter") %>%
log_("Finished")

shop <- simmer("shop") %>%
add_resource("counter") %>%
add_generator("Client", client, function() rexp(1, 1/5))

shop %>% run(until = 30)
```

```
5.71221: Client0: Hello
12.2853: Client1: Hello
12.9215: Client2: Hello
13.8675: Client0: Finished
14.8592: Client3: Hello
22.5608: Client1: Finished
23.534: Client4: Hello
27.1309: Client5: Hello
simmer environment: shop | now: 30 | next: 35.5249008099225
{ Monitor: in memory }
{ Resource: counter | monitored: TRUE | server status: 1(1) | queue status: 3(Inf) }
{ Source: Client | monitored: 1 | n_generated: 7 }
```

Add randomness to the process II

- Monitors of arrivals, Need to compute waiting time. `activity_time` is the service time.

```
shop %>% get_mon_arrivals() %>%
        transform(waiting_time = end_time - start_time - activity_time)
```

```
     name start_time end_time activity_time finished replication waiting_time
1 Client0   5.712207 13.86747      8.155260     TRUE           1     0.000000
2 Client1  12.285332 22.56082      8.693353     TRUE           1     1.582135
```

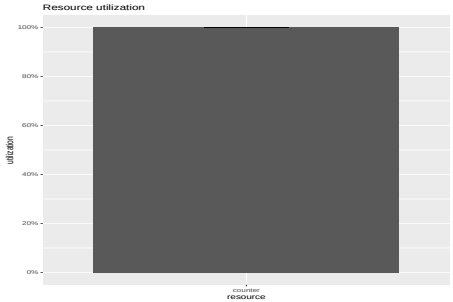
```
shop %>% get_mon_resources() #monitor resources
```

```
  resource      time server queue capacity queue_size system limit replication
1  counter  5.712207      1     0        1        Inf      1   Inf           1
2  counter 12.285332      1     1        1        Inf      2   Inf           1
3  counter 12.921452      1     2        1        Inf      3   Inf           1
4  counter 13.867467      1     1        1        Inf      2   Inf           1
5  counter 14.859203      1     2        1        Inf      3   Inf           1
6  counter 22.560820      1     1        1        Inf      2   Inf           1
7  counter 23.534045      1     2        1        Inf      3   Inf           1
8  counter 27.130878      1     3        1        Inf      4   Inf           1
```

Some visualizations I

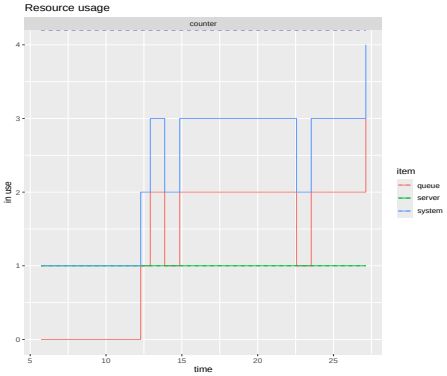
We can visualize different metrics:

```
library("simmer.plot")
resources <- get_mon_resources(shop)
plot(resources, metric = "utilization")
```




```
plot(resources, metric = "usage", steps =T)
```

Some visualizations II



Making replications of a simulation I

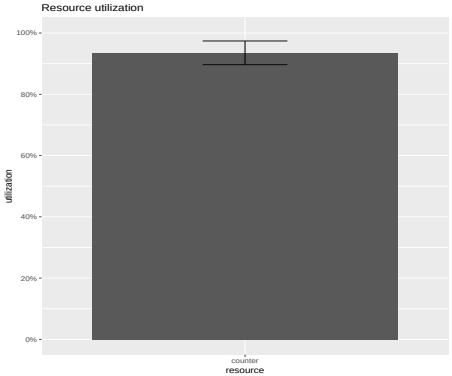
- We can simulate the system many times in order to get several replications and understand the variability of the system:

```
client <- trajectory("client") %>%  # remove text messages
          seize("counter") %>%
          timeout(function() rnorm(1,10,2)) %>%
          release("counter")

envs <- lapply(1:100, function(i){
  simmer("shop") %>%
  add_resource("counter",2) %>%  # two servers
  add_generator("client", client, function() rexp(1,1/5)) %>%
  run(until = 480)  # run 480 minutes = 8 hrs 100 times
  })
```

- Results can be summarized:

```
plot(get_mon_resources(envs), metric = "utilization")
```

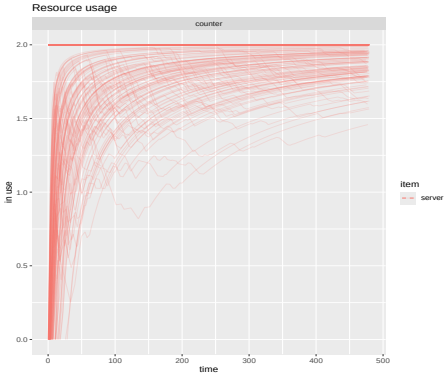
Making replications of a simulation II



- We can assess how busy were the servers in diferent simulations. As the shop opens the two server become more and more busy and at the end of the eight hours they are busy almost all the time

```
plot(get_mon_resources(envs), metric = "usage", items = "server")
```

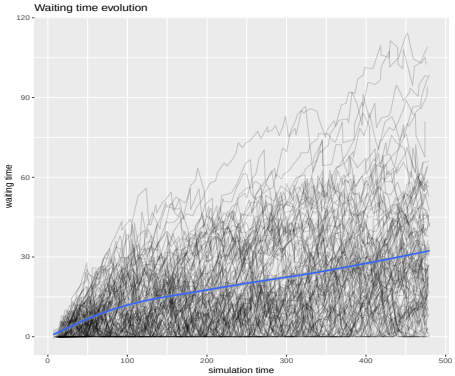
Making replications of a simulation III



Making replications of a simulation IV

- We can look at how long do clients wait in line. Each line is a run:

```
plot(get_mon_arrivals(envs), metric = "waiting_time")
```



Morning Session II

Random number generation

A little of history

- Monte Carlo methods arise mainly during the research in Los Alamos Laboratory during WWII.
- Problems related were associated to difussion of neutrons en fisionable material, as well as criptographic problems, where it was required a trusty source of random numbers.

Von Neumann, Feynman and Ulam

- John Von Neumann and Stanislaw Ulam applied and improve simulation techniques for variance reduction. Further, they propose several methods for random number generation.
- Today, most computer systems include routines or functions to generate successions of random numbers.

Physical generators

- Physical and electronic mechanisms:
  - counters of random events and periodic sampling from electric noise
  - There are about 2,000 patents in 70 years for the generation or physical random numbers.
  - Hotbits: through radioactive decay.
  - IDQ: sources of quantic entropy.
- The most common are based in electric circuits with a source of noise (one resistance or semiconductor diod), that is amplified, sampled and compared with a reference signal to produice sequence of bits, eg, if the noise is less than the reference is 0 and 1 otherwise.
- From sequences of 0's and 1's bytes are generated and translated to numbers.
- Intel Generators (1999-2008): to encrypt credit card information on line.
- Relationship between RNG's and Blockchain/Bitcoin.

Properties of good generators

- Pass statistical tests of independence and randomness.
- Have a strong theoretical support that allows a rigorous analysis
- Reproducible: be able to generate long streams that do not require to be stored in memory.
- Fast and efficient
- To be able to start at different places of a sequence.

Linear Congruential Generators (LCGs)

LCG's: Linear Congruential Generators (Lehmer, 1951)

D. H. Lehmer introduced LCGs in 1948, which are as big roulettes and satisfy, under certain conditions, the requirements.

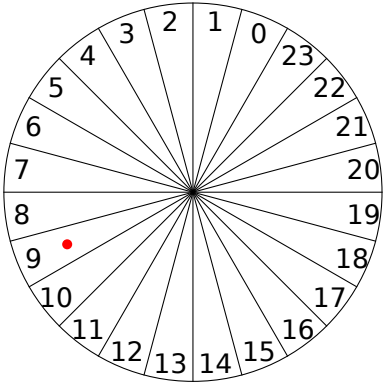
- Four parameters (nonnegative integers):
  - Z_0: a seed
  - m: modulus or divisor
  - a: a multiplier
  - c: an increment
- Generate a sequence of values using:

$$Z_i \equiv (aZ_{i-1} + c) \mod m$$

- Fom that equation, $0 \leq Z_i \leq m - 1$ for all i, the numbers generated are $u_i = Z_i/m$.
- When $c = 0$, it is called a *multiplicative congruential generator (MCG)*.

LCGs

$$m = 24$$
$$Z_i = (3Z_{i-1} + 2) \mod 24$$



$Z_0 = 9$

LCGs

- The sequence generated with LCGs have an explicit formula Z_i in terms of Z_0, a, c y m:

$$Z_i \equiv \left(a^i Z_0 + \frac{c(a^i - 1)}{a - 1} \right) \mod m$$

- The quality of the generator depends on the selection of values a, Z_0, c and m.
- When a value Z_j is repeated for first time, the sequence starts repeating. The length of different values is the cycle or period, which is $\leq m$.
- Usually, m is of order 10^9 or larger.

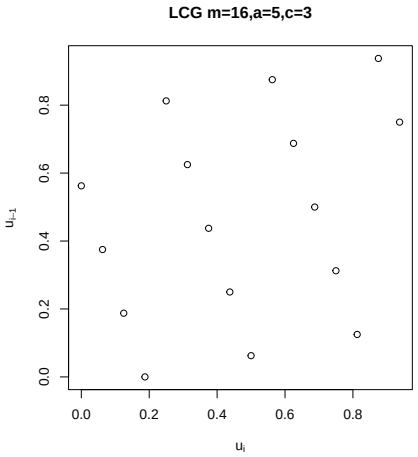
Example: $m = 16$, $a = 5$, $c = 3$, $Z_0 = 7$.

This generator has period of m=16.

$$Z_i \equiv (5Z_{i-1} + 3) \mod 16$$

i	Z_i	U_i
0	7	0.438
1	6	0.375
2	1	0.063
3	8	0.500
4	11	0.688
5	10	0.625
6	5	0.313
7	12	0.750
8	15	0.938
9	14	0.875
10	9	0.563
11	0	0.000
12	3	0.188
13	2	0.125
14	13	0.813
15	4	0.250
16	7	0.438
17	6	0.375
18	1	0.063
19	8	0.500

Ejemplo: $m = 16$, $a = 5$, $c = 3$, $Z_0 = 7$.




```r
lgc  <- function(m=16,a=5,c=3,z0=7){
z <- z0
i <- 1
repeat {
i <- i+1
z[i] <- (a*z[i-1]+c) %% m
if(i>m) break
}
return(z/m)
}

z <- lgc()
par(pty="s")
plot(z[2:17], z[1:16],
xlab = expression(u[i]),
ylab = expression(u[i-1]),
main = "LCG m=16,a=5,c=3")
```

Optimal selection of parameters I

- If the data were a random sample, then the points should be distributed uniformly in the unit square.
- If we ignore the sequential order of an LCG of complete period, the numbers will look like uniform.
- However, the structure of an LCG is very regular and this can affect the serial correlation of the sequence.
- Marsaglia (1968) showed that “random numbers fall mainly in the planes”.
- The following theorem gives the optimal selection of parameters:

Optimal selection of parameters II

Full period Theorem (Hull & Dobell, 1962)

A LCG $Z_i \equiv (aZ_{i-1} + c) \mod m$ has complete period if and only if the following three conditions hold:

1. c and m are relative primes.
2. If q is a prime number that divides m, then q divides $a - 1$
3. If 4 divides m, then 4 divides $a - 1$.

- When $c = 0$, full period cannot be achieved. Therefore, the maximal period of a MCG is $m - 1$.
- The result is not easy to verify for large and relevant values of m, a, c, etc.
- Proof based on number theory (See: SIAM Review **4** (1962) 230-254).

Example Hull & Dobell

- $m = 10^{10}$, $a = 3141592621$, $c = 2718281829$, $Z_0 = 5772156648$
- In the computer, be careful of large integer arithmetic. Some packages in `R` allow to do arithmetic with large numbers:
  - `numbers`: functions from Number theory. in particular `primeFactors()`, `Primes()`

```
library(numbers)
Primes(1,100)
 [1]  2  3  5  7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
primeFactors(2718281829)
[1]       3     157 5771299
```

  - `gmp`: Multiple precision arithmethic. Allows arithmetic with large numbers.

```
suppressMessages(library(gmp))
factorize(3141592621)
Big Integer ('bigz') object of length 3:
[1] 137   239   95947
```

Combined Algoritm of Wichmann & Hill (1982,1984)

Has a period of order 10^{12}.The generator is created combinintg three LCGs:

$$\begin{aligned} x_i &\equiv 171 x_{i-1} \mod 30269 \\ y_i &\equiv 172 y_{i-1} \mod 30307 \\ z_i &\equiv 170 z_{i-1} \mod 30323 \end{aligned}$$

In this case

$$u_i = \left(\frac{x_i}{30269} + \frac{y_i}{30307} + \frac{z_i}{30323} \right) - \left[\frac{x_i}{30269} + \frac{y_i}{30307} + \frac{z_i}{30323} \right]$$

This is the default algorithm in python 2.3.

Random numbers in R

- R has many functions to generate random numbers, based on LCG, as well as other types of algorithms.
- The default generator in Ris based on the Mersenne-Twister algorithm (Matsumoto & Nishimura (1997)), who has a period of $2^{19937} - 1$. The method is based on more sophisticated algorithms. Python 2.4 and Matlab use this algorithm by default too. (see RandStream.list for a list of algoritms available in Matlab).
- To change the RNG in R:

```
>RNGkind("Super") #for Marsaglia's super-duper
```

- In R the library randtoolbox has many other generators and statistical tests (See documentation).

Other methods to generate random numbers

- Multiple Recursive Generators (MRG's):

$$Z_i \equiv (a_1 Z_{i-1} + a_2 Z_{i-2} + \cdots + a_{i-k} Z_{i-k}) \mod m$$

- Lagged Fibonacci generators: choose r, s and m carefully:

$$Z_i \equiv (Z_{i-r} + Z_{i-s}) \mod m$$

- Inverse Congruential Generators (ICG): For m prime and defining $0^{-1} = 0$:

$$Z_i \equiv \begin{cases} (a_0 + a_1 Z_{i-1}^{-1}) \mod m & \text{if } Z_{i-1} \neq 0 \\ a_0 & \text{if } Z_{i-1} = 0 \end{cases}$$

- If m has the form 2^k,are known as *linear feedback shift register* (LFSR)
- KISS (Keep it simple Stupid) (Marsaglia & Zaman, 1993) consists in a combination of 4 subgenerators each one of 32 bits to complete a period of 2^{123}:
  - a LCG mod 2^{32}
  - a generator linear binary
  - Two multiply with carry mod 2^{16}

  NO criptographic quality and not recommended to encript data.
- The mother of all generators (Marsaglia)

Random Variable Generation

Method of Inverse Transformation

Theorem of Inverse Transformation

If $X \sim F$ and F is *continuous and strictly increasing*, then $Y = F(X) \sim \mathcal{U}(0,1)$.

Proof.

Let G the distribution of Y. Then if $y \in (0,1)$,

$$\begin{aligned} G(y) &= P(Y \leq y) = P(F(X) \leq y) \\ &= P(X \leq F^{-1}(y)) \\ &= F(F^{-1}(y)) = y \end{aligned}$$

Then $G(y) = y$ in $(0,1)$. This is the uniform distribution $\square$

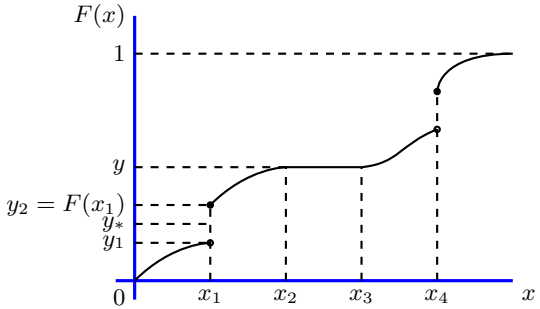
Quantile Function Theorem I

Generalization of Inverse Transformation theorem

- If $X \sim F$ with F discontinuous or non strictly increasing, the previous theorem works, but the proof is a little more elaborated.
- We need a more ample definition of an inverse function. If F is strictly increasing the inverse is well defined if

$$F^{-1}(y) = x \iff F(x) = y.$$

However, if F is constant in some interval (like in (x_2, x_3) in the figure), then the condition doesn't work. Then we need to define a generalized inverse



Quantile Function

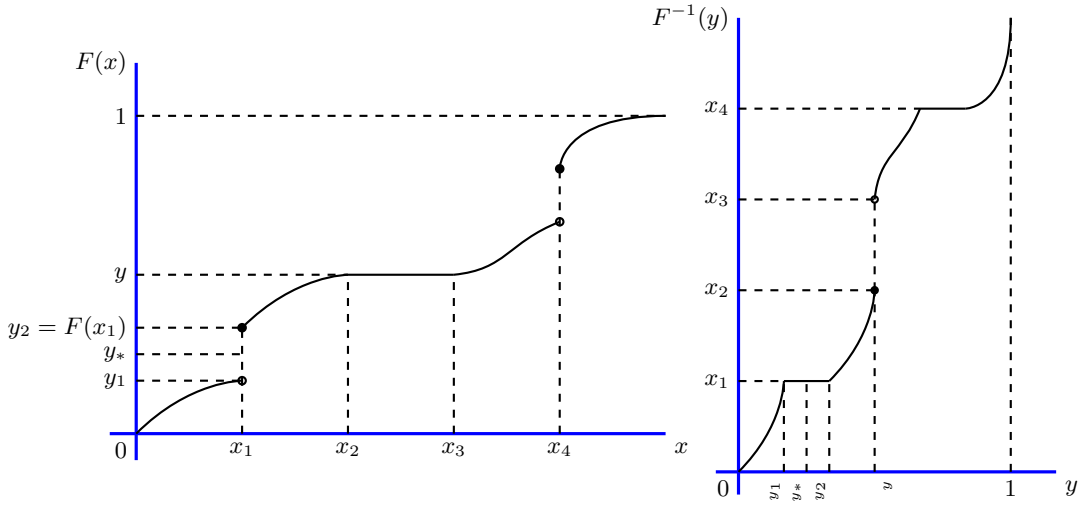
The generalized inverse of F (or quantile function) is defined as

$$F^{-1}(y) = \inf\{x|F(x) \geq y\}.$$

- Then, in the previous figure observation y is well definedto be $F^{-1}(y) = x_2$.
- For any $y_* \in [y_1, y_2]$, $F^{-1}(y_*) = \inf\{x|F(x) \geq y_*\} = x_1$, since F is right-continuous and $F(y_*)$ is greater than any y_* in that interval.
- From the figure, it is important to note the following characteristics
  - F^{-1} is no decreasing.
  - Jumps F translate to flat segments in F^{-1} and vice-versa.
  - F is right continuous, while F^{-1} is left-continuous.

Quantile Function Theorem III

Generalization of Inverse Transformation theorem



Quantile Function Theorem IV

Generalization of Inverse Transformation theorem

Quantile Function Theorem

Let $F$the distribution of a random variable X. Define F^{-1} as the quantile function $F^{-1}(y) = \inf\{x|F(x) \geq y\}, \quad 0 \leq y \leq 1$, and $U \sim \mathcal{U}(0,1)$, then the random variable $X = F^{-1}(U)$ has distribution F.

Application of the theorems

We can define an algorithm in two cases to obtain $X \sim F$ from a sample of uniform random numbers

Inversion Algorithm

- Continuous case:
  1. Generate $u \sim \mathcal{U}(0,1)$
  2. Obtain $X \sim F^{-1}(u)$
- Discrete case: If $x \in \{x_1, \ldots, x_n\}$, the set of ordered points of discontinuity of F (n can be ∞), then the quantile function is $F^{-1}(u) = x_{(i)}$ where $F(x_{(i-1)}) < u \leq F(x_{(i)})$.
  1. Generate $u \sim \mathcal{U}(0,1)$
  2. Take $x = x_{(i)}$, where $F(x_{(i-1)}) < u \leq F(x_{(i)})$.

See Luc Devroyé's book to implement algorithm for complicated cases.

Examples

a. $X \sim exp(\beta)$. Then X has density $f(x) = \frac{1}{\beta} e^{-x/\beta} I(x > 0)$, $\beta > 0$. the distribution is:

$$F(y) = \int_0^y \frac{1}{\beta} e^{-x/\beta}\, dx = \int_0^{y/\beta} e^{-z}\, dz = 1 - e^{-y/\beta}$$

Then $X = F^{-1}(U) = -\beta \log(1-U) = -\beta \log(U)$. We can simplify substituting $1-U$ by U since $U \sim \mathcal{U}(0,1) \Rightarrow 1 - U \sim \mathcal{U}(0,1)$.

Examples I

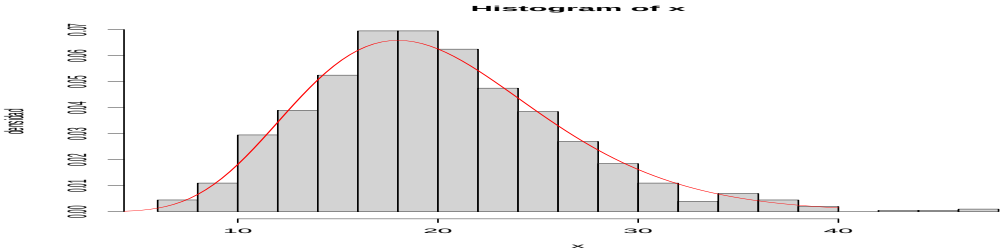
b. If $X \sim \mathcal{G}(\alpha, \beta)$, then X has density $f(x) = \frac{1}{\Gamma(\alpha)\beta^{\alpha}} x^{\alpha-1} e^{-x/\beta}$.
If $Y_i \sim exp(\beta)$ and $Y_1, \ldots, Y_n$ are independent, then $X = \sum_{i=1}^{n} Y_i \sim \mathcal{G}(n, \beta)$. This is the *Erlang distribution*.
Then, generate

$$X = -\beta \sum_{i=1}^{n} \log(1 - u_i) = -\beta \log(\prod_{i=1}^{n} u_i)$$

To obtain a sample from $\mathcal{G}(10, 2)$, we can use the following code. In R it can be defined β as a scale parameter (`scale`) or rate (`rate` = 1/`scale`).

```
x <- NULL; n <- 1000; alfa <-10; beta <- 2
for(i in 1:n) x[i] <- -2*sum(log(runif(alfa)))
hist(x, prob = T, breaks=20, ylab = "densidad")
curve(dgamma(x, shape=10, scale = beta), from = 0, to = 40, add = T, col = "red")
```

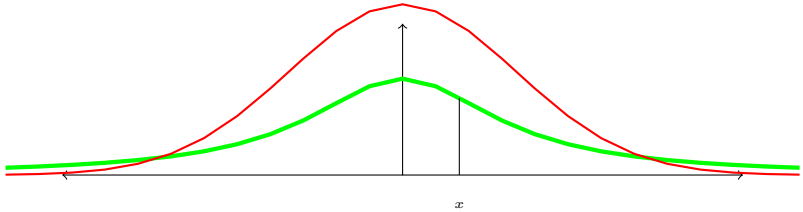
Examples II



c. Consider $\chi^2_{(n)} \equiv \mathcal{G}\,(n/2, 2)$.

Examples

d. **Cauchy distribution**. $X \sim Cauchy(0,1)$ has density $f(x) = \frac{1}{\pi(1+x^2)}$.



Then the distribution function has the form:

$$F(x) = \int_{-\infty}^{x} \frac{1}{\pi(1+v^2)} dv = \frac{1}{2} + \frac{1}{\pi} \int_{0}^{x} \frac{1}{1+v^2} dv = \frac{1}{2} + \frac{1}{\pi} tan^{-1}(x)$$

Take $X = F^{-1}(u) = \tan(\pi(u - \frac{1}{2}))$.

Examples I

e. Assume X has a discrete probability mass function (pmf) given by:

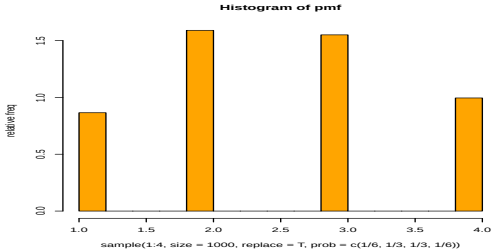
X	1	2	3	4
$P(X = x)$	$1/6$	$1/3$	$1/3$	$1/6$

Then generate $u \sim \mathcal{U}(0, 1)$ if

$$u \in \begin{cases} [0, 1/6] \Rightarrow x = 1 \\ (1/6, 1/2] \Rightarrow x = 2 \\ (1/2, 5/6] \Rightarrow x = 3 \\ (5/6, 1] \Rightarrow x = 4 \end{cases}$$

```
hist(sample(1:4, size = 1000, replace = T,
prob = c(1/6,1/3,1/3,1/6)), col = 'orange',
main = "Histogram of pmf",
ylab = "relative freq", prob=T)
```

Examples II

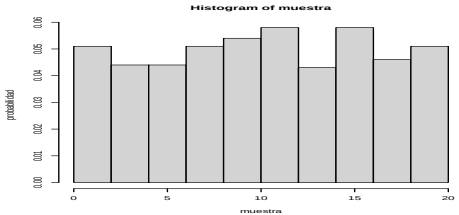

Histogram of pmf
relative freq
0.0
0.5
1.0
1.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
sample(1:4, size = 1000, replace = T, prob = c(1/6, 1/3, 1/3, 1/6))

Examples I

f. For a sample of a uniform discrete in $\{1, \ldots, n\}$ with $P(Y = k) = \frac{1}{n}$, generate $U \sim \mathcal{U}(0, 1)$ and define $Y = [Un] + 1$, where $f(x) = [x]$ is the integer part function. In this case we also can use `R` function `sample` to generate the sample (can generate with or without replacement).

```
n <- 20
muestra <- sample(1:n,1000,replace=T)
head(muestra)

[1]  6  5 14 16 12  6

hist(muestra, prob = T, ylab = "probabilidad")
```

Examples II


Histogram of muestra
probabilidad
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
5
10
15
20
muestra

Examples

g. Geometric distribution

- The pmf of geometric is $f(x) = p(1-p)^x$, $x = 0, 1, 2, \ldots$.
- The cdf is $F(x) = 1 - q^{x+1}$ where $q = 1 - p$. Then, with the inverse transform method we need to solve for x:

$$1 - q^x < u \leq 1 - q^{x+1}$$

- The inequality simplifies to $x < \log(1-u)/\log(q) \leq x + 1$, with solution $x + 1 = \lceil \frac{\log(1-u)}{\log(q)} \rceil$[1]. Fos example, for $p = 1/4$:

```
n <- 1000
p <- 0.25
u <- runif(n)
k <- ceiling(log(1-u)/log(1-p))-1 #se puede simplificar a floor(log(u)/log(1-p))
l <- floor(log(1-u)/log(1-p))
k[1:20]
 [1]  0  7  5  1  2  0 12 12  4  1  6  0  2  4  0  0  1  1 15  7
l[1:20]
 [1]  0  7  5  1  2  0 12 12  4  1  6  0  2  4  0  0  1  1 15  7
```

[1] $\lceil t \rceil$ is the function with returns the least integer that is greater or equal to t). The function $\lfloor t \rfloor$ return the greatest integer which is least than or equal to t. Then, $\lceil 3.5 \rceil = 4$ y $\lfloor 3.5 \rfloor = 3$.

Examples I

h. Poisson distribution.

- The pmf of Poisson is of the form $f(x) = e^{-\lambda}\frac{\lambda^x}{x!}, \quad x = 0, 1, 2, 3, \ldots$
- For the Poisson cdf, we don't have an explicit formula for x such that $F(x) < u \leq F(x+1)$, since F is a series.
- We can use recursive formulas:

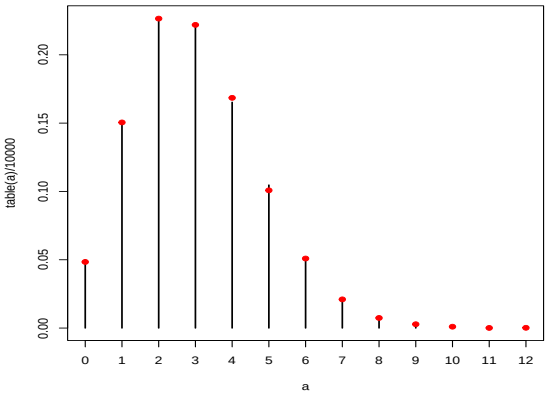
$$f(x+1) = \frac{\lambda}{x+1} f(x), \qquad F(x+1) = F(x) + f(x+1)$$

- Then, generate $u \sim \mathcal{U}(0,1)$ and search in a table or vector with the values of $F(x)$ to find the value of x which is solution of $F(x-1) < u \leq F(x)$ (Search method).

Examples II

```r
a <- dPoisson(lambda = 3, n = 10000)
b <- rpois(10000, lambda = 3)
plot(table(a)/10000)
points(sort(unique(b)), table(b)/10000, col ="red", pch = 19)
```

```r
dPoisson <- function(lambda,n){
# recursive function to generate samples from P(lambda)
# Generate a vector with the values

f <- Fn <- NULL
f[1] <- exp(-lambda) #densidad en 0
Fn[1] <- f[1] #distribución en 0
# Generate vector
for (i in 2:(n+1)){
f[i] <- lambda*f[i-1]/(i-1)
Fn[i] <- Fn[i-1] + f[i]
}
u <- runif(n) # muestra de uniformes
x <- NULL
for(i in 1:n)x <- append(x, sum(Fn < u[i]))
return(x)
}
```



Composition or mixture method I

- Sometimes f or F can be expressed as a convex linear combination of two or more cdf's:

$$F = \sum_{i=1}^{n} p_i F_i, \qquad p_i > 0, \quad \sum_{i=1}^{n} p_i = 1$$

- Mixtures are common in financial applications, marketing applications and in classification problems.

Composition sampling

Sampling method:

1. Sample a value j such that $P(J = j) = p_j$.
2. Obtain X with distribution F_j.

Composition or mixture method II

Proof.

X has distribution:

$$F(x) = P(X \leq x) = \sum_{i=1}^{n} P(X \leq x | J = j) P(J = j) = \sum_{i=1}^{n} F_j(x) p_j$$

- A random variable X is a continuous mixture if the cdf of X is of the form:

$$F_X(x) = \int_{-\infty}^{\infty} F_{X|Y=y}(x) f_y(y)\, dy$$

for a family $X|Y = y$ indexed by the reals y and weigth function $f_Y(y)$ which is a probability density.

Examples

a. Double exponential. the pdf is $f(x) = \frac{1}{2}e^{-|x|}$. f can be written as

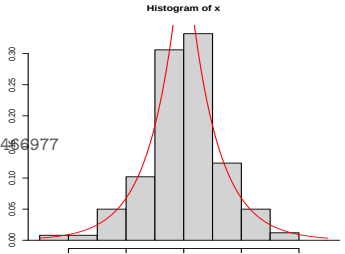
$$f(x) = \frac{1}{2}e^{x}I(x \leq 0) + \frac{1}{2}e^{-x}I(x > 0)$$

Then the algorithm is:

- Generate $u_1, u_2 \sim \mathcal{U}(0, 1)$
- If $u_1 \leq \frac{1}{2}$ then $X = log(u_2)$, otherwise $X = -log(u_2)$.

```
hist(x, xlim = c(-5,5), prob = T, breaks = 20)
curve(0.5*exp(ifelse(x<0,1,-1)*x), from = -5, to = 5, add = T
```

```
u <- matrix(runif(1000), ncol = 2)
x <- ifelse(u[,1] < 0.5, -1, 1)*log(u[,2])
head(x,5)

[1]  0.104767023 -0.592735391  0.251244540  0.004640096  0.948466977
```



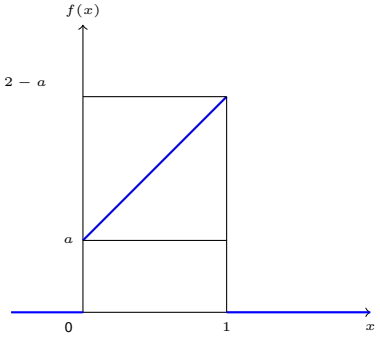
Examples

b. Triangular or trapezoidal.

$f(x) = [a + 2(1-a)x]I_{(0,1)}(x)$ for $0 < a < 1$. f can be rewritten as

$$f(x) = aI_{(0,1)}(x) + (1-a)2xI_{(0,1)}(x) = af_1(x) + (1-a)f_2(x)$$

where $f_1(x)$ is uniform, $f_2(x)$ is the pdf of $\sqrt{U}$ o $X = \max\{U_1, U_2\}$.

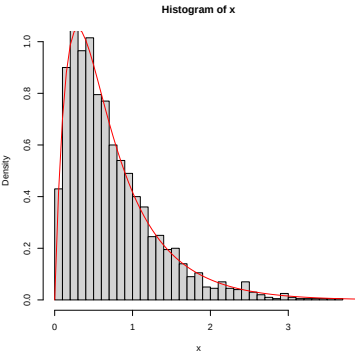


Examples

c. Mixture of gamma distributions. Let $X_1 \sim \mathcal{G}(2,2)$ and $X_2 \sim \mathcal{G}(2,4)$ independent. Obtain a sample from the mixture $F = 0.5F_{X_1} + 0.5F_{X_2}$.

```
n <- 2000
x1 <- rgamma(n,2,2)
x2 <- rgamma(n,2,4)
u <- runif(n)
k <- as.integer(u > 0.5)
x <- k*x1 + (1-k)*x2
```

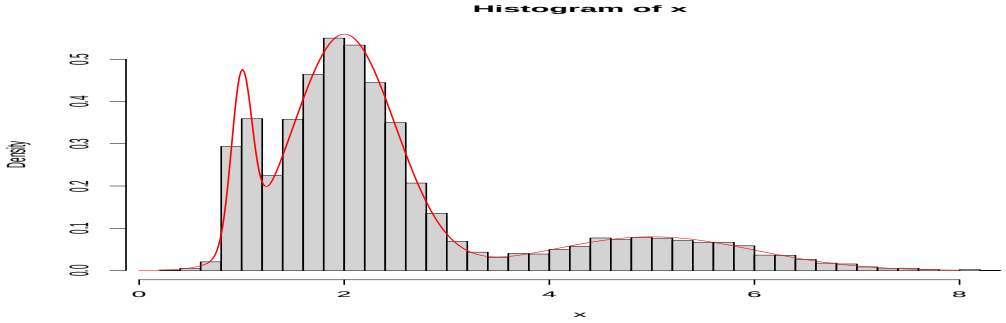
```
hist(x, prob = T, ylim = c(0,1), breaks = 30)
curve(0.5*dgamma(x,2,2)+0.5*dgamma(x,2,4),
      from = 0, to = 5, add = T, col = "red")
```



Examples

d. Normal mixture. Let $X_1 \sim \mathcal{N}\left(1, 0.1^2\right)$, $X_2 \sim \mathcal{N}\left(2, 0.5^2\right)$ and $X_3 \sim \mathcal{N}\left(5, 1\right)$ independent. Obtain a sample of size $n = 10,000$ from the mixture $F = 0.1F_{X_1} + 0.7F_{X_2} + 0.2F_{X_3}$.

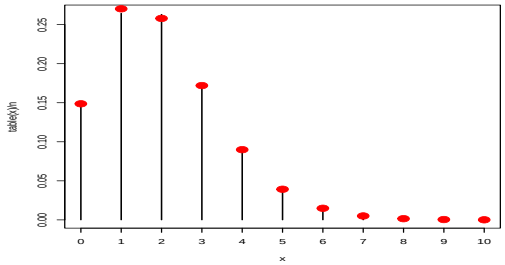
```
n <- 10000
y <- sample(1:3, size = n, prob = c(0.1,0.7,0.2), replace = T)
x <- rnorm(n, mean = ifelse(y == 1, 1, ifelse(y == 2, 2, 5)),
sd = ifelse(y == 1, 0.01, ifelse(y == 2, 0.5, 1)))
hist(x, prob = T, breaks = 50)
curve(0.1*dnorm(x, 1, sd = 0.1) + 0.7*dnorm(x,2, sd = 0.5) + 0.2*dnorm(x, 5, sd = 1),
from = -0, to = 8, add = T, col = "red", n = 500)
```



Examples

e. Poisson-Gamma mixture. in this case, weights are from the gamma distribution and the mixture is continuous: The negative binomial distribution is a mixture of distributions $x|\lambda \sim \mathcal{P}(\lambda)$, with $\lambda \sim \mathcal{G}(k, \beta)$. Then $X \sim Nbin(k, p)$ where $p = \frac{\beta}{1+\beta}$.

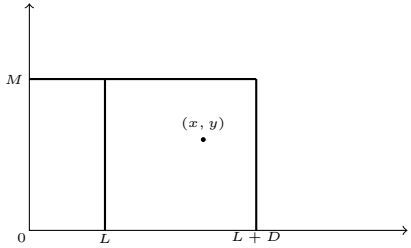
```
n <- 10000; k <- 20; beta <- 10
lambda <- rgamma(n, k, beta) # obtain lambda
x <- rpois(n, lambda) # genera una poisson
plot(table(x)/n)
points(unique(x), dnbinom(unique(x), k, beta/(1+beta)), col = "red", cex = 2, pch = 19)
```



Acceptance-Reject (AR) method (Von Neumman, 1951)

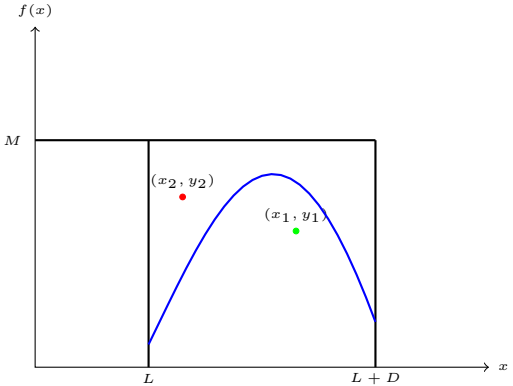
- Assume u_1, u_2 are iid $\mathcal{U}(0,1)$.
- Define: $X = L + Du_1$, $Y = Mu_2$.
- Then (X, Y) is uniform on $[L, L+D] \times [0, M]$:

$$
\begin{aligned}
P(X \leq x, Y \leq y) &= P(X \leq x)P(Y \leq y) \\
&= P(L + DU_1 \leq x)P(MU_2 \leq y) \\
&= P(U_1 \leq \frac{x-L}{D})P(U_2 \leq \frac{y}{M}) \\
&= \frac{x-L}{D} \cdot \frac{y}{M}
\end{aligned}
$$



AR method

- For a random variable with support $[L, L+D]$ and bounded density by M
  1. Obtain (X, Y) as above
  2. If $Y \leq f(X)$, use X.
  3. If $Y > f(X)$, reject (X, Y) and repeat 1.



Notes

- Efficiency depends on:
  (a) the ease of comparison
  (b) the rejection proportion, that is:

$$P(\text{reject}) = 1 - \frac{\text{area of } f(x)}{\text{area rectangle}}$$

- Then, while more tight is the rectangle to $f(x)$, better efficiency.

Improved AR method

- Suppose that the objective density is to sample f.
- instead of a rectangle $[L, L+D] \times [0, M]$, use as "envelope" a density g from which is easy to sample.
- We can find a constant c such that $cg(x) \geq f(x) \quad \forall x$. In this case we said that g *majorize* f.
- Then the improved algorithm:

Improved AR Algorithm

1. Sample $Y \sim g$,
2. Sample $u \sim \mathcal{U}(0,1)$.
3. Accept $X = Y$ if $ucg(x) \leq f(x) \iff u \leq \frac{f(x)}{cg(x)}$, otherwise, reject Y and repeat 1.

- The acceptance probability is $1/c$. The iterations for acceptance is a rv $W \sim geo(1/c)$. Therefore the average number of iterations is $c = E(W)$.

Examples

a. Generate a sample of $x \sim f(x) = 20x(1-x)^3 I_{(0,1)}(x) \equiv \mathcal{B}e\,(2,4)$. Use as majorizing function a rectangle $g(x) \equiv \mathcal{U}\,(0,1)$. To choose a minimum constant c such that $\frac{f(x)}{g(x)} \leq c$, use calculus.

$$\frac{f(x)}{g(x)} = 20x(1-x)^3 = h(x)$$

$h'(x) = 20(1-x)^3 - 60x(1-x)^2 = 0 \Rightarrow x = 1/4$. Then $c = \frac{20}{4}(1-1/4)^3 = 2.109375$ and $\frac{f(x)}{g(x)} \leq 2.109375$. Algorithm is:

- Generate $u_1, u_2 \sim \mathcal{U}\,(0,1)$.
- If $u_2 \leq 9.48u_1(1-u_1)^3 \Rightarrow x = u_1$, otherwise, reject

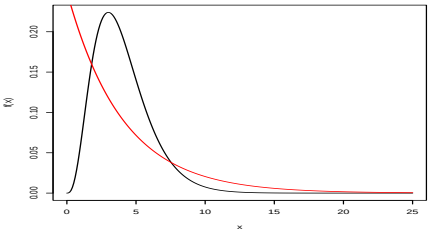
The number of times we repeat step 1 before acceptance is $c = 135/64 \approx 2.11$.

Examples I

b. Exponential envelope for Gamma

- Sample from pdf $f(x) \sim \mathcal{G}(\alpha, 1)$, using an exponential as majorizing function.
- Using $\exp(\alpha)$ with pdf $g(y) = \frac{1}{\alpha} e^{-y/\alpha}$. We find c that optimizes g to cover f.

```
alfa <- 4
f <- function(x){dgamma(x,alfa,1)}
g <- function(x){dexp(x,1/alfa)}
x <- seq(0,25,0.1) # succesion of x values
plot(x,f(x),type="l",lwd=2)
lines(x,g(x),col="red",lwd=2)
```



Examples II

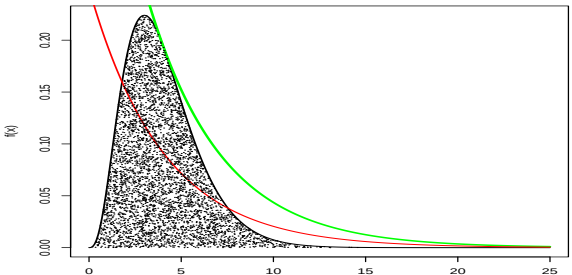
- Note that $h(x) = \frac{f(x)}{g(x)} = \alpha\Gamma(\alpha) \times \frac{x^{\alpha-1}e^{-x}}{e^{-x/\alpha}} = \alpha\Gamma(\alpha)x^{\alpha-1}e^{\frac{1-\alpha}{\alpha}x}$. We optimize $h(x)$. using `optimize` in R.

```
v <- optimize(f = function(x){f(x)/g(x)}, interval = c(0,25),
maximum = T)
v
$maximum
[1] 3.999982

$objective
[1] 2.124248
c <- v$objective
```

- The following graph shows the points that in the relevant area, and the proportion of generated points´.

Examples III

```r
n <- 10000 # generated points
plot(x,f(x), type = "l",lwd = 2)
lines(x, c*g(x), col="green", lwd = 3) # exponential majorized
lines(x, g(x), col = "red", lwd = 2)       # exponential original
u1 <- runif(n) # u1, u2 independent.
u2 <- runif(n)
y <- -alfa*log(1-u2)
ind <- u1 <= f(y)/(c*g(y)) # select accepted points
w <- y[ind]
points(w, u1[ind]*c*g(w), pch = 16, cex = 0.3)
```




```r
table(ind)/n
```

```
ind
 FALSE   TRUE
0.5259 0.4741
```

```r
1/c # accepted points
```

```
[1] 0.4707548
```

Method of convolutions

- The distribution of a sum of two or more random variables $S = X_1 + \cdots + X_n$ is named *convolution* from the original random variables and is written as:

$$f_1 * f_2 * \cdots * f_n$$

- The convolution method consists of summing random variables to obtain a new random variable.
- What is important then is not the pdf of the target variable, but the relationship it has with other variables that are easier to generate.

Examples I

1. Erlang random variables. Is a paticular case of $Gamma(\alpha, \beta)$, where α is a positive integer. Sum of exponentials is Erlang:

$$\sum_{i=1}^{n} Y_i \sim \mathcal{G}(n, \beta)$$

   where $Y_i \sim exp(\beta)$.
2. Binomial. We know that sum of n Bernoullis with parameter p independents, are binomial with parameters n and p.
3. If $Y_i \sim \chi^2_{(1)}$ y $Y_1, \ldots, Y_n$ iid, then $\sum_{i=1}^{n} Y_i \sim \chi^2_{(n)}$.
4. $X_1, \ldots, X_n \sim \mathcal{G}(\alpha_i, \beta)$ independents $\Rightarrow \sum_{i=1}^{n} X_i \sim \mathcal{G}(\sum_{i=1}^{n} \alpha_i, \beta)$
5. Sum of normals is normal.
6. Negative Binomial NegBin(k,p) is the convolution of k iid geometric with parameter p.

Truncated distributions

- If F is a cdf with pdf f, the truncated density on (a, b) is given by:

$$f^*(x) = \frac{f(x)}{F(b) - F(a)} I_{(a,b)}(x)$$

with truncated cdf

$$F^*(x) = \frac{F(x) - F(a)}{F(b) - F(a)} I_{(a,b)}(x) + I_{\{x \geq b\}}(x)$$

To generate a sample from F^*, we have to move the sample of uniforms to the interval of interest:

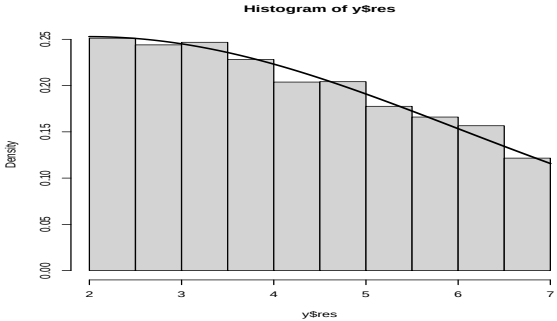
1. Generate $u \sim \mathcal{U}(0, 1)$.
2. Define $\nu = F(a) + (F(b) - F(a))u$. Then $\nu \sim \mathcal{U}(F(a), F(b))$.
3. Define $x = F^{-1}(\nu)$.

Example I

- Want a sample from a $X \sim \mathcal{N}\left(\mu, \sigma^2\right)$ truncated on (a, b).

```
mtrunc <- function(mu, sigma, a, b, n){
        # Obtain a sample from normal from inverse theorem
        u <- runif(n)
        Fa <- pnorm(a, mean = mu, sd = sigma)
        Fb <- pnorm(b, mean = mu, sd = sigma)
        v <-  Fa + (Fb - Fa)*u
        return(list(res=qnorm(v, mean = mu, sd = sigma), Fa = Fa, Fb=Fb))
}

# example
mu <- 2; sigma <- 4; a<- 2; b <- 7; n <- 10000
y <- mtrunc(mu = mu, sigma = sigma, a = a, b = b, n = n)
x <- seq(a, b, length.out = n)
hist(y$res, prob = T)
lines(x, dnorm(x, mean = mu, sd = sigma)/(y$Fb-y$Fa), lwd = 3)
```

Example II


Histogram of y$res
Density
0.00
0.05
0.10
0.15
0.20
0.25
2
3
4
5
6
7
y$res

Order statistics generation I

To generate order statistics $X_{(i)}$ from a cdf F, we use properties from the beta distribution:

Algorithm to generate order statistics

Let $X_1, \ldots, X_n \sim F$ independent.

- If n is small, define $Y_i = X_{(i)}$ (order the sample directly).
- If n is large (say, $n \geq 50$),
  1. generate $\nu \sim \mathcal{B}e\,(i, n-i+1)$
  2. define $X = F^{-1}(\nu)$

If $X \sim F$ then the $i-$th order statistic has cdf:

$$F^*(x_{(i)}) = \sum_{k=i}^{n} \binom{n}{k} F(x_{(i)})^k [1 - F(x_{(i)})]^{n-k}$$

Normal distribution

- For X normal with mean μ and variance σ^2 the cdf is:

$$\Phi_{(\mu,\sigma^2)}(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt$$

- A standard normal $Z \sim \mathcal{N}(0,1)$, is easy to transform: $X = \mu + \sigma Z$. So we can concentrate on standard normals.

Methods based in the CLT

A historic method to generate normal random variables uses the Central Limit Theorem (CLT). Let $U_1, U_2, \ldots, U_k$ a random sample from uniform on (0,1).Then, $E(U_i) = 1/2$ and Var$(U_i) = 1/12$. Because of the CLT:

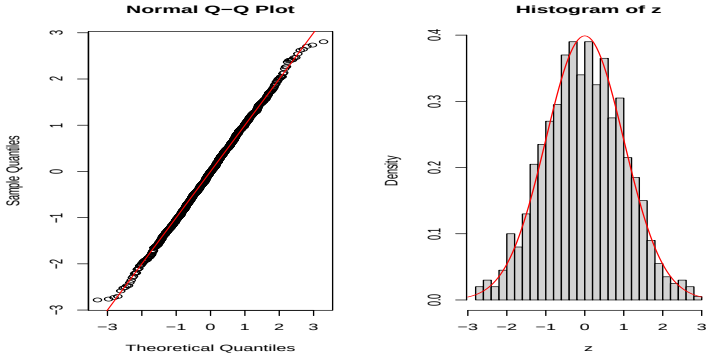
$$Z = \frac{\sum_{i=1}^{k} U_j - k/2}{\sqrt{k/12}} \overset{k \to \infty}{\sim} \mathcal{N}(0,1)$$

If $k = 12$ the equation simplifies. Is this a good method? Depends on k. And always is just an approximation.

Example

Generate a sample of $n = 1000$ standard normals.

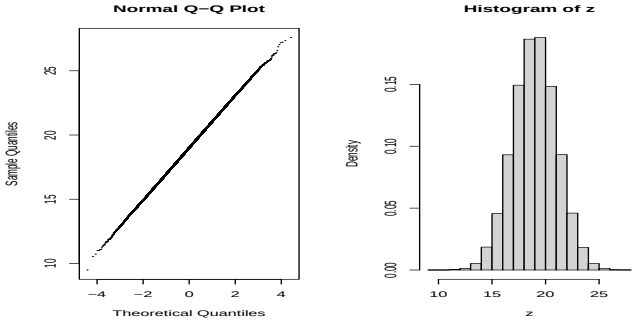
```
n <- 1000; k <-12;
u <- runif(k*n)  # we requiere 12 obs for each sample
z <- NULL
for(i in 1:n) z[i] <- (sum(u[k*(i-1) + 1:k])-6)
par(mfrow = c(1,2))
qqnorm(z); abline(a = 0, b = 1, col = "red")
hist(z, breaks = 20, prob = T); curve(dnorm(x),from=-3,to=3,col="red", add=T)
```



Ejemplo (cont.)

If we repeat the exercise with k larger, the fit improves, but also the time increases.

```
system.time(genera.normal(n=100000,k=12))

   user  system elapsed
  0.086   0.001   0.086

system.time(genera.normal(n=100000,k=50, graf=T))
```

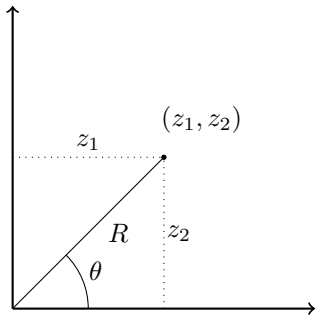



```
   user  system elapsed
  0.319   0.016   0.335
```

Method of Box-Müller (1958)

The method is based on a transformation from polar to euclidean coordinates:

$$(u_1, u_2) \rightarrow (z_1, z_2)$$



Supose $z_1 \perp\!\!\!\perp z_2$. Then

- $z_1^2 + z_2^2 \sim \chi^2_{(2)} \equiv \mathcal{G}\,(2/2, 2) \equiv \exp(2)$
- Distance $d((z_1, z_2), (0, 0)) = ||(z_1, z_2)|| = \sqrt{z_1^2 + z_2^2} = R$
- $\theta \sim \mathcal{U}\,(0, 2\pi)$ for symmetry of normal contours.

In terms of z_1 y z_2:

- $\sin(\theta) = z_2/R$ y
- $\cos(\theta) = z_1/R$ Entonces
- $\tan\theta = z_2/z_1 \implies \theta = tan^{-1}(z_2/z_1)$.

Using u_1 to generate R and u_2 to generate θ, $R = \sqrt{-2\log u_1}$ (since $R^2 \sim \exp(2)$) y $\theta = 2\pi u_2$. Therefore

Box-Müller Transformation

If $U_1 \perp\!\!\!\perp U_2 \sim \mathcal{U}\,(0, 1)$

$$\begin{aligned} Z_1 &= \sqrt{-2\log u_1}\,cos(2\pi u_2) \\ Z_2 &= \sqrt{-2\log u_1}\,sen(2\pi u_2) \end{aligned}$$

Then $Z_1 \perp\!\!\!\perp Z_2 \sim \mathcal{N}\,(0, 1)$.

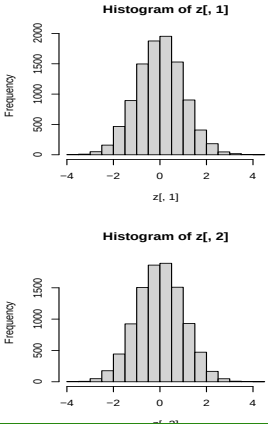
Example

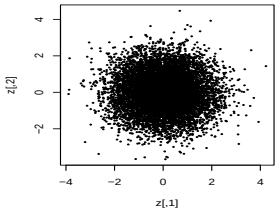
In the following graphs we show the joint and marginals

```
layout(matrix(c(1,3,2,3),nrow=2,byrow=T))
hist(z[,1], breaks = 20); hist(z[,2], breaks = 20)
par(pty = "s")
plot(z, pch = 16, cex = 0.5, asp = 1)
```

```
seed <- 10; n <- 10000
BM <- function(u){ #u es un par de uniformes
R <- sqrt(-2*log(u[1])); th <- 2*pi*u[2]
z<- R*c(cos(th),sin(th))
return(z)
}
z <- matrix(0, nrow = n, ncol = 2)
system.time( for(i in 1:n) z[i,] <-
         BM(c(runif(1), runif(1))) )
```

```
   user  system elapsed
  0.043   0.000   0.044
```





Beta distribution

- The random variable $X \sim \mathcal{B}e\,(\alpha, \beta)$ has pdf:

$$f(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \quad \alpha, \beta > 0, x \in [0,1]$$

- If $Y_1 \perp\!\!\!\perp Y_2$, $Y_1 \sim \mathcal{G}\,(\alpha, 1)$, $Y_2 \sim \mathcal{G}\,(\beta, 1)$, respectively, then

$$X \sim \frac{Y_1}{Y_1 + Y_2} \sim \mathcal{B}e\,(\alpha, \beta)$$

- When α and β are positive integers, then we can use order statistics:

Generate $\mathcal{B}e\,(m, n)$, when $m, n \in \mathbb{N}$

1. Generate $m + n - 1$ uniforms independents $u_1, u_2, \ldots, u_{n+m-1}$.
2. Calculate $X = u_{(m)}$. Then $X \sim \mathcal{B}e\,(m, n)$

- The total number of comparisons required to find $u_{(m)}$ depends of the algorithm selected to order the sample, but usually is inefficient for m and n large.

Poisson distribution I

- A Poisson random variable N with mean $\lambda > 0$ has pmf:

$$P(N = n) = \frac{e^{-\lambda}\lambda^n}{n!} \quad n = 0, 1, \ldots$$

- N can be seen as the number of arrivals in a Poisson process in a time unit.
- The interarrival times in a Poisson Process follows a exponential distribution with rate λ. Then there is a relationship between arrival times and the Poisson distribution:

$$N = n \iff \sum_{i=1}^{n} T_i \leq 1 < \sum_{i=1}^{n+1} T_i$$

This means that in a time unit there was n events. Then we can use the exponential for generating Poisson random variables.

Poisson distribution I

Generate Poisson random variables

1. Define $n = 0$, $P = 1$
2. Generate $u_{n+1} \sim \mathcal{U}(0,1)$ and define $P \leftarrow P u_{n+1}$
3. If $P < e^{-\lambda}$, accept $N = n$. Otherwise, reject n, increase to $n+1$ and return to step 2.

Efficiency of Poisson procedure

- How many random numbers are required, in average to generate a Poisson random variate?
- If $N = n$, then we require $n + 1$ numbers, so that the average number is given by

$$E(N + 1) = \lambda + 1$$

  which is large if λ is large.
- When λ is large (say, greater than 15), we can use a normal approximation that works well.

$$Z = \frac{N - \lambda}{\sqrt{\lambda}} \overset{a}{\sim} \mathcal{N}(0, 1)$$

  Then:

$$N \overset{a}{=} \ulcorner \lambda + \sqrt{\lambda} Z \urcorner$$

Example

- Generate a sample of 100 Poisson random variates with mean $\lambda = 3$.

```
N <- NULL; k <- 100
for(i in 1:k){
n <- 0; P <- runif(1)
while(P > exp(-3)){u <- runif(1); P <- P*u;  n <- n+1;}
N[i] <- n
}
N
  [1] 3 4 0 2 2 2 5 5 3 4 3 3 2 4 0 6 1 4 2 1 0 1 5 5 2 3 5 2 2 5 2 3 4 1 2 2 3
 [38] 5 4 3 3 2 3 4 4 1 6 5 3 5 3 2 1 0 2 6 7 2 1 4 0 6 4 5 2 3 6 6 4 2 3 3 0 5
 [75] 4 3 1 3 0 4 3 0 4 4 2 4 3 3 0 2 2 2 0 4 4 1 0 3 2 2
```

- Generate 100 Poisson, with mean $\lambda = 20$.

```
z <- rnorm(100) #haciendo trampa, tomando normales
N <- ceiling(20 + sqrt(20)*z)
N
  [1] 22 12 18 18 23 18 14 15 23 18 20 30 21 21 24 20 19 25 24 18 25 29 31 30 20
 [26] 20 24 24 18 15 20 15 22 18 13 17 14 24 21 22 17 19 30 22 20 15 20 22 26 19
 [51] 16 28 29 25 14 24 21 18 20 23 22 24 20 27 17 13 27 26 18 20 20 23 19 30 21
 [76] 12 21 16 18 24 24 19 25 23 19 17 17 18 19 11 17 23 17 14 21 21 26 23 22 14
```

NORTA (Normal to Anything) I

- The objective is to generate a random vector $\mathbf{X} = (X_1, \ldots, X_p)$ with the following properties:
  - $X_i \sim F_i, \quad i = 1, \ldots, p$
  - $Cor(X_i) = \Sigma_X$ given
- This method represents the vector **X** as a transformationfrom a multivariate normal vector $(Z_1, \ldots, Z_n)$ with correlation matriz given by $\mathbf{\Sigma}_Z$:

$$X = \left(F_1^{-1}(U_1), \ldots, F_p^{-1}(U_p)\right)$$

  where $U_i = \Phi(Z_i)$.
- Then **Z** is transform to a vector **U** of uniform variables.
- The matrix $\mathbf{\Sigma}_Z$ is chosen with numerical criteria given in (Cario & Nelson 1997).
- Then the problem reduces to solve $p(p-1)/2$ independent problems: for each $i \neq j$, find the value $\rho_Z(i,j)$ for which exists a function c_{ij} such that $c_{ij}(\rho_Z(i,j)) = \rho_X(i,j)$. In general, we require to do a numerical searchto find c_{ij}.

NORTA (Normal to Anything) II

- This is a direct application of the concept of *copula*

NORTA

1. Obtain the factorization **M** of Σ_Z such that $\mathbf{MM}' = \Sigma_Z$.
2. Generate $\mathbf{W} = (W_1, \ldots, W_p)'$ which elements are iid standard normals.
3. Let $Z \leftarrow \mathbf{MW}$
4. Obtain **X**, where $X_i \leftarrow F_{\mathbf{X}}^{-1}[\Phi(Z_i)]$, for $i = 1, \ldots, p$.

Example I

- It is recommended to use an R package: `SimCorMultRes` is one of them. sample vectors $\mathbf{X} = (X_1, X_2, X_3)$ with $X_1 \sim \mathcal{N}(1, 9)$, $X_2 \sim Bin(10, 0.5)$ and $X_3 \sim \mathcal{G}(3, 1)$, using as an example a correlation matrix with Toeplitz simetric structure:

$$\begin{pmatrix} a & b & c \\ b & a & b \\ c & b & a \end{pmatrix}$$

Example II

```r
library(SimCorMultRes)
set.seed(1)
n <- 1000
(Rho <- toeplitz(c(1, 0.8, 0.7))) # proposed cor matrix
```

```
     [,1] [,2] [,3]
[1,]  1.0  0.8  0.7
[2,]  0.8  1.0  0.8
[3,]  0.7  0.8  1.0
```

```r
dists <- c("qnorm", "qbinom", "qgamma")
qpars <- list(c(mean = 1, sd = 9),
              c(size  = 10, prob = 0.5),
              c( shape = 3, scale = 1))
X <- rnorta(R = n, cor.matrix = Rho, distr = dists, qparameters = qpars)
# estimated correlation matrix
cor(X)
```

```
          [,1]      [,2]      [,3]
[1,] 1.0000000 0.7861698 0.6860057
[2,] 0.7861698 1.0000000 0.7660968
[3,] 0.6860057 0.7660968 1.0000000
```

```r
colMeans(X) # means of components
```

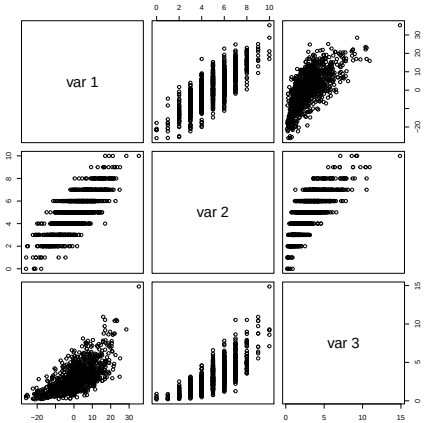
```
[1] 0.8951667 4.9760000 3.0307006
```

```r
apply(X, 2, sd) # standard deviation
```

```
[1] 9.314243 1.658365 1.839702
```

Example

```
pairs(X)
```



Evening Session I

Input and Output analysis

Input analysis

Input analysis is the analysis of input data to identify patterns of random components so they can be mimicked using streams of random variates. There are several ways to obtain input data:

- **Using historical data**: in call center, record call data for several days of operation. Drawbacks:
  - Expensive. Limited amount of information
  - Process not in operation cannot be recorded
  - Lack of potential scenarios
  - data may not include atypical operation conditions
- **Artificially generated data**: collect a representative real data sample and postulate parametric families of distributions.

Steps to estimate input data distributions

1. Identify a distribution family from the sample of real data.
2. Estimate distribution parameters to arrive at a hypothesis for a specify distribution that is appropriate.
3. Perform statistical goodness-of-fit tests to investigate if the hypothesis may be rejected. In that case, start again in step 1.

Some goodness-of-fit tests:

- Kolmogorov-Smirnov test
- Chi-square test
- qq-plot

Output analysis I

- The stochastic behavior of the output of a process is due to the randomness of its inputs and internal components.
- Therefore, the output of a simulation model also should be treated as random variables. Examples:
  - waiting time of jobs
  - cycle times
  - throughput
  - utilization
  - costs
- A simulation is a computer-based statistical sampling experiment:
  - To estimate the characteristics (moments) of output variables given some input conditions and configurations.
  - Counterfactual analysis: To compare characteristics of output variables given some input conditions and configurations.
- It is important to determine adequate sampling and sample sizes.

Output analysis II

- The output process of almost all simulations are nonstationary and autocorrelated, so no iid samples are obtained directly.
- Let $\{Y_i\}$ an output stochastic process from a single simulation run. Consider that we have make n independent replications of size m with some conditions, using different streams of random numbers, resulting in the array of observations:

$$
\begin{matrix}
y_{11}, \ldots, y_{1i}, \ldots, y_{1m} \\
y_{21}, \ldots, y_{2i}, \ldots, y_{2m} \\
\vdots, \ldots, \vdots, \ldots, \vdots \\
y_{n1}, \ldots, y_{ni}, \ldots, y_{nm}
\end{matrix}
$$

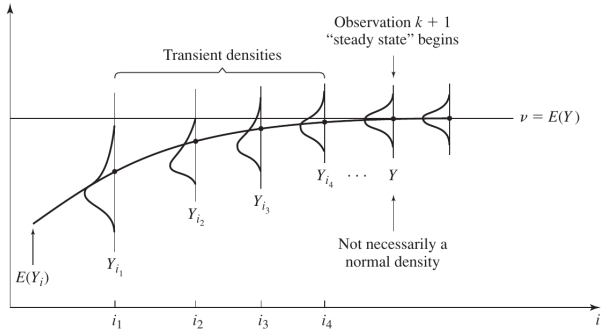
  where rows are replicates and columns are observations of each replicate. Note that rows are not iid, but columns are iid observations of Y_i for $i = 1, \ldots, m$
- The goal of the output analysis is to use observations y_{ij} ($i = 1, \ldots, m$, $j = 1, \ldots, n$) to draw inferences about the distributions of $Y_1, \ldots, Y_m$.

Transient and steady-state behavior I

- Let $\{Y_i\}$ an output stochastic process, and

$$F_i(y|I) = P(Y_i \leq y|I)$$

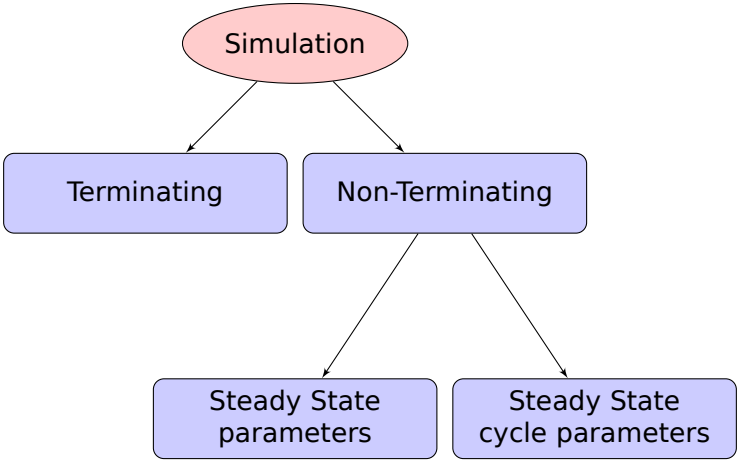
- $F_i(y|I)$ is the transient distribution of the output process at time i for initial conditions I. Note that $F_i(y|I)$ will be different for each value of i and each set of initial conditions I.



Transient and steady-state behavior II

- For fixed y and I, $\{F_i(y|I)\}$ is a sequence of numbers. If $F_i(y|I) \to F(y)$ as $i \to \infty$, for all y, I, F is the steady distribution of Y_i, that does not depend on I. This property is similar to the behvior of Markov chain process or in general of any Markovian Process.

Types of simulation with respect to output analysis I



Types of simulation with respect to output analysis II

According to the output obtained from the simulation, is the analysis done.

- **Terminating simulation**: There is an event that specifies the length of each replication. Comparable random variables from different runs are iid (usual statistical analysis). Examples:
  - restaurant operation by day
  - waiting line reaches a target
- **Non Terminating simulation**: There is no natural event to specify the length of a run. Initial conditions change. A measure performance is said to be *steady-state parameter* if it is a characteristic of a steady-state distribution of an output stochastic process $\{Y_i\}$. Examples:
  - credit approval
  - order fulfillment
  - policy underwriting
  - If $\{Y_i\}$ does not have an steady-state distribution, we can divide the time into same time length intervals or cycles so that Y_i are comparable for each cycle. Then a measure of performance is *steady-state cycle parameter* if it is a characteristic of Y_i^C based on the steady-state distribution $Y^C \sim F^C$.
  - There can be other cases but there will be not treated here.

Statistical analysis of terminating simulations I

- n independent replications of a terminating simulation. The sample $y_{1i}, y_{2i}, \ldots, y_{ni}$ is a random sample of the ith observation and if i is large enough we can treat as a sample from the steady-state distribution. Then we can compute, for example:

$$\bar{y} \pm t_{(n-1,1-\alpha/2)} \sqrt{s^2/n}$$

  where s^2 is an estimate from the sample of the variance of Y.
- An approximate expression for the total number of replications required to obtain an absolute error of ϵ is:

$$n_\alpha(\epsilon) = \min\{i \geq n | t_{(i-1,1-\alpha/2)} \sqrt{s_n^2/i} \leq \epsilon\}$$

  or the smallest $n_\alpha(\epsilon)$ such that

$$n_\alpha(\epsilon) \geq s_n^2 \left(\frac{z_{1-\alpha/2}}{\epsilon} \right)^2$$

  The accuracy will depend on how close is the variance estimate s_n^2 to $\text{Var}(Y)$.

Statistical analysis of non terminating simulations I

- Assume that $F_i(y|I) \to F(y)$ as $i \to \infty$, where Y is the steady-state random variable with distribution F.
- One difficulty in estimate a characteristic θ of Y is that $F_i \neq F$ since it will no be possible to choose I to represent steady-state beahvior.
- So we can warming-up the model using observations after l runs. For example, if $\theta = \mathsf{E}(Y) = \lim_{i\to\infty} \mathsf{E}(Y_i)$, then use as estimate

$$\bar{Y}(m,l) = \frac{\sum_{i=l+1}^{m} Y_i}{m-l}$$

wich we expect to be less biased than $\bar{Y}(m)$.

Counterfactual analysis

Counterfactual analysis I

- In a counterfactual analysis we evaluate scenarios that have not existed or initiatives that are not implemented: hypothetical scenarios to understand the potential outcomes of different actions.
- We can also compare the output of several different simulation models that might represent competing system designs or alternative operating policies to decide which which can be better implemented.
- A basic requirement for using statistical methods for comparing alternative configurations is to collect iid observations with expectation equal to the measurement of performance.
  - In the case of terminating simulations, do independent replications.
  - In the case of non-terminating simulations, we need to apply the deletion/replication or warm-uo technique.
- We assume that the different configurations are given. In other situation is required the application of techniques of design of experiments.

Compare two systems on some performance measure (response) I

- Assume normality of the response. For $i = 1, 2$ let $y_{i1}, y_{i2}, \ldots, y_{in_i}$ a sample of n_i iid observations from system i.
- Let $\mu_i = \mathrm{E}(Y_{ij})$ is the expected response under system i. We want a confidence interval for $\zeta = \mu_1 - \mu_2$.
- Independence of Y_{1j} and Y_{2j} depends on how the simulations are executed and determines which of the following two options is best to estimate a confidence interval.

Paired-t confidence interval I

- If $n_1 = n_2 = n$, we can pair Y_{1j} and Y_{2j} to define $Z_j = Y_{1j} - Y_{2j}$.
- Then $Z_1, \ldots, Z_j$ are iid random variables and $\mathsf{E}(Z_j) = \zeta$. Then take

$$\bar{Z}(n) \pm t_{(n-1,1-\alpha/2)} \sqrt{s(n)^2/n}$$

  where $s(n)^2 = \frac{\sum_{j=1}^{n} (Z_j - \bar{Z}(n))^2}{n-1}$
- If Z_j are normal, the confidence interval is exact. Otherwise, rely on the CLT.
- No need to assume $Y_{1j} \perp\!\!\!\perp Y_{2j}$, or $\mathsf{Var}(Y_{1j}) = \mathsf{Var}(Y_{2j})$
- We can also use some nonparametric tests.

Modified two-sample t-test I

- For this case assume that $Y_{1j} \perp\!\!\!\perp Y_{2j}$, $j = 1, 2, \ldots, n$ and both samples are normally distributed.
- Use Welch-Satterthwaite test:

$$\bar{Y}_1(n_1) - \bar{Y}_2(n_2) \pm t_{(g,1-\alpha/2)} \sqrt{\frac{s_1(n_1)^2}{n_1} + \frac{s_2(n_2)^2}{n_2}}$$

where $g = \frac{\left(\frac{s_1(n_1)^2}{n_1} + \frac{s_2(n_2)^2}{n_2}\right)^2}{\frac{(s_1(n_1)^2/n_1)^2}{(n_1-1)} + \frac{(s_2(n_2)^2/n_2)^2}{(n_2-1)}}$ is an approximate $100(1-\alpha)$% confidence interval for ζ.

Confidence intervals for comparing more than two systems I

- in the case we consider several confidence interval statements simultaneously, we need to adjust the significance level so the overall confidence level of all intervals' covering their target is at least $1-\alpha$. We use Bonferroni's inequality:

$$P\left(\bigcup_{i=1}^{m} B_i\right) \leq \sum_{i=1}^{m} P(B_i)$$

  We adjust for each component $1-\alpha$ for $1-\alpha/m$
- This is a problem of comparison of k means, in one of the following options:
  1. Comparison with a standard: one of the model variants is a "standard", lets consider it variant 1. We want to compare the $k-1$ differences $\mu_i - \mu_1$. In this case $m = k-1$. We can use paired t-test or two sample-t accordingly.
  2. All pairwise comparisons: $\mu_{i_1} - \mu_{i_2}$ for all $i_1, i_2 \in \{1, \ldots, k\}$, $i_1 < i_2$. In this case use $m = \frac{k(k-1)}{2}$

Evening Session II

Case 1: Queue model

Example: Waiting line

- Consider a service unit with one server (teller, ticket box, airport runway , etc.)
- Clientes line up to receive service.
- We want to estimate the mean time in the line, measure as the average since arrival until starts the service.
- The variables that determine the system state are:
  - State of the server (idle, occupied)
  - The number of clients waiting in the line $\{0, 1, 2, \ldots\}$
  - The interarrival time of clients in the line
  - Service time of each client
- Two types of relevant events:
  - client arrival
  - completion of service (departure of a client)

Example 1: Waiting Line

Definitions:

$t_i =$ Time of arrival of i-th client.

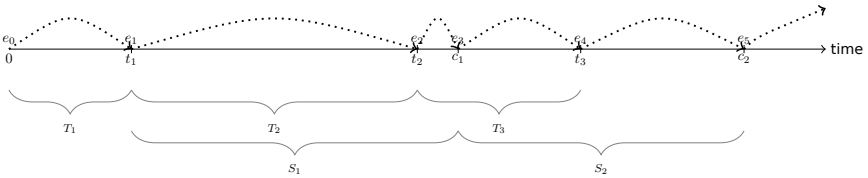
$T_i = t_i - t_{i-1} =$ interarrival times of $i-1$ and i clients

$S_i =$ time of service client i.

$D_i =$ waiting time in line of client i.

$c_i = t_i + D_i + S_i =$ time elapsed between arrival of client i and its departure from the system.

$e_i =$ time of ocurrence of i-th event of any kind (simulation clock).



System Performance Measures I

- Assume that system stops running when the n-th client enters service.
- The time cycle then is random.
- Three measures of performance are considered, one for the clients and two for the server:
  1. The **average waiting time in line** of the n clients:

$$\bar{D}_n = \frac{\sum_{i=1}^{n} D_i}{n}$$

(clients arriving when no one is waiting have $D_i = 0$).

System Performance Measures II

2. The **average number of clients in the line**, q_n. This variable depends on the random time required to observe n clients.
   To consider this random time, let $Q(t) =$ number of clients in line at time t, $Q(t) \in \{0, 1, 2, \ldots\}$. Let T_n the total time to observe n variables $\{D_1, \ldots, D_n\}$. Then

$$q_n = \sum_{i=0}^{\infty} i \frac{T_i}{T_n}$$

   where T_i is the time wehen the line has length i (note $T_n = \sum_{i=0}^{\infty} T_i$). Then:

$$\hat{q}_n = \frac{\sum_{i=0}^{\infty} iT_i}{T_n} = \frac{\int_0^{T_n} Q(t)\, dt}{T_n}$$

   We can compute the estimate $\hat{q}_n$ as the sum of rectabgle areas.
3. The **expected utilization of the server**, u_n. This is the proportion of simulation time that server was busy. Define $B(t) = I(\text{service at time } t)$, then:

$$\hat{u}_n = \frac{\int_0^{T_n} B(t)\, dt}{T_n}$$

System Performance Measures III

- Other termination rules are possible. For example, observe the system a fixed period, so total number of clients served is random.
- Simulation will give the estimates, and if possible, will allow to visualize the behavior of the functions $Q(t)$ and $B(t)$.

Components of a Discrete Event Simulation model I

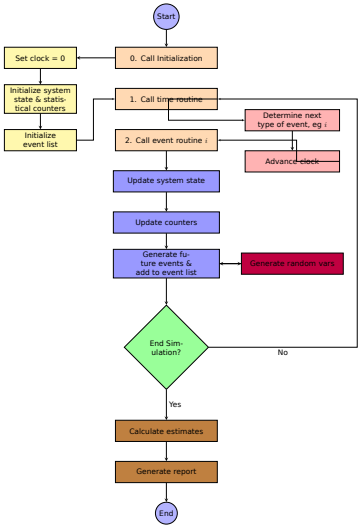
The color code corresponds to the activities that are shown in the following flow charts that connect logical relations among the components of the model.

- *System state*: State variables required to describe the system.
- *Simulation clock*: variable that records the simulation run time.
- *Event List*: List with times of ocurrence of each relevant event.
- *Statistical counters*: Variables that keep record of system performance.
- *initialization routine*: Subprogram that initializes simulation (variables, lists, etc.) at time 0.
- *Time routine*: Subprogram that determines the following event from the event list and advances the simulation clock at the time in which such event occurs.
- *Event routine*: Subprogram that updates the system state when a given type of event occurs (cab be a routine for each type of event).
- *Library routine*: A set of subprograms used to generate random variables with the specified distributions in the simulation model.

Components of a Discrete Event Simulation model II

- *Report generator*: Subprogram that computes the performance measures selected to report and produce a summary of the results when simulation concludes.
- *Main program*: Subprogram (`mm1` next) that calls the subprograms when required.

Process flow



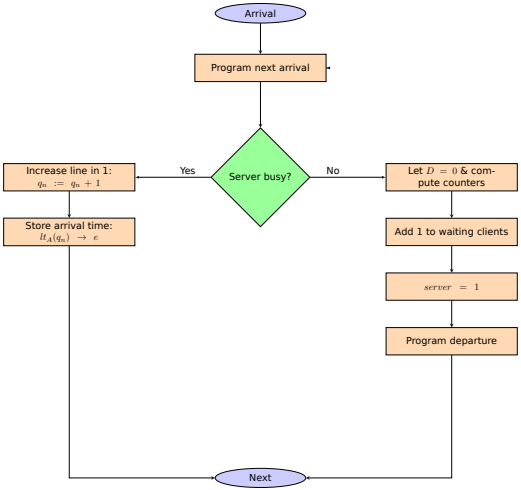
Assumptions model $M/M/1$ I

- System executes until n clients wait in line and simulation ends when client n enters service.
- Interarrival timesare iid exponential random variables with mean λ_A. The exponential distribution with mean λ_A has pdf:

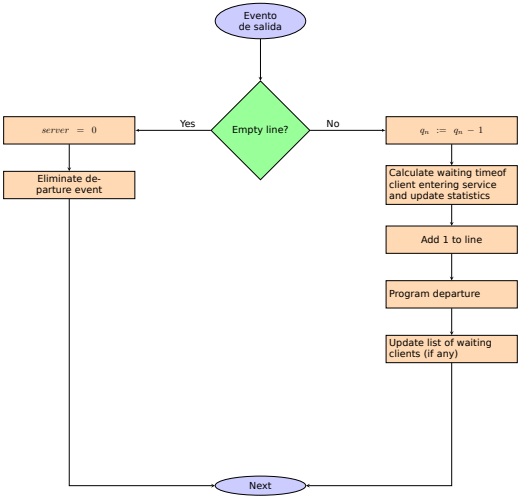
$$f(x) = \frac{1}{\lambda_A} e^{-x/\lambda_A} I(x \geq 0)$$

- Service times are model as iid exponential with meand λ_S.
- Two types of relevant events:
  1. Arrival of a client
  2. Departure of a client after service.

Logic event 1: Arrival of a client I



Logic of departure I



Variables and functions I

Model parameters:

- `lambdaA` and `lambdaS` are the average times of arrival and service, respectively.
- `n` = number of clients entering to the system. Fixed.

Principal variables:

- `reloj` = e = simulation clock
- `sig_tipo_evento` = Next type of event (1= arrival, 2= departure)
- `tiempo_sig_evento` = vector of dim 2, with arrival time and departure time
- `Di` = D_i waiting time in line client i
- `total_esperas` = T_n = Sum of waiting times D_i
- `q_t` = length of line in t

Auxiliar variables:

- `lt_A` = lt_A vector list of arrival times (dynamic length)
- `le` = occurred event list, with occurrence time and type, and the length of the line at those times

Variables and functions II

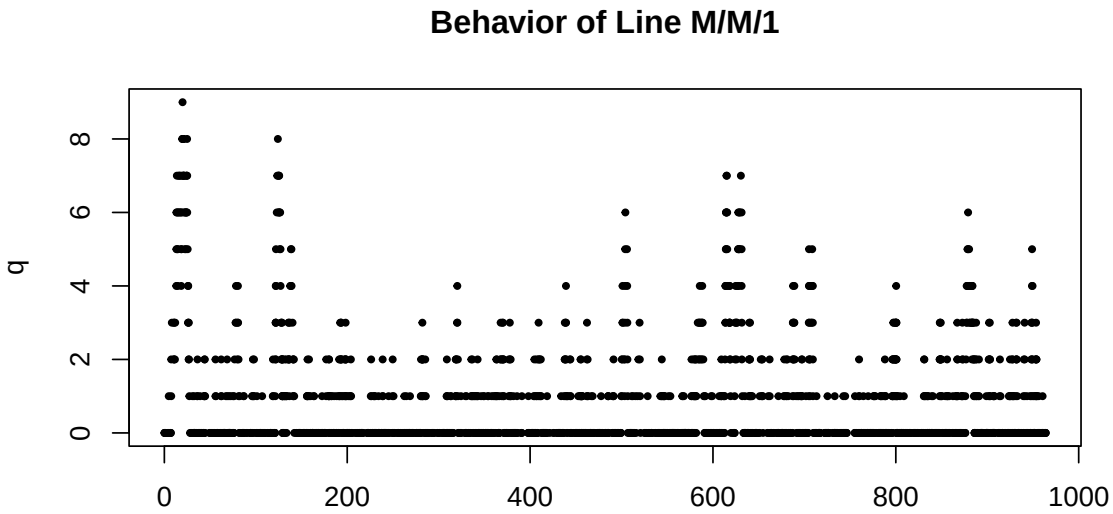
- `area_q` = aux variable to compute the average length of the line
- `area_status_servidor` = aux variable to compute serve use
- `tiempo_ultimo_evento` = record of the last event before advancing clock
- `tiempo_desde_ultimo_evento` = compute the interval of time between the current event and the last event
- `clientes_enespera` = Number of clients that are on line

Program execution I

The next simulation give us a sample of size 1 of the process $M/M/1$. To be able to get conclusions we need to execute the simulation several times to obtain an adequate sample of the estimated parameters.
The program `Queue.R` is in the following github link

```
source("https://raw.githubusercontent.com/jvega68/ISI/master/Queue/Queue.R")
a <- mm1(lambdaA = 1, lambdaS = 0.5, n = 1000)
plot(a$le[,1], a$le[,3], type="p", pch = 19, cex = 0.5,
     xlab = "time",
     ylab = "q",
     main = "Behavior of Line M/M/1")
```

Program execution II



Comparing to Theory $M/M/1$ I

Theory tell us that $M/M/1$ queues with arrival and service time exponential with parameters $\lambda_A = 1$ and $\lambda_S = 0.5$ have expected values:

- Expected utilization of server $u = \frac{\lambda_S}{\lambda_A} = \frac{1}{2}$.

```
a$utilizacion
[1] 0.5168962
```

- Average length of queue: $q_n = \frac{\lambda_S^2}{\lambda_A(\lambda_A - \lambda_S)} = 0.5$

```
a$longitud.promedio.fila
[1] 0.5942199
```

- Average waiting time: $D_n = \frac{\lambda_S^2}{\lambda_A - \lambda_S} = 0.5$

```
a$promedio.espera
[1] 0.5728257
```

Case 2: Inventory model

Ejemplo 2: Simulation of inventory I

- Consider the inventory of a product, for example, $500 banknotes. The banknotes are in bags of 5,000 pieces, (so a bag's amount of money is 12.5 million pesos) How many bags have to be produced to satisfy the demand during n months?
- Demand has the following characteristics:
  1. $D \sim F_D$ is size of demand (number of bags required)
  2. $T_i \sim \exp(\lambda)$ time bewteen demands
  3. Cost: At the beggining of each month the inventory is reviewed and the size of the production is determined. If order Z units, the cost is:

$$\texttt{Cost} = \begin{cases} 0 & i = 0 \\ K + iZ & \text{otherwise} \end{cases}$$

     where K is the fixed cost and i is the unit cost.
  4. Time of delivery of the order (*lead time*)is $U(a, b)$

Ejemplo 2: Simulation of inventory II

5. There is a reorder policy of type (s, S), where s is an indicator of the minimum level of inventory and S is the maximum level of inventory

$$Z = \begin{cases} S - I & I \leq s \\ 0 & I > s \end{cases}$$

where I is the inventory level at the beggining of the month

6. Is supposed that demand not-satisfed (backlog) is attended in the future so inventory level can be negative.

- Other costs considered in the model are:
  - Storage cost: will be of h per bag (rents, security, insurance, humidity control, etc). Let $I^+(t) = \max\{I(t), 0\}$ inventory level at time t. Then the average cost per month is $h\bar{I^+}$ where:

$$\bar{I^+} = \frac{1}{n} \int_0^n I^+(t)\, dt$$

Ejemplo 2: Simulation of inventory III

- Cost for delay: π per bag (reputation, records, etc.). Let $I^-(t) = \max\{-I(t), 0\}$ the backlog. Then the average cost per month is $\pi \bar{I^-}$, where

$$\bar{I^-} = \frac{1}{n} \int_0^n I^-(t)\, dt$$

The state variables of the system are:

- $I(t)$ level of inventory at t
- Order size
- times of events

and the event types:

1. Order arrival from the printworks
2. Demand D
3. End of simulation: n months
4. Assessment of the inventory at the beggining of each month

Example

```
source("https://raw.githubusercontent.com/jvega68/ISI/master/Inventory/Inventory.R")
 (a <- minv(prin = F))

  policy_s policy_S total_cost average_order_cost average_stock_cost
1       20       40     126.45              97.55               9.18
2       20       60     117.17              85.64              17.20
3       20       80     123.06              85.82              26.62
4       20      100     125.61              81.12              37.10
5       40       60     126.42              98.67              25.28
6       40       80     126.46              91.32              33.70
7       40      100     132.15              87.40              43.88
8       60       80     146.45             102.67              43.70
9       60      100     145.40              92.88              52.52
  average_backlog_cost
1                19.73
2                14.33
3                10.62
4                 7.40
5                 2.47
6                 1.45
7                 0.87
8                 0.08
9                 0.00
```