Computational Statistics

Lecture 3: Continuous Random Variables

Raymond Bisdorff

University of Luxembourg

8 novembre 2019

Content of Lecture 3

Probability distributions in R-core

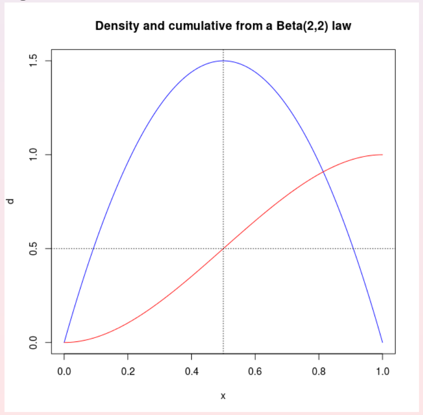
Distribution	R-name	Parameters	Default Values
Beta	`beta`	shape1,shape2	
Binomial	`binom`	size,prob	
Cauchy	`cauchy`	location, scale	0,1
Chi-square	`chisq`	df	
Exponential	`exp`	1/mean	
F	`f`	df1, df2	
Gamma	`gamma`	shape, 1/scale	NA, 1
Geometric	`geom`	prob	
Hypergeometric	`hyper`	m,n,k	
Log-normal	`lnorm`	mean,sd	0,1
Logistics	`logis`	location,scale	0,1
Gaussian	`normal`	mean, sd	0,1
Poisson	`pois`	lambda	
Student	`t`	df	
Uniform	`unif`	min,max	0,1
Weibull	`weibull`	shape	

Each R-name may be prefixed with *d*, *p*, *q*, and *r*, to deliver the corresponding density (*df*), cumulative probability distribution (*cdf*), the quantiles fct (cdf^{-1}), and a random instance generator, like `runif` for instance.

Graphing probability distributions

Checking, for instance, the shape of the *density function* (*df*) and/or of the *cumulative distribution function* (*cdf*) of a beta(2,2) law may be done with the following R commands :

```
> x = seq(0,1, length=100)
> d = dbeta(x,2,2) # beta df
> p = pbeta(x,2,2) # beta cdf
> plot(x,d,type="n")
> lines(x,d,col="blue")
> lines(x,p,col="red")
> abline(h=0.5,lty="dotted"
> abline(v=0.5,lty="dotted"
```



Graphing probability distributions

Exercise (Centrally peaked distributions)

Construct a graph in R on the real interval $[-5, 5]$ which superposes the standard normal distribution $\mathcal{N}(0, 1)$, the student t-distributions $t(6, 0, 1)$ and $t(4, 0, 1)$, the Cauchy distribution $\mathcal{C}(0, 1)$ and the logistic distribution $\mathcal{L}(0, 1)$.

Exercise (Distributions on the positive half-line)

Construct a graph in R on the half-line $[0, 10]$ which superposes the exponential distribution $Exp(0.5)$, the Fischer F-distribution $F(10, 4)$, the Lognormal distribution $\mathcal{LN}(1, 1)$, the Gamma distribution $\Gamma(3, 1)$, and the Chi-Square Distribution $\chi^2(df = 5)$.

Generating uniform simulation data with R

The basic uniform generator in R is `runif` with required number *nSim* of values to be generated. The range of a uniform random variable $X \sim \mathcal{U}(2, 5)$ may be indicated with the min (default $= 0$) and max (default $= 1$) parameters like this :

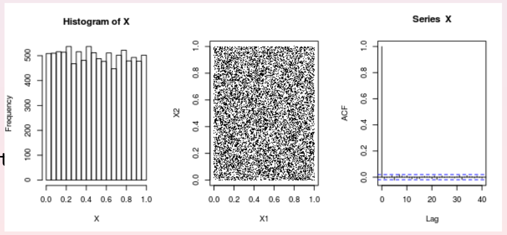
```
> nSim = 10^4
> set.seed(1)   # initializing the generator
> X = runif(nSim, min=2, max=5)
```

The commands will produce a vector X containing 10^4 values generated from a uniform law of range 2 to 5.

Checking the quality of the uniform generator

Checking the quality of a uniform random sequence X may be done with a histogram, a plot of the pair $(X[i], X[i + 1])$, and the estimated autocorrelation function $acf(X)$. Try the following R commands :

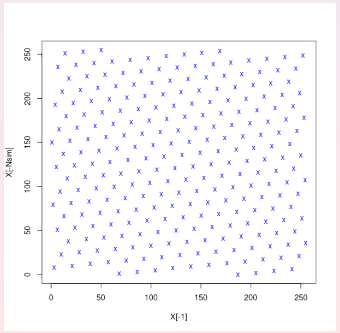
```
> par(mfrow=c(1,3))
> hist(X)
> X1 = X[-nSim] # skip 1rst
> X2 = X[-1]     # skip last
> plot(X1,X2,pch=".")     # scat
> acf(X)
```



The spectral test for RNGs

An especially important way to check the quality of a uniform random number generator is given by the spectral test. If we have a sequence $\langle U_n \rangle$ of period m, the basic idea is to analyse the positions of the set of all m points $\{(U_n, U_{n+1}, ..., U_{n+t-1})\}$ for $0 \geq n \geq m$ in t-dimensional space. For instance, consider the following $t = 2$ and $t = 3$ tests for a linear congruational generator :

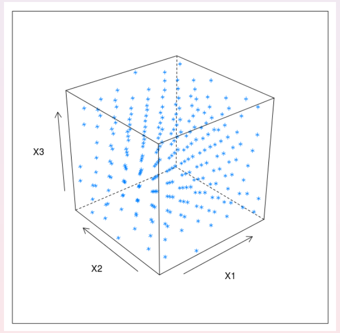
```
> nSim = 256
> X=rep(0,nSim)
> for (i in 2:nSsim){
>    X[i] = (137*X[i-1]+187)%%256 }
> plot(X[-1],X[-nSim],col="blue",\
      type="p",pch="x",lwd=2)
```



The spectral test for RNGs – continue

With the same LCGRNG we obtain in a three-dimensional spectral test the follwoing result :

```
> nSim = 256
> X=rep(0,nSim)
> for (i in 2:nSim){
>   X[i] = (137*X[i-1]+187)%%256 }
> X1 = X[3:256]
> X2 = X[2:255]
> X3 = X[1:254]
> library("lattice")
> cloud(X3 ~ X1 + X2,type="p")
```



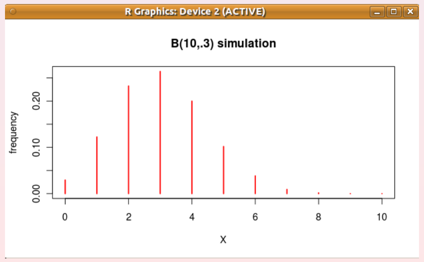
Exercise

Compare with the results of the spectral test for the default Mersenne Twister generator.

Discrete inverse transform

To generate $X \sim \mathcal{P}(\theta)$ where $\mathcal{P}(\theta)$ is a discrete random variable defined on integer values $0, 1, 2, ..., \theta$, we store once for all the discrete cumulated probabilities : $p_0 = P(X \leqslant 0)$, ..., $p_\theta = P(X \leqslant \theta)$. With $U \sim \mathcal{U}(0,1)$, one may take : $X = k$ if $p_{k-1} < U < p_k$ for $k = 1, ..., \theta$.
Here the R code to generate a variable $X \sim \mathcal{B}(10, 0.3)$:

```
> P = pbinom(0:10,10,.3)
> X = rep(0,nSim)
> for (i in 1:nSim){
+    u = runif(1)
+    X[i] = sum(P < u) }
> freq = hist(X,breaks=seq(0,11),
+  right=F)
> attach(freq)
> plot(breaks[-11],density,"h"
+  main="B(10,.3) simulation")
```



Discrete empirical random laws

Exercise

You are requested to draw a sample of 1000 random integers in the range $[0; 9]$ *along the following empirical probability distribution :*

0	1	2	3	4
0.0478	0.3349	0.2392	0.1435	0.0957

5	6	7	8	9
0.0670	0.0478	0.0096	0.0096	0.048

1. *Write a Python program for generating this sample and store the resulting random sequence in a csv file,*
2. *Generate this sample with R,*
3. *Compare both sample distributions with the empirical one.*

The continuous inverse transform

If random variable X has density function f_X and cumulative density function (cdf) F_X, we have the relation :

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

If we set $U := F_X \sim \mathcal{U}(0,1)$ and assume that the cdf F_X has an **analytical inverse F_X^{-1}** then :

$$\begin{aligned} P(U \leqslant u) &= P(F_X \leqslant F_X(x)) \\ &= P[F_X^{-1}(F_X) \leqslant F_X^{-1}(F_X(x))] \\ &= P(X \leqslant x) \end{aligned}$$

Now, if $F_X^{-1}(u) := \inf\{ x \mid F_X(x) \geqslant u\}$ then $F_X^{-1}(U) \sim X$.

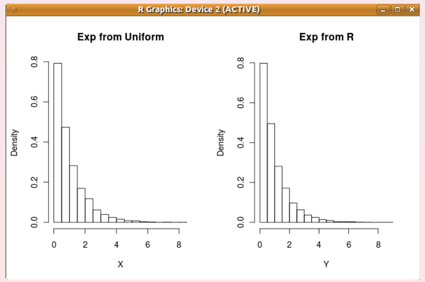
The inverse transform of the standard exponential probability law

Suppose $X \sim \lambda e^{-\lambda x}$ with $\lambda = 1$. Then $F_X = 1 - e^{-x}$. Solving for x in $u = 1 - e^{-x}$ gives $x = -\log(1-u)$. Therefore, if $U \sim \mathcal{U}(0,1)$, then $1 - U \sim U$ and

$$X \; = \; -\log U \; \sim \; e^{-x}$$

Try the following R commands :

```
> nSim = 10^4
> U = runif(nSim)
> X = -log(U)    # transform
> Y = rexp(nSim) # R builtin
> par(mfrow=c(1,2))
> hist(X,freq=F,
+     main="Exp from Uniform")
> hist(Y,freq=F,
+     main="Exp from R")
```



Standard exponential law based generators

Suppose we have a generator for the standard exponential law based on uniform random number generator.
If variables X_i's are independent e^{-x} distributed variables, then the Chi-square, Gamma and Beta distributions can be simulated as follows :

$$Y = 2\sum_{i=1}^{n} X_i \quad \sim \quad \chi^2(df = 2n)$$

$$Y = \beta\sum_{i=1}^{a} X_i \quad \sim \quad \mathcal{G}(a,\beta)$$

$$Y = \frac{\sum_{i=1}^{a} X_i}{\sum_{i=1}^{a+b} X_i} \quad \sim \quad \mathcal{B}(a,b)$$

The normal probability distribution

- A very special role in simulations is played by the "*normal*" or "*normally*" distributed random variables.
- A random variable $X \in \mathbb{R}$ is "*normally*" distributed, or Gaussian, with mean $E(X) = \mu$ and standrad deviation $\sqrt{V(X)} = \sigma$:

$$X \sim \mathcal{N}(\mu, \sigma),$$

  when

$$P(X \leq x) \;=\; \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}\left[\frac{(t-\mu)}{\sigma}\right]^2} dt.$$

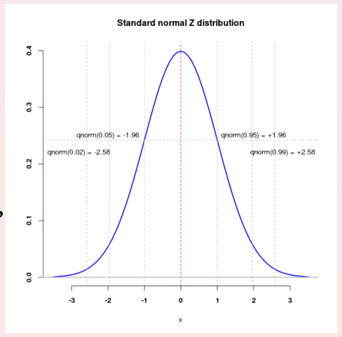
- A *standard* Gaussian variable is a Gaussian variable, denoted Z, with zero mean ($\mu = 0$) and unit standard deviation ($\sigma = 1$).

Quantiles and tolerance intervals of a Gaussian variable

$\mu \pm 1.96\sigma$ or $z \pm 1.96$ gathers 95% of the observations
$\mu \pm 2.58\sigma$ or $z \pm 2.58$ gathers 99% of the observations
$\mu \pm 3.29\sigma$ or $z \pm 3.29$ gathers 99.9% of the observations

$\mu \pm 1\sigma$ or $z \pm 1$ gathers 68.3% of the observations
$\mu \pm 2\sigma$ or $z \pm 2$ gathers 95.5% of the observations
$\mu \pm 3\sigma$ or $z \pm 3$ gathers 99.7% of the observations

```
> mu = 0
> sig = 1
> low = mu - 3.5*sig
> up = mu + 3.5*sig
> x = seq(low,up, by=0.1)
> d = dnorm(x, mean=mu,sd=sig)
> plot(x,d,type="l",lwd=2,xlim=c(low,up),
+ ylab=" ",col="blue",
+ main="Standard normal Z distribution")
> abline(v=mu,lwd=1,lty=2,col="red")
```



Important properties I

1. If $X_1 \sim \mathcal{N}(\mu_1, \sigma_1)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2)$ are two Gaussian variables, then $X_1 + X_2 \sim \mathcal{N}\left(\mu = \mu_1 + \mu_2, \sigma = \sqrt{\sigma_1^2 + \sigma_2^2}\,\right)$.
2. If Z_1 and Z_2 are two independent standard Gaussian variables, then
$$Z \;=\; \frac{Z_1 + Z_2}{\sqrt{2}} \quad\sim\quad \mathcal{N}(0, 1).$$
3. If $Z_1, ..., Z_n$ are n mutually independent standard Gaussian variables, then
$$Z \;=\; \frac{Z_1 + ... + Z_n}{\sqrt{n}} \quad\sim\quad \mathcal{N}(0, 1).$$

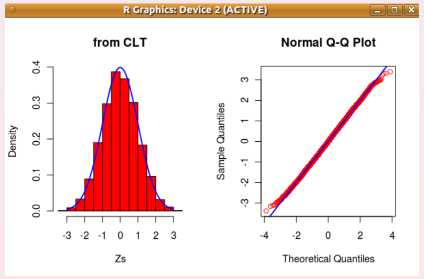
Important properties II

1. If X_i, for $i = 1...n$, are i.i.d. Gaussian $\mathcal{N}(\mu, \sigma)$ variables then
$$\begin{aligned} X_1 + ... + X_n &\sim \mathcal{N}(n\mu, \sqrt{n}\sigma), \\ \frac{(X_1 + ... + X_n)}{n} &\sim \mathcal{N}(\mu, \sigma/\sqrt{n}), \\ \frac{(X_i - \mu)}{\sigma} &\sim \mathcal{N}(0, 1), \\ \frac{(\overline{X_i} - \mu)}{\sigma}\sqrt{n} &\sim \mathcal{N}(0, 1). \end{aligned}$$
2. If $Z_1, ..., Z_n$ are n mutually independent standard Gaussian variables, then
$$X \;=\; \sum_{i=1}^{n} Z_i^2 \quad\sim\quad \chi^2(df = n).$$

The Central Limit Theorem – CLT

The sum of n independently distributed random variables X_1, X_2, ... , X_n, when n gets large, tends toward a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ where $\mu = E(\sum_{i=1}^{n} X_i)$ and $\sigma = \sqrt{V(\sum_{i=1}^{n} X_i)}$.

```
> nSim = 10^4
> X1 = runif(nSim)
> ...
> X10 = runif(nSim)
> X = X1 + X2 + ... + X10
> mu = mean(X); sigma = sd(X)
> Zs = (X-mu)/sigma
> par(mfrow=c(1,2))
> hist(Zs, freq=F,main="from CLT")
> p = seq(-3,3,lengh=500)
> lines(p,dnorm(p))
> qqnorm(Zs), abline(0,1)
```



"Normal" does not mean "normally" observed !

- The name "*normal distribution*" was introduced in 1893 by Karl Pearson ; the distribution was originally discovered in 1721 by A. De Moivre, and later rediscovered and thoroughly independently studied by Laplace (1749–1827) and Gauss (1777–1855).
- The very importance of the Gaussian comes indeed essentially from its mathematical properties which position this distribution via the CLT and the Large Number Laws in the center of mathematical statistics and measure theory.
- Examples of nearly normal random variables are, however, very rarely observed in Nature. Even in the presence of the CLT, extensive sampling from natural data very often reveal systematic differences with a Gaussian distribution ; usually due to showing much heavier distribution tails.

Exercise

We have seen previously that twice the sum of n independent standard exponential variables is distributed like a chi-square variable with 2*n degrees of freedom.*
Similarly, we have seen that the sum of squares of n independent standard Gaussian variables is again distributed like a chi-square variable with n degrees of freedom.

Questions :

1. *What is hence the formal relationship between standard exponential and standard Gaussian variables ?*
2. *Illustrate graphically your previous result with a suitable Monte Carlo simulation experiment.*

A didactical Gaussian random number generator

The inverse of a Gaussian *cdf* has, contrary to the exponential *cdf*, no closed analytic form. One simple way, however to achieve the simulation of the standard Gaussian variable $Z \sim \mathcal{N}(0,1)$ uses the Box-Muller algorithm.
It is based on the observation that, if U_1 and U_2 are two independent and identically $\mathcal{U}(0,1)$ distributed uniform random variables, then :

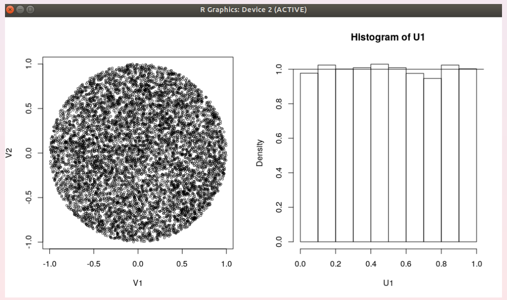
$$\begin{aligned} Z_1 &= R\cos(\Theta) = \sqrt{-2\log(U_1)}\cos(2\pi U_2), \\ Z_2 &= R\sin(\Theta) = \sqrt{-2\log(U_1)}\sin(2\pi U_2). \end{aligned}$$

where $R = \sqrt{Z_1^2 + Z_2^2}$ and $\Theta = 2\pi U_2$ are resp. the length of a vector and its angle with respect to the x-axis in a Cartesian system whith standard Gaussian coordinates (Z_1, Z_2).

The Box-Muller algorithm I

i) If $V_1 \sim \mathcal{U}(-1,1)$ and $V_2 \sim \mathcal{U}(-1,1)$ with $0 < U_1 = (V_1^2 + V_2^2) < 1$, the pairs (V_1, V_2) give uniformly random positions within a unit circle and U_1 is $\mathcal{U}(0,1)$ distributed.

```
> nSim = 10^4
> v1 = runif(nSim,-1,1)
> v2 = runif(nSim,-1,1)
> r2 = v1*v1 + v2*v2
> V1 = v1[r2<1]
> V2 = v2[r2<1]
> plot(V1,V2,pch="°")
> abline(v=0,h=0,lty=2)
> U1 = V1*V1 + V2*V2
> hist(U1,freq=F)
> abline(h=1.0)
```



The Box-Muller algorithm II

ii) By noticing that $R^2 = (Z_1^2 + Z_2^2)$ takes value in $[0, \infty]$ and :

$$(Z_1^2 + Z_2^2) \;\sim\; \chi^2(R^2, df = 2) \;\sim\; 2e^{(-R^2)}\,,$$

we may simulate R by the exponential inverse transform :

$$R \;=\; \sqrt{-2\log(U_1)},$$

The Box-Muller algorithm III

iii) Furthermore,

$$\cos(\Theta) \;=\; V_1/\sqrt{U_1}$$

and,

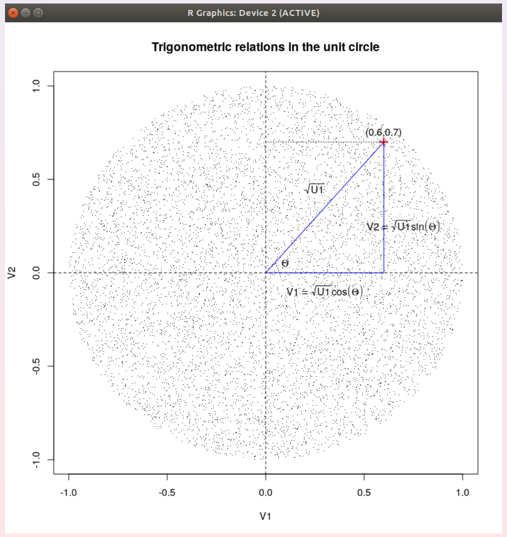
$$\sin(\Theta) \;=\; V_2/\sqrt{U_1},$$

such that :

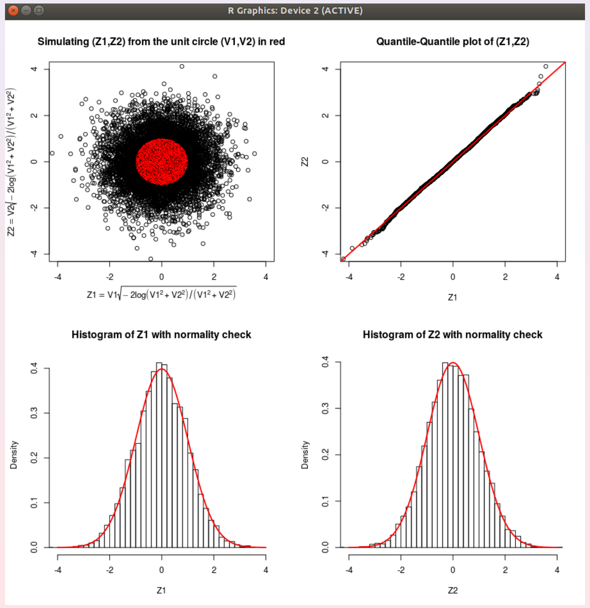
$$Z_1 \;=\; \sqrt{-2\log(U_1)/U_1} \times V1\,,$$

and

$$Z_2 \;=\; \sqrt{-2\log(U_1)/U_1} \times V2\,.$$


R Graphics: Device 2 (ACTIVE)
Trigonometric relations in the unit circle
(0.6,0.7)
√U1
V2 = √U1 sin(Θ)
Θ
V1 = √U1 cos(Θ)
V2
V1
-1.0
-0.5
0.0
0.5
1.0

Checking the Box-Muller algorithm



Exercise (Box-Muller implementation)

Questions :

1. *Implement a Gaussian random number generator in Python and in R based on the Box-Muller algorithm.*
2. *Illustrate graphically in R the distribution of your generator when simulating a standard Gaussian variable and a Gaussian variable with mean=2 and stddev=2.*
3. *Compare your results with the in-built generators both in R and in Python.*

Multivariate Gaussian random variables

A multivariate random variable of dimension m generates a vector $\mathbf{x}$ of $m \geqslant 1$ random numbers. We are interested here in the special case of multivariate Gaussian variables being defined by the multideminsional Gaussian density function :

$$\mathcal{N}(\mathbf{x}|\mu, \mathbf{\Sigma}) \;=\; \frac{1}{(2\pi)^{m/2}\,\det(\mathbf{\Sigma})^{1/2}} \exp\Big[-\frac{1}{2}(\mathbf{x}-\mu)\cdot\mathbf{\Sigma}^{-1}\cdot(\mathbf{x}-\mu)\Big]$$

where the parameter μ is a vector containing the multivariate mean, and the parameter $\mathbf{\Sigma}$, a symmetrical, positive-definite matrix, is the distribution's covariance.
In case $m = 1$ we recover the unidimensional formula seen before.

Simulating a multivariate Gaussian variable

In case of $m = 1$, we may easily generate a random number x from a $\mathcal{N}(\mu, \sigma)$ law by drawing a standard Gaussian number z from $\mathcal{N}(0, 1)$ and applying the transform :

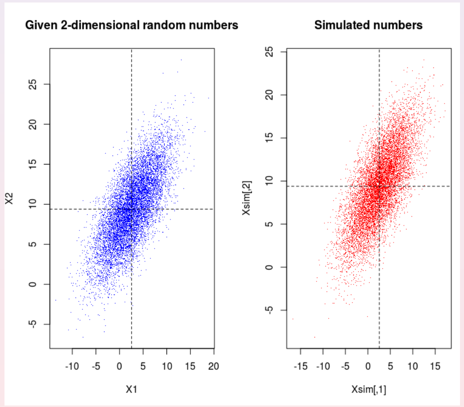
$$x \;=\; \sigma \cdot z + \mu$$

In the general case $m > 1$, we first draw a random vector $\mathbf{y}$ of dimension m from independent standard $\mathcal{N}(0, 1)$ generators.
If $\mathbf{LL}^t$ is the Choleski decomposition of the given covaraince $\mathbf{\Sigma}$, where $\mathbf{L}$ is the "*square root*" of $\mathbf{\Sigma}$ (the multivariate standard deviation $\sqrt{\mathbf{\Sigma}}$), we obtain a random vector $\mathbf{x}$ from the $\mathcal{N}(\mu, \mathbf{\Sigma})$ law in a similar way :

$$\mathbf{x} \;=\; \mathbf{Ly} + \mu$$

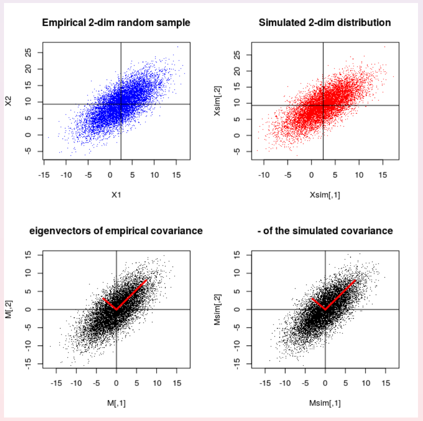
Simulating a multivariate Gaussian in R

```
par(mfrow=c(1,2))
nSim = 10^4
X1 = rnorm(nSim,2.5,4)
X2 = 0.75*X1+1.5*rnorm(nSim,5,2)
X = cbind(X1,X2)
plot(X,pch=".",col="blue")
abline(v=mean(X1),h=mean(X2))
L = chol(cov(X))
#         X1       X2
# X1 4.054996 3.017995
# X2 0.000000 2.999869
Xsim = cbind(rep(0,nSim),
             rep(0,nSim))
for (s in 1:nSim) {
  ts = t(rnorm(2)) %*% L
       + t(c(mean(X1),
             mean(X2)))
  Xsim[s,1] = ts[1]
  Xsim[s,2] = ts[2]}
plot(Xsim,pch=".",col="red")
abline(v=mean(Xsim[,1]),h=mean(Xsim[,2]))
```



Simulating a multivariate Gaussian in R

```
M=cbind(X1-mean(X1),z-mean(X2))
plot(M,ylim=c(-15,15),xlim=c(-17,17),pch=".")
abline(v=0,h=0)
eigCoV=eigen(cov(X))
B=diag(2*sqrt(eigCoV$values))
     %*% t(eigCoV$vectors)
ax1=rbind(B[1,],c(0,0))
ax2=rbind(B[2,],c(0,0))
lines(ax1,col="red",lwd=3)
lines(ax2,col="red",lwd=3)
Msim=cbind(Xsim[,1]-mean(Xsim[,1]),
      Xsim[,2]-mean(Xsim[,2]))
plot(Msim,ylim=c(-15,15),
      xlim=c(-17,17),pch=".")
abline(v=0,h=0)
eigCoVs = eigen(cov(Xsim))
Bx = diag(2*sqrt(eigCoVs$values))
     %*%t(eigCoVs$vectors)
ax1 = rbind(Bx[1,],c(0,0))
ax2 = rbind(Bx[2,],c(0,0))
lines(ax1,col="red",lwd=3)
lines(ax2,col="red",lwd=3)
```



Exercise

The above simulation procedure, using the empirical mean and covariance, does work well in principle only for multivariate Gaussian variables. Indeed all linear combinations of Gaussians are themselves again normally distributed and completely defined by their mean and covariance structure.
But the procedure does not usually work well for non Gaussian multivariate random variables.

Question :

Try the above simulation procedure with different other types of continuous random variables (uniform, triangular, exponential, Cauchy, Beta, etc) in order to find an appropriate example that illustrates well the potential failure of this simulation procedure.