

Simulación

Métodos particulares para generación de variables aleatorias.

Jorge de la Vega Góngora

Departamento de Estadística,
Instituto Tecnológico Autónomo de México

Semana 6



Métodos particulares

Distribución Normal

- Probablemente, una de las más usadas (y abusadas) distribuciones de probabilidad es la distribución Normal con media μ y varianza σ^2 :

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

- Una variable normal estándar $Z \sim \mathcal{N}(0, 1)$, se puede transformar fácilmente: $X = \mu + \sigma Z$. Por eso nos podemos concentrar en generar normales estándar.

Métodos basados en el Teorema del Límite Central

Un método histórico para generar distribuciones normales está basado en el Teorema del Límite Central (TLC). Sea U_1, U_2, \dots, U_k una muestra aleatoria uniforme est醤dar. En este caso, $E(U_i) = 1/2$ y $\text{Var}(U_i) = 1/12$. Entonces por el TLC:

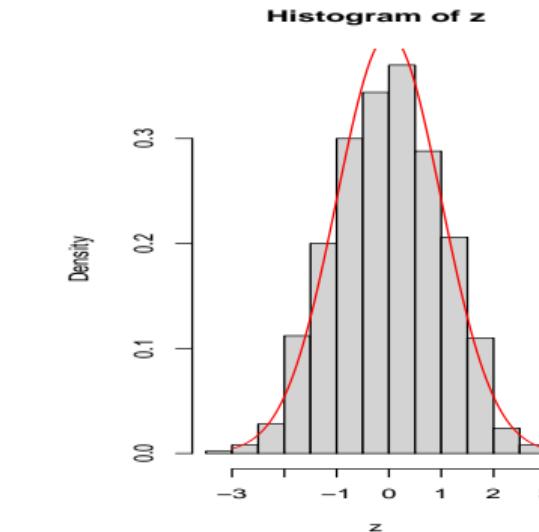
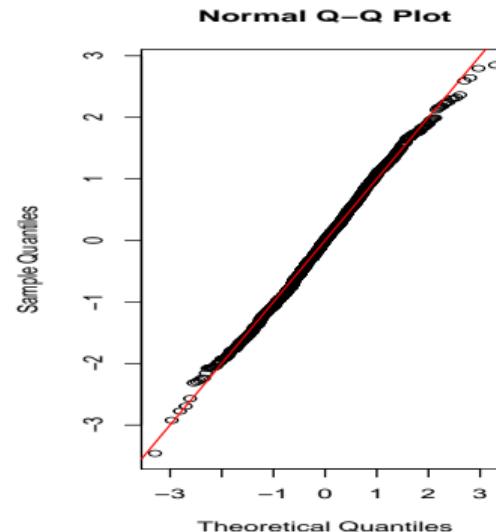
$$Z = \frac{\sum_{i=1}^k U_i - k/2}{\sqrt{k/12}} \xrightarrow{k \rightarrow \infty} \mathcal{N}(0, 1)$$

Si $k = 12$ la ecuación se simplifica. ¿Qué tan bueno es el método? Esto depende de k , sin embargo, no importa que tan grande sea k , el método sólo es aproximado.

Ejemplo

A continuación generamos una muestra de $n = 1000$ números aleatorios con el método propuesto.
¿Detectan algún problema?

```
n <- 1000; k <-12;  
u <- runif(k*n) #se requieren 12 observaciones para cada una de muestra, que es de tamaño n=1000  
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)
```



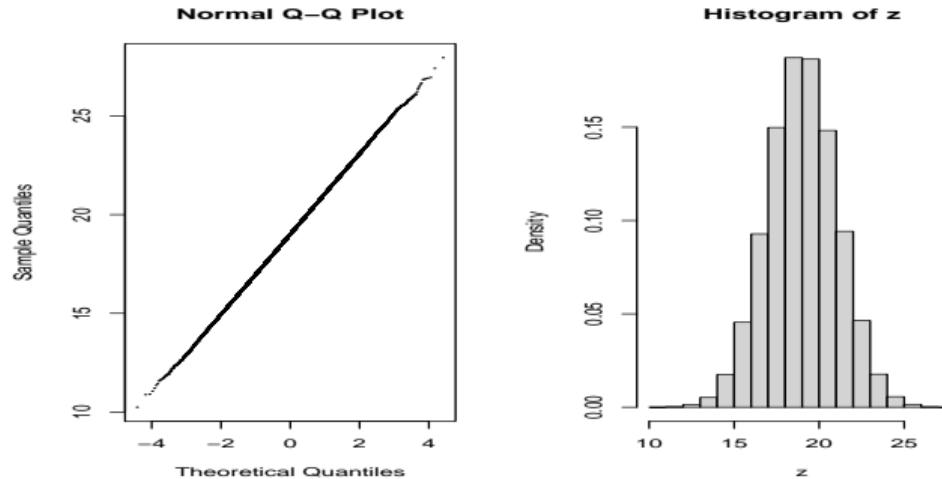
Ejemplo (cont.)

Si repetimos el ejercicio con una k mayor, vemos que el ajuste mejora, pero el tiempo de ejecución es mayor también

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

```
user    system elapsed  
0.250    0.020   0.271
```

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

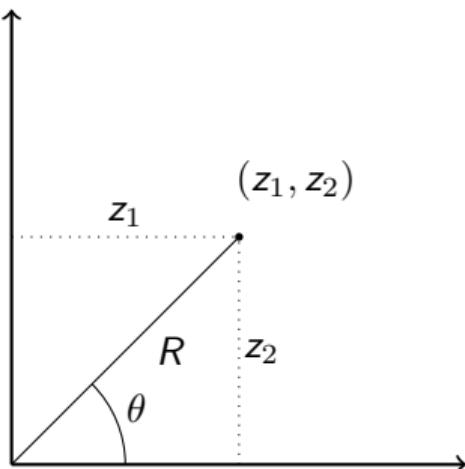


```
user    system elapsed  
0.681    0.000   0.681
```

Método de Box-Müller (1958)

El método se basa en la transformación de coordenadas polares a cartesianas:

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



Supongan que $z_1 \perp\!\!\!\perp z_2$. Entonces

- $z_1^2 + z_2^2 \sim \chi^2_{(2)} \equiv \mathcal{G}(2/2, 2) \equiv \exp(2)$
- La distancia
 $d((z_1, z_2), (0, 0)) = \|(z_1, z_2)\| = \sqrt{z_1^2 + z_2^2} = R$
- $\theta \sim \mathcal{U}(0, 2\pi)$ por la simetría de los contornos de la densidad bivariada.

En términos de z_1 y z_2 :

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

Usando u_1 para generar R y u_2 para generar θ , $R = \sqrt{-2 \log u_1}$ (ya que $R^2 \sim \exp(2)$) y $\theta = 2\pi u_2$. Por lo tanto

Transformación de Box-Müller

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

$$\begin{aligned} z_1 &= \sqrt{-2 \log u_1} \cos(2\pi u_2) \\ z_2 &= \sqrt{-2 \log u_1} \sin(2\pi u_2) \end{aligned}$$

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

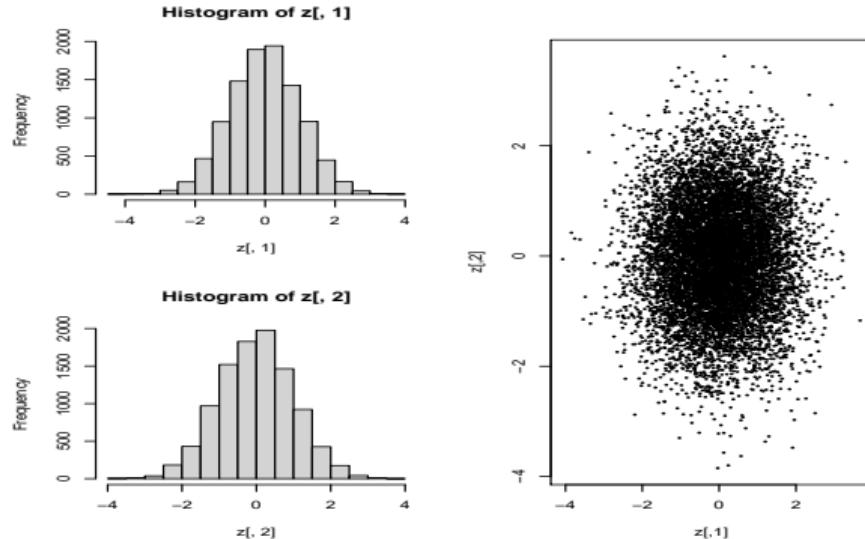
Ejemplo

En las siguientes gráficas se muestran las marginales y la conjunta

```
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
0.111    0.000    0.111
```

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



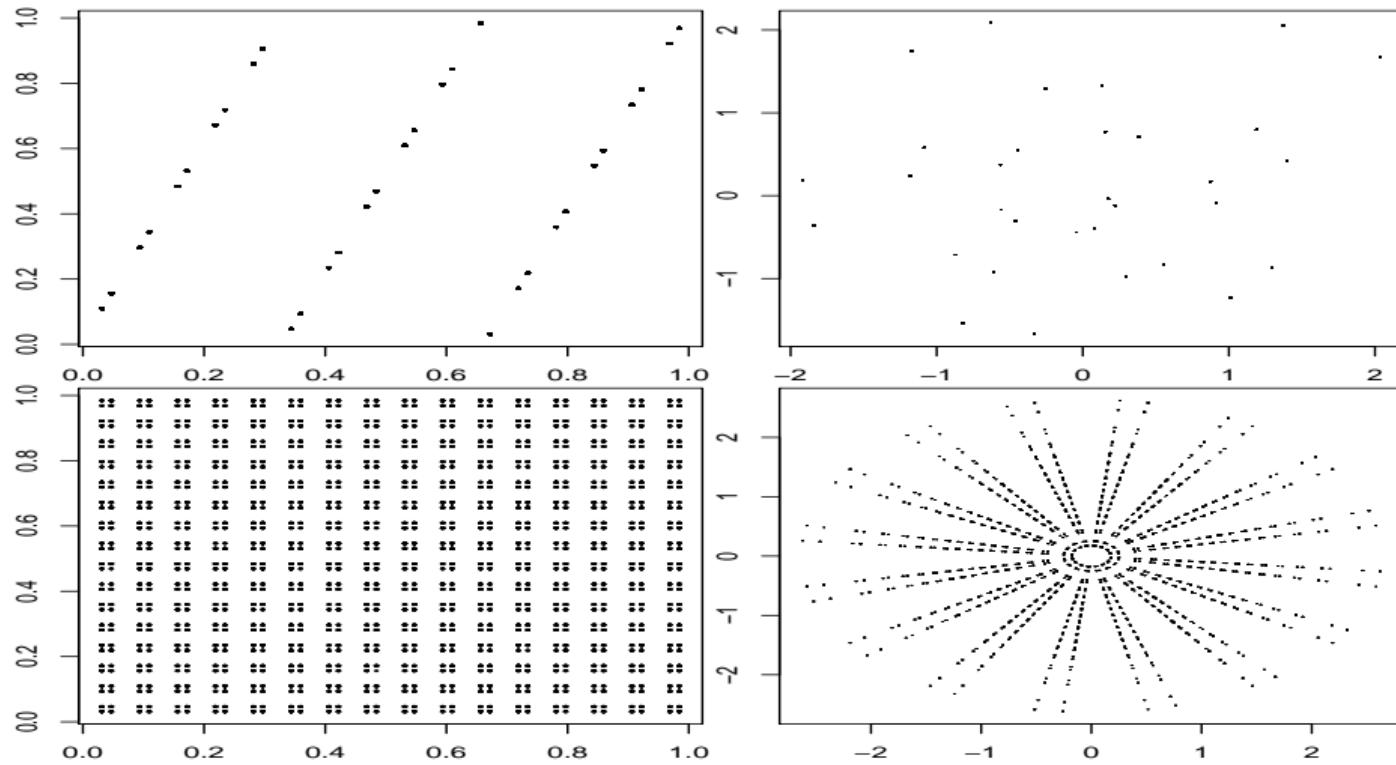
Comentarios al Método de Box-Müller

- El método de Box y Müller es muy costoso computacionalmente. Requiere el cálculo de varias funciones complejas: $\sqrt{}$, \log , \sin , \cos . Sin embargo, es muy fácil de implementar en cualquier computadora, usando C, Fortran, Java, Python, R, etc.
- La calidad de los normales dependerán de los números uniformes que se utilicen. En el siguiente ejemplo, usaremos un generador congruencial malo (RANDU).
- Algunos problemas se pueden resolver permutando los números uniformes antes de usarlos.

Ejemplo |

```
seed <- 10
RANDU <- function() {seed <- (3 * seed+1) %% (64); seed/(64)} #generador RANDU
u <- NULL; n <- 10000
for(i in 1:n) u <- c(u,RANDU()) #genera muestra de n uniformes RANDU
R <- sqrt(-2*log(u[1:(n-1)])) #Ahora obten normales con Box-Muller
th <- u[2:n]
z1 <- R*cos(2*pi*th); z2 <- R*sin(2*pi*th)
par(mfrow=c(2,2), oma = c(1,1,1,1) + 0.1, mar=c(1,1,1,1) + 0.1)
plot(u[1:(n-1)],u[2:n],pch=16,cex=0.4)
plot(z1,z2,pch=16,cex=0.2)
#Ahora permutamos para mejorar la cobertura.
v <- sample(u); R <- sqrt(-2*log(v[1:(n-1)])); th <- v[2:n]
plot(v[1:(n-1)],v[2:n],pch=16,cex=0.4)
plot(R*cos(2*pi*th), R*sin(2*pi*th), pch=16, cex=0.2)
```

Ejemplo II



Método Polar (Marsaglia, 1964) I

El método polar se basa en la idea original de Box-Müller, pero elimina el cálculo de las funciones trigonométricas, generando un método más eficiente.

Método Polar

- ① Muestrea $u_1 \perp\!\!\!\perp u_2 \sim \mathcal{U}(0, 1)$.
- ② Definir: $V_1 = 2u_1 - 1$ y $V_2 = 2u_2 - 1$. Entonces $V_i \sim \mathcal{U}(-1, 1)$.
- ③ Si $S = V_1^2 + V_2^2 > 1$ se rechaza el punto. Regresa a (1). Si $V_1^2 + V_2^2 \leq 1$ transforma:

$$Z_1 = V_1 \sqrt{\frac{-2 \log S}{S}}, Z_2 = V_2 \sqrt{\frac{-2 \log S}{S}}$$

Método Polar (Marsaglia, 1964) II

Demostración.

La transformación a V_1 y V_2 y aplicar el método de aceptación-rechazo nos permite seleccionar puntos al azar en el disco unitario. Los pasos 1 y 2 del algoritmo, permiten esa selección.

Ahora, sea $S = V_1^2 + V_2^2$. Entonces:

$$P(S \leq s) = P(V_1^2 + V_2^2 \leq s) = \frac{\pi(\sqrt{s})^2}{\pi(1)^2} = s$$

Por lo tanto, $S \sim \mathcal{U}(0, 1)$. Por otra parte, del resultado de Box-Müller sabemos que

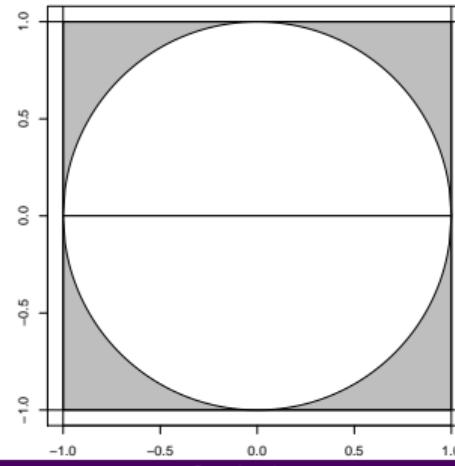
$$\begin{pmatrix} R\cos(\theta) \\ R\sin(\theta) \end{pmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_2)$$

Como S es uniforme, sabemos que $R^2 = -2 \log S \sim \exp(2)$. Por otra parte, $\sin(\theta) = V_2/\sqrt{S}$ y $\cos(\theta) = V_1/\sqrt{S}$. Sustituyendo en las ecuaciones de Box-Müller, se obtiene que

$$\begin{pmatrix} V_1 \sqrt{\frac{-2 \log S}{S}} \\ V_2 \sqrt{\frac{-2 \log S}{S}} \end{pmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_2)$$

Comentarios al Método Polar

- El tiempo ahorrado reemplazando funciones trigonométricas es importante comparado con el uso de aceptación y rechazo.
- Obtenemos nuevas uniformes sólo si caemos en la región sombrada, que tiene área $\frac{4-\pi}{4} = 1 - \pi/4 = 0.215$. Entonces se rechaza 21.5 % del tiempo.
- Entonces se requieren en promedio (basados en la distribución geométrica) $4/\pi = 1.27$ uniformes por normal: $X = \text{número de rechazos antes de aceptar}$. Entonces $X \sim \text{geo}(\pi/4)$ y por lo tanto $E(X) = 1/(\pi/4) = 4/\pi \approx 1.27$.



Distribución Beta

- La variable aleatoria $X \sim Be(\alpha, \beta)$ tiene función de densidad de probabilidad:

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

- Si partimos de que $Y_1 \perp\!\!\!\perp Y_2$, $Y_1 \sim \mathcal{G}(\alpha, 1)$, $Y_2 \sim \mathcal{G}(\beta, 1)$, respectivamente, entonces

$$X \sim \frac{Y_1}{Y_1 + Y_2} \sim Be(\alpha, \beta)$$

- Otro enfoque es cuando α y β son enteros, entonces podemos usar estadísticas de orden:

Generando $Be(m, n)$, cuando $m, n \in \mathbb{N}$

- ① Genera $m+n-1$ uniformes independientes $u_1, u_2, \dots, u_{n+m-1}$.
- ② Calcular $X = u_{(m)}$. Entonces $X \sim Be(m, n)$

- El número total de comparaciones necesarias para encontrar $u_{(m)}$ depende del algoritmo de ordenamiento seleccionado, pero usualmente lo mejor que se puede lograr es algo del orden $O((m+n-1) \log(m+n-1))$, por lo que el procedimiento no es eficiente para m y n grandes.

Distribución Poisson I

- Una variable Poisson N con media $\lambda > 0$ tiene función de masa de probabilidad:

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

- N se puede interpretar como el número de llegadas de un proceso Poisson en una unidad de tiempo.
- Los tiempos de interarribo en un proceso Poisson siguen una distribución exponencial con tasa λ . Entonces hay una relación entre los tiempos de interarribo y la distribución Poisson:

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

Lo anterior significa que en una unidad de tiempo, hubo n eventos. Entonces podemos usar la exponencial para generar variables aleatorias Poisson.

Variable aleatoria Poisson

- ① Definir $n = 0$, $P = 1$
- ② Generar $u_{n+1} \sim \mathcal{U}(0, 1)$ y definir $P \leftarrow Pu_{n+1}$
- ③ Si $P < e^{-\lambda}$, acepta $N = n$. Si no, rechaza n , incrementa a $n + 1$ y regresa a 2.

Distribución Poisson II

Demostración.

Si $T_i = -(1/\lambda) \log(u_i)$, por la relación mencionada

$$\sum_{i=1}^n \frac{-\log u_i}{\lambda} \leq 1 < \sum_{i=1}^{n+1} \frac{-\log u_i}{\lambda}$$

Multiplicando por $-\lambda$ y agrupando suma de logaritmos:

$$\log \prod_{i=1}^n u_i \geq -\lambda > \log \prod_{i=1}^{n+1} u_i$$

Por último, exponenciando ambos lados, obtenemos el resultado, notando que P es el producto acumulado requerido.

$$\prod_{i=1}^n u_i \geq e^{-\lambda} > \prod_{i=1}^{n+1} u_i$$



Eficiencia del procedimiento Poisson

- ¿Cuántos números aleatorios, en promedio serán requeridos para generar una variable Poisson?
- Si $N = n$, entonces se requieren $n + 1$ números, así que el número promedio está dado por

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

que es muy grande si la media de la Poisson es muy grande.

- Cuando λ es grande (digamos mayor a 15), podemos usar una aproximación normal que trabaja bien.

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

Entonces aproximadamente:

$$N \stackrel{a}{=} \lceil \lambda + \sqrt{\lambda} Z \rceil$$

donde $\lceil y \rceil$ es la función que redondea y al entero más cercano.

Ejemplo Poisson

- Genera una muestra de 100 variables Poisson con media $\lambda = 3$.

Solución.

```
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 2 1 3 3 1 4 2 3 3 5 0 5 0 1 5 4 2 5 3 1 1 5 1 3 4 6 0 2 0 3 1 5 1 3 1 5
[38] 4 5 1 1 7 2 2 3 4 1 2 1 3 2 1 2 1 3 2 3 2 1 3 1 5 3 3 3 2 3 3 1 1 3 1 3 6
[75] 3 4 0 6 5 2 6 3 4 3 3 5 1 2 3 5 3 5 6 2 1 4 1 4 4 2
```



- Genera 100 Poisson, con media $\lambda = 20$.

Solución.

```
z <- rnorm(100) #haciendo trampa, tomando normales
N <- ceiling(20 + sqrt(20)*z)
N
[1] 20 21 22 20 23 18 20 25 19 16 20 16 19 19 17 19 15 21 16 22 21 16 21 16 21
[26] 17 20 25 21 20 24 25 16 24 15 15 21 23 24 18 18 12 14 20 22 22 25 17 17 19
[51] 12 14 19 19 17 19 13 17 12 9 18 14 21 17 17 27 27 21 26 15 17 14 11 20 24
[76] 16 21 21 24 25 19 18 16 17 13 12 22 15 21 9 20 20 22 22 28 18 22 23 26 26
```



Distribuciones multivariadas

- La necesidad de generar vectores aleatorios es muy común en la práctica, pero hay nuevos problemas que se tienen que resolver.
- Dentro de las aplicaciones de vectores aleatorios, se encuentran los siguientes:
 - Tiempos de proceso para un producto en una serie de estaciones. Estos tiempos son dependientes entre sí.
 - Variables de control relevantes para un cliente en los módulos de pedido, caja y carga de una tienda.
 - Cantidad de k ítems que una fábrica demanda de un sistema multi-inventario.
 - Rendimientos de los instrumentos de un portafolio de inversión.

Problema general I

- La generación de vectores aleatorios es un tema mucho más complejo que el de variables aleatorias individuales. Por ejemplo:
 - Se pueden tener distribuciones conjuntas no normales con marginales normales
 - Marginales de una familia de distribuciones (por ejemplo, gammas) no necesariamente provienen de una distribución conjunta única
 - Estructuras de dependencia se pueden dar más allá de la correlación o covarianza entre las variables. Por ejemplo, un tipo de dependencia útil entre variables se puede dar en las colas de las distribuciones.
- Como métodos generales se puede utilizar el método de Aceptación-Rechazo en mayores dimensiones.
- Otro método general, podría ser condicionar algunas relaciones:
 - 1 $X_1 \sim F_1$
 - 2 $X_2 \sim F_2(\cdot|x_1)$
 - :
 - p $X_p \sim F_p(\cdot|X_1 = x_1, \dots, X_{p-1} = x_{p-1})$
- $p+1$ regresar el vector $X = (X_1, \dots, X_p)$

Problema general II

- Como ejemplo, considerar generar una muestra de una Cauchy multivariada:

$$f(\mathbf{x}) = \pi^{-\frac{p+1}{2}} \Gamma\left(\frac{p+1}{2}\right) (1 + \mathbf{x}'\mathbf{x})^{-\frac{p+1}{2}}, \quad \mathbf{x} \in \mathbb{R}^p$$

- Cada componente X_i de \mathbf{X} tiene distribución Cauchy univariada y la distribución condicional de $X_m | X_1 = x_1, \dots, X_{m-1} = x_{m-1}$ es

$$\frac{1 + \sum_{i=1}^{m-1} X_i}{\sqrt{m}} t_m \text{ donde } t_m \sim t_{(m)}$$

Algoritmo para distribución multinomial

La distribución multinomial es la del vector de dimensión k , $X = (X_1, \dots, X_k)$ donde cada componente corresponde a una de k categorías, y se realizan n ensayos. La distribución multinomial tiene la siguiente función de masa de probabilidad:

$$P(X_1 = x_1, X_2 = x_2, \dots, X_k = x_k) = \binom{n}{x_1, \dots, x_k} \prod_{i=1}^k p_i^{x_i}$$

Como se puede ver, es una generalización de una variable aleatoria binomial.

Algoritmo para multinomial

- ① Define $r_j = \sum_{l=1}^j p_l$, para obtener ($r_1 = p_1, r_2 = p_1 + p_2, \dots, r_k = 1$)
- ② Genera $u \sim \mathcal{U}(0, 1)$
- ③ Compara u con r_1, r_2, \dots hasta que $u > r_j$ y $u \leq r_{j+1}$
- ④ Entonces seleccionamos la $j + 1$ -ésima celda de la multinomial. Define $X = \epsilon_k$, donde ϵ_k es un vector un 1 en la entrada k y 0 en el resto.
- ⑤ La suma de los vectores $X = \sum_{i=1}^n \epsilon_i$ tiene la distribución deseada

Ejemplo multinomial, $n = 10, k = 4$

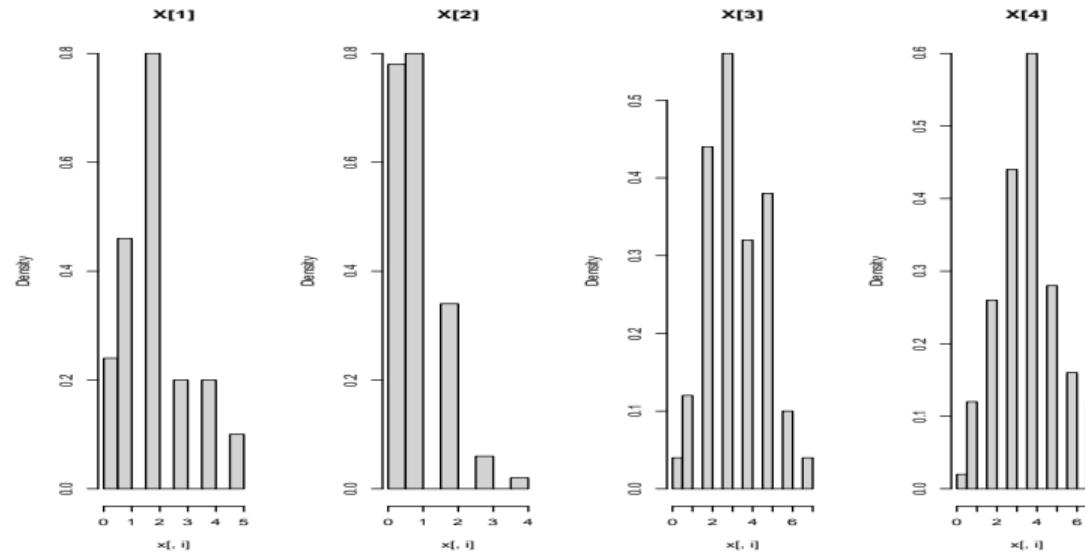
Consideremos una distribución multinomial con $p = (0.2, 0.1, 0.3, 0.4)$, $n = 10$, así que $k = 4$. Este ejercicio genera una muestra de tamaño $m = 100$ de un vector multinomial.

```
buscar <- function(x,cats){  
z <- NULL  
for (i in 1:cats) z[i] <- length(which(x==i)) #cuenta cuantas observaciones en cat i  
z}  
  
p <- c(0.2,0.1,0.3,0.4)  
n <- 10  
rmultinomial <- function(m,n,p){  
r <- cumsum(p)  
X <- matrix(0,nrow=m,ncol=length(p))  
for(i in 1:m){  
u <- runif(n)  
celdas <- findInterval(u,r) + 1 #esta función compara cada u con el vector r  
X[i,] <- buscar(celdas,length(p)) #conteo de categorías.  
}  
X  
}  
x <- rmultinomial(100,n,p)  
head(x)  
  
[,1] [,2] [,3] [,4]  
[1,]    2     1     2     5  
[2,]    2     2     4     2  
[3,]    4     2     0     4  
[4,]    2     1     5     2  
[5,]    2     0     3     5  
[6,]    2     1     3     4
```

Comentarios al ejercicio anterior

- La función `buscar` cuenta el número de observaciones que cayeron en cada una de las categorías
- cada una de las marginales del vector X , X_i es una binomial con parámetros n y p_i .

```
par(mfcol=c(1,4)) for (i in 1:length(p)) hist(x[,i], breaks = 10, prob = T, main = paste0('X[,i]'))
```



Normal Multivariada

- Un vector aleatorio $\mathbf{X} = (X_1, \dots, X_d)$ tiene una distribución normal multivariada que se denota como $\mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ si la densidad de \mathbf{X} es:

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right), \quad \mathbf{x} \in \mathbb{R}^d$$

donde $\boldsymbol{\mu}$ es el vector de medias, y $\boldsymbol{\Sigma}$ es una matriz simétrica definida positiva de $d \times d$.

- Para generar una normal multivariada, se siguen dos pasos:
 - Se genera un vector $\mathbf{Z} = (Z_1, \dots, Z_d)$, donde Z_1, \dots, Z_d son normales estándares independientes.
 - Se transforma \mathbf{Z} para que tenga el vector de medias y matriz de covarianzas dados. Esta transformación requiere factorizar la matriz de covarianzas $\boldsymbol{\Sigma}$.
- Para estandarizar, necesitamos descomponer la matriz de covarianzas en su “raíz cuadrada”: tenemos que encontrar $\mathbf{B} \ni \mathbf{B}\mathbf{B}' = \boldsymbol{\Sigma}$. Entonces

$$\mathbf{Z} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{I}_p) \implies \mathbf{X} = \boldsymbol{\mu} + \mathbf{B}\mathbf{Z} \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Métodos para descomponer Σ

- La descomposición de Σ se puede conseguir por tres métodos (con su función en R:
 - Método de descomposición espectral (eigenvalores): eigen.
 - Factorización de Cholesky chol.
 - descomposición en valor singular svd.
- Usualmente, no se aplica una transformación lineal a cada vector aleatorio en la muestra de manera individual; más bien se hace de golpe sobre todos los datos:
 - Supongan que $Z = (Z_{ij})$ es una matriz de $n \times d$ donde Z_{ij} son iid $\mathcal{N}(0, 1)$. Los renglones de esta matriz son las observaciones de la distribución normal multivariada. La transformación que se aplica a la matriz de datos es

$$\mathbf{X} = \mathbf{Z}\mathbf{Q} + \mathbf{e}\boldsymbol{\mu}'$$

donde $\mathbf{Q}'\mathbf{Q} = \Sigma$, e es un vector de unos.

- Los renglones de \mathbf{X} son las observaciones aleatorias que tienen la media y varianza multivariada deseada.

Algoritmo para generar muestras de normal multivariada

Para generar una muestra de tamaño n de $\mathcal{N}_d(\mu, \Sigma)$:

- ① Generar una matriz Z de $n \times d$ con nd normales estándar independientes (n vectores aleatorios en \mathbb{R}^d).
- ② Calcular una factorización $\Sigma = Q'Q$.
- ③ Aplicar la transformación $X = ZQ + e\mu'$
- ④ X es una matriz con la muestra deseada en cada renglón.

Ejemplo |

Obtener una muestra de un vector normal multivariado de orden 3: $\mathbf{X} = (X_1, X_2, X_3) \sim \mathcal{N}_3(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, con

$$\boldsymbol{\mu} = (1, 2, 3)' \text{ y } \boldsymbol{\Sigma} = \begin{pmatrix} 2.5 & 0.75 & 0.175 \\ 0.75 & 0.7 & 0.135 \\ 0.175 & 0.135 & 0.43 \end{pmatrix}.$$

Con el método de descomposición espectral:

```
rmvn.eigen <- function(n,mu,Sigma){  
  d <- length(mu)  
  ev <- eigen(Sigma, symmetric=T)  
  lambda <- ev$values; V <- ev$vectors  
  Q <- V %*% diag(sqrt(lambda)) %*% t(V)  
  Z <- matrix(rnorm(n*d),nrow=n, ncol=d)  
  X <- Z %*% Q + matrix(mu,n,d,byrow=T)  
  X  
}
```

Con el método de descomposición de Cholesky

```
rmvn.chol <- function(n,mu,Sigma){  
  d <- length(mu)  
  Q <- chol(Sigma)  
  Z <- matrix(rnorm(n*d),nrow=n)  
  X <- Z %*% Q + matrix(mu,n,d,byrow=T)  
  X  
}
```

Ejemplo II

Con el método de descomposición en valor singular.

```
rmvn.svd <- function(n,mu,Sigma){  
  d <- length(mu)  
  S <- svd(Sigma)  
  Q <- S$u %*% diag(sqrt(S$d)) %*% t(S$v)  
  Z <- matrix(rnorm(n*d),nrow=n, ncol=d)  
  X <- Z %*% Q + matrix(mu,n,d,byrow=T)  
  X  
}
```

Ejemplo

Generamos la muestra con las funciones creadas

```
Sigma <- matrix(c(2.5, 0.75, 0.175,      0.75, 0.7, 0.135, 0.175, 0.135, 0.43), byrow=T, nrow=3)
mu <- c(1:3)
set.seed(1000)
system.time(for(i in 1:100) X1 <- rmvn.eigen(1000, mu, Sigma))

  user  system elapsed
0.134    0.044   0.055

set.seed(1000); system.time(for(i in 1:100) X2 <- rmvn.svd(1000, mu, Sigma))

  user  system elapsed
0.058    0.000   0.065

set.seed(1000); system.time(for(i in 1:100) X3 <- rmvn.chol(1000, mu, Sigma))

  user  system elapsed
0.044    0.000   0.046

head(X1, 2); head(X2, 2); head(X3, 2)

      [,1]      [,2]      [,3]
[1,] 1.261355 2.962566 3.467948
[2,] 2.427646 2.486649 3.792581
      [,1]      [,2]      [,3]
[1,] 1.261355 2.962566 3.467948
[2,] 2.427646 2.486649 3.792581
      [,1]      [,2]      [,3]
[1,] 0.8188025 2.800563 3.505366
[2,] 2.3273613 2.512565 3.821431
```

Distribución uniforme en \mathbb{S}^d

- La d -Esfera es el conjunto

$$\mathbb{S}^d = \{\mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{x}\| = \sqrt{\mathbf{x}'\mathbf{x}} = 1\}$$

- Vectores aleatorios uniformes distribuidos en \mathbb{S}^d tienen direcciones distribuidas uniformes.
- Utilizamos la siguiente propiedad de normalidad:

Distribución en \mathbb{S}^d

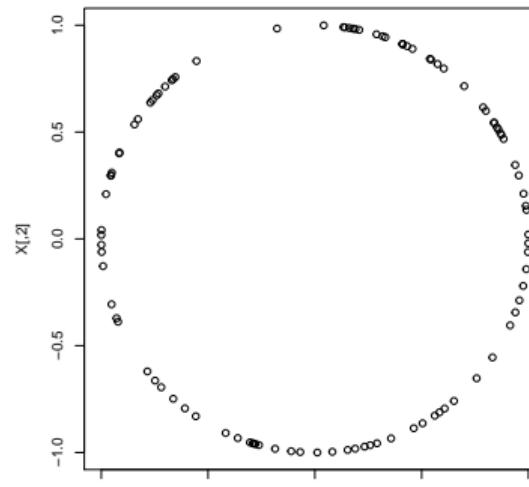
Si X_1, \dots, X_d son iid $\mathcal{N}(0, 1)$, entonces \mathbf{U} se distribuye uniforme en \mathbb{S}^d , donde cada componente es de la forma:

$$U_i = \frac{X_j}{\|\mathbf{X}\|}, \quad i = 1, \dots, d$$

Ejemplo

Función para generar puntos en la esfera unitaria \mathbb{S}^d :

```
runif.esfera <- function(n,d){  
  M <- matrix(rnorm(n*d),nrow=n)  
  L <- apply(M,1,function(x){sqrt(sum(x*x))})  
  D <- diag(1/L)  
  return(D %*% M)  
}  
X <- runif.esfera(100,2)  
par(pty="s")  
plot(X)
```



NORTA (Normal to Anything) I

- El objetivo es generar un vector aleatorio $X = (X_1, \dots, X_p)$ con las siguientes propiedades:
 - $X_i \sim F_i, \quad i = 1, \dots, p$
 - $\text{Cor}(X_i) = \Sigma_X$ dada
- Este método representa a X como una transformación de un vector normal multivariado (Z_1, \dots, Z_n) con matriz de correlación dada por Σ_Z :

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

donde $U_i = \Phi(Z_i)$. Entonces el vector Z es transformado a un vector U de variables uniformes.

- La matriz Σ_Z se escoge siguiendo criterios numéricos descritos en ([Cario & Nelson 1997](#)). Entonces el problema se reduce a resolver $p(p - 1)/2$ problemas independientes: para cada $i \neq j$, encontrar el valor $\rho_Z(i, j)$ para el cual existe una función c_{ij} tal que $c_{ij}(\rho_Z(i, j)) = \rho_X(i, j)$. En general, se requiere hacer una búsqueda numérica para encontrar las funciones c_{ij} .
- Esta es una aplicación directa del concepto de *cópula* que veremos más adelante.

NORTA (Normal to Anything) II

Algoritmo NORTA

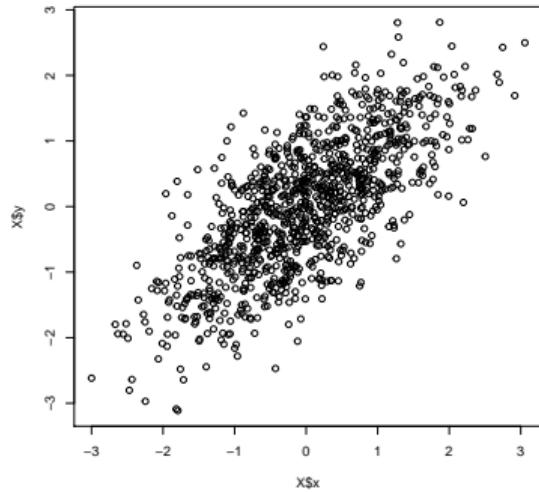
- ① Obtener la factorización \mathbf{M} de Σ_Z tal que $\mathbf{M}\mathbf{M}' = \Sigma_Z$.
- ② Generar $\mathbf{W} = (W_1, \dots, W_p)'$ cuyos elementos son iid normales estándar.
- ③ Hacer $Z \leftarrow \mathbf{MW}$
- ④ Obtener \mathbf{X} , donde $X_i \leftarrow F_{\mathbf{X}}^{-1}[\Phi(Z_i)]$, para $i = 1, \dots, p$.

Ejemplo NORTA I

- 1 Primero generamos dos normales con correlación 0.7

```
library(MASS)
n <- 1000
mu <- rep(0,4)
Sigma <- matrix(0.7, nrow = 4, ncol = 4) + diag(4)*0.3
z <- mvrnorm(n = n, mu = mu, Sigma = Sigma)
X <- data.frame(x = z[,1], y= z[,2])
plot(X$x,X$y)
```

Ejemplo NORTA II

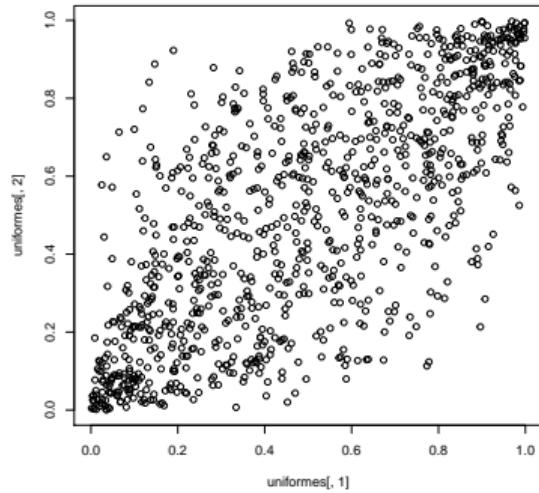


Ejemplo NORTA III

- ② Usando el método NORTA para generar uniformes:

```
uniformes <- qunif(pnorm(z))
cor(uniformes) # no tiene la correlación exacta pero cercana
[,1]      [,2]      [,3]      [,4]
[1,] 1.0000000 0.7086322 0.6842798 0.6952540
[2,] 0.7086322 1.0000000 0.7198221 0.6807372
[3,] 0.6842798 0.7198221 1.0000000 0.6771189
[4,] 0.6952540 0.6807372 0.6771189 1.0000000
plot(uniformes[,1],uniformes[,2])
```

Ejemplo NORTA IV



Ejemplo NORTA V

- ③ Para generar datos, por ejemplo de una distribución Poisson

```
poisson5 <- qpois(pnorm(z), lambda = 5)
cor(poisson5)

[,1]      [,2]      [,3]      [,4]
[1,] 1.0000000 0.7098691 0.6904797 0.6935174
[2,] 0.7098691 1.0000000 0.7102541 0.6706911
[3,] 0.6904797 0.7102541 1.0000000 0.6730499
[4,] 0.6935174 0.6706911 0.6730499 1.0000000

plot(poisson5[,1],poisson5[,2])
```

Ejemplo NORTA VI

