CENTRO DE INVESTIGACIÓN EN MATEMÁTICAS

El Lenguaje de Programación R

Rogelio Ramos Quiroga
(CIMAT)

3^{er} Verano de la Probabilidad y Estadística, 27 junio – 24 de julio de 2010

- R es un lenguaje para el cómputo estadístico, con muy buenas herramientas para la producción de gráficas de gran calidad; además ofrece un ambiente de programación con los elementos estándar de un lenguaje: Ciclos, acciones condicionales, estructuras de lectura y salida de datos, funciones con argumentos vectoriales, así como una gran cantidad de funciones y paquetes que le añaden funcionalidad.
- Si bien, R no cuenta con una interfaz de menús como Minitab, Statistica, JMP u otros paquetes estadísticos; creemos que es relativamente fácil, con una curva rápida de aprendizaje.
- R es la implementación GNU del lenguaje S. Puede obtenerse libremente de `http://cran.r-project.org`
- S es un lenguaje premiado, en 1998, por la ACM (Association of Computer Machinery) y fue desarrollado en los Laboratorios Bell en los 80's por un equipo lidereado por John Chambers.
- R incluye varios manuales y, en particular, recomendamos la lectura de *An Introduction to* R. En el sitio de distribución de R se pueden encontrar una gran cantidad de notas, tutoriales, etc. que también recomendamos explorar.
- Precisamente por la razón dada en el punto anterior, las presentes notas no pretenden ser un manual de R. El objetivo es proporcionar una colección de ejemplos y ejercicios que, esperamos, nos serviran para aprender el lenguaje.
- También es importante puntualizar que, si bien usamos muchos ejemplos de Estadística y Análisis Numérico, no pretendemos implicar que estamos ofreciendo las mejores técnicas de estas áreas (ni las mejores implementaciones); simplemente usamos aquellas que sean sencillas y que nos sirvan para ilustrar técnicas de programación y cálculo en R.

En términos generales, el curso cubrirá:

- Sesión 1: introducción a R, método de newton, gráficas
- Prácticas 1: operaciones básicas, funciones, teorema central del límite
- Sesión 2: gauss-seidel, regla de simpson, runge-kutta, estimador de parzen
- Prácticas 2: prueba de hipótesis, matrices, recursividad, potencia de una prueba, paquete tcltk, caminatas aleatorias, máxima verosimilitud
- Sesión 3: k-medias, escalamiento multidimensional, recocido simulado
- Prácticas 3: funciones, gráficas, prueba mann-whitney, mínimos cuadrados, splines
- Sesión 4: kolmogorov-smirnov, aceptación de lotes por muestreo, gráficos de control, interpolador de lagrange
- Prácticas 4: lilliefors, newton-raphson, regresión no lineal, integración monte carlo
- Sesión 5: generación de números aleatorios, robbins-monro, valores propios
- Prácticas 5: intervalos de confianza, gráficas, análisis exploratorio de datos, sliders, algoritmo em

LUNES

Notas Sesión 1

En nuestra primera sesión con R explicaremos los elementos básicos:

- Instalación de R
- Como entrar y salir del software
- Como entrar al editor de programas (scripts)
- Como ejecutar instrucciones
- Los elementos básicos: átomos, vectores, matrices, arreglos, data frames, listas
- Operaciones básicas
- Lectura de datos de archivos externos
- Introducción a gráficas
- Elaboración de funciones
- Introducción a elementos de programación

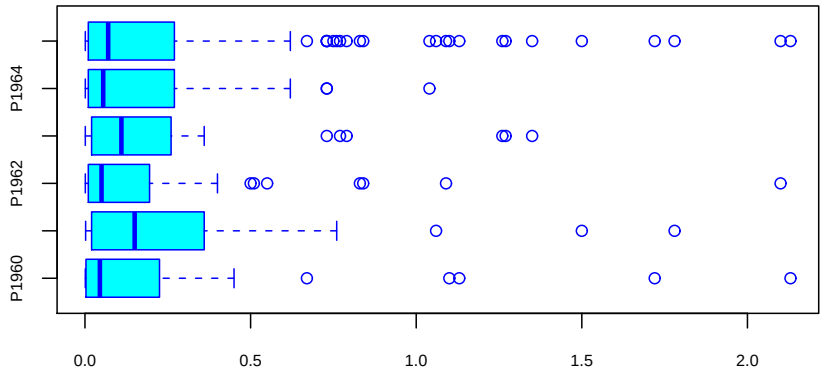
Procederemos reproduciendo y explicando las siguientes instrucciones:

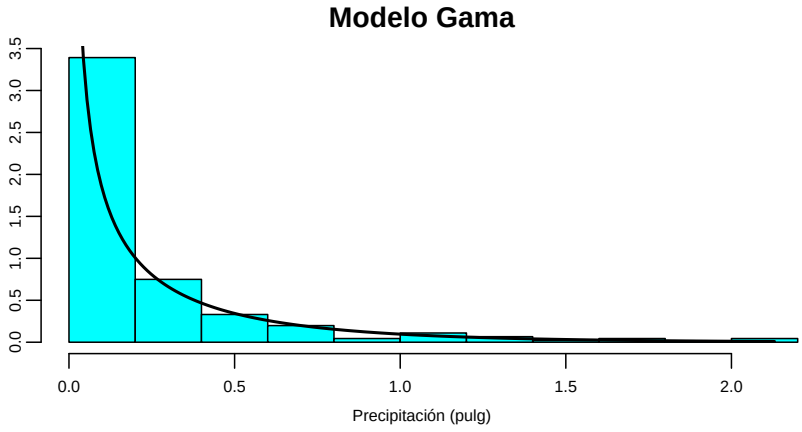
```
# Sesion 1:  Lunes 28 junio 2010
# Un vistazo general a R.
# R viene acompanado de varios manuales, recomendamos:
# Introduction to R, de Venables y Smith.


########################################
# Parte I.- Objetos basicos
# Algunos objetos: numeros, vectores, matrices, data frames,
#                  arreglos, listas

a  <- 2
b  <- c(4,8,0,16)                    # vector
c  <- 1:14                            # vector
d  <- matrix( 1:10, nrow=2, byrow=T ) # matriz
d  <- matrix( 1:10, ncol=2, byrow=T )
d  <- matrix( 1:100000, ncol=5 )

dim(d)                                # dimensiones de matriz d
remove(d)                             # remueve un objeto
ls()                                  # enlista objetos del directorio
remove(a,c)                           # remueve varios
remove( list=ls() )                   # remueve todos

a  <- seq( from=5, to=23, by=.5 )     # una secuencia (o vector)
a  <- seq( from=5, to=23, length=100 )
b  <- rnorm(100)                      # generacion de variables normales
b  <- runif(100)                      # generacion de uniformes
hist(b)                               # un histograma simple

c  <- cbind(a,b)                      # matriz
d  <- rbind(a,b)                      # matriz
dt <- t(d)                            # transpuesta
```

```
e  <- cbind(c,dt)
e  <- e[1:20,]                        # para poner nombre a renglones y cols
rownames(e) <- NULL                   # de una matriz

colnames(e) <- c("nom1","nom2","nom3","nom4")
rownames(e) <- paste("obs",1:20, sep="")
dimnames(e) <- list( paste("obs",1:20, sep=""),c("nom1","nom2","nom3","nom4"))

f  <- letters[1:20]                   # vector de caracteres
g  <- LETTER[1:20]
h  <- data.frame(e,f,g)               # data frame
dim(h)
names(h) <- c("var1","var2","var3","var4","var5","var6")
h
remove(h)

data(iris)                            # datos usados por Fisher (discriminante)
iris                                  # matriz
# extraccion de elementos
iris[3,4]
iris[,2]
iris[67,]
iris[67,c(2,4)]
iris[,5]
iris[1:50,1:4]

aa <- array(1:24,dim=c(2,3,4))        # arreglo
bb <- array(0,dim=c(50,4,3))          # arreglo
bb[,,1] <- as.matrix( iris[iris[,5]=="setosa",1:4] )
bb[,,2] <- as.matrix( iris[iris[,5]=="versicolor",1:4] )
bb[,,3] <- as.matrix( iris[iris[,5]=="virginica",1:4] )

# algunos resumenes numericos
aa <- rnorm(100,mean=20,sd=3)         # generamos 100 normales
summary(aa)                           # resumen numerico
min(aa)                               # minimo
mean(aa)                              # media
quantile(aa,probs=c(.25,.5,.75))      # algunos cuantiles
median(aa)                            # mediana
max(aa)                               # mximo
range(aa)                             # rango: min y max
sd(aa)                                # desviacion estandar
var(aa)                               # varianza
sum( (aa-mean(aa))^2 )/(length(aa)-1) # varianza

# listas
bb <- list( sumario=summary(aa), media=mean(aa),
            letras=letters[8:15], basura=runif(200) )
bb$letras                             # como acceder a los elementos de lista
bb[[3]]                               # o, equivalentemente
```

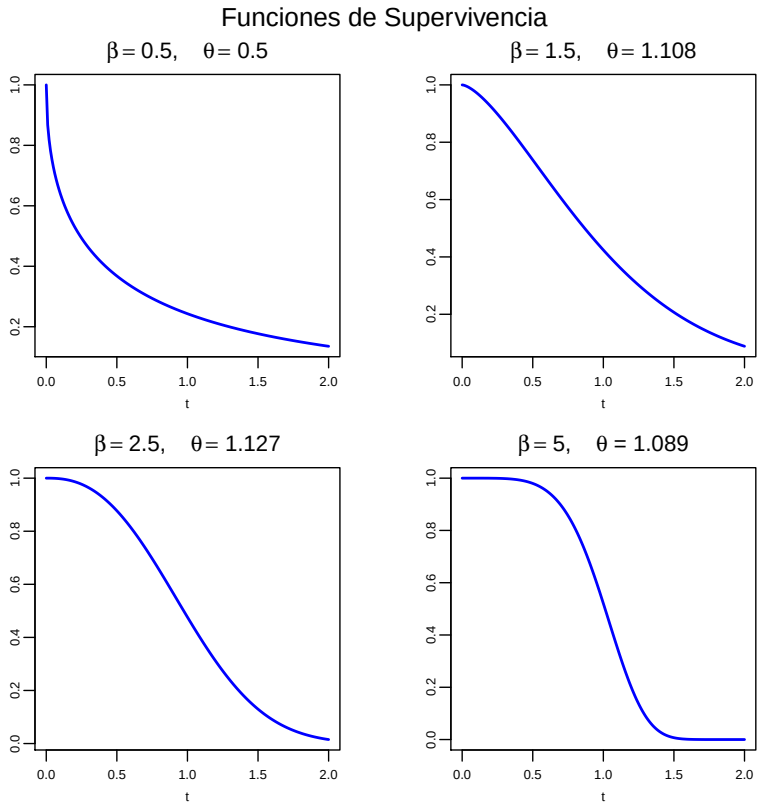
```
#####################################
# Parte II.- Operaciones basicas

a  <- 1:20
b  <- 31:50
a+b                                  # operaciones con vectores
a-b
a/b
a*b
a^b
a %% 2
a %% 3
cbind(a, a%%3)
(a+b) %% 2
(a*b) %% 2

a  <- 1:20
b  <- 31:40
a+b                                  # reciclado ok
b  <- 30:40                          # lo hace pero te avisa
a+b

a  <- matrix( 1:8, ncol=2 )
b  <- matrix( 9:16, nrow=2, byrow=T )
a+b
a %*% b
b %*% a
# la mayoria de las funciones de R actuan elemento a elemento
a^2
log(a)

# algunas funciones utiles: which, ifelse
a  <- rnorm(100)
b  <- which(a>0)                     # cuales son positivos
c  <- a[b]                           # los extraemos
cc <- a[a>0]                         # o, equivalentemente
aa <- ifelse(a<5,1,0)                # si los elementos de a son < 5 ponemos 0, si no 1
# algunas funciones utiles: read.csv, scan
# tipicamente: read.csv para importar datos de Excel
#              scan para importar datos de archivos .txt

x <- rnorm(100)
y <- rnorm(100)
plot(x,y)
plot(x,y,xlab="Eje x",cex.lab=.7,cex.axis=.7,mgp=c(1.5,.5,0),pch=24 )
abline(h=0,col="red")
abline(v=0,col="blue")

dat    <- scan(
 "c:\\Documents and Settings\\My Documents\\lluvia.txt",na.strings="*")
dat    <- as.vector(na.omit(dat))

datos  <- read.csv(
 "c:\\Documents and Settings\\My Documents\\lluvia.csv",header=FALSE)
```

```
dat <- as.vector(na.omit(as.vector(as.matrix(datos))))  # igual que antes
xr  <- range(dat)
aa  <- list(P1960=datos[,1],P1961=datos[,2],P1962=datos[,3],
            P1963=datos[,4],P1964=datos[,5],Todos=dat)
par(mfrow=c(2,1),mar=c(3, 3, 2, 2))
boxplot(aa, horizontal=TRUE, xlab="Precipitacion (pulg)", border="blue",
        col="cyan",cex.axis=.7,cex.lab=.7, mgp=c(1.5,.5,0))

n    <- length(dat)           # 227
xb   <- mean(dat)             # 0.2244
xv   <- (n-1)*var(dat)/n      # 0.1332
lam  <- xb/xv                 # 1.6842
alf  <- xb^2/xv               # 0.3779
hist(dat,prob=T,main="Modelo Gama", xlab="Precipitacion (pulg)",ylab="",
     col="cyan", xlim=xr,cex.axis=.7,cex.lab=.7, mgp=c(1.5,.5,0))
xx   <- seq(xr[1],xr[2],length=200)
lines(xx,dgamma(xx,shape=alf,rate=lam), lwd=2)
```



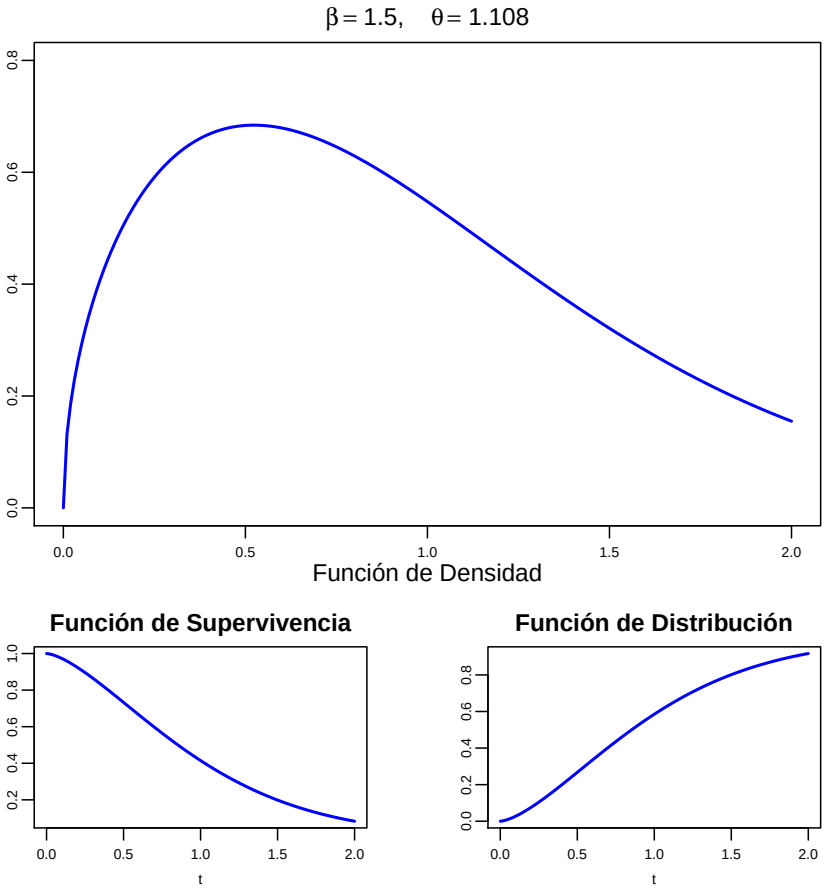


```
#####################################
# Parte III.- Graficas

# Distribucion Weibull
tt <- seq(0,2,length=200)
par(mfrow=c(2,2),mar=c(3, 3, 2, 2), oma=c(0,0,2,0))
b1 <- .5;  t1 <- 1/gamma(1+1/b1)
plot(tt, exp(-(tt/t1)^b1), xlim=c(0,2), cex.axis=.7, cex.lab=.7,
     mgp=c(1.5,.5,0), lwd=2, col="blue", type="l", xlab="t", ylab="",
     main= expression(paste(beta == .5, ",    ", theta == .5)))
b2 <- 1.5;  t2 <- 1/gamma(1+1/b2)
plot(tt, exp(-(tt/t2)^b2), xlim=c(0,2), cex.axis=.7, cex.lab=.7,
     mgp=c(1.5,.5,0), lwd=2, col="blue", type="l", xlab="t", ylab="",
     main= expression(paste(beta == 1.5, ",    ", theta == 1.108)))
b3 <- 2.5;  t3 <- 1/gamma(1+1/b3)
plot(tt, exp(-(tt/t3)^b3), xlim=c(0,2), cex.axis=.7, cex.lab=.7,
     mgp=c(1.5,.5,0), lwd=2, col="blue", type="l", xlab="t", ylab="",
     main= expression(paste(beta == 2.5, ",    ", theta == 1.127)))
b4 <- 5;  t4 <- 1/gamma(1+1/b4)
plot(tt, exp(-(tt/t4)^b4), xlim=c(0,2), cex.axis=.7, cex.lab=.7,
     mgp=c(1.5,.5,0), lwd=2, col="blue", type="l", xlab="t", ylab="",
     main=
 substitute(paste(beta == 5, ",    ", theta, " = ",t4),list(t4=round(t4,3))))
mtext("Funciones de Supervivencia", outer=TRUE, cex=1.2)

# (en esta grafica, notar en la ultima, el uso de substitute)
```



Uso de la función `layout()`:

```
layout( matrix(c(1,1,2,3),ncol=2,byrow=T), heights=c(2,1))
par(mar=c(3, 3, 2, 2))
b2 <- 1.5;  t2 <- 1/gamma(1+1/b4)
plot(tt, dweibull(tt,shape=b2, scale=t2), xlim=c(0,2), cex.axis=.7, cex.lab=1.2,
     mgp=c(1.5,.5,0), lwd=2, col="blue", type="l", xlab="Funcion de Densidad",
     ylab="", ylim=c(0,.8),main= expression(paste(beta == 1.5, ",    ", theta == 1.108)))
plot(tt, exp(-(tt/t2)^b2), xlim=c(0,2), cex.axis=.7, cex.lab=.7,
     mgp=c(1.5,.5,0), lwd=2, col="blue", type="l", xlab="t", ylab="",
     main= "Funcion de Supervivencia")
plot(tt, 1-exp(-(tt/t2)^b2), xlim=c(0,2), cex.axis=.7, cex.lab=.7,
     mgp=c(1.5,.5,0), lwd=2, col="blue", type="l", xlab="t", ylab="",
     main= "Funcion de Distribucion")
```



Considere las variables aleatorias independientes $X_1, X_2, X_3, \cdots$ tales que

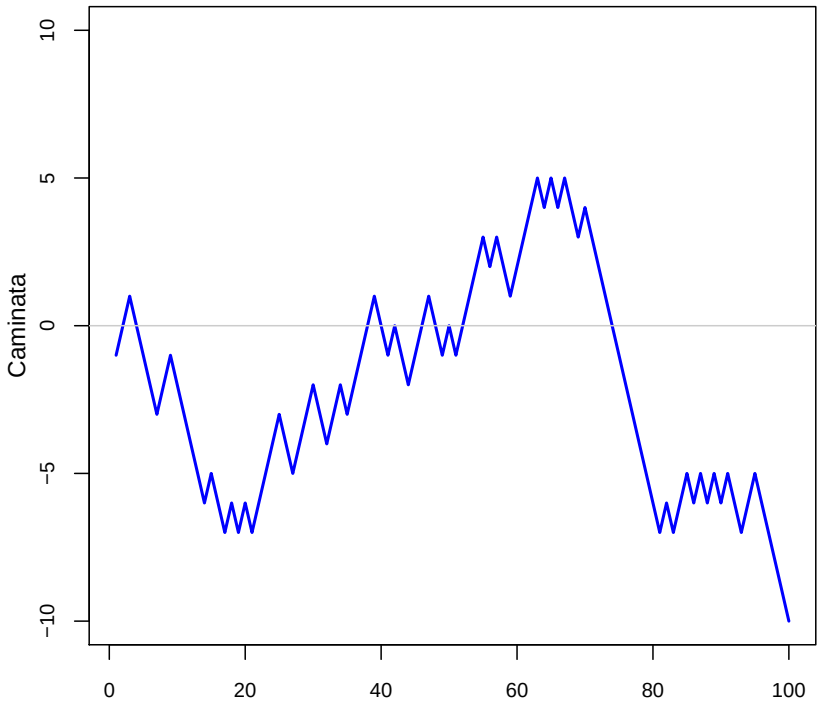
$$X_i = \begin{cases} -1 & \text{con probabilidad } 1/2 \\ 1 & \text{con probabilidad } 1/2 \end{cases}$$

Si a estas variables X_i's las pensamos como los movimientos de un partícula (-1 = moverse hacia abajo y 1 = moverse hacia arriba), entonces la suma de estos movimientos describe una trayectoria que es llamada "Caminata Aleatoria". La caminata se define como

$$S_j = X_1 + X_2 + \cdots + X_j$$

El siguiente programa produce una animación de una caminata aleatoria:

```
# caminata aleatoria

n   <- 100
xx  <- 1:n
cam <- cumsum(sample(c(-1,1),size=n,replace=T))

camina <- function(k){
 plot(1:k, cam[1:k], xlim=c(1,n), ylim=c(-n/10,n/10), type="l",
      col="blue",lwd=2,mgp=c(2,1,0),ylab="Caminata",xlab="", cex.axis=.8)
 abline(h=0,col=gray(.8))
 Sys.sleep(0.1) }         # Sys.sleep() controla la rapidez de la animacion

trash <- sapply(xx,camina)
```



Método de Newton. Suponga que deseamos encontrar una raíz de la función $f(x)$, esto es, deseamos encontrar un valor, x^*, tal que $f(x^*) = 0$. El método de Newton es un método iterativo que -usualmente- converge a una raíz. Consiste en arrancar en punto inicial x_0, aproximar linealmente a la función $f(x)$ alrededor de x_0, encontrar la raíz de esta aproximación e iterar estos pasos hasta convergencia.

La aproximación lineal alrededor de $x = x_0$ es

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0)$$

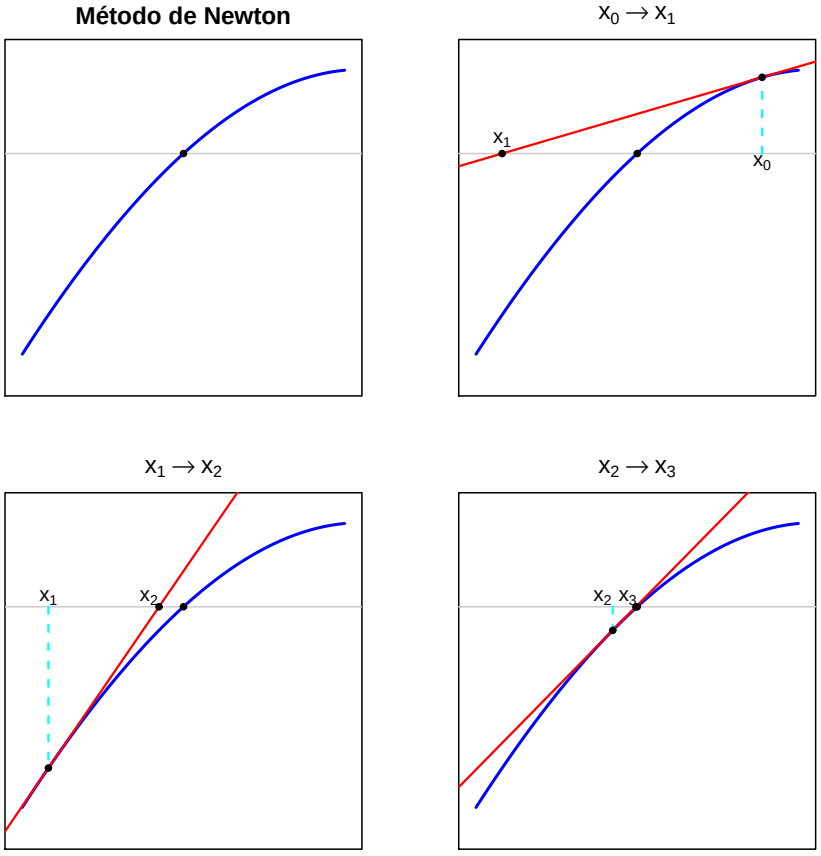
como encontrar directamente la raíz de $f(x)$ puede ser difícil, entonces el método de Newton lo que hace es encontrar la raíz de la aproximación lineal, lo cual es muy fácil:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Ahora consideramos que x_1 es nuestro nuevo punto inicial e iteramos el procedimiento. Ejemplificamos esto con la función

$$f(x) = 2x - x^2 - 0.1x^3$$

usando como punto inicial $x_0 = 0.62$




```
# Metodo de Newton para encontrar una raiz.

f  <- function(x){ 2*x-x^2-.1*x^3 }
fp <- function(x){ 2-2*x-.3*x^2 }

par(mfrow=c(2,2),mar=c(2,2,2,2))
```

```
xx <- seq(-.8,.8,length=200)
yy <- 2*xx-xx^2-.1*xx^3
plot(xx,yy,type="l",xlab="",ylab="",xlim=c(-.82,.82),ylim=c(-2.5,1.1),
 col="blue",lwd=2,xaxt="n",yaxt="n", main=c("Mtodo de Newton"))
abline(h=0,col=gray(.8))
points(0,0,pch=20)

x0 <- 0.62
x1 <- x0 - f(x0)/fp(x0)

plot(xx,yy,type="l",xlab="",ylab="",xlim=c(-.82,.82),ylim=c(-2.5,1.1),
 main=expression(x[0] %->% x[1]),col="blue",lwd=2,xaxt="n",yaxt="n")
abline(h=0,col=gray(.8))
abline(a=f(x0)-fp(x0)*x0, b=fp(x0), col="red",lwd=1.5)
segments(x0,0,x0,f(x0),lty=2,col="cyan",lwd=1.5)
points(c(x0,x1,0),c(f(x0),0,0),pch=20)
text(x0,-.1,expression(x[0]))
text(x1,.15,expression(x[1]))

x0 <- x1
x1 <- x0 - f(x0)/fp(x0)

plot(xx,yy,type="l",xlab="",ylab="",xlim=c(-.82,.82),ylim=c(-2.5,1.1),
 main=expression(x[1] %->% x[2]),col="blue",lwd=2,xaxt="n",yaxt="n")
abline(h=0,col=gray(.8))
abline(a=f(x0)-fp(x0)*x0, b=fp(x0), col="red",lwd=1.5)
segments(x0,0,x0,f(x0),lty=2,col="cyan",lwd=1.5)
points(c(x0,x1,0),c(f(x0),0,0),pch=20)
text(x0,.1,expression(x[1]))
text(x1-.05,.1,expression(x[2]))

x0 <- x1
x1 <- x0 - f(x0)/fp(x0)

plot(xx,yy,type="l",xlab="",ylab="",xlim=c(-.82,.82),ylim=c(-2.5,1.1),
 main=expression(x[2] %->% x[3]),col="blue",lwd=2,xaxt="n",yaxt="n")
abline(h=0,col=gray(.8))
abline(a=f(x0)-fp(x0)*x0, b=fp(x0), col="red",lwd=1.5)
segments(x0,0,x0,f(x0),lty=2,col="cyan",lwd=1.5)
points(c(x0,x1,0),c(f(x0),0,0),pch=20)
text(x0-.05,.1,expression(x[2]))
text(x1-.04,.1,expression(x[3]))
```

La estructura del algoritmo es:

- Iniciar en x_0
- Para $k = 1, 2, \cdots$, hacer
$$x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})}$$
- Parar si $|x_k - x_{k-1}| < \delta$

donde δ es cierta tolerancia que nosotros fijamos.

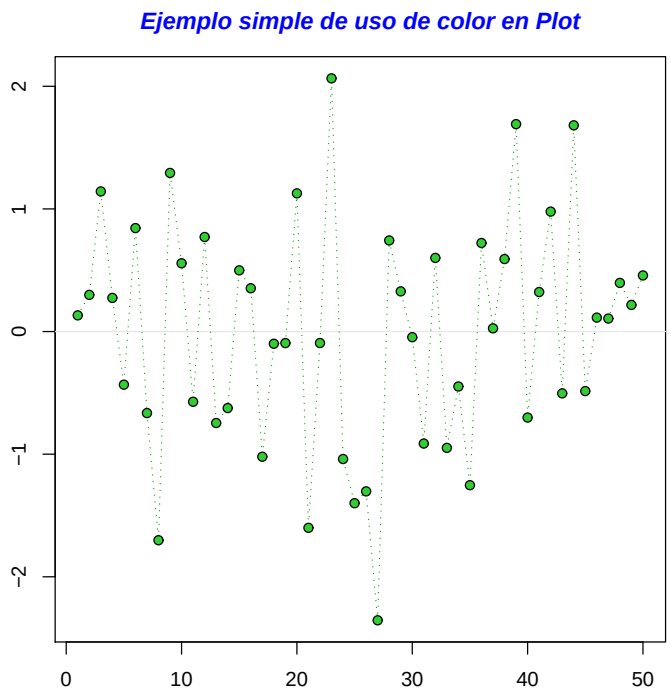
Ahora mostramos un programa en R que incorpora estructuras de ciclos y control de paro para este algoritmo

```
f  <- function(x){ 2*x-x^2-.1*x^3 }
fp <- function(x){ 2-2*x-.3*x^2 }

x      <- 0.62      # valor inicial
delta  <- 1e-6      # tolerancia
iterM  <- 1000      # numero maximo de iteraciones
tolera <- 1         # inicializar tolera
itera  <- 0         # inicializar itera
histo  <- x         # inicializar historial de iteraciones

while( (tolera>delta) & (itera<iterM) ){
 xold   <- x
 x      <- xold - f(xold)/fp(xold)
 tolera <- abs( x - xold )
 histo  <- c(histo,x)
 itera  <- itera + 1 }
length(histo)          # 7 iteraciones
histo

#[1]  6.200000e-01 -6.702017e-01 -1.213367e-01 -6.418085e-03 -2.043842e-05
#[6] -2.088594e-10 -2.181112e-20
```

Notas Sesión 1: Aprovechando los demos.

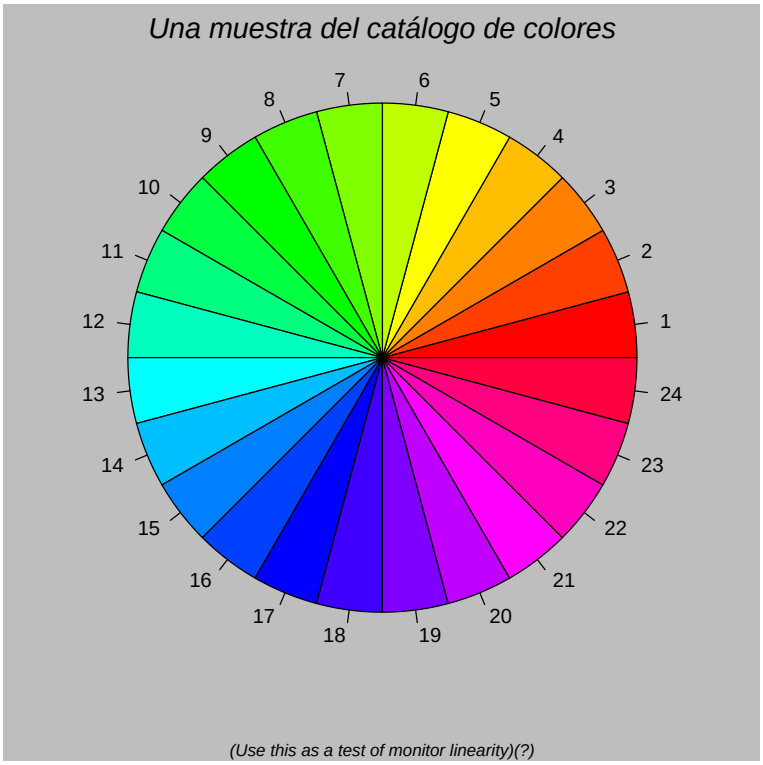
Los siguientes ejemplos fueron tomados de `demo(graphics)`:

- Ejemplo 1: Una gráfica simple, ilustrando el uso de colores en sus distintos elementos.



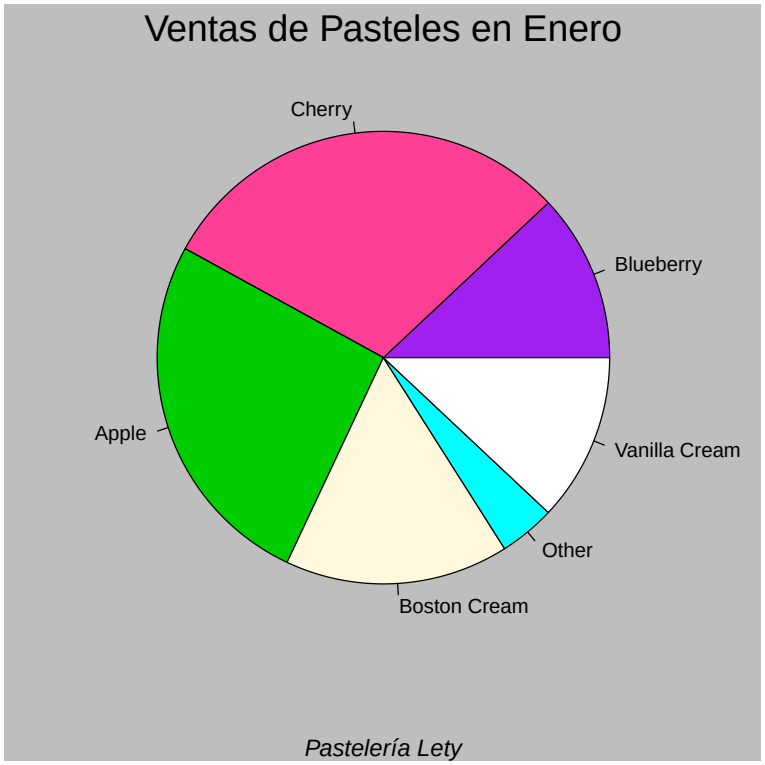

```
opar <- par(bg = "white")              # guardar parametros

x    <- rnorm(50)
plot(x, ann = FALSE, type = "n")
abline(h = 0, col = gray(.90))
lines(x, col = "green4", lty = "dotted")
points(x, bg = "limegreen", pch = 21)
title(main = "Ejemplo simple de uso de color en Plot",
      xlab = "Informacion con un color desvanecido",
      col.main = "blue", col.lab = gray(.7),
      cex.main = 1.2, cex.lab = 1.0, font.main = 4, font.lab = 3)
```

- Ejemplo 2: Diagrama de pastel.



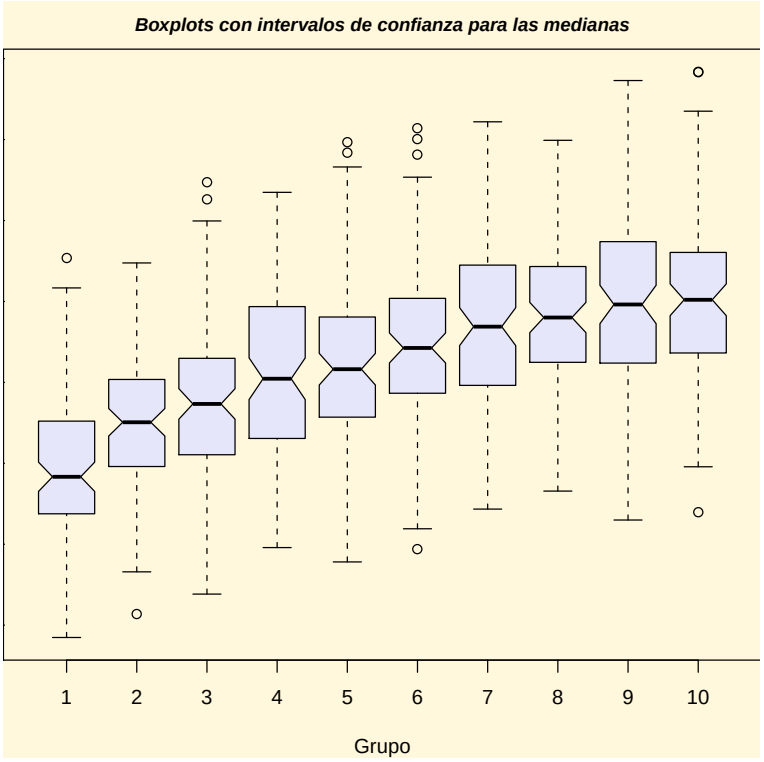

```
par(bg = "gray")
pie(rep(1,24), col = rainbow(24), radius = 0.9)
title(main = "Una muestra del catalogo de colores",
      cex.main = 1.4, font.main = 3)
title(xlab = "(Use this as a test of monitor linearity)(?)",
       cex.lab = 0.8, font.lab = 3)
```

- Ejemplo 3: Diagrama de pastel (de nuevo).



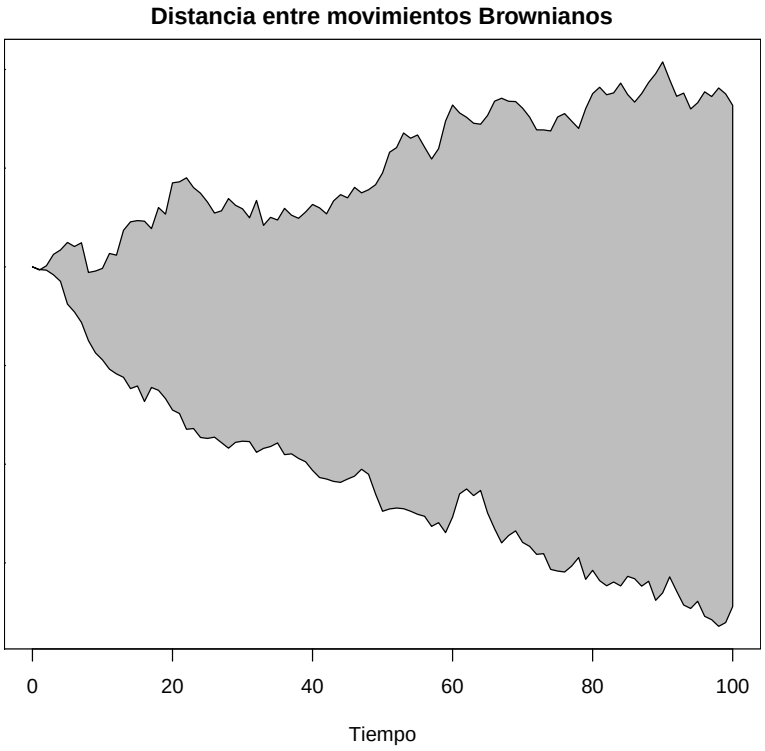

```
pie.sales <- c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)
names(pie.sales) <- c("Blueberry", "Cherry",
                      "Apple", "Boston Cream", "Other", "Vanilla Cream")
pie(pie.sales,
     col = c("purple","violetred1","green3","cornsilk","cyan","white"))
title(main = "Ventas de Pasteles en Enero", cex.main = 1.8, font.main = 1)
title(xlab = "Pasteleria Lety", cex.lab = 1.2, font.lab = 3)
```

- Ejemplo 4: Boxplots.



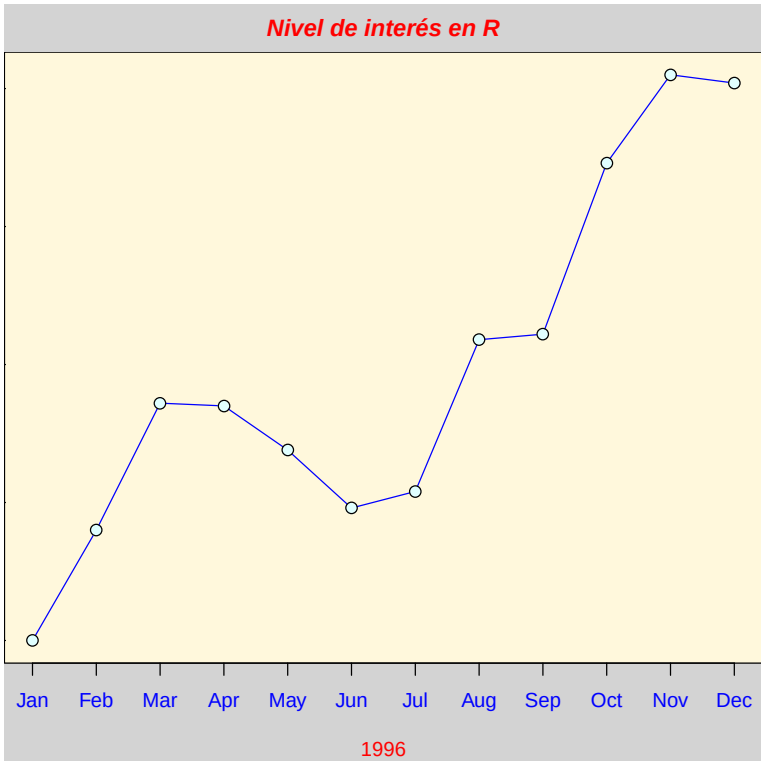

```
par(bg="cornsilk")

n <- 10
g <- gl(n, 100, n*100)
x <- rnorm(n*100) + sqrt(as.numeric(g))
boxplot(split(x,g), col="lavender", notch=TRUE)
title(main="Boxplots con intervalos de confianza para las medianas",
      xlab="Grupo", font.main=4, font.lab=1, cex.main=.9)
```

- Ejemplo 5: Área sombreada entre dos gráficas.



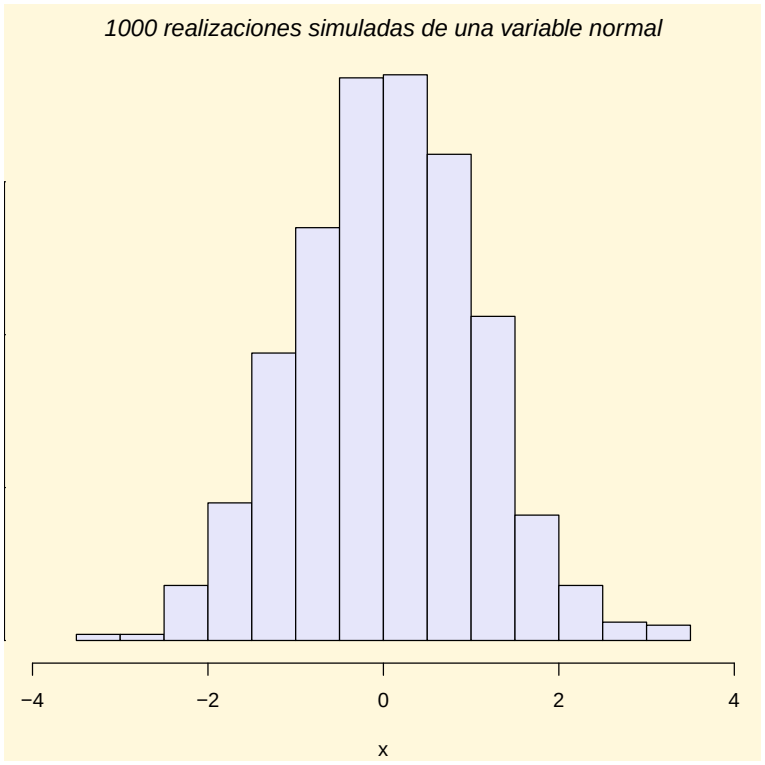

```
par(bg="white")
n   <- 100
x   <- c(0,cumsum(rnorm(n)))
y   <- c(0,cumsum(rnorm(n)))
xx  <- c(0:n, n:0)
yy  <- c(x, rev(y))
plot(xx, yy, type="n", xlab="Tiempo", ylab="Distancia")
polygon(xx, yy, col="gray")
title("Distancia entre movimientos Brownianos")
```

- Ejemplo 6: Gráficas tipo Excel, o algo parecido.



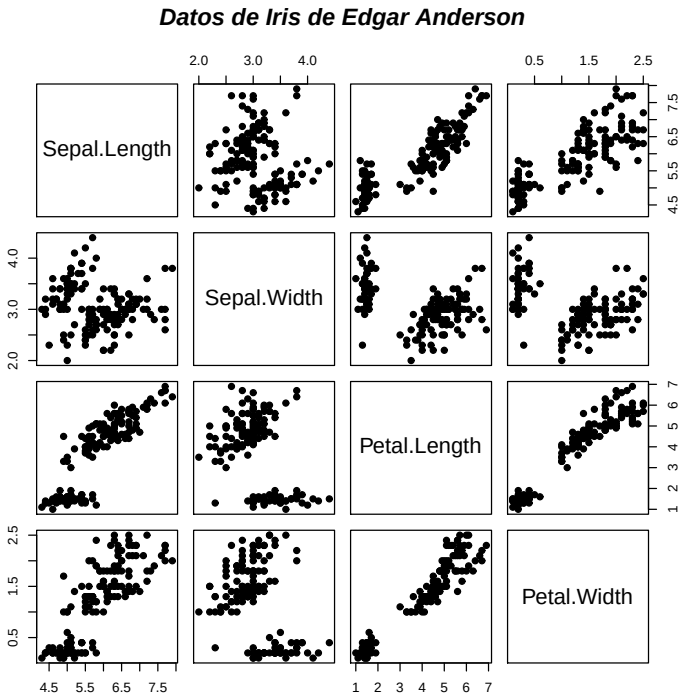

```
x   <- c(0.00,0.40,0.86,0.85, 0.69, 0.48, 0.54, 1.09, 1.11, 1.73, 2.05, 2.02)
par(bg="lightgray")
plot(x, type="n", axes=FALSE, ann=FALSE)
usr <- par("usr")  # c(x1,x2,y1,y2) con coordenadas de region de graficacion
rect(usr[1], usr[3], usr[2], usr[4], col="cornsilk", border="black")
lines(x, col="blue")
points(x, pch=21, bg="lightcyan", cex=1.25)
axis(2, col.axis="blue", las=1)
axis(1, at=1:12, lab=month.abb, col.axis="blue")
title(main= "Nivel de interes en R", font.main=4, col.main="red")
title(xlab= "1996", col.lab="red")
```

- Ejemplo 7: Histograma.



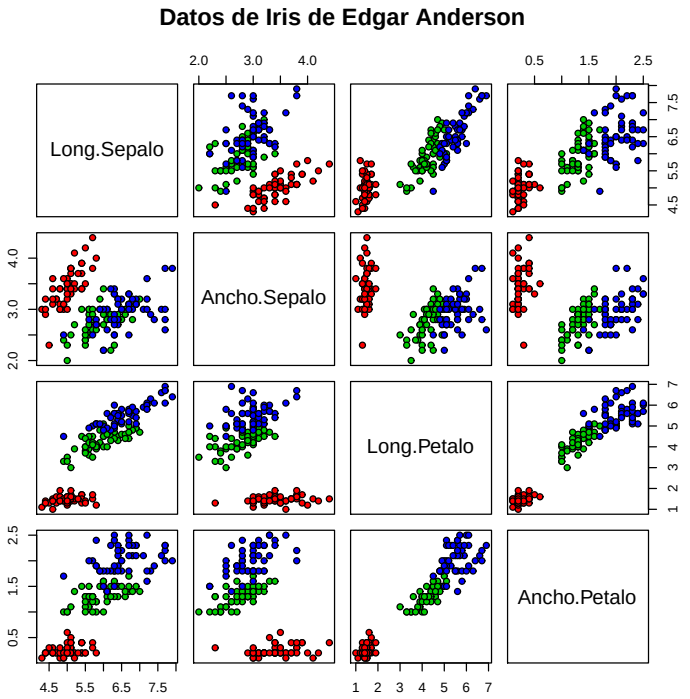

```
par(bg="cornsilk")
x <- rnorm(1000)
hist(x, xlim=range(-4, 4, x), col="lavender", main="", ylab="Frecuencia")
title(main="1000 realizaciones simuladas de una variable normal", font.main=3)
```

- Ejemplo 8: Gráfica por parejas.



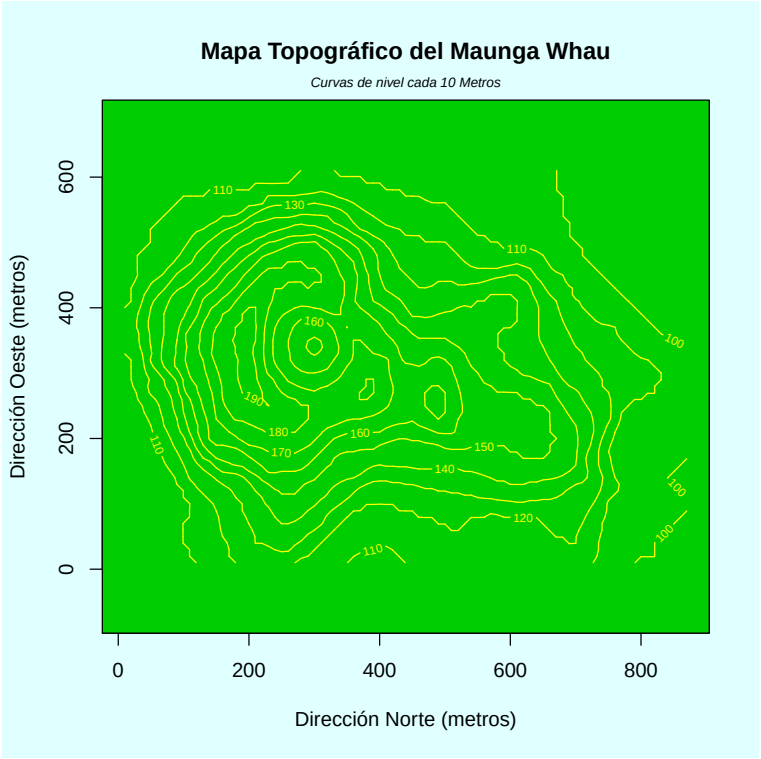

```
pairs(iris[1:4], main="Datos de Iris de Edgar Anderson", font.main=4, pch=19)
```

- Ejemplo 9: Gráfica por parejas (colores diferentes para cada especie).



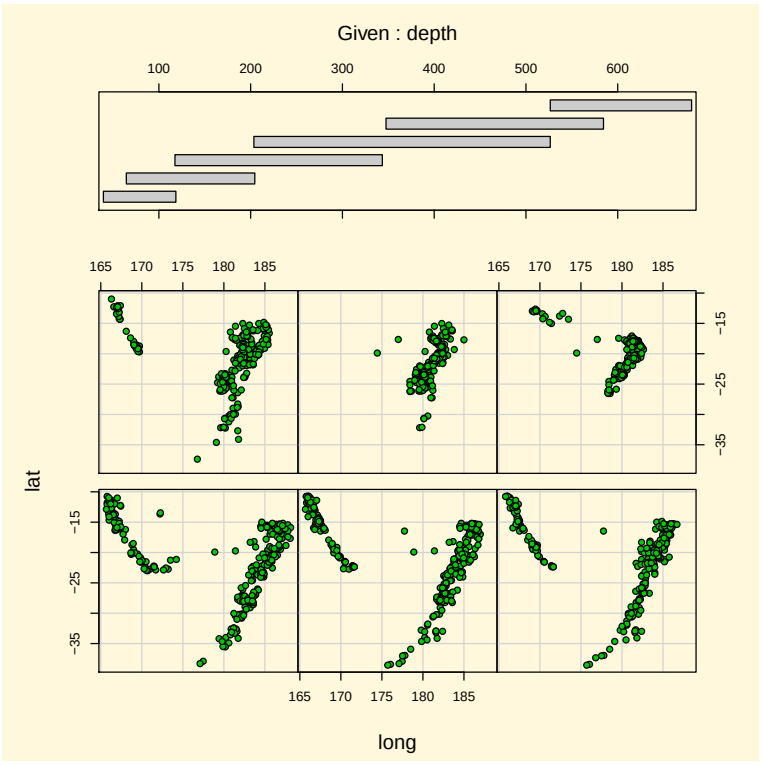

```
aa <- iris
names(aa) <- c("Long.Sepalo","Ancho.Sepalo",
               "Long.Petalo","Ancho.Petalo","Especie")
pairs(aa[1:4], main="Datos de Iris de Edgar Anderson", pch=21,
       bg = c("red", "green3", "blue")[unclass(iris$Species)])
```

- Ejemplo 10: Gráfica de contornos.




```
# volcano es la matriz 87 x 61 de elevaciones del volcan Maunga Whau en NZ

x      <- 10*1:nrow(volcano)
y      <- 10*1:ncol(volcano)
lev    <- pretty(range(volcano), 10)
par(bg = "lightcyan")
pin    <- par("pin")
xdelta <- diff(range(x))
ydelta <- diff(range(y))
xscale <- pin[1]/xdelta
yscale <- pin[2]/ydelta
scale  <- min(xscale, yscale)
xadd   <- 0.5*(pin[1]/scale - xdelta)
yadd   <- 0.5*(pin[2]/scale - ydelta)
plot(numeric(0), numeric(0),
      xlim = range(x)+c(-1,1)*xadd, ylim = range(y)+c(-1,1)*yadd,
      type = "n", ann = FALSE)
usr <- par("usr")
rect(usr[1], usr[3], usr[2], usr[4], col="green3")
contour(x, y, volcano, levels = lev, col="yellow", lty="solid", add=TRUE)
title("Mapa Topografico del Maunga Whau", font= 4)
title(xlab = "Direccin Norte (metros)", ylab = "Direccin Oeste (metros)",
      font= 3)
mtext("Curvas de nivel cada 10 Metros", side=3, line=0.35, outer=FALSE,
       at = mean(par("usr")[1:2]), cex=0.7, font=3)
```

- Ejemplo 11: Gráficas condicionales.




```
# El conjunto de datos quakes es un data frame con 1000 observaciones
# en 5 variables:
#  lat      = Latitud del evento
#  long     = Longitud
#  depth    = Profundidad (km)
#  mag      = Magnitud en escala de Richter
#  stations = Numero de estaciones reportando el evento

par(bg="cornsilk")
coplot(lat ~ long | depth, data = quakes, pch = 21, bg = "green3")
```

Prácticas Sesión 1

(Los ejercicios del 1 al 6 están relacionados).

1. Genere una muestra de tamaño 50 de la distribución normal con media 5 y desviación estándar 2. La función para generar la muestra es `rnorm(n, mean=, sd=)` donde:

`n`	=	es el tamaño de la muestra
`mean`	=	media
`sd`	=	desviación estándar

   Dé a esta muestra formato de matriz `A` 10×5 y de estructura de datos `A.df` (data frame)

2. Dé nombre a los renglones y columnas de la matriz `A`

3. Dé nombre a los renglones y columnas de `A.df`

4. Fije los primeros dos elementos del renglón 2 y el tercer y quinto elemento del renglón 5 de `A` como valores `NA`. Encuentre la media de aquellos elementos de `A` que no sean `NA`.

5. Genere una segunda muestra de tamaño 25 de una disribución normal con media 10 y desviación estándar 1. Guarde esta muestra en una matriz 5×5 llamada `B`. Combine las matrices A y B para formar una matriz `C` de 15×5.

6. Encuentre el número de elementos en cada renglón de `C` que están entre 5 y 10. Calcule la media de cada renglón y columna de `C` (hay que tener cuidado pues `C` contiene valores `NA`) Reemplaze los valores `NA` de `C` por el valor de la media de los elementos no `NA` de la misma columna.

7. Considere el conjunto de datos `iris`. Estos datos consisten de mediciones de largo y ancho de los pétalos y sépalos de 150 flores de iris (50 flores de cada una de tres especies diferentes: Setosa, Versicolor y Virginica). Efectúe un análisis exploratorio para comparar esas tres especies.

8. Sea x el vector de longitudes de pétalos de flores de la variedad Setosa y sea y el correspondiente vector de longitudes de pétalos, pero de la variedad Versicolor. Suponiendo homogeneidad de varianzas:

   (a) Use los datos x y y y la función `t.test()` para contrastar la hipótesis nula de que las longitudes medias poblacionales son iguales contra la hipótesis alterna de que no lo son.

   (b) Replique la salida de la función `t.test()` pero haciendo los cálculos directamente (por supuesto, usando R).

9. Los primeros términos de la sucesión de Fibonacci son

   $$1 \quad 1 \quad 2 \quad 3 \quad 5 \quad 8 \quad 13 \quad 21$$

   Escriba un programa, `Fibo(n)`, que calcule los términos de esa sucesión; esto es, el valor que regrese `Fibo(n)` debe ser el n-ésimo término de la sucesión de Fibonacci.

10. Los primeros términos de la sucesión de números primos son

$$2 \quad 3 \quad 5 \quad 7 \quad 11 \quad 13 \quad 17 \quad 19$$

Escriba un programa, `Primo(n)`, que calcule los términos de esa sucesión; esto es, el valor que regrese `Primo(n)` debe ser el n-ésimo término de la sucesión de números primos.

11. Si, por ejemplo, `x <- c(1,2,3,4)`, `y <- c(5,4,3,1)` y `z <- c(4,3,1,2)`, entonces `pmin(x,y,z)` (parallel minimum) dá como resultado `1,2,1,1`. Indique como se puede hacer esta misma operación usando `apply`.

12. El estatuto `segments(x1,y1,x2,y2)` traza el segmento de recta que une los puntos (x_1, y_1) y (x_2, y_2). Considere una caminata aleatoria de la siguiente forma: Arrancamos en el punto inicial (x_1, y_1), luego nos vamos al punto (x_2, y_2) definido por `x2 <- x1 + rnorm(1)` y `y2 <- y1 + rnorm(1)`. Escriba una función, `Caminata(n=100)` que simule una caminata de n pasos en el plano (Puede definir de antemano una región de graficación, por ejemplo entre -10 y 10, tanto para el eje x como el eje y).

13. Suponga que $x_{11}, x_{21}, \cdots, x_{n1}$ son realizaciones observadas de una variable aleatoria x_1 y $x_{12}, x_{22}, \cdots, x_{n2}$, son las correspondientes observaciones de una variable x_2. La covarianza entre x_1 y x_2 puede estimarse mediante

$$s_{12} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{i1} - \bar{x}_1)(x_{i2} - \bar{x}_2)$$

donde $\bar{x}_j = \sum_i x_{ij}/n$.

(a) Suponga que $X_1 = (x_{11}, x_{21}, \cdots, x_{n1})^T$ y $X_2 = (x_{12}, x_{22}, \cdots, x_{n2})^T$. Muestre que

$$s_{12} = \frac{1}{n-1} X_1^T \left(I - \frac{1}{n} J \right) X_2$$

donde I es la matriz idéntica y J es una matriz cuadrada con todos sus elementos iguales a 1; ambas de tamaño $n \times n$.

(b) Suponga que en R tenemos dos vectores, `x1` y `x2`, de longitud n. Escriba los comandos necesarios para calcular su covarianza. Compare con la función `cov()`.

14. Reproduzca en R las figuras 3.2.3 y 3.2.4 de las hojas 108 y 109 del libro de Casella y Berger (1a edición) (son gráficas de diferentes funciones de densidad Beta).

15. El **Teorema Central del Límite** nos dice que si $X_1, X_2, \cdots$ es una sucesión de variables aleatorias i.i.d. cada una con media μ y varianza σ^2 entonces

$$\sqrt{n} \frac{\bar{X}_n - \mu}{\sigma} \quad \texttt{tiene una distribución límite normal estándar.}$$

Usando simulación vea que este resultado es razonable.... (en clase comentaremos como estructurar una simulación que justifique el resultado).

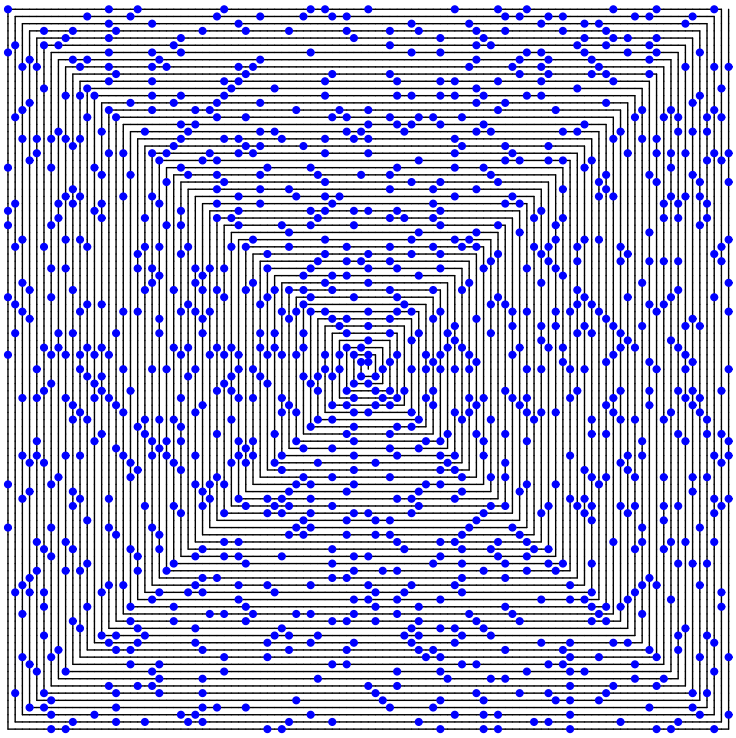
16. ¿Para que se puede usar la función `barplot()`?. Dé ejemplos.

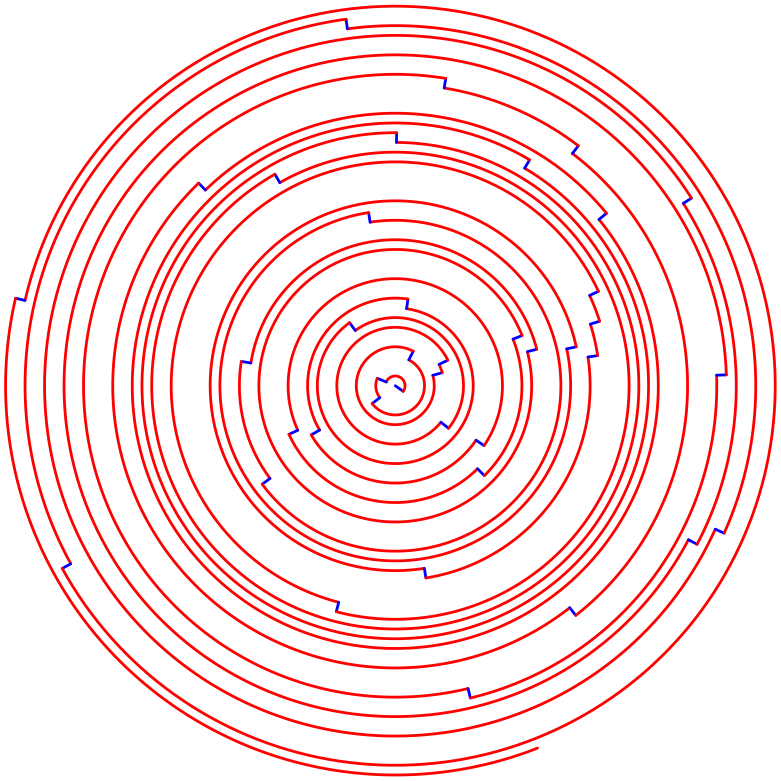
17. Un apostador tiene $1,023 dólares y tiene un método seguro para ganar jugando a la ruleta. Apuesta $1 al color rojo (con probabilidad de ganar igual a 18/38), si gana entonces recibe la cantidad que apostó más una cantidad igual (i.e. recibe un total igual al doble de lo que apostó) y se retira. Ahora, si pierde, entonces en la siguiente vuelta apuesta el doble de lo que apostó la última vez. En el momento que gane por primera vez el se retirará de la masa. Claro que hay una pequeña probabilidad de que se vaya a bancarrota, sin embargo, es tan pequeña que se puede ignorar (según él). ¿Cuál es la probabilidad de que gane?, ¿Cuánto ganaría?, ¿Cuál es la probabilidad de que pierda?, ¿Cuánto perdería?, si este sistema lo pusiera en práctica, a la larga, ¿Cuál sería su balance financiero?. Estas preguntas pueden contestarse usando procedimientos analíticos, sin embargo, queremos contestarlas usando simulación.

18. Si tiro tres dados, ¿Cuál es la probabilidad de obtener exactamente un 6?. Use simulación (que involucre el uso d ela función `sample()`) y compare con la solución analítica.

Apéndice Sesión 1

Espiral de Ulam





Apéndice Sesión 1

```
#######################################################################
# Figura 1: Espiral de Ulam
# prim = Un programa para calcular los primeros n primos
prim <- function(n){
 if(n==1){return(2)}
 primos <- 2
 notyet <- TRUE
 probar <- 2
 while(notyet){
  probar <- probar + 1
  pritst <- primos[ primos<=sqrt(probar) ]
  aa     <- (probar %% pritst)
  if( any(aa==0) ){next}
  primos <- c(primos,probar)
  if( length(primos)==floor(n) ){ return(primos) }}}
m  <- 100
pp <- prim( (m+1)^2 )
ii <- seq(3,m+1,by=2)
jj <- length(ii)
par(mar=c(0,0,0,0)+1);   xylim <- c(1,m+1)
plot(1,1,xlim=xylim,ylim=xylim,type="n",xaxt="n",yaxt="n",bty="n",xlab="",ylab="")
aa <- c(floor(m/2)+1,floor(m/2)+1)
for(k in 1:jj){
 r  <- ii[k]
 co <- cbind(c(rep(r,r),(r-1):2,rep(1,r),2:(r-1)),c(r:1,rep(1,r-2),1:r,rep(r,r-2)))
 co <- co + (jj-k)
 n  <- dim(co)[1]
 uu <- (r^2):((r-2)^2)
 rr <- is.element(uu[-(n+1)],pp)
 bb <- co[n,]
 segments(aa[1],aa[2],bb[1],bb[2],col="black",lwd=1)
 aa <- co[1,]
 for(i in 1:(n-1)){ segments(co[i,1],co[i,2],co[i+1,1],co[i+1,2],col="black",lwd=1)}
 points(co[rr,1],co[rr,2],col="blue",pch=20) }
title("Espiral de Ulam",cex=.9,line=-.3)
#######################################################################
# Figura 2: Laberinto circular
M <- 40;   m <- 120;  n <- M;  xylim <- .95*c(-M,M)
par(mar=c(0,0,0,0)+.6)
plot(0,0,type="n",xlim=xylim,ylim=xylim,xaxt="n",yaxt="n",xlab="",ylab="",bty="n")
pp   <- c(0,0)
tet1 <- runif(1,min=0,max=2*pi)
for( r in 1:n ){
 qq   <- r*c(cos(tet1),sin(tet1))
 segments(pp[1],pp[2],qq[1],qq[2], col="blue",lwd=2)
 tet2 <- tet1 + runif(1,min=0,max=2*pi)
 ts   <- seq(tet1,tet2,length=200)
 nc   <- r*cbind( cos(ts), sin(ts) )
 lines( nc[,1], nc[,2], col="red",lwd=2 )
 tet1 <- tet2
 pp   <- nc[200,] }
#######################################################################
```

MARTES

Notas Sesión 2

Gauss-Seidel. Consideremos el sistema de ecuaciones lineales $Ax = b$. Escribamos a A como $A = L + U$, donde L tiene los elementos diagonales y abajo de la diagonal de A y U tiene los elementos de A que se encuentran por arriba de la diagonal. Entonces

$$Ax = b \quad \Rightarrow \quad (L+U)x = b \quad \Rightarrow \quad Lx = b - Ux$$

De esta última ecuación se sigue una idea de como resolver el sistema en forma iterativa: Empezar con algún valor para x, resolver el sistema triangular resultante e iterar (en cursos de Análisis Numérico se vé que este esquema converge para sistemas con A positiva definida o A con diagonal dominante). El algoritmo Gauss-Seidel es:

- Iniciar en x^0
- Para $k = 1, 2, \cdots$, resolver para x^k: $Lx^k = b - Ux^{k-1}$
- Parar hasta que se cumple algún criterio de paro

Veamos a detalle la estructura del algoritmo para un caso pequeño.

$$Ax = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

Tenemos entonces que, en cada paso, el sistema a resolver es:

$$\begin{bmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1^k \\ x_2^k \\ x_3^k \\ x_4^k \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} - \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} \\ 0 & 0 & a_{23} & a_{24} \\ 0 & 0 & 0 & a_{34} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1^{k-1} \\ x_2^{k-1} \\ x_3^{k-1} \\ x_4^{k-1} \end{bmatrix}$$

equivalentemente

$$\begin{array}{rcrcrcrcl} a_{11}x_1^k & & & & & & = & b_1 - a_{12}x_2^{k-1} - a_{13}x_3^{k-1} - a_{14}x_4^{k-1} \\ a_{21}x_1^k & + & a_{22}x_2^k & & & & = & b_2 - a_{23}x_3^{k-1} - a_{24}x_4^{k-1} \\ a_{31}x_1^k & + & a_{32}x_2^k & + & a_{33}x_3^k & & = & b_3 - a_{34}x_4^{k-1} \\ a_{41}x_1^k & + & a_{42}x_2^k & + & a_{43}x_3^k & + & a_{44}x_4^k & = & b_4 \end{array}$$

un sistema triangular es fácil de resolver

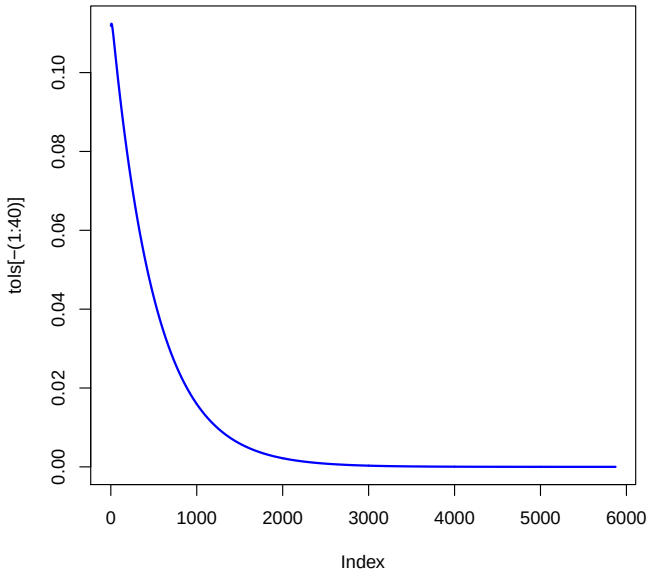
$$\begin{aligned} x_1^k &= \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^{k-1} - a_{13}x_3^{k-1} - a_{14}x_4^{k-1}\right) \\ x_2^k &= \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^k - a_{23}x_3^{k-1} - a_{24}x_4^{k-1}\right) \\ x_3^k &= \frac{1}{a_{33}}\left(b_3 - a_{31}x_1^k - a_{32}x_2^k - a_{34}x_4^{k-1}\right) \\ x_4^k &= \frac{1}{a_{33}}\left(b_4 - a_{41}x_1^k - a_{42}x_2^k - a_{43}x_3^k\right) \end{aligned}$$

esto es,

$$\begin{aligned} x_1^k &= \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^{k-1} - a_{13}x_3^{k-1} - a_{14}x_4^{k-1}\right) \\ x_i^k &= \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^k - \sum_{j=i+1}^{n} a_{ij}x_j^{k-1}\right), \quad i = 2, 3 \\ x_4^k &= \frac{1}{a_{33}}\left(b_4 - a_{41}x_1^k - a_{42}x_2^k - a_{43}x_3^k\right) \end{aligned}$$

A continuación mostramos una implementación en R del Gauss-Seidel.

```
set.seed(75757)
n  <- 10
A  <- matrix(runif(n^2),n,n);    A  <- t(A)%*%A   # (haciendo A p.d.)
b  <- runif(n)
x      <- rep(0,n)  # valores de inicio
tolm   <- 1e-6      # tolerancia (norma minima de delta)
iterM  <- 10000     # numero maximo de iteraciones
tolera <- 1         # inicializar tolera
itera  <- 0         # inicializar itera
histo  <- x         # inicializar historial de iteraciones
tols   <- 1         # inicializar historial tolerancias
while( (tolera>tolm)&(itera<iterM) ){
 xold   <- x
 x[1]   <- ( b[1] - sum(A[1,-1]*x[-1]) ) / A[1,1]
 for(i in 2:(n-1)){
  rr    <- 1:(i-1)
  ss    <- 1:i
  x[i]  <- ( b[i] - sum(A[i,rr]*x[rr]) - sum(A[i,-ss]*x[-ss]) ) / A[i,i] }
 x[n]   <- ( b[n] - sum(A[n,-n]*x[-n]) ) / A[n,n]
 delta  <- x - xold
 tolera <- sqrt( sum(delta*delta) )
 histo  <- rbind(histo,x)
 tols   <- c(tols,tolera)
 itera  <- itera + 1 }
dim(histo)[1]           # 5913 iteraciones!
histo[dim(histo)[1],]
# comparar con:  solve(A,b)
# examinando la velocidad de convergencia (lenta)
plot(tols[-(1:40)],type="l",lwd=2,col="blue")
```

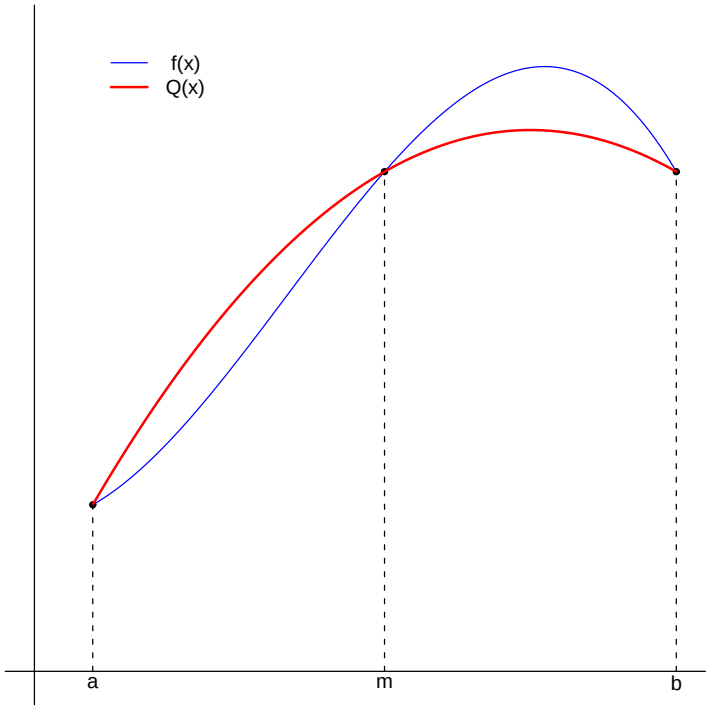


Nota: R no es para este tipo de problemas (muchos ciclos!), sólo queremos ejemplificar el uso del lenguaje.

Regla de Simpson. Abordamos ahora el problema de integración numérica. Un método popular es el basado en la Regla de Simpson:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]$$

Esta fórmula da una aproximación a la integral de $f(x)$ en el intervalo (a,b). Está basada en integrar una cuadrática, $Q(x)$, que coincide con f en los tres puntos a, $m=(a+b)/2$ y b.



El código correspondiente, que no usaremos, de todos modos lo incluimos enseguida:

```
x  <- seq(-1,1,length=200)
y  <- 1+2*x-x^2-x^3
yc <- 1+x-x^2
par(mar=c(0,0,0,0))
plot(x,y,type="l",xlim=c(-1.3,1.1),ylim=c(-2.2,2),xaxt="n",yaxt="n",xlab="",
 ylab="",bty="n",col="blue")
points(c(-1,0,1),c(f1,f2,f3),pch=20)
lines(x,yc,col="red",lwd=2)
segments(-1.3 , -2   ,  1.1 , -2)
segments(-1.2 , -2.2 , -1.2 ,  2)
segments(-1   , -2   , -1   , -1 , lty=2)
segments( 0   , -2   ,  0   ,  1 , lty=2)
segments( 1   , -2   ,  1   ,  1 , lty=2)
text(-1,-2.07,"a")
text( 0,-2.07,"m")
text( 1,-2.07,"b")
legend(-1,1.8,legend=c(" f(x)","Q(x)"),lwd=c(1,2),col=c("blue","red"),bty="n")
```

La Regla de Simpson, propiamente se refiere a la aplicación de la aproximación, dada arriba, a los elementos de una partición del intervalo (a,b). Suponga que particionamos el intervalo en:

$$a=x_0<x_1<x_2<x_3<x_4<\cdots<x_j<x_{j+1}<x_{j+2}<\cdots<x_{2n}=b$$

con los puntos igualmente espaciados y $h = x_{j+2} - x_j$. Entonces

$$\int_a^b f(x)\,dx = \int_{x_0}^{x_2} f(x)\,dx + \int_{x_2}^{x_4} f(x)\,dx + \cdots + \int_{x_{2n-2}}^{x_{2n}} f(x)\,dx$$

usando la aproximación, tenemos

$$\begin{aligned}\int_a^b f(x)\,dx &= \sum_j \int_{x_j}^{x_{j+2}} f(x)\,dx, \qquad j = 0, 2, 4, \cdots, 2n-2 \\ &\approx \sum_j \frac{h}{3}\left[f(x_j) + 4f(x_{j+1}) + f(x_{j+2})\right]\end{aligned}$$

esto es,

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots + 4f(x_{2n-1}) + f(x_{2n})\right]$$

lo cual se ve como un producto punto y esto es lo que contiene el siguiente código en R que calcula

$$\int_1^{50} \log(x)\,dx$$

```
# Regla de Simpson
f <- function(x){ return(log(x)) }
n   <- 600
a   <- 1
b   <- 50
x   <- seq(a,b,length=n+1)
h   <- (b-a)/n
coe <- c(1,rep(c(4,2),(n-2)/2),4,1)
int <- h*sum(coe*f(x))/3
int
integrate(f,a,b)
```

Ecuaciones Diferenciales. Consideremos la ecuación diferencial:

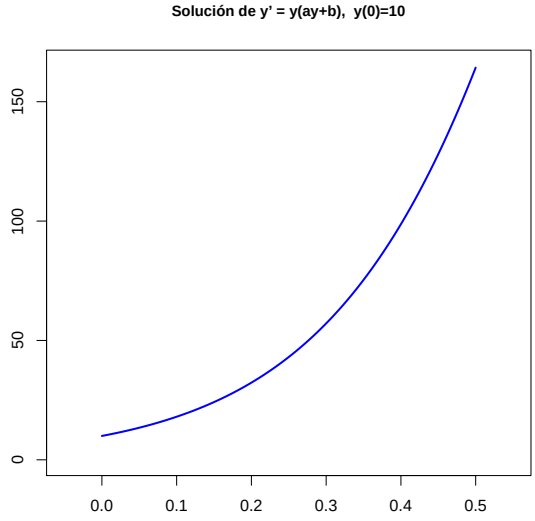
$$\frac{dy}{dx} = y(ay + b), \qquad x_0 = 0, \quad y_0 = y(x_0) = 10$$

donde $a = -0.007$ y $b = 6$. Queremos estimar el valor de y cuando $x = 0.5$. Para esta ecuación diferencial es fácil encontrar su solución analítica, dada por:

$$y = \frac{b}{e^{-b(x+C_0)} - a}, \qquad \text{donde} \quad C_0 = -\frac{1}{b}\log\left(\frac{ay_0 + b}{y_0}\right)$$

de modo que $y(x = 0.5) = 164.277$. Una gráfica de la solución la obtenemos mediante:

```
a  <- -0.007
b  <- 6
y0 <- 10
h  <- 0.5
C0 <- -log((a*y0+b)/y0)/b
xx <- seq(0,h,length=200)
yy <- b/(exp(-b*(xx+C0))-a)

plot(xx,yy,type="l",xlab="",ylab="",lwd=2,col="blue",cex.main=.9,
 ylim=c(0,165),xlim=c(-.05,.55),
 main="Solucion de y' = y(ay+b),  y(0)=10")
```



Método de Euler. El ejemplo que vimos de ecuación diferencial es de la forma

$$\frac{dy}{dx} = f(x,y), \qquad y(x_0) = y_0$$

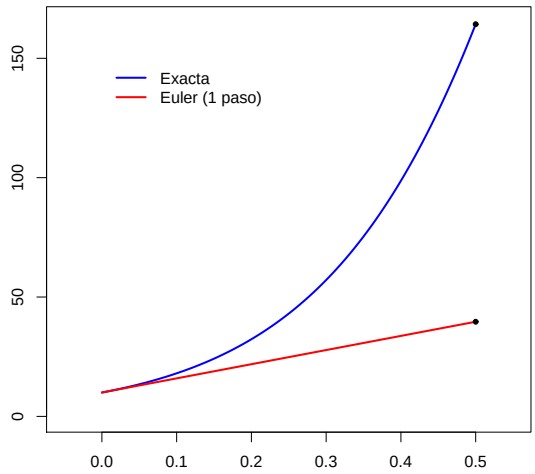
integrando, obtenemos

$$dy = f(x,y)dx \;\Rightarrow\; \int_{y_0}^{y(x)} dy = \int_{x_0}^{x} f(t,y(t))dt \;\Rightarrow\; y(x) = y_0 + \int_{x_0}^{x} f(t,y(t))dt$$

Queremos $y_1 = y(x_1)$, donde $x_1 = 0.5$. Ahora, si suponemos que $f(x,y(x)) \approx f(x_0,y_0)$ para toda x en $[x_0,x_1]$, entonces

$$y_1 = y(x_1) \approx y_0 + hf(x_0,y_0), \qquad \text{donde} \quad h = x_1 - x_0$$

A esta aproximación, $y_1 = y_0 + hf(x_0,y_0)$, se le llama "Fórmula de Euler". Es claro que esta aproximación será adecuada sólo si h es pequeño.



```
# Euler, un solo paso
a  <- -0.007
b  <- 6
h  <- .5
x0 <- 0
y0 <- 10
C0 <- -log((a*y0+b)/y0)/b
x1 <- 0.5
ya <- b/(exp(-b*(x1+C0))-a)    # 164.2768

ff <- function(x,y){ return( y*(a*y+b) ) }

y1 <- y0+h*ff(x0,y0)  # 39.65

# grafica de Euler y analitica
a  <- -0.007
b  <- 6
y0 <- 10
C0 <- -log((a*y0+b)/y0)/b
xx <- seq(0,h,length=200)
yy <- b/(exp(-b*(xx+C0))-a)

plot(xx,yy,type="l",xlab="",ylab="",lwd=2,col="blue",cex.main=.9,
 ylim=c(0,165),xlim=c(-.05,.55),
 main="Solucion de y' = y(ay+b),  y(0)=10")
segments(x0,y0,x1,y1,col="red",lwd=2)
points(c(x1,x1),c(ya,y1),pch=20)
legend(0,150,legend=c("Exacta","Euler (1 paso)"),col=c("blue","red"),
 lwd=2,bty="n")
```

Runge-Kutta (de 2 etapas). Dada la ecuación $dy = f(x,y)dx$, con condiciones iniciales $y(x_0) = y_0$, vimos que la solución está dada por

$$y(x) = y_0 + \int_{x_0}^{x} f(t, y(t))dt$$

Definamos $g(t) \equiv f(t, y(t))$. Necesitamos integrar a $g(t)$ en el intervalo (x_0, x_1). Ahora, en vez de suponer a g constante, como en el método de Euler, usamos una aproximación trapezoidal.

Es fácil ver que la ecuación de la recta que pasa por los puntos $(x_0, g(x_0))$ y $(x_1, g(x_1))$ es

$$g(x) \approx g(x_0) + \frac{g(x_1) - g(x_0)}{x_1 - x_0}(x - x_0)$$

entonces

$$\begin{aligned} y(x) &= y_0 + \int_{x_0}^{x} f(t, y(t))dt = y_0 + \int_{x_0}^{x} g(t)dt \\ &\approx y_0 + \int_{x_0}^{x} \left(g(x_0) + \frac{g(x_1) - g(x_0)}{x_1 - x_0}(t - x_0) \right) dt \end{aligned}$$

Ahora, despúes de integrar, tenemos que

$$y(x_1) \approx y_0 + \frac{h}{2}\left[g(x_0) + g(x_1)\right]$$

y aquí hay un detalle importante, $g(x_0) = f(x_0, y(x_0)) = f(x_0, y_0)$ pero $g(x_1) = f(x_1, y(x_1)) = f(x_1, y_1)$ y no conocemos y_1 (pues es precisamente lo tratamos de evaluar!), entonces el método Runge-Kutta consiste en

sustituír ahí la aproximación de Euler $y_1 = y_0 + hf(x_0, y_0)$, donde $h = x_1 - x_0$:

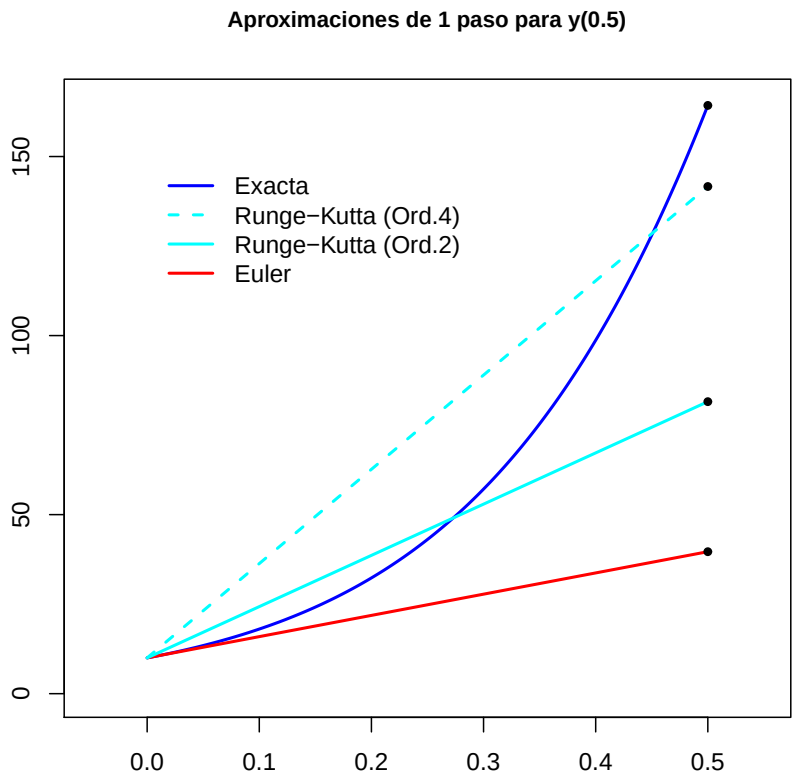
$$y(x_1) \approx y_0 + \frac{h}{2}\left[f(x_0, y_0) + f(x_1, y_1)\right] = y_0 + \frac{h}{2}\left[f(x_0, y_0) + f(x_0 + h, y_0 + hf(x_0, y_0))\right]$$

Resumiendo, el llamado método Runge-Kutta de segundo orden consiste en calcular:

$$\begin{aligned} y_1 = y(x_1) &= y_0 + \frac{1}{2}(k_1 + k_2) \\ \text{donde} \qquad k_1 &= hf(x_0, y_0) \\ k_2 &= hf(x_0 + h, y_0 + k_1) \end{aligned}$$

Enseguida tenemos una implementación en R de este método:

```
# Runge-Kutta (de 2 etapas o 2o orden), un solo paso
# y'=6y - .007y^2
a    <- -0.007
b    <- 6
ff   <- function(x,y){ return( y*(a*y+b) ) }
h    <- .5
x0   <- 0
x1   <- 0.5
x    <- seq(x0,x1,by=h)
m    <- length(x)
y    <- rep(0,m)
y[1] <- 10                        # condiciones iniciales
y0   <- 10
C0   <- -log((a*y0+b)/y0)/b
y1   <- y0+h*ff(x0,y0)            #  39.65    Euler
ya   <- b/(exp(-b*(x1+C0))-a)     # 164.2768  Exacto

for( j in 2:m ){
 f0   <- ff(x[j-1],y[j-1])
 k1   <- h*f0
 k2   <- h*ff(x[j],y[j-1]+k1)
 y[j] <- y[j-1]+(k1+k2)/2 }       # Runge-Kutta-2
yrk2 <- y[m]                      # 81.54879

# grafica de Euler y analitica y RK2
xx <- seq(0,h,length=200)
yy <- b/(exp(-b*(xx+C0))-a)

plot(xx,yy,type="l",xlab="",ylab="",lwd=2,col="blue",cex.main=.9,
 ylim=c(0,165),xlim=c(-.05,.55),
 main="Solucin de y' = y(ay+b),  y(0)=10")
segments(x0,y0,x1,y1,col="red",lwd=2)
segments(x0,y0,x1,yrk2,col="cyan",lwd=2)
points(c(x1,x1,x1),c(ya,y1,yrk2),pch=20)
legend(0,150,legend=
 c("Exacta","Euler (1 paso)","Runge-Kutta (Ord.2)"),
 col=c("blue","red","cyan"),lwd=2,bty="n")
```



(Nota: El método Runge-Kutta de orden 4 lo dejaremos para la sesión de prácticas)

Estimación de Densidades. Los datos del archivo `geyser` se refieren a mediciones tomadas en el géiser "Olf Faithful" de un parque en Estados Unidos. Se tienen registros del tiempo entre erupción y erupción, así como del tiempo de duración. Deseamos explorar la naturaleza de la variable "waiting" (tiempos entre erupciones).

```
library(MASS)
# datos geyser, 299 x 2 waiting, duration
attach(geyser)
```

El estimador más simple de la densidad es el Histograma que todos hemos visto alguna vez. Los estimadores de **Parzen** son una generalización del histograma. Suponga que tenemos datos $x_1, x_2, \cdots, x_n$ y deseamos estimar la densidad en un valor particular de $x = x_0$. Es intuitivo que si hay muchos valores de x_i's cercanos a x_0 entonces la densidad en x_0 es alta, y si no, entonces la densidad debe ser pequeña. Los estimadores de Parzen implementan esta idea simple.

El estimador de Parzen con kernel, K y ancho de banda h es

$$\widehat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

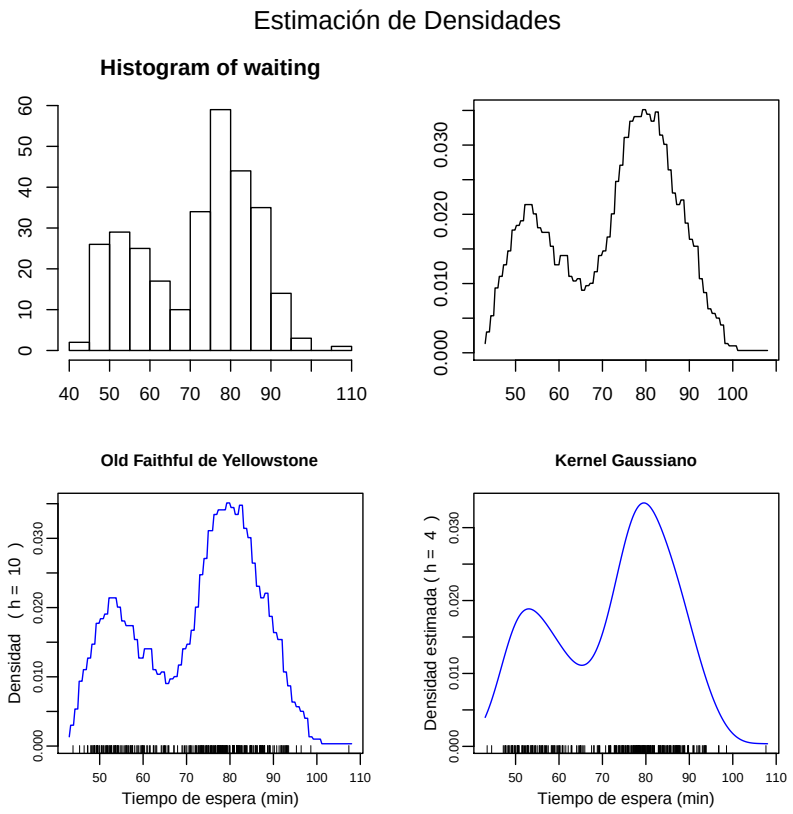
donde $K(u)$ es una densidad adecuada. En particular, el kernel rectangular es

$$K(u) = \begin{cases} 1 & \text{si } |u| < \frac{1}{2} \\ 0 & \text{de otra forma} \end{cases}$$

Uno de los kerneles más usado es el Gaussiano, definido por:

$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}$$

La siguiente gráfica muestra el histograma de los datos de tiempos de espera, luego tenemos dos implementaciones del estimador con ventana rectangular y finalmente está el estimador de Parzen con kernel Gaussiano.

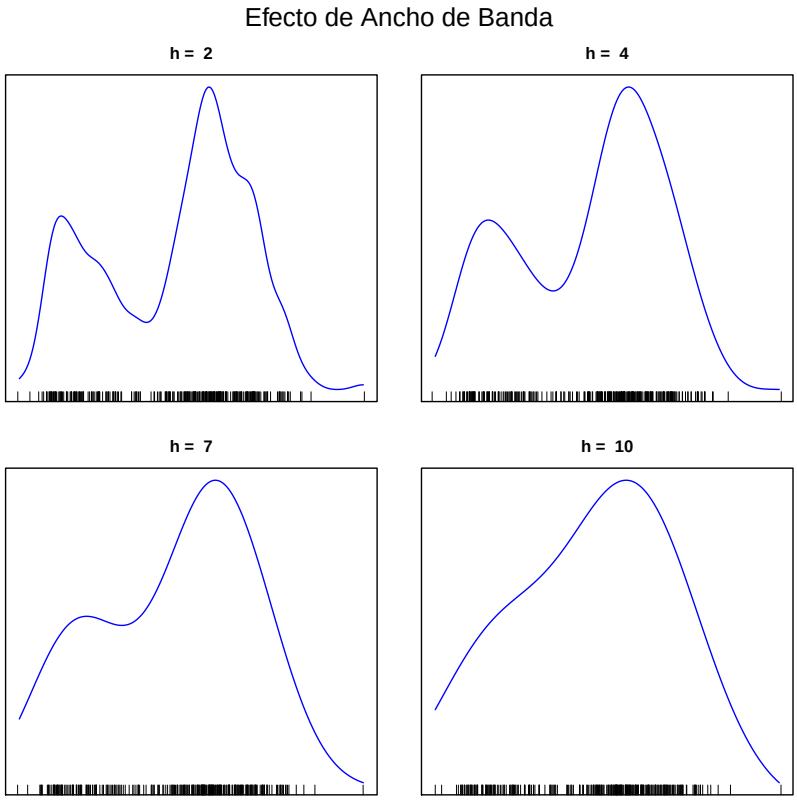


El código correspondiente en R:

```
h <- 10
n <- length(waiting)
k <- function(x){
 aa <- ifelse( (waiting<x+h/2) & (waiting>x-h/2),1,0)
 return( sum(aa)/(n*h) )}

m  <- 200
xx <- seq(min(waiting),max(waiting),length=m)
ff <- rep(0,m)
for(j in 1:m){ ff[j] <- k(xx[j]) }

par(mfrow=c(2,2), mar=c(3, 3, 3, 2), oma=c(0,0,2,0))
# Version 1: El histograma
hist(waiting)

# Version 2: Parzen con kernel rectangular
plot(xx,ff,type="l")

# Version 3: Igual que version 2 pero mas adornado.
```

```
plot(xx,ff,type="l",mgp=c(1.5,.5,0), xlab="Tiempo de espera (min)",
 ylab=paste("Densidad   ( h = ",h," )"),cex.lab=.9, cex.axis=.7, col="blue",
 main="Old Faithful de Yellowstone",cex.main=.9)
rug(jitter(waiting,amount=1))

# Usando un kernel gaussiano
h <- 4
n <- length(waiting)
k <- function(x){ return(sum(dnorm( (x-waiting)/h ))/(n*h)) }
m  <- 200
xx <- seq(min(waiting),max(waiting),length=m)
ff <- rep(0,m)
for(j in 1:m){ ff[j] <- k(xx[j]) }
# Version 4: Parzen con kernel Gaussiano
plot(xx,ff,type="l",mgp=c(1.5,.5,0), xlab="Tiempo de espera (min)",
 ylab=paste("Densidad estimada ( h = ",h," )"),
 cex.lab=.9, cex.axis=.7, col="blue",
 main="Kernel Gaussiano",cex.main=.9)
rug(jitter(waiting,amount=1))
mtext("Estimacion de Densidades", outer=TRUE, cex=1.2)
```

En la práctica siempre se nos presenta el problema de decidir el ancho de banda que debemos usar. Hay varias técnicas para ello, pero aquí simplemente ilustramos cuál es el efecto de usar diferentes anchos de banda.




```
# Seleccion de ancho de banda (Silverman)
sig <- sd(waiting)
uu  <- quantile(waiting,p=c(.25,.75))              # tambien: IQR()
ho  <- 0.9 * min(sig,(uu[2]-uu[1])/1.34) / n^0.2   # 3.998
```

```
# Comparacion de anchos de banda (kernel gaussiano)
par(mfrow=c(2,2), mar=c(1, 1, 2, 1), oma=c(0,0,2,0))
for(h in c(2,4,7,10)){
 ff <- rep(0,m)
 for(j in 1:m){ ff[j] <- k(xx[j]) }
 plot(xx,ff,type="l",mgp=c(1.5,.5,0), xlab="",yaxt="n",xaxt="n",
  ylab="",cex.lab=.9, cex.axis=.9, col="blue",
  main=paste( "h = ", h ),cex.main=.9)
 rug(jitter(waiting,amount=1)) }
mtext("Efecto de Ancho de Banda", outer=TRUE, cex=1.2)
```

En general, mientras más grande es el ancho de banda, más suavizado es el estimador y viceversa, mientras más pequeño es, más variable resulta. Lo ideal es usar un ancho de banda que sea un balance entre características globales y locales.

Prácticas Sesión 2

1. Considere la siguiente secuencia de símbolos

$$A \quad A \quad B \quad B \quad B \quad A \quad B \quad B \quad B \quad B \quad A \quad A \quad B \quad A$$

diremos que, en esta secuencia, hay 7 "rachas":

$$\underline{A \quad A} \quad \underline{B \quad B \quad B} \quad \underline{A} \quad \underline{B \quad B \quad B \quad B} \quad \underline{A \quad A} \quad \underline{B} \quad \underline{A}$$

El número de rachas en una secuencia nos da un indicador del grado de aleatoriedad en el que los símbolos A's y B's se encuentran ordenados. Muchas rachas o muy pocas rachas nos dirían que los símbolos **no** aparecen al azar; por ejemplo, 13 y 2 rachas en, respectivamente:

$$\underline{B} \; \underline{A} \; \underline{B} \; \underline{A} \; \underline{B} \; \underline{A} \; \underline{B} \; \underline{A} \; \underline{B} \; \underline{A} \; \underline{B} \; \underline{A} \; \underline{B \; B} \quad \text{y} \quad \underline{A \; A \; A \; A \; A \; A} \; \underline{B \; B \; B \; B \; B \; B \; B \; B}$$

Considere las hipótesis

H_0 : Los símbolos están en orden aleatorio vs H_1 : Los símbolos **no** están en orden aleatorio

y considere el estadístico de prueba $r = \#$ de rachas. **Encuentre una región de rechazo para H_0 correspondiente a un nivel de significancia $\alpha = .05$.**

Algunas posibles regiones de rechazo son, por ejemplo, $R = \{2\}$, $R = \{13\}$, $R = \{2, 13\}$, $R = \{2, 3, 12, 13\}$ etc.; la cuestión es, ¿Cuál tiene el tamaño más cercano a .05?, en otras palabras, ¿Cuál región es tal que $P(r \in R) \approx .05$? (la probabilidad se calcula bajo el supuesto de que H_0 es verdadera). Para este problema es claro que necesitamos saber la distribución de r cuando los símbolos están en orden aleatorio. Para este fin usaremos simulación. Por ejemplo, podemos hacer:

```
r     <- rep(0,M)              # donde M es muy grande
for( i in 1:M){
 orden <- sample( c(0,0,0,0,0,0,1,1,1,1,1,1,1,1) )
 r[i]  <- cuenta( orden ) }
```

donde `cuenta` es una función que cuenta las rachas (esta es la parte laboriosa). Al final `r` es un vector de longitud `M` y con ello nos puede dar una buena idea de su distribución; la región de rechazo se obtendría de las colas de esa distribución.

2. Para aprovechar el trabajo del problema anterior, consideremos las últimas tendencias de cierta acción de la bolsa. Suponga que dicha acción está en nuestro portafolio de inversión:

$$\uparrow \quad \uparrow \quad \uparrow \quad \downarrow \quad \uparrow \quad \uparrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \uparrow \quad \downarrow \quad \downarrow \quad \downarrow$$

Suponga que las caídas y bajadas son más o menos del mismo nivel. El comportamiento de la acción, ¿es consistente con una variabilidad aleatoria?

3. Si A es una matriz simétrica $n \times n$, entonces la descomposición espectral de A es $A = VDV^T$, donde V es la matriz $n \times n$ de vectores propios de A y D es una matriz diagonal con los correspondientes valores propios. Para la matriz simétrica A

```
B <- matrix(rnorm(100),ncol=10)
A <- round(t(B)%*%B,3)
```

encuentre las matrices V y D de la descomposición espectral y verifique que $A = VDV^T$ (pueden usar `eigen()`).

4. Las siguientes líneas muestran como resolver el sistema de ecuaciones $Ax = b$, de dos formas:

```
n    <- 1000
A    <- matrix( runif(n^2), ncol=n )
b    <- runif(n)
met1 <- solve(A)%*%b
met2 <- solve(A,b)
```

   Use la función `system.time()` para investigar cual de los dos métodos tiene un mejor desempeño en términos de tiempo de ejecución.

5. La siguiente función calcula, en forma recursiva, el factorial de un número

$$\texttt{fact(n)} = n(n-1)(n-2)\cdots 1$$

```
fact <- function(n){
 if(n==1){ return(1) }
 return( n*fact(n-1) )}
```

   Explique como es que este programa funciona.

6. Los primeros términos de la sucesión de Fibonacci son

$$1 \quad 1 \quad 2 \quad 3 \quad 5 \quad 8 \quad 13 \quad 21$$

   En las prácticas de la Sesión 1 se obtuvieron funciones para calcular el término n-ésimo de esta sucesión.

   (a) Escriba un programa que calcule el n-ésimo término de la sucesión, usando el concepto de funciones recursivas.

   (b) Compare, usando `system.time()`, el desempeño del programa del inciso anterior, contra el desempeño de la correspondiente función desarrollada en la sesión anterior. ¿Es buena idea usar recursión?.

7. Considere los siguientes conjuntos de datos (tomados de Anscombe, F.J. (1973) *American Statistician*)

X_1	10	8	13	9	11	14	6	4	12	7	5
Y_1	8.04	6.95	7.58	8.81	8.33	9.96	7.24	4.26	10.84	4.82	5.68
X_2	10	8	13	9	11	14	6	4	12	7	5
Y_2	9.14	8.14	8.74	8.77	9.26	8.10	6.13	3.10	9.13	7.26	4.74
X_3	10	8	13	9	11	14	6	4	12	7	5
Y_3	7.46	6.77	12.74	7.11	7.81	8.84	6.08	5.39	8.15	6.42	5.73
X_4	8	8	8	8	8	8	8	19	8	8	8
Y_4	6.58	5.76	7.71	8.84	8.47	7.04	5.25	12.50	5.56	7.91	6.89

   (a) Ajuste modelos de regresión lineal simple usando, por ejemplo `summary( lm( y ~ x ) )` . Muestre que los 4 conjuntos de datos tienen la misma media en las x's, la misma media en las y's, la misma correlación entre las x's y y's, el mismo intercepto, la misma pendiente, el mismo R^2 y la misma estimación de la varianza.

   (b) Sin embargo, muestre gráficamente que los 4 conjuntos de datos son muy distintos. En otras palabras, no hay que ponerle mucha fé a los números.

8. **Cálculo de la función potencia**. Suponga que $x_1, x_2, \cdots, x_n$ son i.i.d. $N(\mu, \sigma^2)$, con σ^2 conocida. Deseamos contrastar las hipótesis

$$H_0 : \mu = \mu_0 \qquad \text{versus} \qquad H_1 : \mu > \mu_0$$

Para fijar ideas, suponga $\mu_0 = 10$. Como sabemos, una prueba adecuada de nivel $\alpha = .05$ consiste en:

$$\text{Rechazar} \quad H_0 \quad \text{si} \quad z = \frac{\bar{x} - \mu_0}{\sqrt{\sigma^2/n}} > z_\alpha$$

Recuerde que el nivel de significancia, α, de la prueba, es la probabilidad (que nosotros aceptamos) de cometer un error tipo I. En este problema estamos interesados en investigar el **poder** de la prueba. Esto es, queremos saber, por ejemplo, que si el valor real de μ fuera, digamos, 11, ¿Cuál sería la probabilidad de que rechazemos H_0? (lo cual es algo deseable, pues, en el caso $\mu = 11$, la hipótesis de que $\mu = 10$ sería falsa). Ahora bien, el poder, evaluado en $\mu = \mu_1 = 11$ es

$$\text{Poder}(\mu_1) = P\left(\frac{\bar{x} - \mu_0}{\sqrt{\sigma^2/n}} > z_\alpha \quad | \quad \mu_1 \text{ es la media verdadera} \right)$$

Haciendo operaciones, es fácil ver que el poder se calcula como:

$$\text{Poder}(\mu_1) = P\left(\frac{\bar{x} - \mu_1}{\sqrt{\sigma^2/n}} > z_\alpha - \frac{\mu_1 - \mu_0}{\sqrt{\sigma^2/n}} \quad | \quad \mu_1 \right)$$

de aquí que

$$\text{Poder}(\mu_1) = P\left(Z > z_\alpha - \frac{\mu_1 - \mu_0}{\sqrt{\sigma^2/n}} \right)$$

donde Z es una variable aleatoria normal estándar.

(a) Suponga $\sigma^2 = 4$ y $n = 10$. Construya una gráfica de la función potencia.

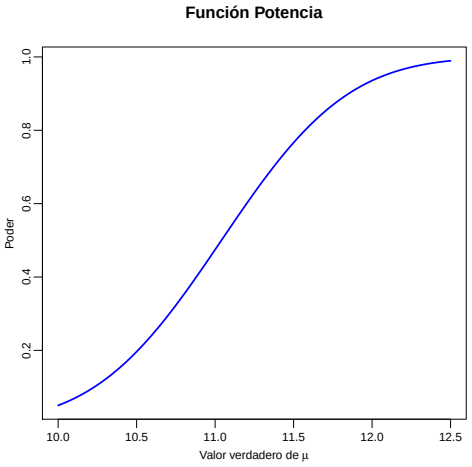
(b) Suponga nuevamente que $\sigma^2 = 4$, pero ahora $n = 30$. Calcule la función potencia y grafique esta función en la gráfica del inciso anterior.

(c) Comente sobre el efecto del tamaño de muestra sobre el poder de la prueba.

(d) Notará que el valor de la función potencia, cuando $\mu = 10$, no es 0 sino que es .05, ¿Porqué?.

(e) ¿Qué tamaño de muestra sería necesario tener si quisieramos tener una potencia de .95 en $\mu = 11$?

[Nota: La gráfica del inciso (a) se verá, más o menos, como la gráfica de la derecha]



9. **Uso del paquete `tcltk`**.

(a) Investige que es lo que hace el siguiente programa

```
library(tcltk)

grafica <- function(...){
 sig  <- as.numeric(tclvalue(sliderval))
```

```
 dat  <- rnorm(1000,mean=0,sd=sig)
 hist(dat,xlim=c(-3.5,3.5), main="Histograma de datos normales simulados",
      cex.main=1, xlab="x", ylab="", freq=FALSE, ylim=c(0,1),
      col="cyan", nclass=20, cex.axis=.7, cex.lab=.7, mgp=c(1.5,.5,0))
 zz <- seq(-3.5,3.5,length=200)
 lines(zz, dnorm(zz,mean=0,sd=sig), lwd=2, col="red") }

sliderinicio <- 1
slidermin    <- 0.2
slidermax    <- 2
sliderstep   <- 0.01
tt           <- tktoplevel()
sliderval    <- tclVar(sliderinicio)
slidervallab <- tklabel(tt, text=as.character(tclvalue(sliderval)))
tkgrid(tklabel(tt, text="Desviacion Estandar = "),
       slidervallab, tklabel(tt,text=""))
tkconfigure(slidervallab, textvariable=sliderval)
slider <- tkscale(tt, from=slidermin, to=slidermax, showvalue=F,
                  variable=sliderval, resolution=sliderstep, command=grafica)
tkgrid(slider)
tkgrid(tklabel(tt,text="valor de sigma"))
tkfocus(tt)
```

(b) Muestre, usando el concepto de "sliders", el efecto del ancho de banda en el estimador de densidades con kernel gaussiano. Use los datos de tiempos entre erupción y erupción del Geyser Old Faithful. En el ejemplo del inciso (a) nuevos datos se generaban para cada valor de la desviación estándar; ahora, los datos siempre serán los mismos pero lo que va a ir cambiando es el estimador de la densidad, para diferentes anchos de banda.

10. **Simulación de una caminata aleatoria**. Consideremos el caso de un tipo permanentemente bajo la influencia del alcohol. Cada día baja del autobús y toma hacia la izquierda rumbo al bar o hacia la derecha rumbo a su casa. Tanto el bar como su casa se encuentran a 2 cuadras de la parada del autobús.

bar		parada de autobús		casa
1	2	3	4	5

Su comportamiento es completamente aleatorio, en cada cuadra que avanza puede suceder que, o sigue avanzando o se regresa una cuadra. Sin embargo, si por fin llega ya sea al bar o a su casa, ahí se quedará. Así, por ejemplo, una posible caminata puede ser:

		3	4	3	2	1		

otra podría ser

3	2	3	4	3	2	3	4	5

En el primer caso, la longitud de la caminata es de 5 y en el segundo de 9. Diremos también que, en el primer caso, los puntos 1, 2, 3, 4 y 5 fueron visitados 1, 1, 2, 1 y 0 veces, respectivamente, mientras que en la segunda caminata, las respectivas frecuencias fueron 0, 2, 4, 2 y 1. Suponga que esta persona se mueve a la izquierda con probabilidad p y hacia la derecha con probabilidad $q = 1-p$. Use simulación para contestar lo siguiente (para fijar ideas, suponga que $p = 0.6$, sin embargo, en el programa que escriba, este valor deberá ser fácilmente cambiable a cualquier otro valor de p).

(a) ¿Cuál es la distribución de las longitudes de las caminatas?.

(b) ¿Con que probabilidades son visitados los puntos 1, 2, 3, 4 y 5?.

(c) Por lógica, uno pensaría que, si $p = 0.6$, entonces el borrachito termina en el bar aproximadamente el 60% de las veces, ¿Es esto así?. ¿Qué pasa si $p = 0.5$?.

Este es un ejemplo de lo que se llama una Cadena de Markov. A los estados 1 y 5 se les llama "absorbentes" y a 2, 3 y 4 se les llama "transitorios".

11. **Estimación de Parámetros usando Máxima Verosimilitud**. Recuerde que la densidad Gama se define como:

$$g(x) = \frac{\lambda^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\lambda x}$$

Los siguientes datos son el resultado de una prueba de vida sobre 8 amortiguadores. Cada amortiguador se probó en un aparato que los sometía a compresiones y estiramientos hasta que se rompían. Se registró el número de ciclos a la falla (las unidades son kilociclos)

Kilociclos a la falla :	190	245	265	300	320	325	370	400

Suponga que en este caso el modelo Gama es razonable; más aún, suponga que el valor de λ es conocido ($\lambda = 0.07$). Deseamos estimar el parámetro α usando Máxima Verosimilitud. Recuerde que la Verosimilitud es

$$L(\alpha) = \Pi_{i=1}^n g(x_i;\alpha) = \frac{\lambda^{n\alpha}}{\Gamma(\alpha)^n} (x_1 \cdot x_2 \cdots x_n)^{\alpha-1} e^{-\lambda \sum x_i}, \qquad \text{con} \quad \lambda = 0.07$$

La correspondiente logverosimilitud es

$$l(\alpha) = n\alpha \log(\lambda) - n \log\Gamma(\alpha) + (\alpha - 1) \sum_{i=1}^n \log(x_i) - \lambda \sum_{i=1}^n x_i, \qquad \text{con} \quad \lambda = 0.07$$

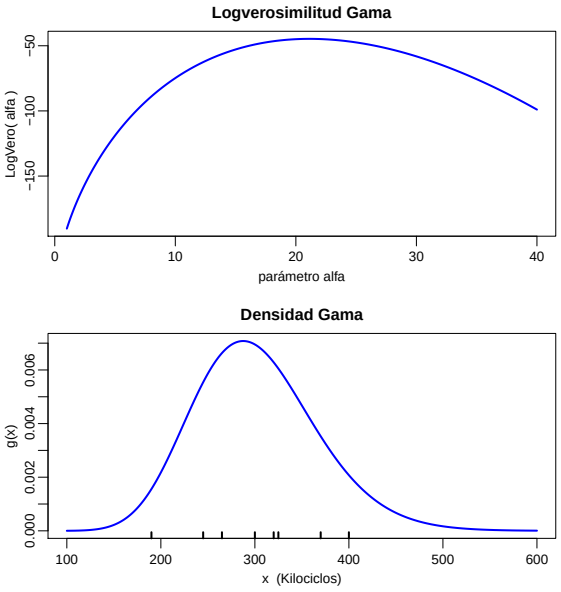
El estimador máximo verosímil de α es aquel valor que maximiza a $l(\alpha)$. La función `nlminb` de R sirve para minimizar funciones, así que la usaremos para minimizar el negativo de $l(\alpha)$.

(a) Haga una gráfica de la logverosimilitud $l(\alpha)$.

(b) Escriba una función que calcule el negativo de la logverosimilitud. Use la función `nlminb` para encontrar el estimador de α. La función `nlminb` tiene varios argumentos, pero con definir dos, en la mayoría de los casos, es suficiente: `nlminb(start,objective)`, donde `start` es un valor inicial para α (por ejemplo, dar el valor `start = 10`) y en `objective` ponemos el nombre de la función que se definió antes como el negativo de la logverosimilitud.

(c) Haga una gráfica de la función de densidad Gamma estimada; esto es, graficar $g(x)$ con los valores de $\lambda = 0.07$ y el valor de α encontrado en el inciso anterior.

[Nota: Las gráficas de los incisos (a) y (c) se ven, más o menos, como en las gráficas de la derecha]



12. **Runge-Kutta de orden 4**. Vimos que el método Runge-Kutta de orden 2 para aproximar $y_1 = y(x_1)$ en la ecuación diferencial $dy = f(x,y)dx$, con condición inicial $y(x_0) = y_0$, está dado por:

$$y_1 = y(x_1) = y_0 + \frac{1}{2}(k_1 + k_2)$$

donde $\quad k_1 = hf(x_0, y_0)$

$$k_2 = hf(x_0 + h, y_0 + k_1)$$

Los métodos Runge-Kutta extienden el método de Euler, bajo la idea de usar más evaluaciones de f dentro de un mismo paso, h. El Runge-Kutta de orden 4 (RK4) es una de estas extensiones –es de los métodos más usados en la práctica– y las ecuaciones correspondientes son:

$$y_1 = y(x_1) = y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

donde
$$k_1 = hf(x_0, y_0)$$
$$k_2 = hf(x_0 + h/2, y_0 + k_1/2)$$
$$k_3 = hf(x_0 + h/2, y_0 + k_2/2)$$
$$k_4 = hf(x_0 + h, y_0 + k_3)$$

Considere la ecuación $y' = y(ay + b)$, donde $a = -0.007$, $b = 6$ y la condición inicial es $y_0 = 10$. Usando el método RK4 encuentre una aproximación para $y_1 = y(0.5)$.

13. Calcule numéricamente el valor de la integral

$$\int_0^\infty \frac{dy}{(\cosh(y) - \rho r)^{n-1}}$$

donde $n = 10, r = 0.5$, y $\rho = 0.2$. Compare los resultados obtenidos usando la Regla de Simpson y lo que se obtiene usando `integrate()`.

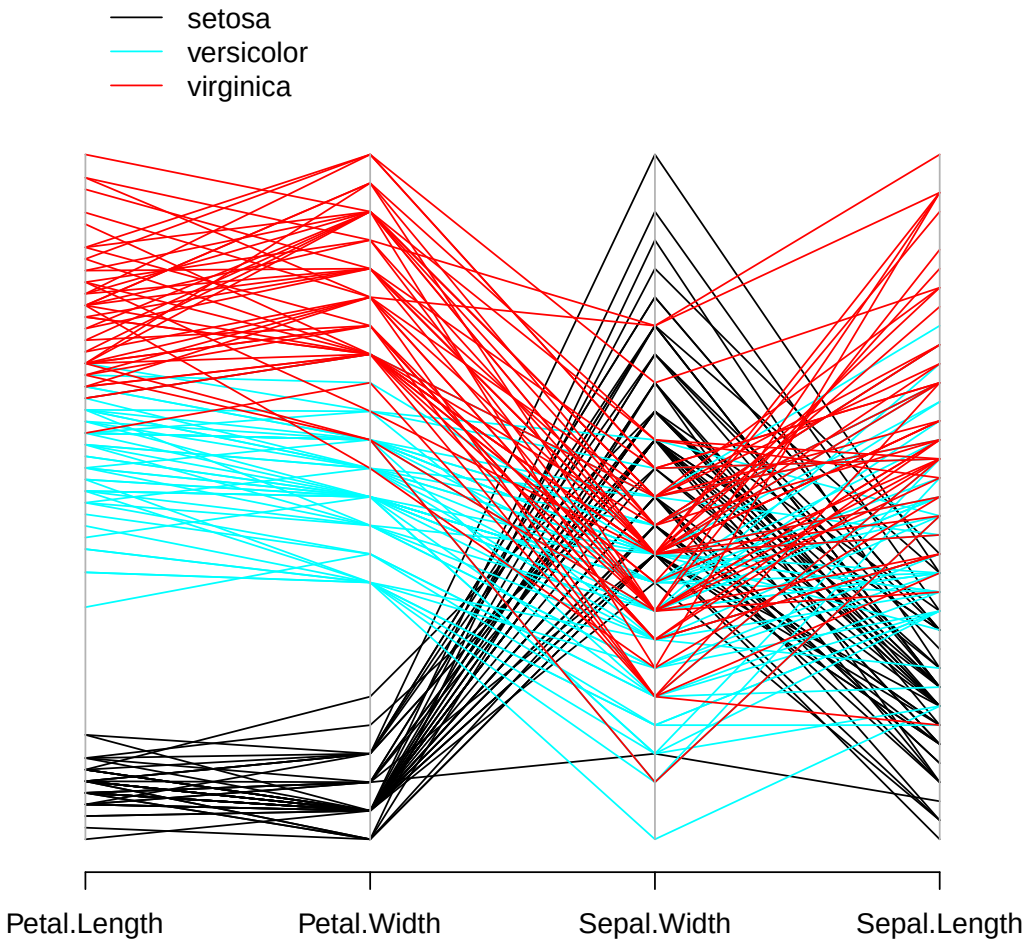
14. ¿Cuántas veces tengo que tirar un dado para estar 99% seguro de que obtengo al menos un 6?. Use simulación para contestar a esta pregunta.

15. **Gráficas con Coordenadas Paralelas.** El conjunto de datos `iris` tiene 150 renglones y 5 columnas. Los primeros 50 renglones tienen medidas de 50 flores iris de la especie Setosa; luego vienen 50 flores iris-versicolor y, finalmente hay 50 flores iris-virginica. Los datos se ven como sigue:

```
> iris[c(1,51,101),]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
51           7.0         3.2          4.7         1.4 versicolor
101          6.3         3.3          6.0         2.5  virginica
```

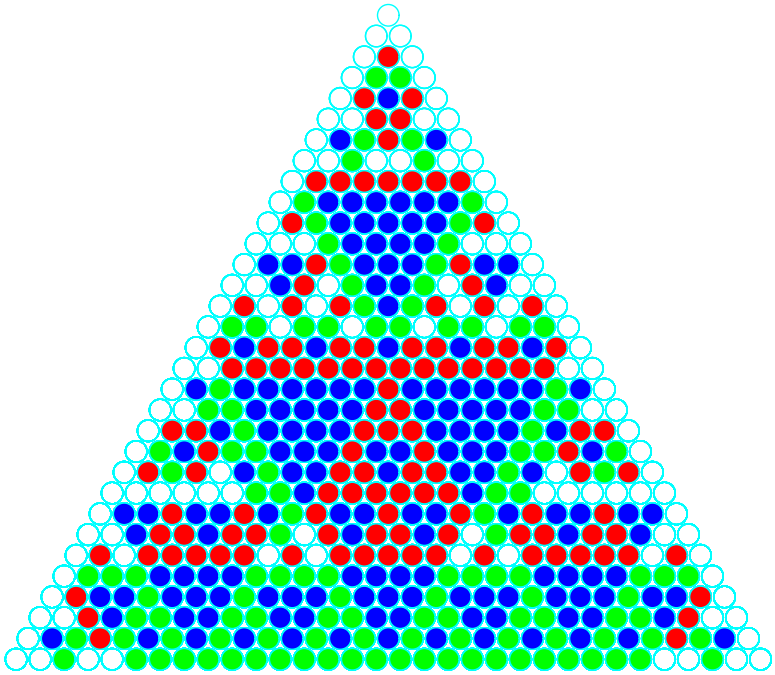
Deseamos explorar el uso de una técnica de visualización de datos multivariados: Las gráficas con coordenadas paralelas ("parallel coordinate plot"). Para ello examine el siguiente código:

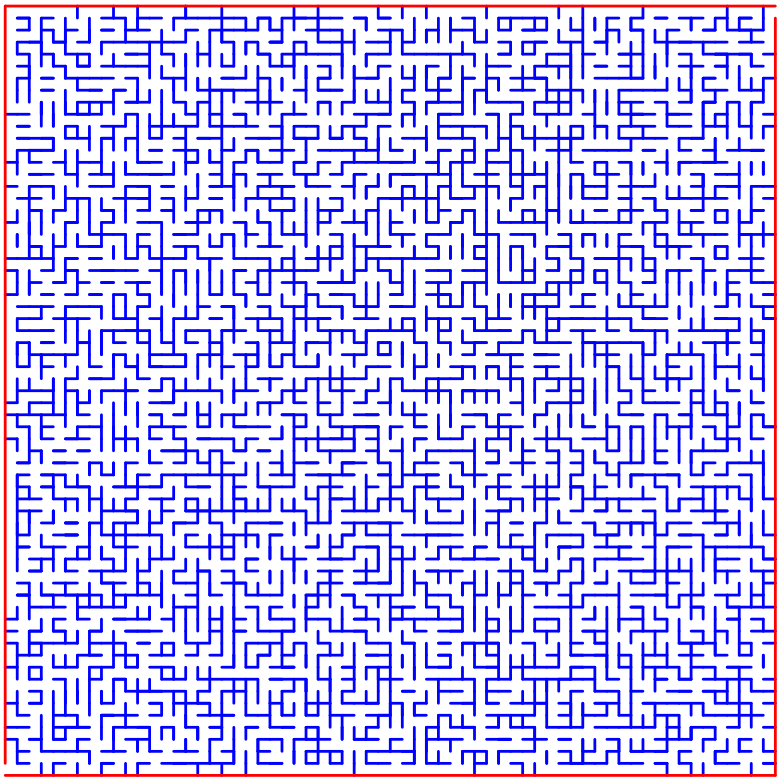
```
library(MASS)
iri <- iris[,c(3,4,2,1)]
par(mar=c(3,2,2,2))
cc  <- c(rep("black",50),rep("cyan",50),rep("red",50))
parcoord(iri,col=cc,pch="",ylim=c(0,1.2))
legend(1,1.25,legend=c("setosa","versicolor","virginica"),
       lty=1,col=c("black","cyan","red"),bty="n")

# Uso de tapply
tapply(iris[,3],iris[,5],summary)
```



Cada una de las 150 flores tiene 4 mediciones; estas 4 mediciones definen una línea quebrada, así que tenemos 150 líneas quebradas en esta gráfica. Es claro que las características de los pétalos son muy distintivas de cada especie; esto es, esas variables tienen "poder discriminatorio".

Apéndice Sesión 2





Apéndice Sesión 2

```
#########################################################################
# Figura 1: Triangulo de Pascal.
# Divisibilidad de coeficientes binomiales: Por 2 rojo, por 3 azul, por ambos verde.
library(grid)
n  <- 32                # n <- 64
x  <- 0.5
y  <- .95
h  <- .8/n
r  <- h/tan(pi/3)
c2 <- c3 <- 1
grid.circle(x=x,y=y,r=.9*r,default.units="npc",gp=gpar(fill="transparent",col="cyan"))
for( i in 2:n ){
 c2 <- (c(0,c2)+c(c2,0))%%2
 c3 <- (c(0,c3)+c(c3,0))%%3
 x  <- x-r
 y  <- y-h
 fi <- ifelse(c2==0,"red","transparent")
 fi <- ifelse(c3==0,"green",fi)
 fi <- ifelse((c2==0)&(c3==0),"blue",fi)
 for( j in 1:i ){
  xx <- seq(x,x+2*(i-1)*r,by=2*r)
  grid.circle(x=xx,y=y,r=.9*r, default.units="npc", gp=gpar(fill=fi,col="cyan"))}}
#########################################################################

#########################################################################
# Figura 2: Laberinto
par( mar=c(0,0,0,0) );   set.seed(632425)
plot(0,0, xlim=c(0,1), ylim=c(0,1), type="n", xaxt="n", yaxt="n",
     ylab="", xlab="", bty="n")
m  <- 6   # cuadrado unitario se divide en 2^m x 2^m cuadritos
x  <- seq(1/(2^m),(2^m-1)/(2^m),by=1/(2^m))
for(i in 1:(2^m-1)){
 for(j in 1:(2^m-1)){
  p <- c(x[i],x[j])
  if( runif(1) < .5 ){              # horizontal
   if( runif(1) < .5 ){             # izquierda
    q <- p - c(1/(2^m),0)
    }else{ q <- p + c(1/(2^m),0) }  # derecha
  }else{                            # vertical
    if( runif(1) < .5 ){
     q <- p - c(0,1/(2^m))          # abajo
    }else{ q <- p + c(0,1/(2^m)) }} # arriba
  segments( p[1], p[2], q[1], q[2], col="blue", lwd=2 )
  if(runif(1)<.2){
   a <- sample(c(0,1),size=1)
   segments(p[1], p[2], p[1]+a/(2^m), p[2]+(1-a)/(2^m), col="blue", lwd=2 )}}}
segments( 0, 1/(2^m), 0, 1, col="red", lwd=2 )
segments( 0, 1, 1, 1, col="red", lwd=2 )
segments( 1, (2^m-1)/(2^m), 1, 0, col="red", lwd=2 )
segments( 1, 0, 0, 0, col="red", lwd=2 )
#########################################################################
```

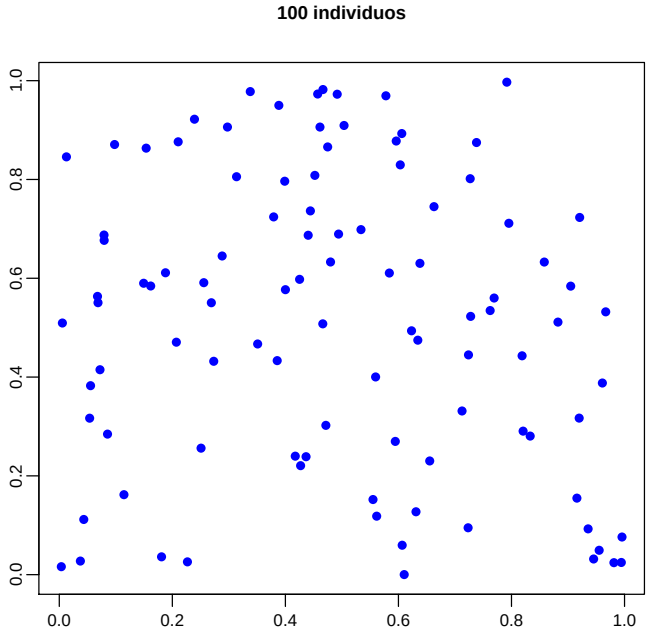
MIÉRCOLES

Notas Sesión 3

Algoritmo k-medias. Suponga que tenemos N individuos descritos por p características; esto es, tenemos

$$\begin{aligned} \text{Individuo 1}: &\quad x_{11}, x_{12}, \cdots, x_{1p} \\ \text{Individuo 2}: &\quad x_{21}, x_{22}, \cdots, x_{2p} \\ &\cdots \\ \text{Individuo N}: &\quad x_{N1}, x_{N2}, \cdots, x_{Np} \end{aligned}$$

Estos N individuos los visualizamos como N puntos en $\mathbb{R}^p$ y, en estudios exploratorios, puede ser de interés encontrar grupos de individuos con características similares con la esperanza de que estos grupos puedan aportar conocimiento en el contexto del problema. La técnica de k-medias es simple y muy usada. Supongamos que la siguiente gráfica muestra a 100 individuos en $\mathbb{R}^2$:

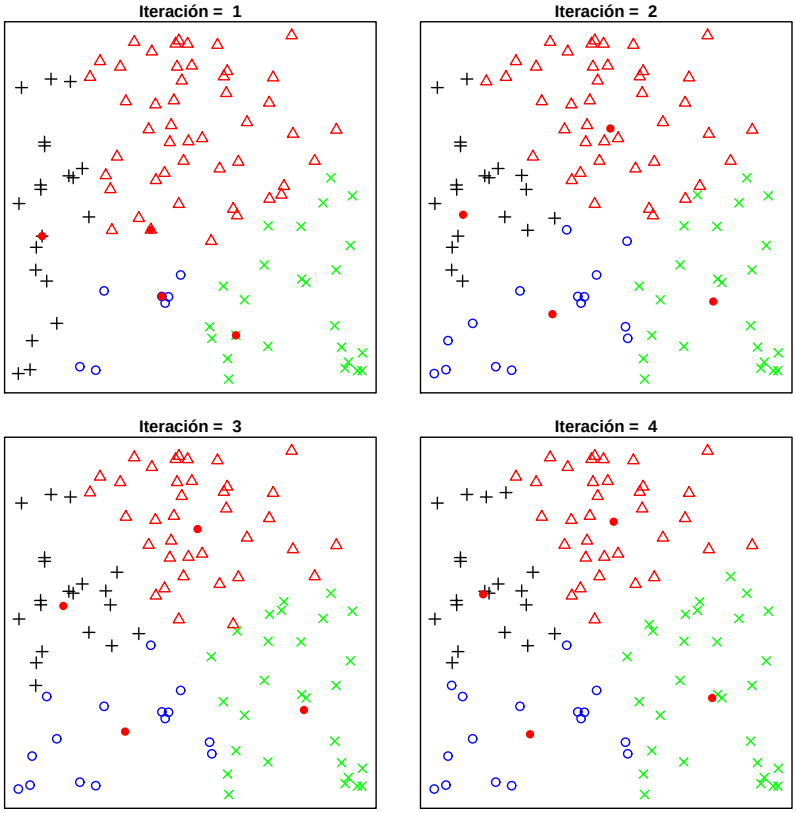



```
set.seed(565656)
n     <- 100
X     <- matrix(runif(2*n),n,2)
plot(X[,1],X[,2], col="blue",cex.main=.9,pch=16,xlab="",
 ylab="",mgp=c(1.5,.5,0), cex.axis=.8,main="100 individuos")
```

Supongamos que queremos encontrar $K = 4$ grupos. El algoritmo procede como sigue:

1. Elegir K puntos al azar (de los N) para que sean los centroides de los K grupos iniciales
2. Calcular la distancia de cada punto a cada uno de los K centroides
3. Asignar cada individuo al grupo asociado al centroide más cercano
4. Recalcular los centroides e iterar los pasos 2,3,4.

El algoritmo termina hasta que se logre algún tipo de equilibrio en la formación de los grupos. A continuación mostramos las 4 primeras iteraciones del algoritmo.

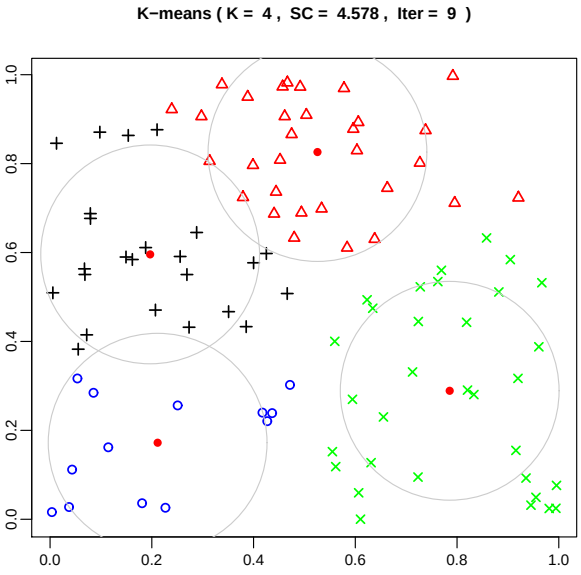



```
par(mfrow=c(2,2),mar=c(1, 1, 1, 1))

set.seed(565656)
n      <- 100
X      <- matrix(runif(2*n),n,2)
K      <- 4
cent   <- X[sample(n,K),]

for(i in 1:4){
 dista <- matrix(0,n,K)
 for(j in 1:n){ dista[j,] <- colSums(  (X[j,]-t(cent))^2  ) }
 cual  <- function(x){ which(x==min(x))[1] }
 grupo <- apply( dista, 1, cual )
 cols  <- c("blue","red","black","green","cyan","orange","yellow","brown")
 cc    <- cols[grupo]
 plot(X[,1],X[,2],pch=grupo, col=cc,cex.main=.9,xaxt="n",yaxt="n",
  xlab="",ylab="",mgp=c(1.5,.5,0), cex.axis=.8,
  main=paste("Iteracin = ",i))
 points(cent[,1],cent[,2],pch=16,col="red")
 for( j in 1:K ){
  cent[j,] <- colMeans(X[grupo==j,]) } }
```

Este programa está escrito para un cierto número fijo de iteraciones. A continuación presentamos una modificación que incorpora un criterio de paro (¿cuál es este criterio?).

```
dis  <- function(x){
 aa <- colSums((x-t(cent))^2)
 return(which(aa==min(aa))[1])}
```

```
set.seed(565656)
n      <- 100
X      <- matrix(runif(2*n),n,2)
K      <- 4
cent   <- X[sample(n,K),]
cols   <- c("blue","red","black","green","cyan","orange","yellow","brown")
tol    <- 1e-6
seguir <- TRUE
obj    <- NULL
iter   <- 0
while(seguir){
 iter  <- iter+1
 cento <- cent
 grupo <- apply(X,1,dis)
 for(i in 1:K){ cent[i,] <- colMeans(X[grupo==i,]) }
 crit  <- sum((cent-cento)^2)
 if( crit<tol ){ seguir <- FALSE } }
vv   <- function(A){sum((scale(A,scale=F))^2)}
bb   <- sum( by(X,grupo,vv) )
cc   <- cols[grupo]
plot(X[,1],X[,2],pch=grupo, col=cc,cex.main=.9,
 xlab="",ylab="",mgp=c(1.5,.5,0), cex.axis=.8, lwd=1.5,
 main=paste("K-means ( K = ",K,",  SC = ",round(bb,3),",  Iter = ",iter," )"))
symbols(cent[,1],cent[,2],circles=rep(1,K),add=T,fg=gray(.8))
points(cent[,1],cent[,2],pch=16,col="red")
```



Debemos anotar que si a $k-$medias le pedimos 4 grupos, se obtendrán 4 grupos. Si le pedimos 5, encontrará 5. Así que no debemos forzar interpretaciones acerca de la naturaleza de los grupos. $k-$ medias es una técnica exploratoria.

Escalamiento Multidimensional. La siguiente tabla muestra datos de 22 compañías productoras de electricidad en Estados Unidos. Nos interesa producir una imágen bidimensional que, en algún sentido, sea una buena representación de los datos. Este problema cae dentro de la clase de los llamados problemas de "reducción de dimensionalidad". Una de las técnicas más usadas para esta clase de problemas es el Análisis de Componentes Principales, lo cual no trataremos en este curso. Aquí estaremos interesados en aplicar la técnica de Escalamiento Multidimensional.

Co.	X1	X2	X3	X4	X5	X6	X7	X8
Arizona Public Service	1.06	9.2	151	54.4	1.6	9077	0.0	0.628
Boston Edison Co.	0.89	10.3	202	57.9	2.2	5088	25.3	1.555
Central Louisiana Electric Co.	1.43	15.4	113	53.0	3.4	9212	0.0	1.058
Commonwealth Edison Co.	1.02	11.2	168	56.0	0.3	6423	34.3	0.700
Consolidated Edison Co. (NY)	1.49	8.8	192	51.2	1.0	3300	15.6	2.044
Florida Power & Light Co.	1.32	13.5	111	60.0	-2.2	11127	22.5	1.241
Hawaiian Electric Co.	1.22	12.2	175	67.6	2.2	7642	0.0	1.652
Idaho Power Co.	1.10	9.2	245	57.0	3.3	13082	0.0	0.309
Kentucky Utilities Co.	1.34	13.0	168	60.4	7.2	8406	0.0	0.862
Madison Gas & Electric Co.	1.12	12.4	197	53.0	2.7	6455	39.2	0.623
Nevada Power Co.	0.75	7.5	173	51.5	6.5	17441	0.0	0.768
New England Electric Co.	1.13	10.9	178	62.0	3.7	6154	0.0	1.897
Northern States Power Co.	1.15	12.7	199	53.7	6.4	7179	50.2	0.527
Oklahoma Gas & Electric Co.	1.09	12.0	96	49.8	1.4	9673	0.0	0.588
Pacific Gas & Electric Co.	0.96	7.6	164	62.2	-0.1	6468	0.9	1.400
Puget Sound Power & Light Co.	1.16	9.9	252	56.0	9.2	15991	0.0	0.620
San Diego Gas & Electric Co.	0.76	6.4	136	61.9	9.0	5714	8.3	1.920
The Southern Co.	1.05	12.6	150	56.7	2.7	10140	0.0	1.108
Texas Utilities Co.	1.16	11.7	104	54.0	-2.1	13507	0.0	0.636
Wisconsin Electric Power Co.	1.20	11.8	148	59.9	3.5	7287	41.1	0.702
United Illuminating Co.	1.04	8.6	204	61.0	3.5	6650	0.0	2.116
Virginia Electric & Power Co.	1.07	9.3	174	54.3	5.9	10093	26.6	1.306

El elemento básico en Escalamiento Multidimensional es una matriz de disimilaridades

$$\Delta = \begin{pmatrix} \delta_{11} & \delta_{12} & \cdots & \delta_{1n} \\ \delta_{21} & \delta_{22} & \cdots & \delta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{n1} & \delta_{n2} & \cdots & \delta_{nn} \end{pmatrix}$$

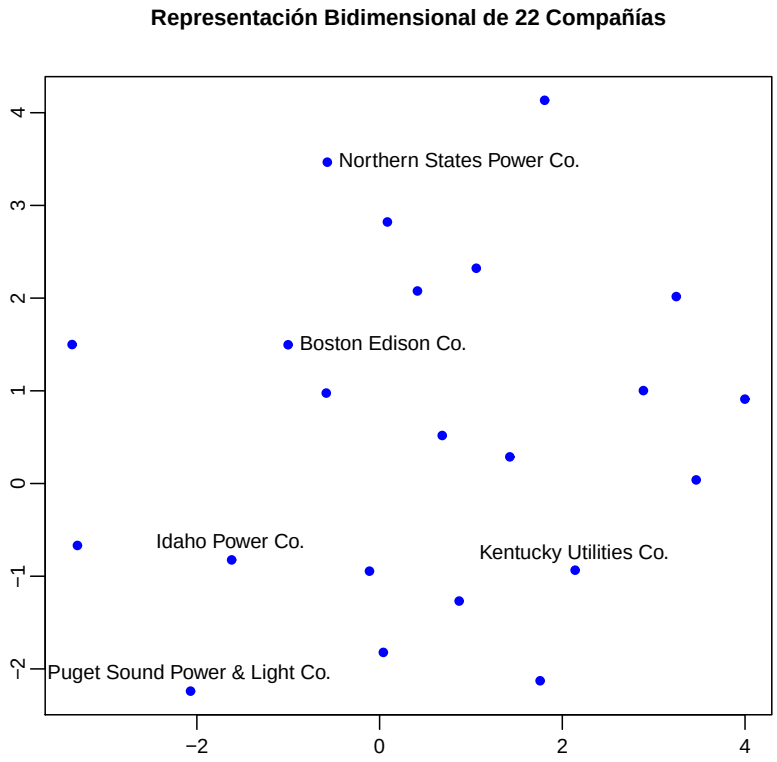
y queremos encontrar vectores $x_1, x_2, \cdots, x_n$ (típicamente en $\mathbb{R}^2$) tales que sean representantes de los n vectores originales (en el caso de los datos de las $n = 22$ compañías, tendríamos que estos vectores originales están en $\mathbb{R}^8$). Una forma de lograr esto es mediante la minimización de, por ejemplo:

$$\sum_{i<j} (||x_i - x_j|| - \delta_{ij})^2$$

donde las variables sobre las cuáles se hace la minimización son $x_1, \cdots, x_n$. A continuación presentamos una implementación en R de estas ideas:

```
# Escalamiento Multidimensional
datos  <- read.csv(
 "c:\\Documents and Settings\\Verano2010\\PublicUtilitiesData.csv",header=TRUE)
X    <- datos[,-1]
X    <- scale(X)
n    <- dim(X)[1]
dX   <- dist(X)
```

```
stress <- function(p){
 coo <- matrix(p,ncol=2)
 aa  <- dist(coo)
 return(sum((aa-dX)^2))}
ini <- runif(2*n)
out <- nlminb(ini,stress)
out$conv
out$obj
coo <- matrix(out$par,ncol=2)
plot(coo[,1],coo[,2],pch=20, col="blue",cex.main=.9,
 xlab="",ylab="",mgp=c(1.5,.5,0), cex.axis=.8, lwd=1.5,
 main="Representacion Bidimensional de 22 Companias")
identify(coo[,1],coo[,2],n=5,labels=datos[,1],cex=.8)
```



El tipo de datos que usamos para ilustrar Escalamiento Multidimensional no es, en realidad, el tipo de datos típico en donde se usa esta técnica; los usos más prevalentes son en situaciones en las que lo único que tenemos es la matriz de disimilaridades, por ejemplo, que proviene de estudios de mercado en los que a los entrevistados se les presentan parejas de productos y van expresando sus opiniones en cuanto a la semejanza o diferencias entre ellos y, al final, se calcula algún índice de disimilaridad entre todos los productos y se procede a elaborar una representación bidimensional de los productos que facilite llegar a conclusiones acerca de los diferentes productos en competencia.

Recocido simulado. El algoritmo de Recocido Simulado (en inglés: Simulated Annealing) fué introducido por Metrópolis $et\ al.$ en los 50's. Es un algoritmo de optimización donde las direcciones y longitudes de paso son de naturaleza estocástica. Este método es de los precursores de los métodos MCMC que son ampliamente usados en estadística Bayesiana.

Supongamos que deseamos minimizar la función $h(x)$. El algoritmo produce una secuencia

$$x_0, x_1, x_2, \cdots, x_i, \cdots$$

en la que los diferentes valores de las x_i's, por diseño, van explorando la región del mínimo, pero en ocasiones toman excursiones para explorar otras regiones, no necesariamente mejores. Esta característica del algoritmo la hace particularmente valiosa para problemas con mínimos locales, a diferencia de los algoritmos "greedy" que típicamente pueden quedarse atrapados en estos mínimos, el recocido simulado tiene la posibilidad de salirse de estos hoyos.

Este algoritmo de exploración estocástica tiene la siguiente forma:

- Definir el valor inicial x_0
- Definir el programa de "temperatura", para $i = 1, 2, \cdots$

  $$\tau_i = \frac{r}{\log(1+i)}$$

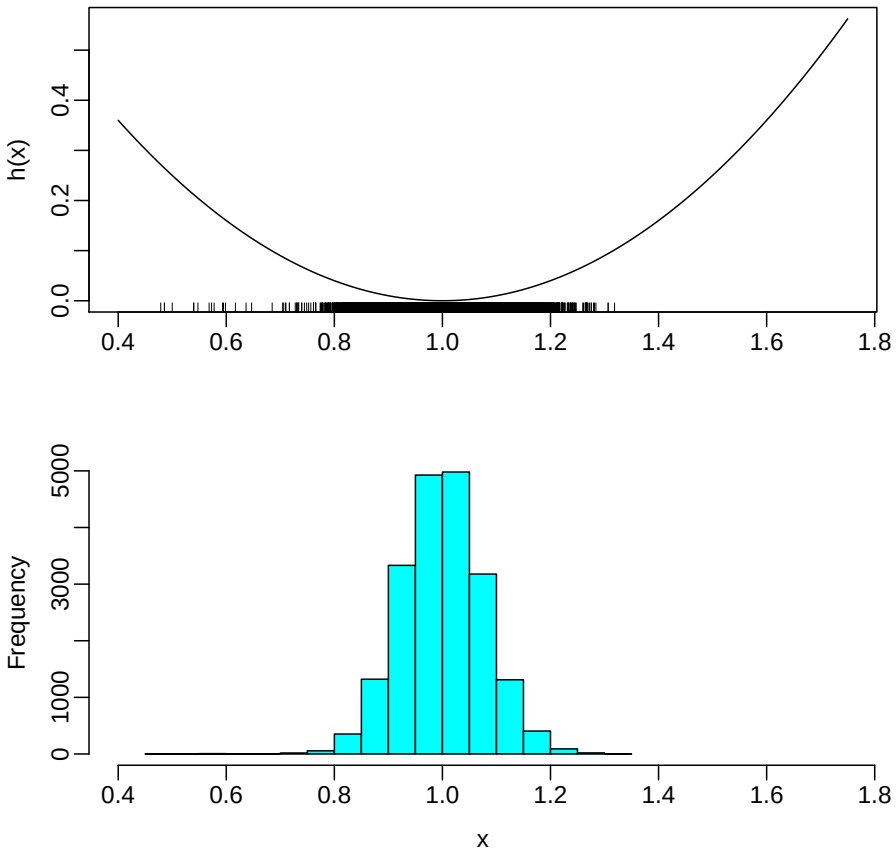
  donde r es un parámetro que deberá adaptarse de acuerdo al problema que se esté resolviendo
- Generar una propuesta de nuevo valor: $x_i \sim U(x_{i-1} - s, x_{i-1} + s)$, aquí s también debe seleccionarse de acuerdo al problema
- Calcular $\Delta h = h(x_{i-1}) - h(x_i)$
- Calcular $\rho = \min\left\{1, e^{\Delta h/\tau_i}\right\}$
- Aceptamos el valor propuesto de x_i con probabilidad ρ (así que nos quedamos donde estábamos con probabilidad $1 - \rho$)

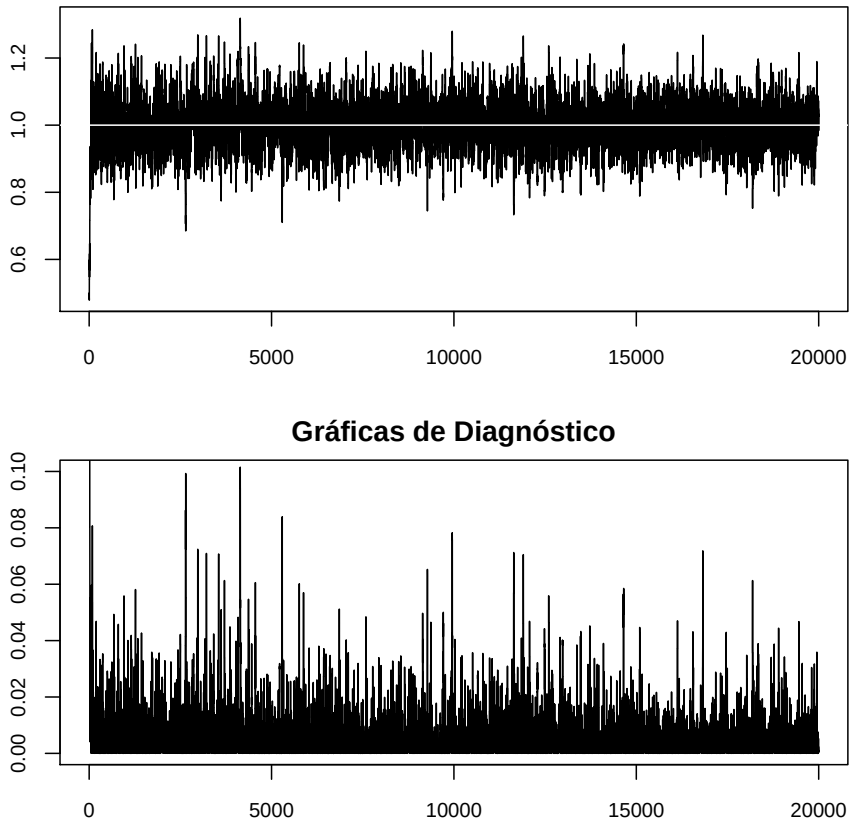
Note que si el valor propuesto de x_i es bueno, esto es, que tenga un menor valor de h con respecto al valor anterior, x_{i-1}; entonces es automáticamente aceptado, pues, en este caso, $\Delta h = h(x_{i-1}) - h(x_i) > 0$ y por lo tanto $e^{\delta h/\tau_i} > 1$ y entonces la probabilidad de aceptarlo es $\rho = 1$. Por otro lado, si el nuevo valor es malo, esto es que incrementa el valor de h, entonces hay una probabilidad (positiva) de que sea aceptado. Esta probabilidad de aceptación de un punto malo está dada por $\rho = e^{\Delta h/\tau_i}$, donde $\Delta h < 0$. Por supuesto, que si el nuevo punto es malísimo entonces Δh es grande (negativamente) y por lo tanto hay una probabilidad baja de aceptar ese malísimo.

Ejemplificamos este procedimiento con un ejemplo sencillo, $h(x) = (x-1)^2$:

```
# Recocido Simulado (Simulated Annealing)

h <- function(x) (x-1)^2

x0 <- 0.5
h0 <- h(x0)
ua <- 0.1
n  <- 20000
r  <- 1
z  <- matrix(0,n,3)
z[1,] <- c(x0,h0,r)

for( i in 2:n ){
 ti <- .1/(log(1+i))          # parametro clave: "temperatura"
```

```
 xt <- runif(1,x0-ua,x0+ua)
 ht <- h(xt)
 dh <- h0 - ht
 r  <- min(exp(dh/ti),1)
 if(runif(1) < r){
  x0 <- xt
  h0 <- ht
  z[i,]  <- c(xt,ht,r)
  } else {
  z[i,]  <- c(x0,h0,r) } }

par(mfrow=c(2,1),mar=c(3.1,3.1,2,1))
xs <- seq(.4,1.75,length=200)
plot(xs,h(xs),type="l",ylab="h(x)",xlab="",mgp=c(2,.5,0),xlim=c(.4,1.75))
rug(z[,1])
hist(z[,1],main="",col="cyan",xlab="x",mgp=c(2,.5,0),xlim=c(.4,1.75))

plot(z[,1],type="l",ylab="",cex.axis=.8)
abline(h=1,col="white")
plot(z[,2],type="l",ylab="",ylim=c(0,.1),cex.axis=.8,
 main="Graficas de Diagnostico")
```



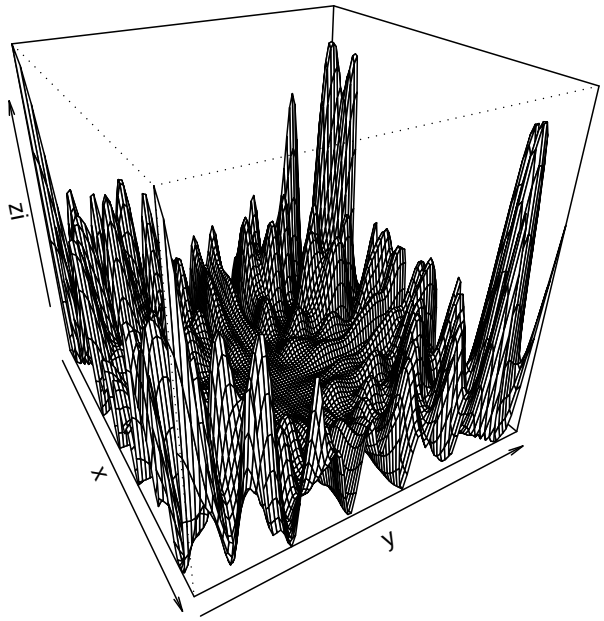


Consideremos ahora la minimización de la función:

$$\begin{aligned} h(x,y) =& (x\text{Sen}(20y) + y\text{Sen}(20x))^2 \ \text{Cosh}(\text{Sen}(10x)x) \\ &+ (x\text{Cos}(10y) - y\text{Sen}(10x))^2 \ \text{Cosh}(\text{Cos}(20x)x) \end{aligned}$$

cuya gráfica la obtenemos con:

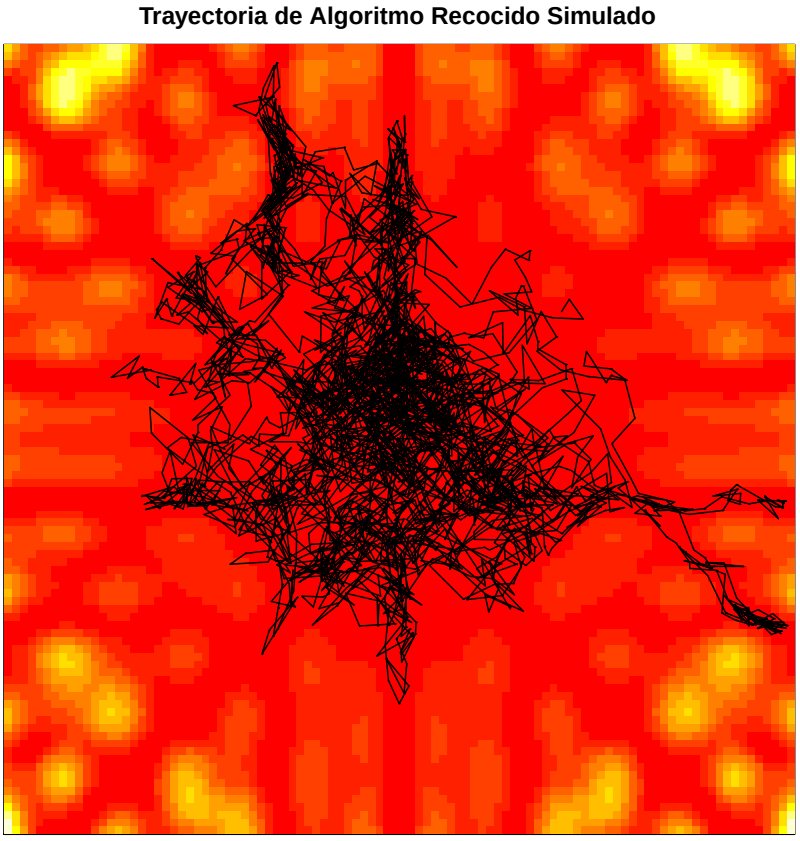
```
h <- function(x,y){
 (x*sin(20*y) + y*sin(20*x))^2 * cosh(sin(10*x)*x) +
 (x*cos(10*y) - y*sin(10*x))^2 * cosh(cos(20*y)*y) }
x  <- y <- seq(-1,1,length=100)
zi <- outer(x,y,"h")
persp(x,y,zi,theta=60,phi=30)
contour(x,y,zi)
```



Una implementación del algoritmo para minimizar esta función, la tenemos enseguida:

```
set.seed(84611)
x0 <- 0.5
y0 <- 0.4
h0 <- h(x0,y0)
ua <- 0.1
n  <- 5000
z  <- matrix(0,n,4)
z[1,] <- c(x0,y0,h0,1)
rem   <- 1
for( i in 2:n ){
 ti <- 1/(log(1+i))
 xt <- runif(1,x0-ua,x0+ua)
 yt <- runif(1,y0-ua,y0+ua)
 if( abs(xt) > 1 | abs(yt) > 1 ){
  rem <- c(rem,i)
  next }
 ht <- h(xt,yt)
 dh <- h0 - ht
 r  <- min(exp(dh/ti),1)
 if(runif(1) < r){
  x0 <- xt
  y0 <- yt
  h0 <- ht
```

```
  z[i,]  <- c(xt,yt,ht,r)
  } else {
  z[i,]  <- c(x0,y0,h0,r) } }
z <- z[-rem,]
m <- dim(z)[1]
x  <- y <- seq(-1,1,length=100)
zi <- outer(x,y,"h")
par(mar=c(2,2,2,2))
image(x,y,zi,xaxt="n",yaxt="n",ylab="",xlab="",cex.main=1,
 main="Trayectoria de Algoritmo Recocido Simulado")
for(i in 1:(m-1)){
 segments(z[i,1],z[i,2],z[i+1,1],z[i+1,2]) }
```



Prácticas Sesión 3

1. Sea $t_1, t_2, \cdots, t_n$ una muestra aleatoria de tiempos de vida de un cierto tipo de componentes. Deseamos ver si es razonable que éstos tiempos puedan considerarse Exponencialmente distribuídos. Para ello, podemos usar el estadístico de Bartlett

$$B_n = \frac{2n}{1 + \frac{n+1}{6n}} \left[\log(\bar{t}) - \frac{1}{n} \sum_{i=1}^{n} \log(t_i) \right]$$

Si la hipótesis de que los tiempos son Exponenciales, entonces

$$B_n \sim \chi^2_{n-1}$$

y la hipótesis de Exponencialidad se rechaza para valores grandes en la cola derecha de la χ^2_{n-1}.

(a) Escriba una función de la forma

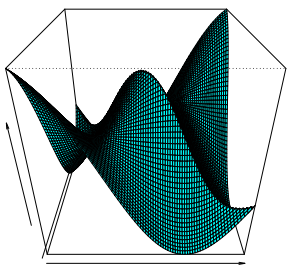
```
Bart <- function(x){
 ...
 (calculos)
 ...
 return( c(Bn,pval) )}
```

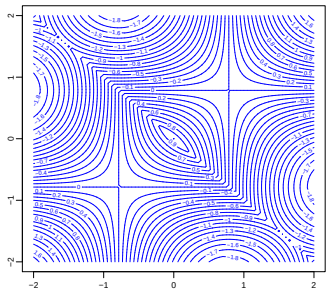
Esta función tiene como argumento el vector `x` de tiempos y debe regresar dos cantidades: El valor del estadístico B_n así como el p-valor asociado a la prueba de Bartlett.

(b) Busque en la Biblioteca algún libro (dar la referencia) (de preferencia, del área de Confiabilidad), donde venga algún conjunto de datos que se supone son Exponenciales. Efectúe la prueba de Bartlett a esos datos. Dé sus conclusiones.

2. Las siguientes gráficas muestran a la función

$$f(x, y) = \text{Cos}|x - y| - \text{Sen}|x + y|, \qquad -2 \leq x \leq 2, \ -2 \leq y \leq 2$$





(a) Usando `persp()` y `contour()`, replique estas gráficas.

(b) Usando fuerza bruta, encuentre los máximos y mínimos locales de esta función en la región indicada.

3. **Prueba de Mann-Whitney.** Consideremos el análogo noparamétrico de la prueba t para dos muestras. Supongamos que tenemos dos muestras de tamaño 4 de dos grupos experimentales:

A :	7	4	9	17
B :	11	6	21	14

Deseamos saber si existe evidencia de que A y B provienen de poblaciones con diferente nivel en la variable bajo estudio. La hipótesis nula es de que los miembros de un grupo no tienen una tendencia a ser mayores que los miembros del otro grupo. La hipótesis alternativa es de que sí existe tal tendencia (ya sea que las A's son mayores que las B's o viceversa).

Primero, ordenamos las observaciones en orden ascendente

4	6	7	9	11	14	17	21
A	B	A	A	B	B	A	B

Ahora, nos fijamos en un grupo, digamos A. Para cada elemento de A contamos el número de B's que la anteceden. Para la primer A no hay B's antes que ella, así que llevamos 0 B's; para la segunda A, hay una B antes que ella, así que llevamos $0+1=1$ B, etc., en total tenemos que el número de B's que anteceden a las A's es $U=0+1+1+3=5$; ahora, si este número fuera muy pequeño, esto sería evidencia de que las A's están por abajo que las B's y, si U fuera grande entonces sería evidencia de que las A's están por arriba de las B's. Voila!, ya tenemos un estadístico de prueba. Finalmente, para poder usar este estadístico para efectuar la prueba de hipótesis necesitamos saber su distribución nula; esto es, ¿Cuál es el comportamiento de U cuando en realidad no hay diferencias entre ambas poblaciones? (y las diferencias observadas son simplemente el producto de variación aleatoria pero que no tiene que ver con los tratamientos A y B).

Los dos conjuntos de 4 observaciones pueden arreglarse de 70 formas diferentes; desde $AAAABBBB$ hasta $BBBBAAAA$, en el primer caso $U=0$ y en el segundo $U=16$, todos los arreglos tienen, bajo H_0, la misma probabilidad de ocurrencia (1/70); ahora solo nos faltaría obtener los otros 68 arreglos, calcular sus respectivos valores de U, para así tener la distribución nula de U... esto es lo que se debería hacer, pero es medio engorroso, así que tomaremos un camino fácil (simulación) que no es del todo correcto, pero que nos puede dar una buena aproximación. Hagamos (¿suena conocido?):

```
r     <- rep(0,M)             # donde M es muy grande
for( i in 1:M){
 orden <- sample( c(0,0,0,0,1,1,1,1) )
 r[i]  <- cuenta( orden ) }
```

donde `cuenta` es una función que cuenta el número de B's que anteceden a las A's (i.e. calcula el valor de U). Al final `r` es un vector de longitud `M` y con ello nos puede dar una buena idea de la distribución de U; la región de rechazo se obtendría de las colas de esa distribución.

(a) Efectúe la prueba de H_0 : No hay tendencias, con los datos de arriba (Use un nivel de significancia lo más cercano a $\alpha=.05$).

(b) Los siguientes datos son grosores de piel (en mm) de la región del bíceps en dos grupos de pacientes (con Síndrome de Crohn y con Enfermedad Celíaca, respectivamente):

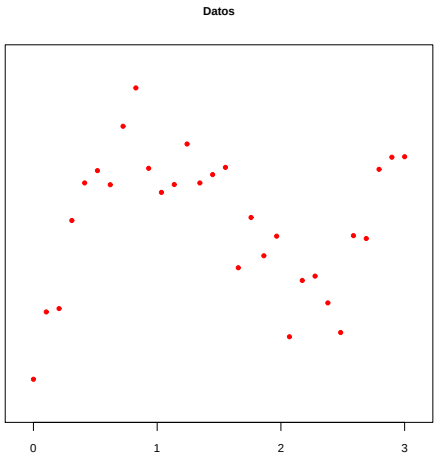
Crohn	1.8	2.3	2.4	2.5	2.8	2.9	3.2	3.6	3.9	4.0
Crohn	4.3	4.4	4.8	5.6	6.0	6.2	6.6	7.0	10.0	10.1
Celíacos	1.7	2.0	2.1	2.2	3.0	3.8	4.2	5.4	7.6	

Compare los grosores de piel en ambos grupos y determine si hay evidencia o no de diferencias entre ellos. Aquí hay que hacer un ejercicio de simulación similar al inciso anterior para calcular la distribución de U.

Nota: Los datos reales del segundo inciso fué modificado para evitar observaciones iguales (empates), en la práctica esto sucede con frecuencia y lo que se hace es contar sólo 1/2 (en vez de 1) en caso de parejas de A's y B's empatadas (en este curso no dicutiremos estas modificaciones). También, en la práctica, no se tiene que calcular cada vez la distribución del estadístico de prueba; éstas distribuciones ya han sido tabuladas ampliamente.

4. El siguiente código genera un conjunto de datos y produce la gráfica de los mismos

```
set.seed(79015)
n    <- 30
x    <- seq(0,3,length=n)
y    <- 3*x-3*x^2+x^3-.1*x^4 + rnorm(n,0,.15)
yr   <- range(y)
yl   <- yr + .10*c(-1,1)*(yr[2]-yr[1])
plot(x, y, xlab="", ylab="", mgp=c(1.5,.5,0), col="red", yaxt="n", xaxt="n",
 cex.axis=.7, main="Datos", cex.main=.8, pch=20, ylim=yl, xlim=c(-.1,3.1))
axis(1,at=0:3, cex.axis=.8)
```



Tenemos entonces, n parejas de puntos $(x_1, y_1), \cdots, (x_n, y_n)$. Suponga que deseamos ajustar un modelo lineal de la forma

$$y_i = \beta_0 + \beta_1 x_i + e_i, \qquad \text{con} \quad e_i \sim \text{ i.i.d. } N(0, \sigma^2),\ i = 1, \cdots, n$$

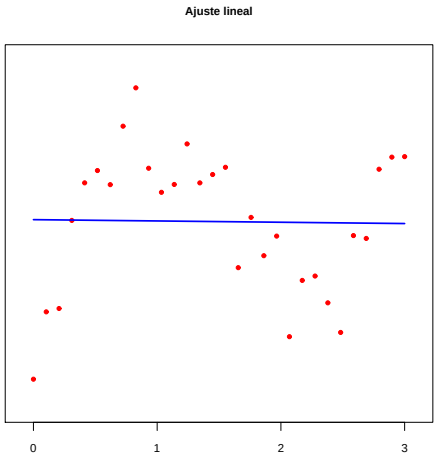
Si escribimos este modelo en forma matricial

$$y = X\beta + e, \quad \text{donde } y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix},\ X = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}_{n\times 2},\ \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix},\ e = \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix}$$

donde $e \sim N_n(0, \sigma^2 I)$. Entonces, la teoría de estimación por mínimos cuadrados (esto es, estimar β minimizando la suma de cuadrados del error, $SCE(\beta) = (y - X\beta)^T (y - X\beta)$), (esto lo verán en su curso de Regresión), esa teoría nos dice que los estimadores de los diferentes parámetros de interés son:

$$\begin{aligned} \widehat{\beta} &= (X^T X)^{-1} X^T y \\ \widehat{\sigma}^2 &= \frac{1}{n-2}(y - X\widehat{\beta})^T (y - X\widehat{\beta}) \\ \widehat{\text{Var}}(\widehat{\beta}) &= \widehat{\sigma}^2 (X^T X)^{-1} \end{aligned}$$

Con los datos generados para este problema, escriba un programa para calcular $\widehat{\beta}$ y $\widehat{\text{Var}}(\widehat{\beta})$ y grafique la recta estimada de regresión, $\widehat{y} = \widehat{\beta}_0 + \widehat{\beta}_1 x$, como se muestra en la siguiente gráfica.



5. En el problema anterior, si deseamos efectuar un ajuste cuadrático, entonces el modelo es

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + e_i, \qquad \text{con} \quad e_i \sim \text{ i.i.d. } N(0, \sigma^2),\ i = 1, \cdots, n$$

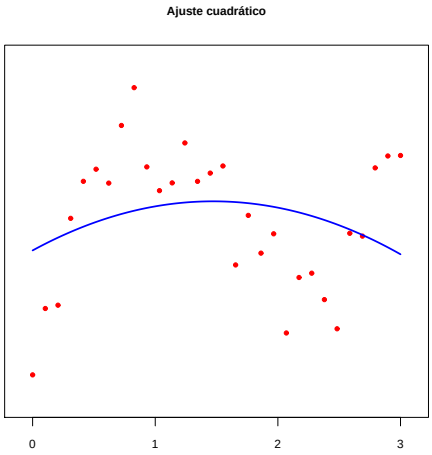
nuevamente, podemos escribir el modelo en forma matricial $y = X\beta + e$, donde ahora

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix}_{n \times 3}, \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

Los estimadores de los diferentes parámetros de interés son:

$$\begin{aligned} \widehat{\beta} &= (X^T X)^{-1} X^T y \\ \widehat{\sigma}^2 &= \frac{1}{n-3}(y - X\widehat{\beta})^T (y - X\widehat{\beta}) \\ \widehat{\text{Var}}(\widehat{\beta}) &= \widehat{\sigma}^2 (X^T X)^{-1} \end{aligned}$$

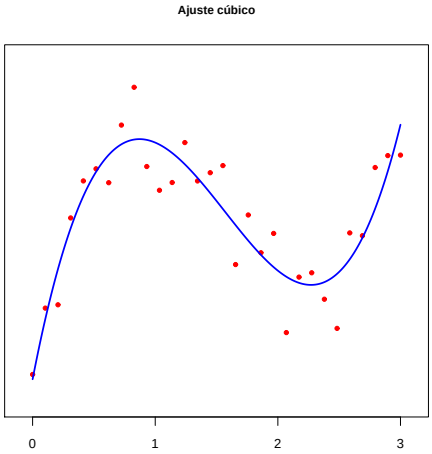
Note que son las mismas expresiones, excepto que el denominador en el estimador de la varianza es $n-3$. Con los datos del problema 1, escriba un programa para calcular $\widehat{\beta}$ y los errores estándar de los elementos de $\widehat{\beta}$ (esto es, las raíces cuadradas de los elementos diagonales de la matriz $\widehat{\text{Var}}(\widehat{\beta})$) y grafique la curva estimada de regresión, $\widehat{y} = \widehat{\beta}_0 + \widehat{\beta}_1 x + \widehat{\beta}_2 x^2$, como se muestra en la siguiente gráfica.



6. En los problemas anteriores vemos que los ajustes no son satisfactorios. Ajuste un modelo cúbico

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + e_i, \qquad \text{con} \quad e_i \sim \text{ i.i.d. } N(0, \sigma^2),\ i = 1, \cdots, n$$

Calcule $\widehat{\beta}$ y sus errores estándar (hay que dividir entre $n-4$) y grafique la curva estimada de regresión, como se muestra en la siguiente gráfica.



¿Hay alguna diferencia importante si ajustara un modelo de orden 4?.

7. Los modelos polinomiales son importantes pero, al mismo tiempo, son algo restrictivos pues es difícil imaginar que el comportamiento de un fenómeno real sea modelable de esa forma en todo el rango de su dominio. Hay ocasiones en las que será razonable pensar que en cierto rango de valores de la variable independiente el comportamiento sigue una relación polinomial y en otro rango es modelable por otro polinomio. En este problema deseamos explorar como hacer el ajuste de un modelo lineal por pedazos.

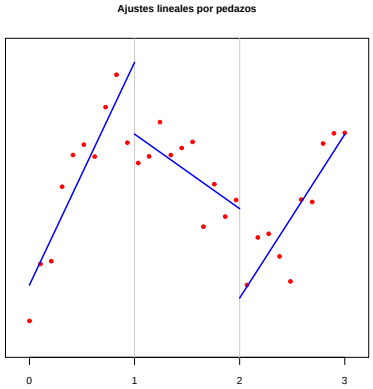
Suponga que queremos ajustar, a los datos del problema 4, el modelo

$$E(y_i|x_i) = \begin{cases} a + bx_i & \text{si} \quad 0 \leq x_i \leq 1 \\ c + dx_i & \text{si} \quad 1 < x_i \leq 2 \\ e + fx_i & \text{si} \quad 2 < x_i \leq 3 \end{cases}$$

Este modelo también puede ponerse en la forma $y = X\beta + e$, pero con

$$X = \begin{bmatrix} 1 & x_1 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{10} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & x_{11} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & x_{20} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x_{21} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 1 & x_{30} \end{bmatrix} \quad \text{y} \quad \beta = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}$$

Calcule $\widehat{\beta}$ y sus errores estándar (hay que dividir entre $n-6$) y grafique las rectas estimadas de regresión, como se muestra en la siguiente gráfica.



8. Desde el punto de vista de modelación, el modelo anterior no es aceptable pues si nos acercamos a $x = 1$ por la izquierda el modelo nos predice un cierto valor para y, pero si nos acercamos a $x = 1$ por la derecha, el modelo nos predice un valor diferente para y, mucho más bajo. Lo que le falta a este modelo es "continuidad". Para hacer que las rectas se "peguen" en los puntos 1 y 2 (a estos puntos se les llama **nodos**), necesitamos que se cumpla que

$$\begin{aligned} a + k_1\, b &= c + k_1\, d \\ c + k_2\, d &= e + k_2\, f \end{aligned}$$

donde $k_1 = 1$ y $k_2 = 2$ son los nodos. En otras palabras, necesitamos que los parámetros del modelo satisfagan el sistema de ecuaciones lineales (2 ecuaciones y 6 incógnitas):

$$\begin{bmatrix} 1 & k_1 & -1 & -k_1 & 0 & 0 \\ 0 & 0 & 1 & k_2 & -1 & -k_2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \equiv \qquad K^T\beta = 0$$

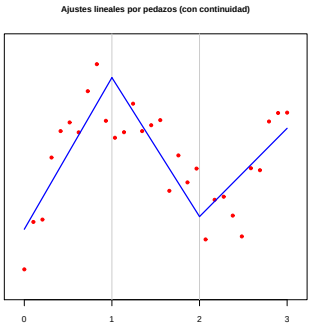
Entonces, para estimar los parámetros del modelo tenemos que resolver el siguiente problema de minimización sujeta a restricciones:

$$\min_{\beta} \ (y - X\beta)^T(y - X\beta) \qquad \text{sujeta a} \quad K^T\beta = 0$$

Este problema puede resolverse en forma explícita usando Multiplicadores de Lagrange y la solución es

$$\tilde{\beta} \;=\; \widehat{\beta} - (X^TX)^{-1}K[\;K^T(X^TX)^{-1}K\;]^{-1}K^T\widehat{\beta}$$

donde $\widehat{\beta} = (X^TX)^{-1}X^Ty$, es el estimador de β sin restricciones. Encuentre el estimador restringido $\tilde{\beta}$ y grafique el modelo resultante, como se muestra en la gráfica:



9. Una vez que ya sabemos como ajustar modelos lineales por pedazos, podemos ahora considerar modelos cúbicos por pedazos y que satisfagan condiciones de continuidad. Suponga que queremos ajustar, a los datos del problema 4, el modelo

$$E(y_i|x_i) = \begin{cases} a + bx_i + cx_i^2 + dx_i^3 & \text{si} \quad 0 \le x_i \le 1 \\ e + fx_i + gx_i^2 + hx_i^3 & \text{si} \quad 1 < x_i \le 2 \\ i + jx_i + kx_i^2 + lx_i^3 & \text{si} \quad 2 < x_i \le 3 \end{cases}$$

Por lo que aprendimos en el problema 7, esto lo podemos poner en un modelo lineal $y = X\beta + e$ con

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{10} & x_{10}^2 & x_{10}^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x_{11} & x_{11}^2 & x_{11}^3 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 1 & x_{20} & x_{20}^2 & x_{20}^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & x_{21} & x_{21}^2 & x_{21}^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & x_{30} & x_{30}^2 & x_{30}^3 \end{bmatrix} \quad \text{y} \quad \beta_{12\times 1} = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \\ k \\ l \end{bmatrix}$$

y como queremos condiciones de continuidad, entonces las ecuaciones son

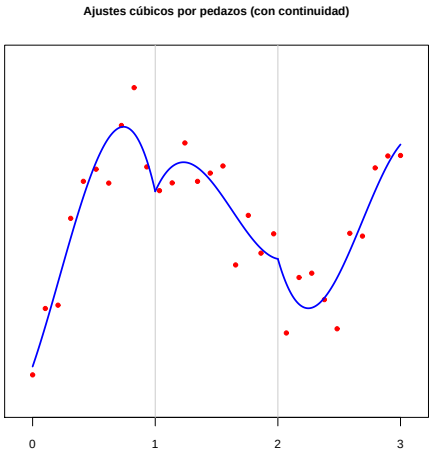
$$\begin{aligned} a + bk_1 + ck_1^2 + dk_1^3 &= e + fk_1 + gk_1^2 + hk_1^3 \\ e + fk_2 + gk_2^2 + hk_2^3 &= i + jk_2 + kk_2^2 + lk_2^3 \end{aligned}$$

En forma matricial, estas condiciones son

$$\begin{bmatrix} 1 & k_1 & k_1^2 & k_1^3 & -1 & -k_1 & -k_1^2 & -k_1^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & k_2 & k_2^2 & k_2^3 & -1 & -k_2 & -k_2^2 & -k_2^3 \end{bmatrix} \beta_{12\times 1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \equiv \quad K^T\beta = 0$$

donde k_1 y k_2 son los nodos.

Nuevamente, como en el problema 8, lo que queremos es estimar β mediante aquel valor que minimice la suma de cuadrados residual, $SCE(\beta) = (y - X\beta)^T(y - X\beta)$, sujeta a las restricciones $K^T\beta = 0$. Ajuste el modelo cúbico por pedazos con restricciones de continuidad y muestrelo gráficamente como enseguida:



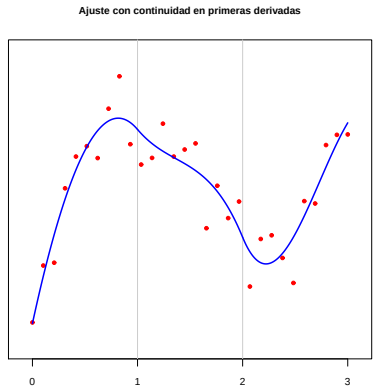
10. Inmediatamente observamos que algo no esta bien en la solución obtenida en el problema 9. Si bien, tenemos continuidad, a la gráfica le falta "suavidad" en los nodos; así que podemos pedir que las derivadas sean continuas ahí. La derivada de la primer cúbica es $b+2cx+3dx^2$, queremos que sea igual a la derivada de la segunda cúbica, $f+2gx+3hx^2$ cuando las evaluemos en $x=k_1$. Similarmente, en el segundo nodo, también queremos que las derivadas sean iguales. Estas observaciones se traducen en las ecuaciones

$$\begin{aligned} b+2ck_1+3dk_1^2 &= f+2gk_1+3hk_1^2 \\ f+2gk_2+3hk_2^2 &= j+2kk_2+3lk_2^2 \end{aligned}$$

De modo que el conjunto total de restricciones es ahora

$$\begin{bmatrix} 1 & k_1 & k_1^2 & k_1^3 & -1 & -k_1 & -k_1^2 & -k_1^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & k_2 & k_2^2 & k_2^3 & -1 & -k_2 & -k_2^2 & -k_2^3 \\ 0 & 1 & 2k_1 & 3k_1^2 & 0 & -1 & -2k_1 & -3k_1^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2k_2 & 3k_2^2 & 0 & -1 & -2k_2 & -3k_2^2 \end{bmatrix} \beta = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \equiv K^T\beta = 0$$

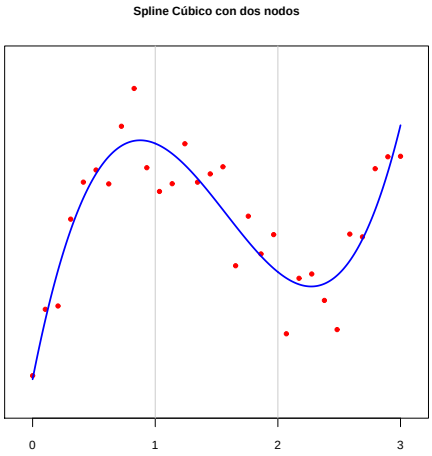
donde k_1 y k_2 son los nodos. Efectuar el ajuste y mostrarlo gráficamnete.



11. Una petición más: Queremos agregar continuidad en la segunda derivada. En otras palabras:

$$\begin{aligned} 2c + 6dk_1 &= 2g + 6hk_1 \\ 2g + 6hk_2 &= 2k + 6lk_2 \end{aligned}$$

Agregue estas dos ecuaciones a las anteriores, efectue el ajuste y muestrelo gráficamente. Al resultado de este ajuste se le llama **Spline Cúbico**.



12. Cuando ajustamos un modelo de regresión de la forma $y = X\beta + e$, lo que hacemos es obtener $\widehat{y}$ el cual es la proyección de y sobre el espacio de columnas generado por la matriz X. Ahora, cuando uno impone condiciones, en realidad lo que estamos haciendo es definiendo un subespacio del espacio de columnas de X. Para clarificar estas ideas, supongamos un modelo de regresión lineal múltiple con dos variables predictoras

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + e_i, \qquad i = 1, \cdots, n$$

La matriz X en este caso es

$$X = \begin{bmatrix} 1 & x_{11} & x_{21} \\ 1 & x_{12} & x_{22} \\ \vdots & \vdots & \vdots \\ 1 & x_{1n} & x_{2n} \end{bmatrix}_{n \times 3}$$

Ahora, si quisieramos ajustar el modelo bajo la condición $\beta_1 = \beta_2$, esto sería equivalente a ajustar el modelo

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_1 x_{2i} + e_i \;=\; \beta_0 + \beta_1 (x_{1i} + x_{2i}) + e_i, \qquad i = 1, \cdots, n$$

cuya nueva matriz X tiene un espacio de columnas que es un subespacio del original

$$X = \begin{bmatrix} 1 & x_{11} + x_{21} \\ 1 & x_{12} + x_{22} \\ \vdots & \vdots \\ 1 & x_{1n} + x_{2n} \end{bmatrix}_{n \times 2}$$

Note que la dimensión del nuevo subespacio es 2 = 3 - 1 = dimensión original - número de restricciones.

Cuando ajustamos un spline cúbico con dos nodos, tenemos que la dimensión original es 12 y el número de restricciones es 6; así que la dimensión del subespacio del spline es 12 - 6 = 6. Ahora, como $\widehat{y}$ es una proyección sobre ese subespacio resultante, no importa cual base sea, la proyección es independiente de la

base; así que es suficiente con conocer esa base y ya esta. Puede verse que una base para el spline cúbico con dos nodos está dada por las columnas de

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{10} & x_{10}^2 & x_{10}^3 & 0 & 0 \\ 1 & x_{11} & x_{11}^2 & x_{11}^3 & (x_{11}-1)^3 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{20} & x_{20}^2 & x_{20}^3 & (x_{20}-1)^3 & 0 \\ 1 & x_{21} & x_{21}^2 & x_{21}^3 & (x_{21}-1)^3 & (x_{21}-2)^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{30} & x_{30}^2 & x_{30}^3 & (x_{30}-1)^3 & (x_{30}-2)^3 \end{bmatrix}$$

Esto es, a los elementos de la base los podemos denotar por $1,\ x,\ x^2,\ x^3,\ (x-k_1)^3_+,\ (x-k_2)^3_+$, donde la notación $(x-k)^3_+$ significa que vale $(x-k)^3$ si $x > k$ y vale 0 de otra forma.

El ajuste del modelo se hace en la forma usual de regresión $\widehat{y} = X\tilde{\beta} = X(X^TX)^{-1}X^Ty$, donde esta X es la que acabamos de definir, correspondiente al subespacio del spline. Entonces, para cualquier x, el valor estimado de y está dado por

$$\widehat{y} = \tilde{\beta}_0 + \tilde{\beta}_1 x + \tilde{\beta}_2 x^2 + \tilde{\beta}_3 x^3 + \tilde{\beta}_4 (x-k_1)^3_+ + \tilde{\beta}_5 (x-k_2)^3_+$$

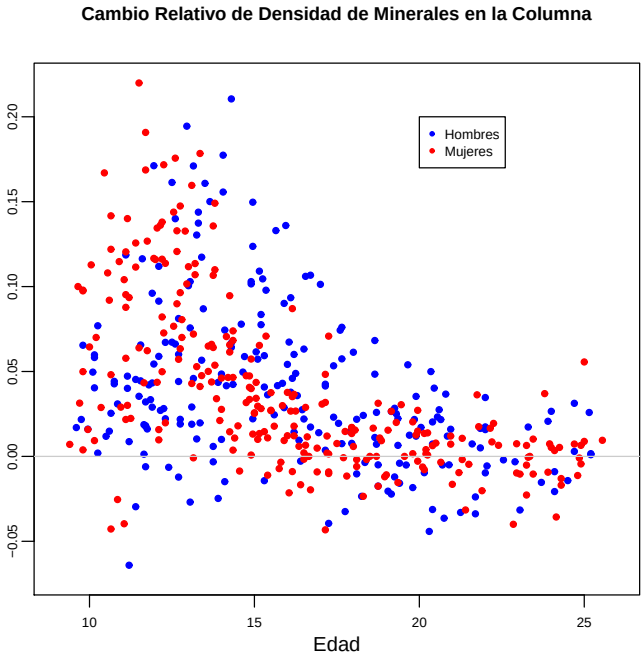
Ajuste el spline cúbico con dos nodos en $k_1 = 1$ y $k_2 = 2$, usando esta nueva matriz X (note que aquí ya no hay necesidad de resolver el problema de mínimos cuadrados con restricciones). Muestre el ajuste gráficamente (la gráfica debe ser idéntica a la obtenida en el problema 8).

Un punto importante que se desprende de este ejercicio es que es muy fácil ajustar splines, por ejemplo, si tuvieramos 4 nodos $k_1 < k_2 < k_3 < k_4$, entonces la base sería

$$1,\ x,\ x^2,\ x^3,\ (x-k_1)^3_+,\ (x-k_2)^3_+,\ (x-k_3)^3_+,\ (x-k_4)^3_+$$

13. El siguiente código produce la gráfica que se muestra enseguida

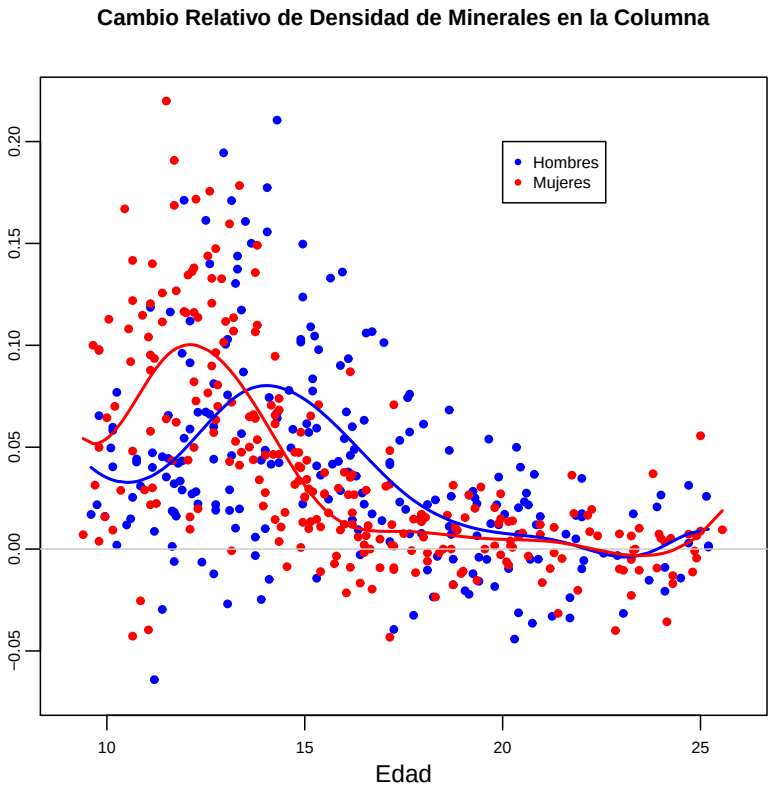
```
datos  <- read.csv("c:\\Documents and Settings\\ ... \\Bone.csv", header=TRUE)
names(datos) <- c("id","edad","genero","spnbmd")
attach(datos)
datH <- datos[genero=="male",]     # 226 x 4
datM <- datos[genero=="female",]   # 259 x 4
detach(datos)

plot(datH[,2],datH[,4], xlab="Edad", ylab="", mgp=c(1.5,.5,0), col="blue",
 cex.axis=.7,ylim=c(-0.07,0.22), xlim=c(9,26),pch=20,
 main="Cambio Relativo de Densidad de Minerales en la Columna", cex.main=.9)
points(datM[,2],datM[,4],col="red",pch=20)
abline(h=0,col=gray(.8))
legend(20,.2,legend=c("Hombres","Mujeres"),pch=20,col=c("blue","red"),cex=.7)
```



Estos datos son los cambios relativos en la densidad de minerales en la columna vertebral de cerca de 500 niños y jóvenes. Estos cambios relativos fueron registrados en visitas consecutivas (aproximadamente separadas un año). Los cambios para los hombres se indican en azul y los de las mujeres en rojo.

Ajuste splines cúbicos para los datos de hombres y mujeres y muestre gráficamente sus resultados. Las gráficas refuerzan el hecho de que el crecimiento de las mujeres antecede al de los hombres (los máximos de las gráficas están separadas por unos dos años).

La gráfica de la siguiente hoja fue construída usando 7 nodos (`nodos <- seq(11.5,23.5,length=7)`). Los splines cúbicos tienen no muy buen comportamiento antes del primer nodo y después del último; hay técnicas para corregir esto (**splines naturales**). Finalmente, hay splines que le evitan a uno decidir donde poner los nodos pues usan un número maximal de nodos pero tienen que usar técnicas de regularización para su implementación (**splines suavizadores**).



14. En 1693, Samuel Pepys le planteó a Issac Newton el siguiente problema: ¿Qués es más probable,

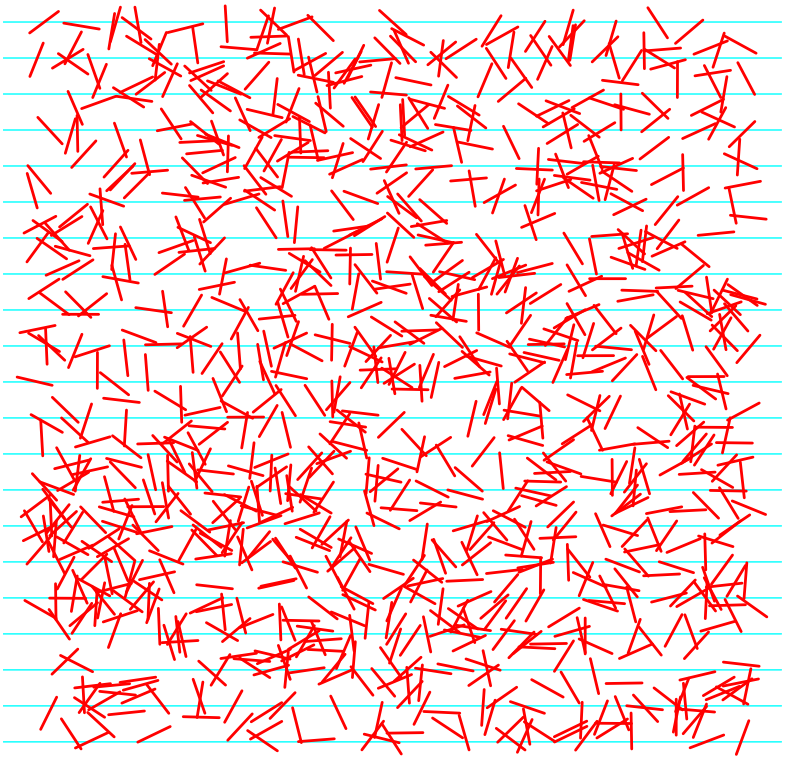
    - Obtener al menos un 6 al tirar 6 dados,
    - obtener al menos dos 6's al tirar 12 dados,
    - u obtener al menos tres 6's al tirar 18 dados?

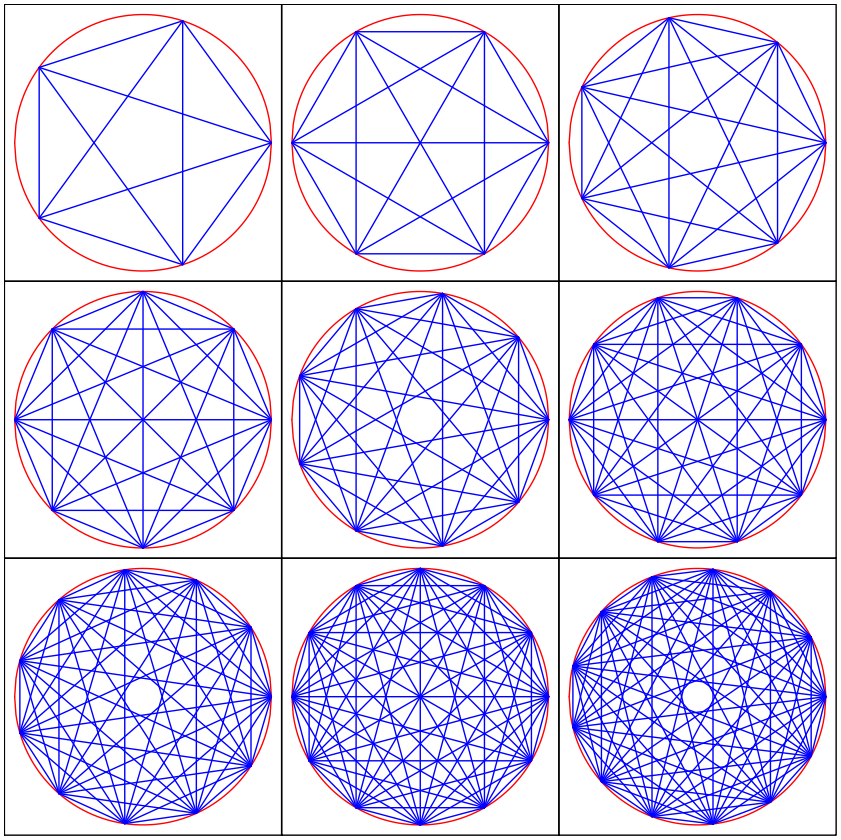
    (buscar "Newton-Pepys problem" en internet para ver el cálculo de las probabilidades exactas). Dé una respuesta usando simulación.

15. (El problema de los cumpleaños.) En un grupo de $M = 20$ alumnos, ¿Cuál es la probabilidad de que al menos dos alumnos cumplan años el mismo día?. ¿Cuál sería esa probabilidad si el grupo tiene $M = 30$ alumnos?. Mejor, ¿Cuál sería esa probabilidad para $M = 2, 3, 4, \cdots, 80$?. Haga una gráfica de las probabilidades calculadas versus M (usar simulación).

Apéndice Sesión 3

π es aproximadamente 3.1201





Apéndice Sesión 3

```
#########################################################################
# Figura 1: Las agujas de Buffon
# (ver http://www.mste.uiuc.edu/reese/buffon/buffon.html)
#
m    <- 20
n    <- 1000
xy   <- matrix( runif(2*n,min=0,max=m),n,2)
tet  <- runif(n,min=0,max=pi)
pen  <- tan( tet )
del  <- 1/(2*sqrt(1+pen^2))
ext  <- cbind(xy[,1]-del,xy[,2]-pen*del,xy[,1]+del,xy[,2]+pen*del)
a    <- xy[,2]-floor(xy[,2])
D    <- ifelse(a<.5, a, 1-a)
hits <- sum(D < sin(tet)/2)
pig  <- round(2*n/hits,4)
par(mar=c(0,1,2,1))
plot(0,0, type="n", xlim=c(0,m), ylim=c(0,m),xaxt="n",
     xlab="", bty="n", mgp=c(1.5,.5,0),yaxt="n",ylab="")
title(main=substitute(paste(pi, "  es aproximadamente  ",pig),list(pig=pig)))
abline( h=0:m, col="cyan" )
for(i in 1:n){
 segments(ext[i,1],ext[i,2],ext[i,3],ext[i,4], lwd=2, col="red")}
#########################################################################

#########################################################################
# Figura 2: Spirografo
spiro <- function(k){
 tet <-(2*pi/k)*(1:k)
 x   <- cbind(cos(tet),sin(tet))
 xx  <- seq(-1,1,length=200)
 yp  <- sqrt(1-xx^2)
 yn  <- -yp
 plot(xx,yp,type="l",xlim=c(-1,1),xlab="",col="red",
     xaxt="n",yaxt="n",ylim=c(-1,1),ylab="")
 lines(xx,yn, col="red")
 for( i in 1:(k-1) ){
  for( j in (i+1):k ){
   segments(x[i,1],x[i,2],x[j,1],x[j,2], col="blue") }}}

par(mfrow=c(3,3),mar=c(0,0,0,0))
for(k in 5:13){spiro(k)}
#########################################################################
```

JUEVES

Notas Sesión 4

Kolmogorov-Smirnov. La prueba Kolmogorov-Smirnov es usada en Estadística para verificar supuestos distribucionales (prueba de Bondad de Ajuste). La idea es simple: La función de distribución empírica es un estimador de la función de distribución verdadera; entonces, si la empírica difiere mucho de la teórica entonces rechazamos que la teórica es la distribución verdadera. Los términos que subrayamos implican dos problemas:

- ¿Cómo medimos discrepancias entre dos funciones?
- Una vez que cuantificamos la discrepancia, ¿Cómo sabemos si es grande o no?

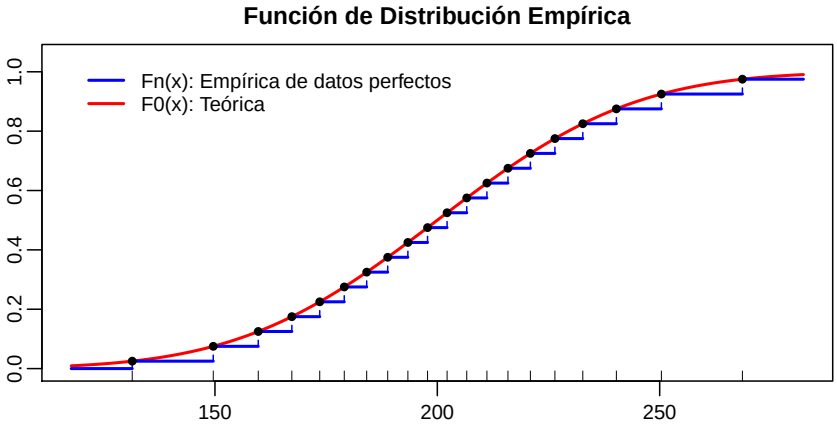
Para contestar la primera pregunta tenemos el estadístico D_n de Kolmogorov-Smirnov

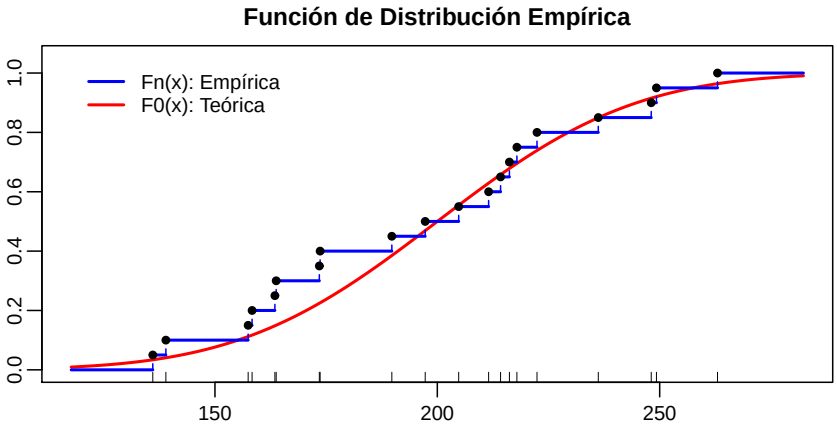
$$D_n = \sup_x \mid F_n(x) - F_0(x) \mid$$

donde $F_n(x)$ es la Función de Distribución Empírica y $F(x)$ es la Función de Distribución Teórica (o Hipotetizada). La función de distribución empírica se define como

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} I(x_i \leq x)$$

En relación a la segunda pregunta, puede mostrarse el resultado (remarcable) que, para el caso continuo, la distribución de D_n no depende de la distribución verdadera $F(x)$; esto es, hay una sola distribución nula de D_n no importando si estamos en el caso Normal, Gama, Weibull, etc.
La siguiente gráfica ejemplifica la construcción de la Distribución Empírica:





```
par(mfrow=c(2,1),mar=c(2,2,2,2))
n    <- 20
med  <- 200
des  <- 35
delt <- .10
M    <- 200
edf  <- ((1:n)-.5)/n
dat  <- qnorm(edf,mean=med,sd=des)
ran  <- max(dat) - min(dat)
lmin <- min(dat) - delt*ran
lmax <- max(dat) + delt*ran
xx   <- seq(lmin,lmax,length=M)
yy   <- pnorm(xx,mean=med,sd=des)
plot(dat,edf, xlim=c(lmin,lmax), ylim=c(0,1.05), mgp=c(1.5,.5,0),
  xlab="", ylab="", cex.axis=.8, cex.lab=.8,type="n",
  main="Funcion de Distribucion Empirica",cex.main=1)
rug(dat)
lines(xx,yy,col="red",lwd=2)
segments(lmin,0,dat[1],0,col="blue",lwd=2)
for(i in 1:(n-1)){ segments(dat[i],edf[i],dat[i+1],edf[i],col="blue",lwd=2) }
segments(dat[n],edf[n],lmax,edf[n],col="blue",lwd=2)
segments(dat[1],0,dat[1],edf[1],col="blue",lty=2)
for(i in 2:n){ segments(dat[i],edf[i-1],dat[i],edf[i],col="blue",lty=2) }
points(dat,edf,pch=20)
legend(lmin,1.05,legend=c("Fn(x): Empirica de datos perfectos","F0(x): Teorica"),
 col=c("blue","red"),lwd=2, cex=.8, bty="n")

edf  <- (1:n)/n
set.seed(5963471)
dat  <- sort(rnorm(n,mean=med,sd=des))
xx   <- seq(lmin,lmax,length=M)
yy   <- pnorm(xx,mean=med,sd=des)
plot(dat,edf, xlim=c(lmin,lmax), ylim=c(0,1.05), mgp=c(1.5,.5,0),
  xlab="", ylab="", cex.axis=.8, cex.lab=.8,type="n",
  main="Funcion de Distribucion Empirica",cex.main=1)
rug(dat)
lines(xx,yy,col="red",lwd=2)
segments(lmin,0,dat[1],0,col="blue",lwd=2)
for(i in 1:(n-1)){ segments(dat[i],edf[i],dat[i+1],edf[i],col="blue",lwd=2) }
segments(dat[n],1,lmax,1,col="blue",lwd=2)
segments(dat[1],0,dat[1],edf[1],col="blue",lty=2)
for(i in 2:n){ segments(dat[i],edf[i-1],dat[i],edf[i],col="blue",lty=2) }
points(dat,edf,pch=20)
legend(lmin,1.05,legend=c("Fn(x): Empirica","F0(x): Teorica"),
 col=c("blue","red"),lwd=2, cex=.8, bty="n")
```

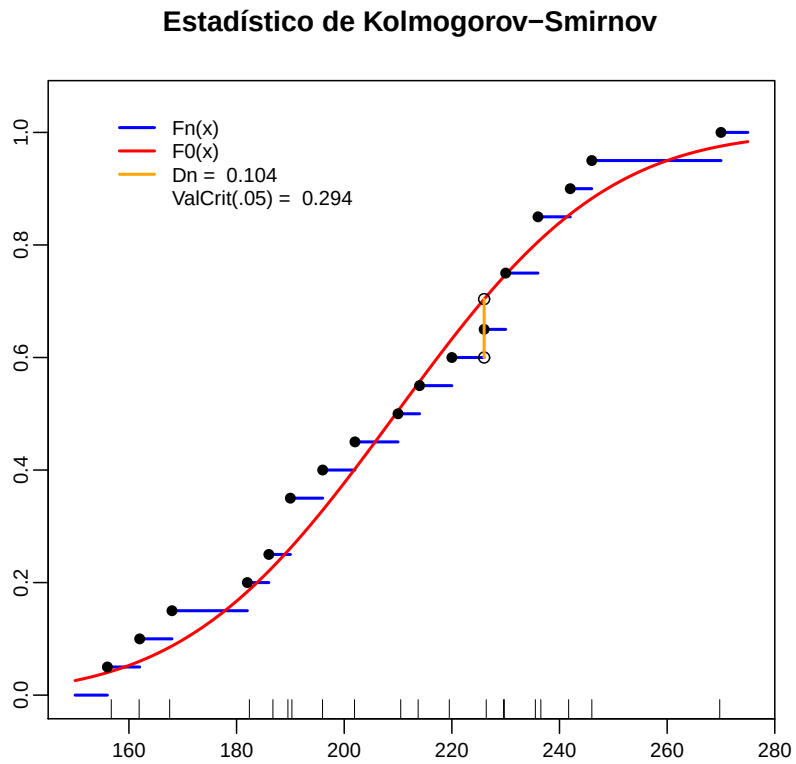
Ahora bien,¿Cómo se efectúa, por ejemplo, una prueba de Normalidad?. Para ello, calculamos el valor del estadístico D_n de Kolmogorov-Smirnov y vemos si es improbablemente grande. Veamos un ejemplo: Los siguientes datos son 20 registros de pesos (en gramos) de pollitos

156	162	168	182	186	190	190	196	202	210
214	220	226	230	230	236	236	242	246	270

Deseamos probar la hipótesis

$$H_0 : \text{Los datos provienen de:} \quad N(\mu, \sigma^2)$$

donde $\mu = 209.6$ y $\sigma = 30.65$ (como explicaremos, es crucial saber de dónde vienen estos valores, pero, por lo pronto, supongamos que estos son los valores en los que estamos interesados). Calculamos primero el valor de D_n y lo comparamos contra el valor de un cuantil de la distribución nula de D_n, como se muestra en la siguinete gráfica:



Como el valor de $D_n = 0.104$ no sobrepasa al valor crítico $D_{n,.05} = 0.294$, entonces no rechazamos la aseveración de que los datos provienen de una distribución normal (en Estadística así es como no toca vivir: No estamos diciendo que los datos son Normales, simplemente decimos que es razonablemente aceptable considerarlos normales pues no hay evidencia de lo contrario... :). El código R usado para obtener la gráfica anterior se muestra enseguida:

```
# Prueba de normalidad. Pesos (en gramos) de 20 pollitos
vc05 <- 0.2940701         # valor obtenido del libro New Cambridge Statistical Tables
dat  <- sort(c(156,162,168,182,186,190,190,196,202,210,
               214,220,226,230,230,236,236,242,246,270))
datu <- unique(dat)
n    <- length(dat)
nu   <- length(datu)
aa   <- table(dat)
edf  <- cumsum(aa)/n
edf0 <- c(0,edf[-nu])
xx   <- seq(150,275,length=200)
yy   <- pnorm(xx,mean=200,sd=35)
ye   <- pnorm(datu,mean=200,sd=35)
yy   <- pnorm(xx,mean=mean(dat),sd=sd(dat))      # usando estimaciones los
ye   <- pnorm(datu,mean=mean(dat),sd=sd(dat))    # valores criticos no validos
```

```
Dn   <- max(max(abs(ye-edf)),max(abs(ye-edf0)))
ii   <- which(Dn == pmax(abs(ye-edf),abs(ye-edf0)))[1]
if( abs(ye[ii]-edf[ii])==Dn ){
 yD <- edf[ii] }else{
 yD <- edf0[ii] }

plot(datu,edf, xlim=c(150,275), ylim=c(0,1.05), mgp=c(1.5,.5,0),
  xlab="", ylab="", cex.axis=.8, cex.lab=.8,type="n",
  main="Estadistico de Kolmogorov-Smirnov")
rug(jitter(dat))
segments(150,0,datu[1],0,col="blue",lwd=2)
for(i in 1:(nu-1)){
 segments(datu[i],edf[i],datu[i+1],edf[i],col="blue",lwd=2)}
segments(datu[nu],1,275,1,col="blue",lwd=2)
lines(xx,yy,col="red",lwd=2)
points(datu,edf,pch=16)
points(c(datu[ii],datu[ii]),c(ye[ii],yD))
segments(datu[ii],ye[ii],datu[ii],yD,lwd=2,col="orange")

legend(155,1.05,legend=c("Fn(x)","F0(x)",paste("Dn = ",round(Dn,3)),
 paste("ValCrit(.05) = ",round(vc05,3))),
 col=c("blue","red","orange","white"),lwd=2, cex=.8, bty="n")
```

Si bien es cierto que la distribución de D_n no depende de la distribución verdadera $F(x)$, en la práctica uno debe especificar completamente todos los parámetros que definen la distribución nula. Usualmente lo que se hace es estimar los parámetros en base a los datos; por ejemplo, con los datos anteriores calculamos $\bar{x} = 209.6$ y $s = 30.65$ y luego estos valores los tomamos como μ y σ respectivamente. Sin embargo, es sabido que estimar los parámetros para definir la distribución nula altera sus propiedades (esto es, la distribución tabulada no es la correcta). Al uso de estimación de parámetros para especificar la nula y usar los cuantiles correctos de D_n se le llama "Prueba de Lilliefors". Es fácil encontrar en internet tablas para esta prueba, aunque también es fácil simular en R la distribución del estadístuco de Lilliefors, como se muestra enseguida

```
# Lilliefors
M  <- 50000
Dn <- rep(0,M)
n  <- 20
y  <- (1:n)/n
y0 <- y - 1/n

for(i in 1:M){
 dat   <- sort(rnorm(n))
 yt    <- pnorm(dat,mean=mean(dat),sd=sd(dat))
 Dn[i] <- max(max(abs(yt-y)),max(abs(yt-y0))) }

quantile(Dn,p=c(.8,.85,.9,.95,.99))

      80%       85%       90%       95%       99%
0.1586585 0.1661773 0.1764335 0.1918497 0.2229242
```

El valor crítico correcto es $D_{n,.05} = 0.19185$, así que la conclusión de que los datos de pesos de pollitos son razonablemente normales no cambia.

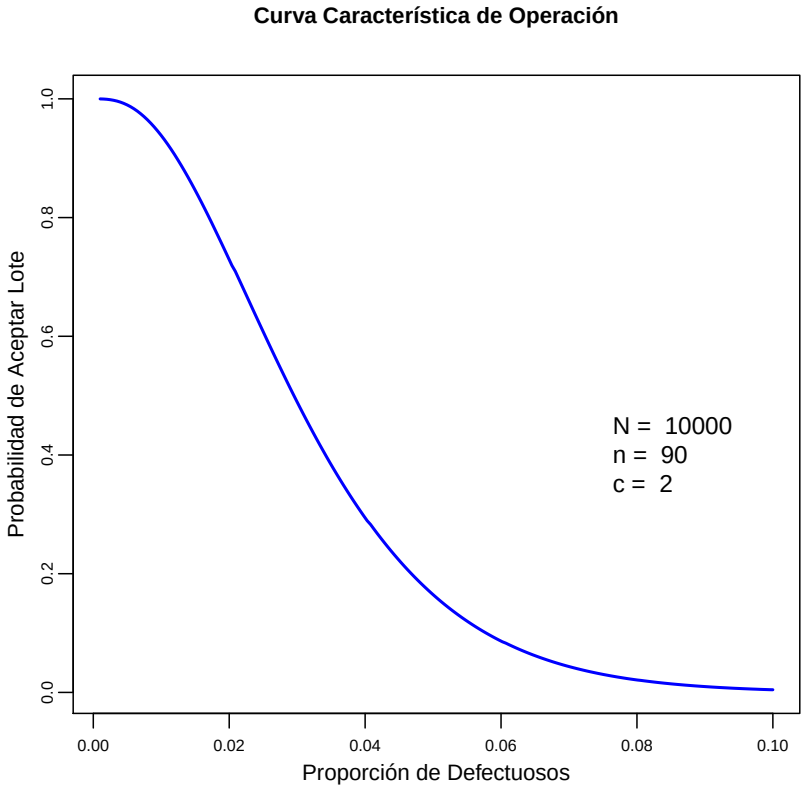
Aceptación de lotes por muestreo. Suponga una compañía manufacturera que recibe lotes de partes de ciertos proveedores. Suponga que un lote es de tamaño $N = 10,000$ y la compañía tiene el derecho de rechazar el lote si considera que no está dentro de ciertas especificaciones de calidad. Es claro que examinar todo el lote de 10,000 piezas está fuera de consideración; una alternativa común es tomar una muestra aleatoria de partes y decidir rechazar el lote si el número de piezas defectuosas es menor que cierto número pequeño aceptable. Ahora bien, ¿De que tamaño debemos tomar la muestra?, ¿Qué valor del número "pequeño" de defectuosos es adecuado?.

Usando simulación deseamos evaluar diferentes esquemas de muestreo. Empezamos considerando un programa de aceptación en el cual seleccionamos una muestra aleatoria de $n = 90$ piezas y rechazaríamos el lote si el número de defectuosos, X, es menor o igual a $c = 2$. En otras palabras, bajo este esquema, consideramos aceptable que el lote pueda tener hasta un 2.2% de defectuosos; ahora bien, si realmente el lote tuviera una proporción mayor, ¿cuál es mi riesgo de aceptarlo?, esto es, ¿Cuál es la probabilidad de que el número de defectuosos, X, sea menor o igual a 2 cuando la proporción real esta por arriba de 2.2%?. Para calcular estas probabilidades usamos la distribución Hipergeométrica.

Suponga que en un lote de tamaño N hay K defectuosos. Si tomo una muestra de tamaño n, la probabilidad de que X sea menor o igual a c es

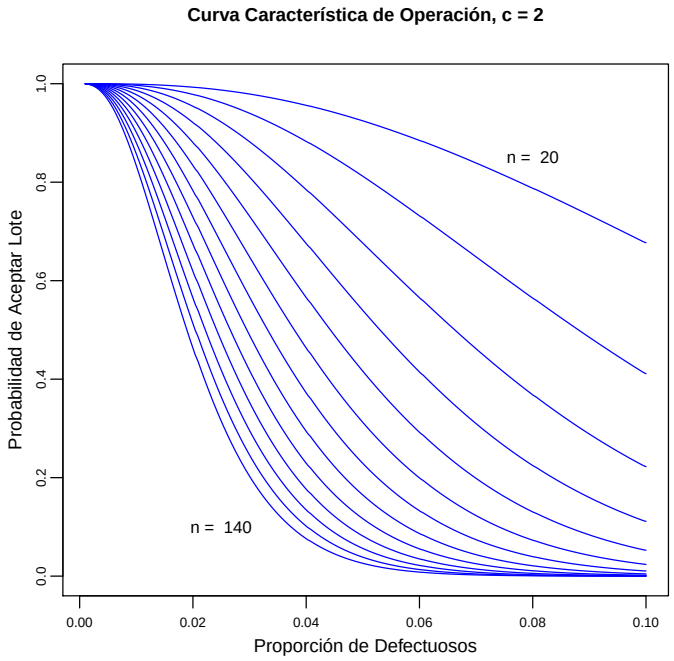
$$P(X \leq c) = \sum_{k=0}^{c} \frac{\binom{N-K}{n-k}\binom{K}{k}}{\binom{N}{n}}$$

La siguiente gráfica muestra el resultado de los cálculos de la fórmula anterior para diferentes valores de K y, para los gerentes de comparas de una compañía, la información contenida en ella es muy valiosa para saber los riesgos de adoptar ese plan de aceptación por muestreo.



```
# Evaluacion de un plan simple, con n=90 y c=2
N   <- 10000                        # Tamano de lote
n   <- 90                           # Tamano de muestra
c   <- 2                            # Aceptar si No. def. <= c
p   <- seq(.001,.10,length=200)
K   <- ceiling(p*N)
acp <- phyper(c, K, N-K, n)
plot(p,acp,type="l",mgp=c(1.5,.5,0), xlab="Proporcion de Defectuosos",
 ylab="Probabilidad de Aceptar Lote",lwd=2,col="blue",
 cex.lab=.9, cex.axis=.7,
 main="Curva Caracteristica de Operacion",cex.main=.9)
legend(.07,.50,bty="n",
 legend=c(paste("N = ",N),paste("n = ",n),paste("c = ",c)))
```

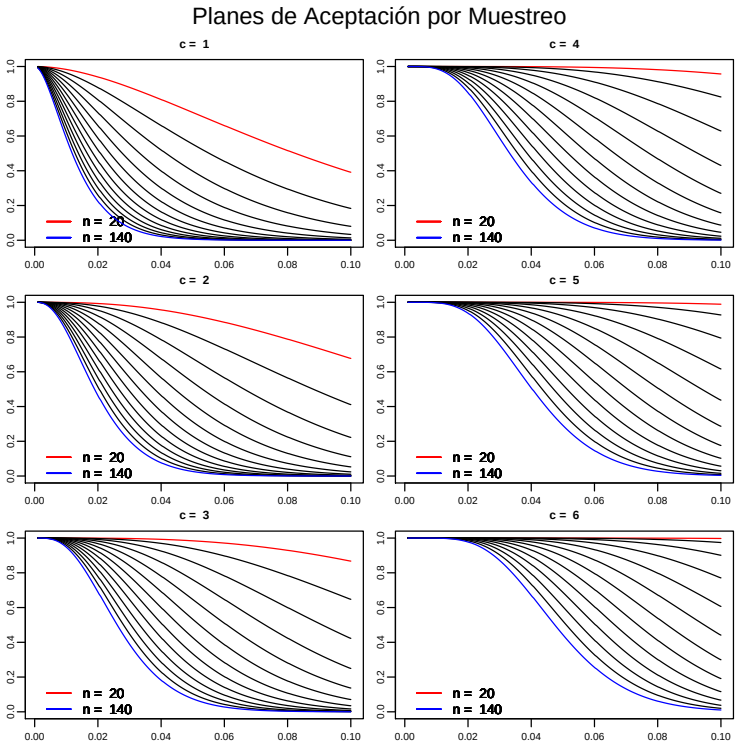
El programa anterior evaúa el plan $n = 90$ y $c = 2$, pero ¿qué pasa si cambio, por ejemplo, el valor de n?




```
# Evaluacion de un plan simple, con c=2 y varios valores de n
N   <- 10000
n   <- seq(20,140,by=10)
M   <- length(n)
c   <- 2
ng  <- 200
p   <- seq(.001,.10,length=ng)
K   <- ceiling(p*N)
acp <- matrix(0,ng,M)
for(j in 1:M){ acp[,j] <- phyper(c, K, N-K, n[j]) }
plot(p,acp[,1],type="n",mgp=c(1.5,.5,0), xlab="Proporcion de Defectuosos",
 ylab="Probabilidad de Aceptar Lote",lwd=2,col="blue",
 cex.lab=.9, cex.axis=.7, ylim=c(0,1),
 main="Curva Caracteristica de Operacion, c = 2",cex.main=.9)
```

```
for(j in 1:M){lines(p,acp[,j],col="blue")}
text(.08,.85,paste("n = ",n[1]),cex=.8)
text(.025,.1,paste("n = ",n[M]),cex=.8)
```

Finalmente, ¿qué pasa si variamos también el valor de c?




```
# Evaluacion de un plan simple, varios valores de n y c
N   <- 10000
n   <- seq(20,140,by=10)
M   <- length(n)
c   <- 1:6
nc  <- length(c)
ng  <- 200
p   <- seq(.001,.10,length=ng)
K   <- ceiling(p*N)
par(mfcol=c(3,2), mar=c(1, 1, 2, 1), oma=c(1,1,2,0))
for(i in 1:nc){
 acp <- matrix(0,ng,M)
 for(j in 1:M){ acp[,j] <- phyper(c[i], K, N-K, n[j]) }
 plot(p,acp[,1],type="n",mgp=c(1.5,.5,0), xlab="",
  ylab="",lwd=2,col="blue",
  cex.lab=.9, cex.axis=.7, ylim=c(0,1),
  main=paste("c = ",c[i]),cex.main=.9)
 for(j in 1:M){
  if(j == 1){lines(p,acp[,j],col="red"); next}
  if(j == M){lines(p,acp[,j],col="blue"); next}
  lines(p,acp[,j])
 legend(0,.2,col=c("red","blue"),bty="n",lty=1,
  legend=c(paste("n = ",n[1]),paste("n = ",n[M]))) }}
mtext("Planes de Aceptacion por Muestreo", outer=TRUE, cex=1.2)
```

Gráficos de Control. Suponga que se tienen datos de un proceso de manufactura que está operando en forma estable. Cada cierto tiempo, se toma una muestra y se registran las medidas de cierta característica de calidad. La estructura de los datos es

$$\begin{array}{lccccc|c} \text{muestra 1:} & x_{11} & x_{12} & \cdots & x_{1m} & \bar{x}_1 \\ \text{muestra 2:} & x_{21} & x_{22} & \cdots & x_{2m} & \bar{x}_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \text{muestra n:} & x_{n1} & x_{n2} & \cdots & x_{nm} & \bar{x}_n \end{array}$$

Suponga que la característica que estamos midiendo (por ejemplo, diaámetros de baleros) puede considerarse que esta distribuída normalmente con media μ y desviación estándar σ. Bajo estos supuestos, y suponiendo muestreo aleatorio, tenemos que

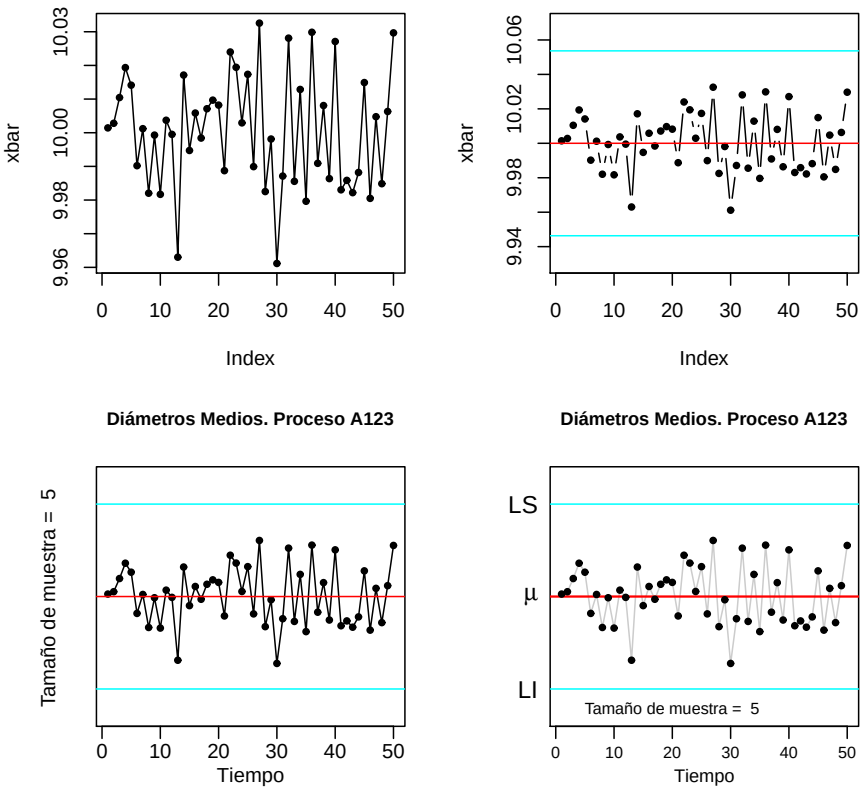
$$\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n \quad \sim \; N(\mu, \sigma^2/n)$$

Así que si graficamos estos valores, tenemos una buena idea de que esperar de la gráfica y ella nos puede servir para, visualmente, indicarnos si el proceso está en control.

En procesos industriales, el producir cartas (o gráficos) de control es una práctica estándar. La gráfica de enseguida mostramos datos simulados y el tipo de gráficos llamados "de control". Mostramos los mismos datos en gráficas con diferentes grados de desarrollo. Los límites de control son de la forma

$$\mu \pm 3\frac{\sigma}{\sqrt{n}}$$

bajo las condiciones establecidas, esperaríamos que solo 3 de cada 1000 muestras tengan un diámetro medio que se salga de los límites de control.



```
# Graficos de Control
# Supongamos datos normales con media y desviacion estandar conocidos.

mu   <- 10
de   <- .04
n    <- 50        # cuantos puntos de muestreo a lo largo del tiempo
m    <- 5         # cuantas observaciones en cada muestra
obs  <- matrix( rnorm(n*m,mean=mu,sd=de), ncol=m, byrow=T )
xbar <- apply(obs,1,mean)
LI   <- mu - 3*de/sqrt(m)
LS   <- mu + 3*de/sqrt(m)
aa   <- LS-LI
ry   <- c(LI-.15*aa,LS+.15*aa)

par(mfrow=c(2,2),mar=c(4,4,4,2))
# Primera version de grafico de control
plot(xbar,type="o",pch=20)     # help(plot) da lista de tipos de graficas
                               # help(points) da lista de "pch"
                               # help(par) da opciones para modificar graficas

# ahora queremos agregar los limites de control
plot(xbar,type="b",pch=20,ylim=ry)
abline(h=c(LI,mu,LS),col=c("cyan","red","cyan"))

# queremos modificar como se ven los ejes
plot(xbar,type="o",pch=20,ylim=ry,yaxt="n", mgp=c(1.5,.5,0),
     xlab="Tiempo",ylab=paste("Tamano de muestra = ",m),
     main="Diametros Medios. Proceso A123", cex.main=.9)
abline(h=c(LI,mu,LS),col=c("cyan","red","cyan"))

# queremos que las lineas no sean tan solidas
plot(xbar,type="l",ylim=ry, yaxt="n", mgp=c(1.5,.5,0),col=gray(.8),
     xlab="Tiempo",ylab="",cex.lab=.9,
     main="Diametros Medios. Proceso A123", cex.main=.9, cex.axis=.8)
abline(h=c(LI,mu,LS),col=c("cyan","red","cyan"),lwd=c(1,1.5,1))
mtext(c("LI","LS"),side=2, at=c(LI,LS), line=.5,las=2)
mtext(substitute(mu),side=2,at=mu,line=.5,las=2)
legend(0,LI,legend=paste("Tamano de muestra = ",m),bty="n",cex=.8)
points(xbar,pch=20)
```

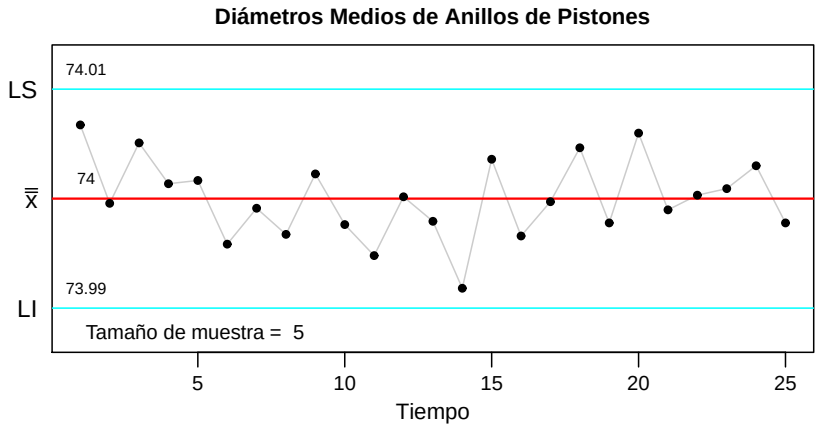
Ahora, en la práctica, los parámetros μ y σ no son conocidos, así que deberán ser estimados de la información que se está recabando o de registros históricos. Los gráficos anteriores son para monitorear la media del proceso, sin embargo también es importante monitorear la variabilidad. Una medida de la variabilidad es el "Rango", esto es, para una muestra dada, el Rango, R, es la diferencia entre el valor máximo y el mínimo. Los límites de control llamados "3-sigmas" tienen la estructura siguiente:

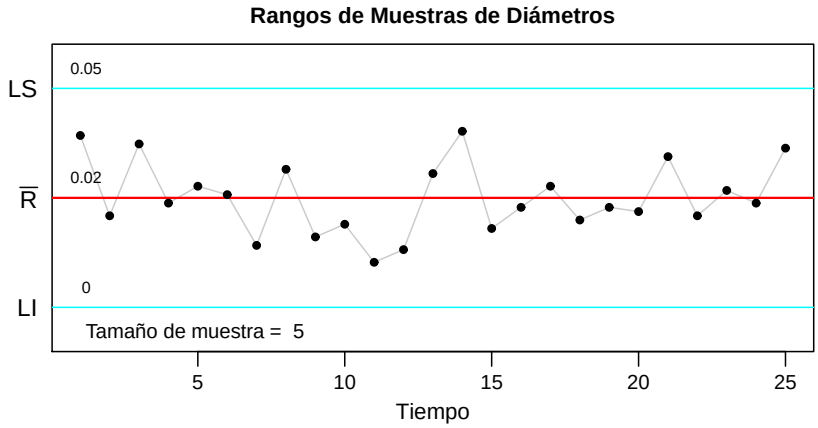
$$\bar{\bar{x}} \pm 3\frac{\bar{R}}{d_2\sqrt{m}}$$

$$\bar{R} \pm 3d_3\frac{\bar{R}}{d_2}$$

donde $\bar{\bar{x}} = \sum_{i=1}^{n} \bar{x}_i/n$, $\bar{R} = \sum_{i=1}^{n} R_i$ y d_2 y d_3 son ciertas constantes que pueden consultarse de tablas estadísticas.

La siguiente gráfica ilustra la construcción de estos gráficos.






```
##########################################
# Datos tomados del libro
# Montgomery, D.C. (1997)
# Introduction to statistical quality control
# (3rd ed.) Wiley
# 25 muestras de 5 observaciones cada una.
# Diametros (mm) de anillos de pistones (pag. 187)

dat  <- scan(
 "c:\\Documents and Settings\\Verano2010\\Montgp187Control.txt")
obs  <- matrix(dat,ncol=5,byrow=T)/1000
xbar <- rowMeans(obs)
rr   <- function(x){
 aa <- range(x)
 return(aa[2]-aa[1])}
rang <- apply(obs,1,rr)

# Construccion de graficos para la media y la dispersion
# Aqui se necesita de un par de valores obtenidos de tablas
d2   <- 2.326
d3   <- 0.864
xbb  <- mean(xbar)
Rb   <- mean(rang)
LIx  <- xbb - 3*Rb/(d2*sqrt(m))
LSx  <- xbb + 3*Rb/(d2*sqrt(m))
```

```
aa   <- LSx-LIx
ryx  <- c(LIx-.15*aa,LSx+.15*aa)
LIr  <- Rb  - 3*d3*Rb/d2
LSr  <- Rb  + 3*d3*Rb/d2
aa   <- LSr-LIr
ryr  <- c(LIr-.15*aa,LSr+.15*aa)

par(mfrow=c(2,1),mar=c(3, 3, 2, 2))
plot(xbar,type="l",ylim=ryx, yaxt="n", mgp=c(1.5,.5,0),col=gray(.8),
     xlab="Tiempo",ylab="",cex.lab=.9,
     main="Diametros Medios de Anillos de Pistones", cex.main=.9, cex.axis=.8)
abline(h=c(LIx,xbb,LSx),col=c("cyan","red","cyan"),lwd=c(1,1.5,1))
mtext(c("LI","LS"),side=2, at=c(LIx,LSx), line=.5,las=2)
mtext(expression(bar(bar(x))),side=2,at=xbb,line=.5,las=2)
legend(0,LIx,legend=paste("Tamano de muestra = ",m),bty="n",cex=.8)
points(xbar, pch=20)
text(x=rep(1.2,3),y=c(LIx,xbb,LSx),
     labels=as.character(round(c(LIx,xbb,LSx),2)),pos=3,cex=.7)

plot(rang,type="l",ylim=ryr, yaxt="n", mgp=c(1.5,.5,0),col=gray(.8),
     xlab="Tiempo",ylab="",cex.lab=.9,
     main="Rangos de Muestras de Diametros", cex.main=.9, cex.axis=.8)
abline(h=c(LIr,Rb,LSr),col=c("cyan","red","cyan"),lwd=c(1,1.5,1))
mtext(c("LI","LS"),side=2, at=c(LIr,LSr), line=.5,las=2)
mtext(expression(bar(R)),side=2,at=Rb,line=.5,las=2)
legend(0,LIr,legend=paste("Tamano de muestra = ",m),bty="n",cex=.8)
points(rang,pch=20)
text(x=rep(1.2,3),y=c(LIr,Rb,LSr),
     labels=as.character(round(c(LIr,Rb,LSr),2)),pos=3,cex=.7)
```

El Interpolador de Lagrange. Los polinomios pueden ser usados para aproximar funciones complicadas o para reconstruír un función cuando sólo conocemos su valor en algunos puntos (i.e. interpolación). Los polinomios de Lagrange son interpoladores que han sido ampliamente usados en las aplicaciones: En problemas de telecomunicaciones, en problemas de cuadratura, en problemas de ecuaciones diferenciales numéricas.

Lagrange: Si $x_0, x_1, \cdots, x_n$ son $n+1$ puntos distintos y f es una función cuyos valores están dados en esos puntos, entonces existe un polinomio único de gardo a lo más n con la propiedad de que

$$f(x_k) = P(x_k), \qquad \text{para cada } \; k = 0, 1, \cdots, n$$

este polinomio está dado por:

$$P(x) = f(x_0)L_0(x) + \cdots + f(x_n)L_n(x)$$

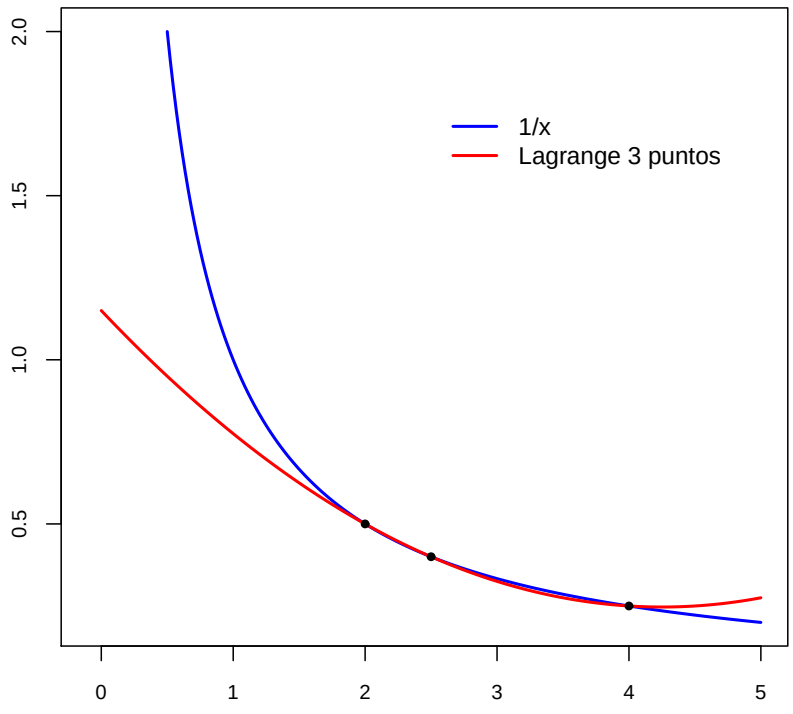
donde

$$L_k(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}, \qquad k = 0, 1, \cdots, n$$

A continuación presentamos una implementación en R.

```
# Interpolador de Lagrange
# Deseamos f, cuando sabemos f(2),f(2.5),f(4)
```

```
f   <- function(x){ return(1/x) }

xd  <- c(2,2.5,4)
fd  <- f(xd)        # (claro, en la practica no tenemos f explicitamente)
n   <- length(xd)-1
x   <- seq(0,5,length=200)

Lag <- 0
for(k in 1:(n+1)){
 Lk  <- 1
 for(i in (1:(n+1))[-k]){
  Lk <- Lk*(x-xd[i])/(xd[k]-xd[i]) }
 Lag <- Lag + fd[k]*Lk }

xx  <- seq(.5,5,length=200)
ff  <- f(xx)
plot(xx,ff,type="l",lwd=2,col="blue",xlim=c(-.1,5),xlab="",ylab="",
 mgp=c(2,1,0),cex.axis=.8)
lines(x,Lag,col="red",lwd=2)
points(xd,fd,pch=20)
legend(2.5,1.8,legend=c("1/x","Lagrange 3 puntos"),bty="n",
 lwd=2,col=c("blue","red"))
```



Prácticas Sesión 4

1. **Prueba Kolmogorov-Smirnov (Bajo estimación de parámetros)**. Suponga que $x_1, x_2, \cdots, x_n$ son una muestra aleatoria de una distribución de probabilidad $F(x)$. En base a esa muestra, deseamos probar la hipótesis

$$H_0 : F(x) = F_0(x) \qquad \text{vs} \qquad H_1 : F(x) \neq F_0(x)$$

La idea básica es construír la Función de Distribución Empírica, $F_n(x)$, y compararla contra la distribución hipotetizada $F_0(x)$. El estadístico natural de prueba es el de Kolmogorov-Smirnov:

$$D_n = \sup_x \mid F_n(x) - F_0(x) \mid$$

Si D_n es grande entonces esto implica que la Empírica esta lejos de la Teórica y, por lo tanto, rechazaríamos la hipótesis nula H_0. Ahora, para juzgar si D_n es grande o no, necesitamos la distribución nula de D_n. Si F_0 está completamente especificada, entonces la distribución nula de D_n es conocida y ha sido ampliamente tabulada, de modo que rechazamos H_0 si $D_n > D_{n,\alpha}$, donde $D_{n,\alpha}$ es un cuantil apropiado de esta distribución nula.

El presente problema consiste en estudiar el comportamiento de D_n para el caso en que F_0 **no está completamente especificada**; en este caso, no se conoce la distribución de D_n, sin embargo la podemos aproximar vía simulación. Consideremos el caso particular en donde F_0 corresponde a la función de distribución Normal. Suponga que la media y varianza de esta distribución no se conocen.

(a) Desarrolle un estudio de simulación para tabular la distribución nula del estadístico de Kolmogorov-Smirnov, bajo el supuesto que estimamos μ y σ^2 a partir de los datos. Esto es, si tengo $x_1, x_2, \cdots, x_n$ y deseo contrastar H_0 vs H_1, entonces calculo $\widehat{\mu} = \bar{x}$ y $\widehat{\sigma}^2 = s^2$; luego calculo D_n como arriba, pero con F_0 la distribución $N(\widehat{\mu}, \widehat{\sigma}^2)$ y entonces el problema es ¿Cuál es la distribución de D_n?. Los elementos mínimos que se deben incluír en la simulación:

- Varios tamaños de muestra.
- Cálculo de cuantiles de la distribución que sean usuales para prueba de hipótesis.

(b) Efectúe un estudio de potencia para ver el poder de D_n ante las alternativas Uniforme(0,1), χ^2 con 3 grados de libertad y t de Student con 3 grados de libertad. En otras palabras, si mis n datos realmente son, por ejemplo, Uniforme(0,1), (pero yo no lo sé) y quiero ver si son normales, entonces calcularía D_n para efectuar la prueba y lo que uno quisiera es que se rechazara H_0 (normalidad), entonces, lo que se tiene que hacer aquí es responder a ¿Cuál es la probabilidad de que rechaze H_0? (i.e. ¿Cuál es la potencia de la prueba K-S ante la hipótesis alterna de que la distribución es Uniforme?) (hacer el estudio similar con las alternativas χ^2 y t).

(c) Cuál prueba es más potente, ¿la Jarque-Bera o la Kolmogorov-Smirnov del inciso (a)? (comparar las potencias de esas pruebas ante las mismas alternativas consideradas en (b)).

(d) Investigue el uso de la función `ks.test()`. ¿Toma en cuenta esta función si los parámetros son estimados a partir de los datos?

En la literatura podrán encontrar mucho acerca de este tema. En particular, la prueba que ustedes desarrollarán en el inciso (a) lleva el nombre de "Prueba de Lilliefors" (El artículo original donde se trata esto, es: Lilliefors, H. (1967) On the Kolmogorov-Smirnov tests for normality with mean and variance unknown. JASA, Vol. 62, pp 399-402 - aquí encontrarán tablas semejantes a las que les pido que hagan).

---

2. **Regresión Nolineal**. Sea $g(x)$ una función real; esto es $g : \mathbb{R}^p \to \mathbb{R}$. Suponga que deseamos minimizar esta función. El primer paso en el método de Newton-Raphson consiste en dar una aproximación de segundo orden para g alrededor de algún valor inicial x_0:

$$g(x) \approx g(x_0) + \frac{\partial^T g(x_0)}{\partial x}(x - x_0) + \frac{1}{2}(x - x_0)^T \frac{\partial^2 g(x_0)}{\partial x \partial^T x}(x - x_0)$$

En vez de minimizar g, el Newton-Raphson lo que hace, en un segundo paso, es minimizar la aproximación del lado derecho. Como esta aproximación es una cuadrática, puede minimizarse en forma directa. Puede verse que el mínimo es

$$x_1 = x_0 - \left[\frac{\partial^2 g(x_0)}{\partial x \partial^T x} \right]^{-1} \frac{\partial g(x_0)}{\partial x}$$

Después, este valor x_1 se convierte ahora en nuestro nuevo punto inicial e iteramos este procedimiento hasta convergencia.

$$x_{k+1} = x_k - \left[\frac{\partial^2 g(x_k)}{\partial x \partial^T x} \right]^{-1} \frac{\partial g(x_k)}{\partial x}$$

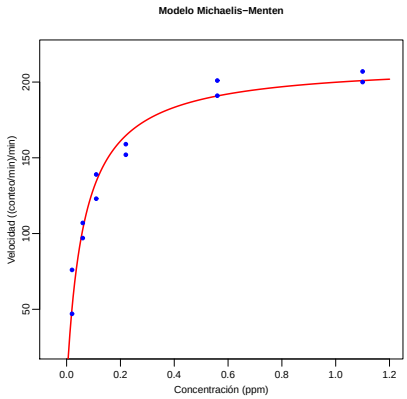
(a) Considere la función de Rosenbrock

$$g(u_1, u_2) = 100(u_2 - u_1^2)^2 + (1 - u_1)^2$$

Escriba un programa para encontrar el mínimo de esta función usando el método de Newton-Raphson (Use valores iniciales: $u_1 = -1.2$ y $u_2 = 1$).

(b) La gráfica de abajo fue hecha con los siguientes comandos:

```
conc <- c(.02,.02,.06,.06,.11,.11,.22,.22,.56,.56,1.1,1.1)
vel  <- c(76,47,97,107,123,139,159,152,191,201,207,200)
plot( conc, vel, xlim=c(-0.05,1.2), ylim=c(25,220),cex.lab=.8,
    xlab = "Concentracion (ppm)", mgp=c(1.5,.5,0), cex.axis=.8,
    ylab = "Velocidad ((conteo/min)/min)", cex.main=.8,
    main = "Modelo Michaelis-Menten", type="n")
t1 <- 212.6837;       t2 <- 0.06412128
xx <- seq(0,1.2,length=200)
yy <- t1*xx/(t2+xx)
lines(xx,yy,lwd=2,col="red")
points(conc,vel,pch=20,col="blue")
```



Se trata de resultados de un experimento en un estudio sobre cinética de reacciones enzimáticas. El modelo Michaelis-Menten relaciona la "velocidad" inicial de una reacción con la concentración del sustrato mediante la ecuación

$$f(x; \theta) = \frac{\theta_1 x}{\theta_2 + x}$$

donde x es la concentración y f es la velocidad media inicial (Ver "Nonlinear regression" en Wikipedia). La interpretación de los parámetros de este modelo es como sigue: Se puede ver fácilmente que θ_1 es la asíntota de la velocidad cuando x (la concentración) se hace grande y que θ_2 es la concentración que corresponde a un medio de la velocidad final. Los valores especícos usados para graficar la curva ($\theta_1 = 212.68$ y $\theta_2 = 0.064$) fueron obtenidos mediante mínimos cuadrados. Esto es, encontrando aquellos valores de $\theta = (\theta_1, \theta_2)^T$ que minimizen la suma de los cuadrados de los errores:

$$S(\theta) = \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2$$

donde las y_i's son las velocidades observadas y las x_i's son las correspondientes concentraciones.

Usando el método de Newton-Raphson, encuentre los estimadores de mínimos cuadrados de θ_1 y θ_2 (esto es, verifique que son $\widehat{\theta}_1 = 212.68$ y $\widehat{\theta}_2 = 0.064$) (Use 200 y 0.1, como valores iniciales para los parámetros).

(c) Siempre es conveniente dar información acerca de la precisión de nuestras estimaciones. Para el caso que nos ocupa, queremos saber los errores estándar correspondientes a las estimaciones $\widehat{\theta}_1 = 212.68$ y $\widehat{\theta}_2 = 0.064$. ¿Cómo hacer esto?.

- Calcule la matriz $n \times 2$:

$$F = \begin{bmatrix} \frac{\partial f(x_1;\theta)}{\partial \theta_1} & \frac{\partial f(x_1;\theta)}{\partial \theta_2} \\ \frac{\partial f(x_2;\theta)}{\partial \theta_1} & \frac{\partial f(x_2;\theta)}{\partial \theta_2} \\ \vdots & \vdots \\ \frac{\partial f(x_n;\theta)}{\partial \theta_1} & \frac{\partial f(x_n;\theta)}{\partial \theta_2} \end{bmatrix}_{\theta=\widehat{\theta}}$$

- Calcule el estimador de la varianza

$$\widehat{\sigma}^2 = \frac{1}{n-2} \sum_{i=1}^{n} (y_i - f(x_i; \widehat{\theta}))^2$$

- Calcule el estimador de la matriz, 2×2, de varianzas y covarianzas de los estimadores

$$\text{Var}(\widehat{\theta}) = \widehat{\sigma}^2 (F^T F)^{-1}$$

- Los errores estándar de $\widehat{\theta}_1$ y $\widehat{\theta}_2$ serán la raíz cuadrada de los elementos diagonales de $\text{Var}(\widehat{\theta})$.

Encuentre los errores estándar asociados a $\widehat{\theta}_1 = 212.68$ y $\widehat{\theta}_2 = 0.064$.

(d) Un modelo alternativo podría ser:

$$f(x; \theta) = \theta_1 \left(1 - e^{-\theta_2 x}\right)$$

Ajuste este modelo exponencial y compárelo con el Michaelis-Menten. ¿Cúal sería mejor?.

(e) Efectúe nuevamente la estimación del modelo Michaelis-Menten, pero ahora usando la función `nls()`.

---

3. **Integración Montecarlo**. La circunferencia inscrita en el cuadrado, que se muestra más adelante a la izquierda fue hecha con los siguientes comandos:

```
h  <- 1;   r  <- 1
xx <- seq(0,2*h,length=200)
yu <- h + sqrt(r^2-(xx-h)^2)
yl <- h - sqrt(r^2-(xx-h)^2)
plot(h,h,xlab="",ylab="",type="n", mgp=c(1.5,.5,0),
 xlim=c(0,2*h), ylim=c(0,2*h), cex.axis=.8, bty="n" )
```

```
lines(xx,yu,lwd=2,col="red")
lines(xx,yl,lwd=2,col="red")
segments(0,0,2*h,0,lwd=2,col="blue")
segments(2*h,0,2*h,2*h,lwd=2,col="blue")
segments(2*h,2*h,0,2*h,lwd=2,col="blue")
segments(0,2*h,0,0,lwd=2,col="blue")
abline(h=h,v=h,col=gray(.8))
text(1.4,.6,"C",cex=1.5,col="red")
```

Si genero un punto al azar dentro del cuadrado, la probabilidad de que ese punto también esté dentro de la circunferencia, es igual al cociente de las dos áreas

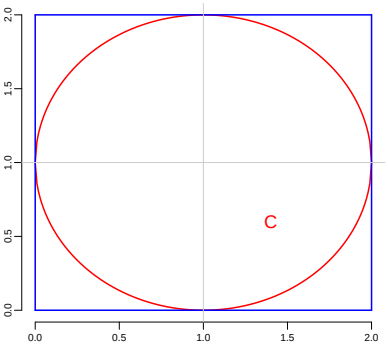
$$p = \frac{A_{\text{circ}}}{A_{\text{cuad}}} = \frac{A_{\text{circ}}}{4}$$

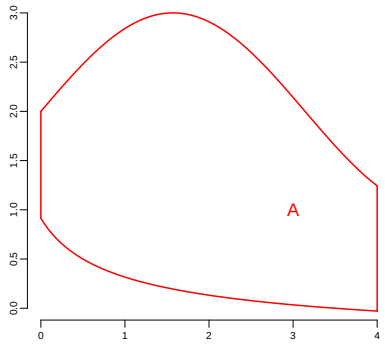
de modo que $A_{\text{circ}} = 4\,p$.

(a) Usando simulación, estime el área del círculo. Para ello, genere x_1, Uniforme entre 0 y 2; luego genere y_1 también Uniforme entre 0 y 2. Si el punto (x_1, y_1) cae dentro de la circunferencia, registramos un 1 (un "exito") si no, entonces registramos un 0 ("fracaso"). Hacemos esto muchas veces y tendremos una secuencia de 1's y 0's, la cual es una muestra aleatoria de una distribución Bernoulli(p). Entonces podemos estimar p mediante $\widehat{p}$, la proporción observada de éxitos y, con ello, podemos estimar el área del círculo.

(b) Usando técnicas similares a las del inciso anterior, estime el área de la región, A, acotada por las curvas en rojo de la gráfica de abajo a la derecha. Las curvas superior e inferior son, respectivamente

$$s(x) = \text{Sen}(x) + 2 \qquad \text{y} \qquad l(x) = \frac{1}{\sqrt{x + 0.5}} - \frac{1}{2}$$





(c) En los dos incisos anteriores es fácil saber, sin necesidad de simulación, los valores correctos de las respectivas áreas; sin embargo, en otras situaciones no es posible conocer los valores verdaderos, así que es conveniente tener una idea del error que se comete con estos métodos de simulación. Si $\widehat{p}$ es la proporción observada de éxitos y es un estimador de la proporción verdadera, p, entonces un intervalo de confianza para p, está dado por

$$\widehat{p} \pm z_{\alpha/2}\sqrt{\frac{\widehat{p}\,(1-\widehat{p})}{n}}$$

de aquí, el error máximo para estimar p, con una confianza del $100(1-\alpha)\%$, está dado por $z_{\alpha/2}\sqrt{\widehat{p}\,(1-\widehat{p})/n}$. Para los incisos anteriores calcule los errores máximos en las estimaciones de las áreas y compárelos con los errores reales.

(d) Consideremos de nuevo el problema del área del círculo. Básicamente, lo que estamos haciendo es calcular la integral doble

$$I = \int_0^2 \int_0^2 g(x,y)\, dx\, dy = \int_0^2 \int_0^2 I_{\{(x,y)\,\in\, \mathsf{C}\}}\, dx\, dy$$

donde $g(x,y) = I_{\{(x,y)\,\in\, \mathsf{C}\}}$ es la función indicadora de la región C, la cuál vale 1 si $(x,y) \in$ C y vale 0 de otra forma. Cuando dijimos "genere x_1, Uniforme entre 0 y 2; luego genere y_1 también Uniforme entre 0 y 2", lo que estábamos haciendo era simular de la Uniforme en el rectángulo $(0,2)\times(0,2)$; esto es, generamos una realización (x,y) de la densidad uniforme bivariada

$$U(x,y) = \begin{cases} \frac{1}{4} & \text{si } 0 \le x \le 2,\ 0 \le y \le 2 \\ 0 & \text{de otra forma} \end{cases}$$

Note que

$$I \;=\; \int_0^2 \int_0^2 \frac{I_{\{(x,y)\,\in\, \mathsf{C}\}}}{U(x,y)}\, U(x,y)\, dx\, dy \;=\; E\left[\frac{I_{\{(X,Y)\,\in\, \mathsf{C}\}}}{U(X,Y)}\right] \;\equiv\; E\left[h(X,Y)\right]$$

esto es, la integral, I, no es otra cosa más que el valor esperado de $h(X,Y)$. Ahora, si tenemos una muestra aleatoria $(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)$, entonces, por la Ley de los Grandes Números, tenemos que

$$\frac{1}{n}\sum_{i=1}^{n} h(x_i,y_i) \;\to E[h(X,Y)] = I$$

y ya está; ésta observación nos dice que si queremos aproximar la integral I, entonces podemos generar muchas (x_i,y_i)'s y promediar todas las $h(x_i,y_i)$'s y esto sería $\widehat{I}$. Entre paréntesis, para el problema en el cuadrado, tenemos que $U(x,y) = 1/4$ y entonces $h(x,y) = 4I_{\{(x,y)\,\in\, \mathsf{C}\}}$.
Tomemos estas ideas y consideremos un problema más simple. Consideremos la integral

$$I = \int_a^b g(x)dx$$

Sea $f(x)$ una densidad (de la cual podamos generar muestras aleatorias); entonces

$$I = \int_a^b g(x)dx = \int_a^b \frac{g(x)}{f(x)}\, f(x)dx \equiv \int_a^b h(x)\, f(x)dx \;=\; E[\,h(X)\,] \;\leftarrow\; \frac{1}{n}\sum_{i=1}^{n} h(x_i)$$

donde $x_1,\cdots,x_n$ son una muestra aleatoria tomada de la densidad $f(x)$.
Considere la integral

$$I = \int_0^{\pi/2} \mathsf{Sen}(x)\, dx$$

- Aproxime el valor de I usando simulación (usando como f a una Uniforme).
- Nuevamente aproxime a I pero ahora usando muestras tomadas de $f(x) = 8x/\pi^2$, para $0 \le x \le \pi/2$. ¿Cuál elección de f fue mejor?.

En este segundo punto, la densidad de donde se muestrea, $f(x)$, es parecida a la función que se desea integrar, $g(x)$; a esto se le llama "muestreo por importancia". En general, para integrales unidimensionales no se usa integración Montecarlo, son preferidos los métodos de Cuadratura; sin embargo, para integrales multidimensionales, los enfoques basados en simulación son muy valiosos pues las fórmulas para usar Cuadratura se vuelven muy complejas e ineficientes, mientras que integración Montecarlo es, en general, sencilla de aplicar.

4. Suponga que una cadena de DNA de longitud a se parte aleatoriamente en dos partes. Use simulación para estimar la distribución del cociente de la longitud de la cadena mayor y la longitud de la cadena menor.

5. Genere un punto al azar en un cuadrado de lado a. La distancia del punto al centro del cuadrado es aleatoria, ¿Que distribución tiene? (use simulación).

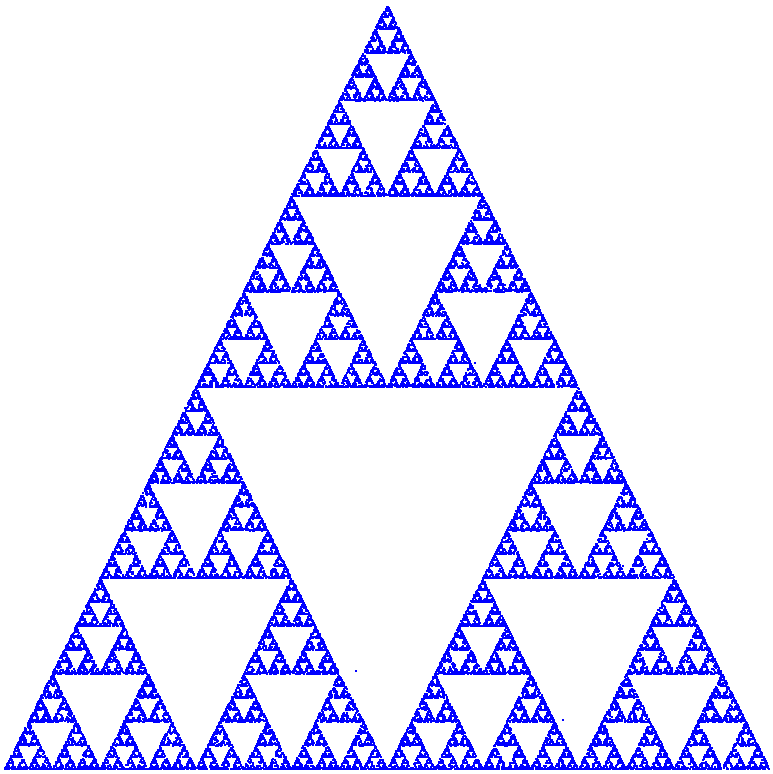
6. Si genero n puntos al azar dentro del cuadrado de lado a y calculo la distancia del centro al punto más cercano, ¿Qué distribución tiene esa distancia?

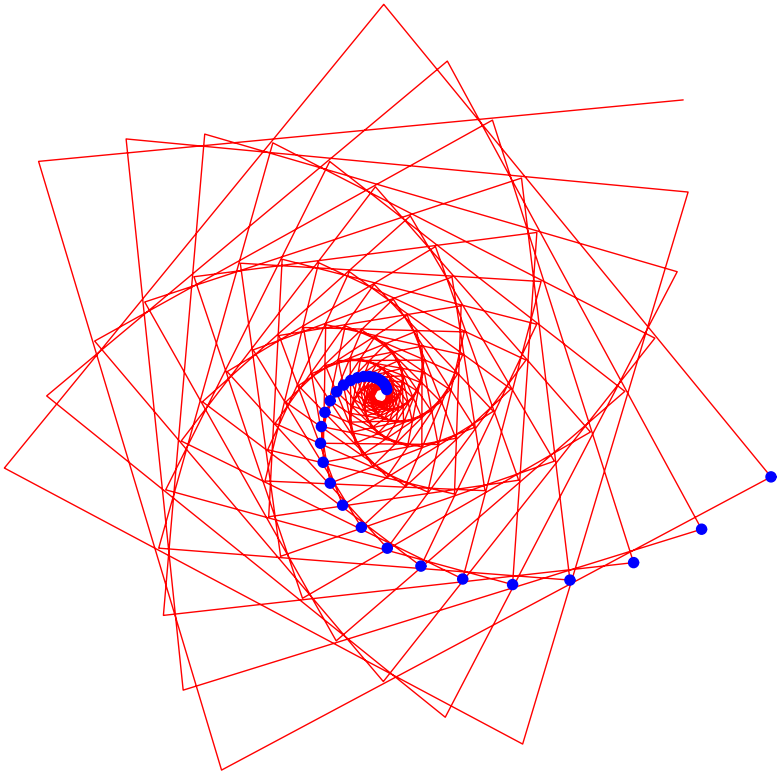
7. Use simulación para investigar que tan probable es que el polinomio cuadrático

$$q(x) = ax^2 + bx + c$$

tenga raíces reales. Los coeficientes considérelos uniformemente distribuídos en $(0, 1)$.

Apéndice Sesión 4





Apéndice Sesión 4

```
#########################################################################
# Figura 1: Sierpinsky
#
n  <- 50000
x  <- matrix(c(0,0,1,1,2,0),ncol=2,byrow=T)
r  <- runif(3)
p0 <- (colSums(r*x))/sum(r)
par(mar=c(0,0,0,0)+.1)
plot(1,1,type="n",xlim=range(x[,1]),ylim=range(x[,2]),
 xlab="", ylab="", xaxt="n", yaxt="n", bty="n")
points(p0[1],p0[2], pch=".", col="blue")
for( i in 1:n ){
 na <- sample( 1:3, size=1 )
 pm <- (x[na,]+ p0)/2
 points(pm[1],pm[2], pch=".", col="blue")
 p0 <- pm }
#########################################################################

#########################################################################
# Figura 2: Ejemplo tomado del libro de Meyn y Tweedie (1993)
# Markov Chains and Stochastic Stability, p.7
GG   <- matrix( c(-0.2,1,-1,-0.2), ncol=2, byrow=T )
evol <- function(p){ GG%*%p }
m    <- 200
tray <- matrix(0,m,2)
tray[1,] <- c(1,1)     # punto inicial
for( i in 2:m ){ tray[i,] <- evol(tray[i-1,]) }
xx <- tray[,1]
yy <- tray[,2]
xr <- range(xx)
yr <- range(yy)
par(mar=c(0,0,0,0)+1)
plot(xx,yy, type="l", bty="n", col="red", xlab="", ylab="",
     xaxt="n", yaxt="n", lwd=1, xlim=.94*xr, ylim=.94*yr )
pps <- seq(1,m, by=7)
points( tray[pps,1],tray[pps,2], col="blue", pch=20, cex=1.5 )
#########################################################################
```

VIERNES

Notas Sesión 5

Generación de variables aleatorias. A lo largo de este curso hemos usado funciones de R que nos permiten generar variables aleatorias, por ejemplo, `runif, rnorm, rexp`. Ahora le daremos un vistazo a los métodos que estan detrás de estos generadores.

Como veremos, todos los generadores de números aleatorios son de naturaleza determinística; es decir, si al generar números aleatorios obtengo la secuencia $u_1, u_2, \cdots, u_n$, entonces, siempre obtendremos esa misma secuencia. Sin embargo, son tales que parecen aleatorios; es por ello que algunas veces son llamados "pseudo-aleatorios".

Las características principales que debe cumplir un generador de números aleatorios uniformes son:

- Distribución uniforme
- Independencia
- Repetibilidad y portabilidad
- Rapidez computacional

Uno de los primeros métodos propuestos fue el método de cuadrados de von Neumann (1951). No es recomendable su uso pero lo incluímos solo por su importancia histórica. El método de cuadrados para generar números aleatorios de 4 dígitos consiste en tomar los 4 dígitos centrales del cuadrado del número anterior en la secuencia, por ejemplo, si iniciamos en 9876:

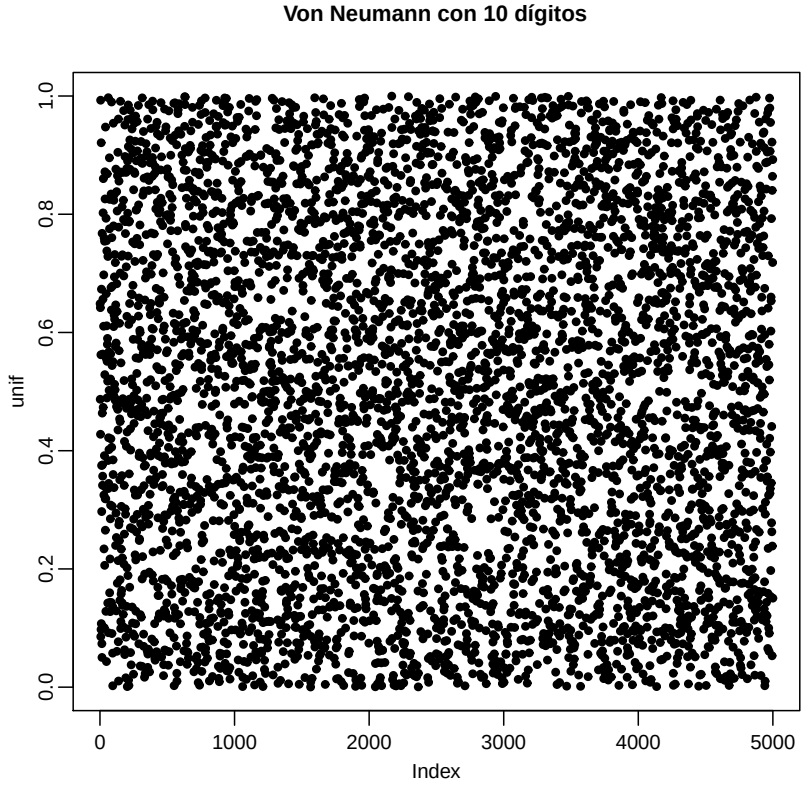
```
9876^2 = 97535376
5353^2 = 28654609
6546^2 = 42850116
8501^2 = 72267001
2670^2 =  7128900
1289^2 =  1661521
6615^2 = 43758225 , etc.
```

La secuencia generada es: 9876, 5353, 6546, 8501, 2670, 1289, 6615, 7582,...

Un programa en R que implementa el método de cuadrados es:

```
# Metodo de los cuadrados de von Neumann
# con 4 digitos (no buena idea)
neumann <- function(seed){ (floor( (seed^2)/10^2 )) %% 10^4 }
RCUAD <- function(n,seed){
 mu  <- rep(0,n)
 for(i in 1:n){
  seed  <- neumann(seed)
  mu[i] <- seed }
 return(mu/10^4) }
seed <- 9876             # cicla al 50-avo
n    <- 100
unif <- RCUAD(n,seed)
# con 10 digitos
neumann <- function(seed){ (floor( (seed^2)/10^5 )) %% 10^10 }
RCUAD <- function(n,seed){
 mu  <- rep(0,n)
 for(i in 1:n){
  seed  <- neumann(seed)
  mu[i] <- seed }
 return(mu/10^10) }
```

```
n    <- 5000
seed <- 9182736450
unif <- RCUAD(n,seed)
plot(unif,pch=20,cex.axis=.8, cex.lab=.8, mgp=c(1.5,.5,0),
     main="Von Neumann con 10 digitos",cex.main=.9)
```



(se ve bien: aleatorio y uniforme)

Métodos Congruenciales. El método de generadores congruenciales es el más ampliamente usado y conocido. Los generadores están basados en la siguiente fórmula:

$$U_i = (aU_{i-1} + c) \mod m$$

donde

$$U_i = \text{números enteros pseudo-aleatorios}$$
$$U_0 = \text{valor de inicio (semilla), se escoge al azar}$$
$$a, c, m = \text{constantes (o parámetros) que definen al generador}$$

Note que para convertir estas variables a variables uniformes (0,1), solo se necesita dividirlas por m, esto es, se usa la secuencia $\{U_i/m\}$. Para ejemplificar, consideremos el funcionamiento del generador con $m = 10$ y

$U_0 = a = c = 7$:

$$\begin{aligned} U_1 &= (7 \times 7 + 7) \bmod 10 = (56) \bmod 10 = 6 \\ U_2 &= (7 \times 6 + 7) \bmod 10 = (49) \bmod 10 = 9 \\ U_3 &= (7 \times 9 + 7) \bmod 10 = (70) \bmod 10 = 0 \\ U_4 &= (7 \times 0 + 7) \bmod 10 = (7) \bmod 10 = 7 \\ U_5 &= (7 \times 7 + 7) \bmod 10 = (56) \bmod 10 = 6 \\ &\cdots \text{etc.} \end{aligned}$$

Claramente la secuencia 7, 6, 9, 0, 7, 6, 9, 0, 7, ... no es muy aleatoria que digamos.

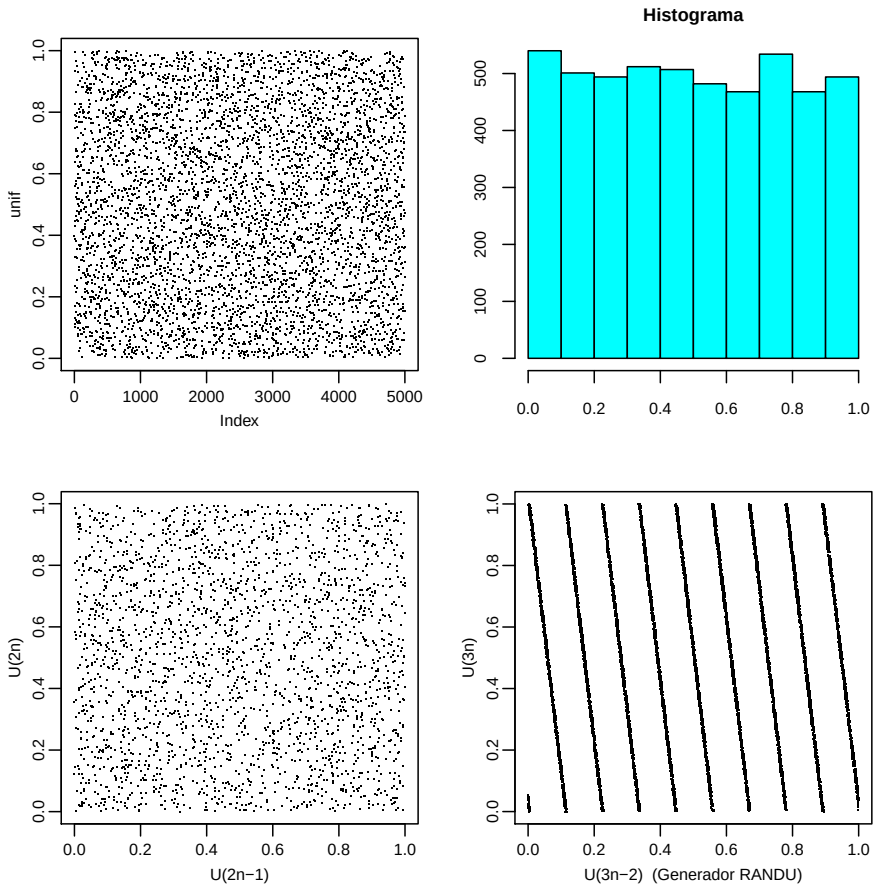
Algunos comentarios acerca de los métodos congruenciales:

- Dado que se utiliza la función "mod m", los posibles valores que produce el algoritmo son los enteros $0, 1, 2, \cdots, m-1$. Lo anterior debido a que por definición "x mod m" es el residuo que queda después de dividir a x entre m.
- Debido a que el entero aleatorio U_i depende solamente del entero aleatorio previo U_{i-1}, una vez que se repita el valor previo, la secuencia entera deberá de repetirse. A tal secuencia repetida se le llama *ciclo*, y su periodo es la longitud del ciclo. La longitud máxima del periodo es m.
- Si se está interesado en generar variables uniformes (0,1), la partición más fina del intervalo (0,1) que provee este generador es: $\{0, 1/m, 2/m, \cdots, (m-1)/m\}$. Por supuesto que no es una distribución uniforme (0,1).
- Los valores de a, c y m determinan la finura con que se hace la partición del intervalo (0,1), la longitud del ciclo, la uniformidad de la distribución marginal y también afectan la propiedad de independencia de la secuencia que se genera.

Consideremos el generador congruencial con parámetros $a = 2^{16}+3$, $c = 0$ y $m = 2^{31}$. Este generador, conocido como RANDU, fue ampliamante usado en los 70's y 80's pues estaba implementado en el sistema operativo de los sistemas VAX y las máquinas IBM/370. Una implementación en R es como sigue:

```
##############
# Usando generadores congruenciales

randu <- function(seed) (a*seed) %% m

RANDU <- function(n,seed){
 a  <<- 2^16 + 3
 m  <<- 2^31
 mu  <- rep(0,n)
 for(i in 1:n){
  seed  <- randu(seed)
  mu[i] <- seed/m }
 return(list(mues=mu,lastseed=seed))}

# Uso de RANDU, generamos 5000 uniformes (0,1):
n    <- 5000
seed <- 45813
unif <- RANDU(n,seed)$mues
par2 <- matrix(unif,ncol=2,byrow=T)

par(mfrow=c(2,2), mar=c(3, 3, 2, 1))
plot(unif,pch=".",cex.axis=.8, cex.lab=.8, mgp=c(1.5,.5,0))
hist(unif,cex.axis=.8, col="cyan", cex.lab=.8, main="Histograma",cex.main=.9)
```

```
plot(par2[,1],par2[,2],pch=".",xlab="U(2n-1)", ylab="U(2n)",
 cex.axis=.8, cex.lab=.8, mgp=c(1.5,.5,0))

seguir <- T
n      <- 60000
seed   <- 45813
sel    <- rep(0,3)
nr     <- 10000

while( seguir ){
 sal  <- RANDU(n,seed)
 rRAN <- matrix(sal$mues,ncol=3,byrow=T)
 sel  <- rbind(sel,rRAN[(rRAN[,2]>.5)&(rRAN[,2]<=.51),])
 if( dim(sel)[1] > nr ) seguir <- F
 seed <- sal$lastseed }
sel <- sel[-1,]

plot(sel[,1],sel[,3],xlab="U(3n-2)  (Generador RANDU)",
 ylab="U(3n)",pch=".", cex.axis=.8, cex.lab=.8,mgp=c(1.5,.5,0))
```



Las primeras tres gráficas muestran el comportamiento de RANDU, y se ven bastante satisfactorias. Sin embargo, la cuarta gráfica muestra un comportamiento extraño no deseable: Si un valor se encuentra entre 0.5 y 0.51, entonces los valores adyacentes están altamente correlacionados.

Los siguientes parámetros han sido recomendados en la literatura por sus propiedades de uniformidad, independencia y facilidad de implementación. Todos ellos usan módulo $2^{31} - 1$ y tienen ciclos maximales de $2^{31} - 1$

(excepto el primero que tiene un ciclo de $2^{31}-2$), para propósitos prácticos $2^{31}-2 = 2,147,483,646$ debe alcanzar para casi cualquier simulación.

$$\begin{aligned} a &= 16,807 \\ a &= 62,809,911 \\ a &= 742,938,285 \\ a &= 950,706,376 \\ a &= 1,226,874,159 \\ a &= 1,343,714,438 \end{aligned}$$

La implementación en R es muy semejante a la presentada para RANDU:

```
RAND1 <- function(n,seed){
 a  <<- 16807
 m  <<- 2^31 - 1
 mu  <- rep(0,n)
 for(i in 1:n){
  seed  <- randu(seed)
  mu[i] <- seed/m }
 return(list(mues=mu,lastseed=seed))}
```

Generador congruencial combinado: Wichmann y Hill (1982) propusieron una combinación de tres generadores congruenciales

$$\begin{aligned} X_i &= 171\, X_{i-1} \mod 30269 \\ Y_i &= 172\, Y_{i-1} \mod 30307 \\ Z_i &= 170\, Z_{i-1} \mod 30323 \end{aligned}$$

y la secuencia de números generados se obtiene usando

$$U_i = \left\{ \frac{X_i}{30269} + \frac{Y_i}{30307} + \frac{Z_i}{30323} \mod 1 \right\}$$

Este generador tiene un ciclo de 6.952×10^{12} (los anteriores son del orden de 2×10^9)

```
randu <- function(seed) (a*seed) %% m
RANWH <- function(n,seed){
 a  <<- c( 171, 172, 170)
 m  <<- c(30269, 30307, 30323)
 mu  <- rep(0,n)
 for(i in 1:n){
  seed  <- randu(seed)
  mu[i] <- (sum(seed/m)) %% 1 }
 return(list(mues=mu,lastseed=seed))}
n    <- 5000
seed <- c(67612,92318,652612)
unif <- RANWH(n,seed)$mues
```

Generación de variables aleatorias no-uniformes. Un resultado importante para la generación de variables aleatorias no-uniformes es el Teorema de la Probabilidad Inversa:

Teorema: Si X es una variable aleatoria continua con función de distribución $F(x)$ y si U es una variable aleatoria uniforme en (0,1), entonces si $W = F^{-1}(U)$, se cumple que $W \stackrel{d}{=} X$

Como ejemplo, apliquemos este resultado a la distribución exponencial:

$$\text{Si } X \sim \text{Exp}(\lambda), \quad \text{entonces} \quad f(x) = \lambda e^{-\lambda x} \ \text{ y } \ F(x) = 1 - e^{-\lambda x}$$

Para usar el teorema, invertimos la función de distribución

$$F(x) = u \quad \Leftrightarrow \quad 1 - e^{-\lambda x} = u \quad \Leftrightarrow \quad x = -\frac{1}{\lambda}\log(1-u)$$

Entonces, para generar exponenciales con parmetro λ, generamos primero uniformes y luego usamos la transformación correspondiente,

$$X_i = -\frac{1}{\lambda}\log(1-U_i) \quad i = 1, 2, \cdots$$

Ahora, la distribución de $1 - U$ es la misma que la distribución de U, entonces el algoritmo para generar exponenciales es:

$$X_i = -\frac{1}{\lambda}\log(U_i) \quad i = 1, 2, \cdots$$

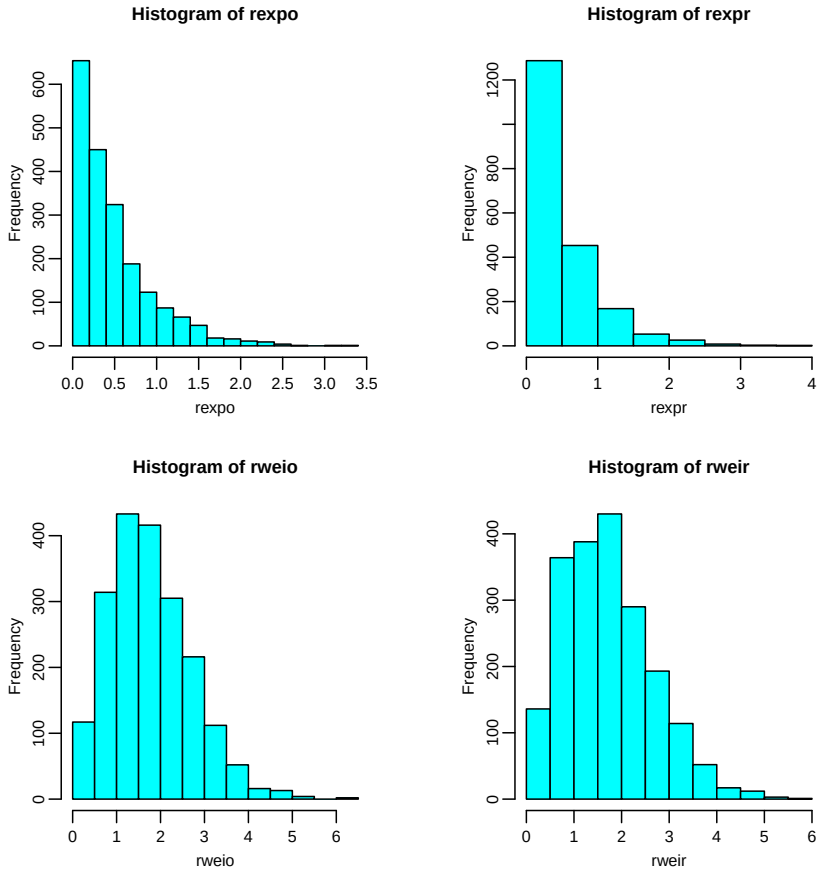
El caso de la distribución Weibull es igualmente sencillo:

$$\text{Si } X \sim \text{Weibull}(\theta, \beta), \quad \text{entonces} \quad f(x) = \frac{\beta}{\theta}\left(\frac{x}{\theta}\right)^{\beta-1} \exp\left\{-\left(\frac{x}{\theta}\right)^{\beta}\right\} \quad \text{y} \quad F(x) = 1 - \exp\left\{-\left(\frac{x}{\theta}\right)^{\beta}\right\}$$

$$F(x) = u \quad \Leftrightarrow \quad 1 - \exp\left\{-\left(\frac{x}{\theta}\right)^{\beta}\right\} = u \quad \Leftrightarrow \quad x = \theta\left[-\log(1-u)\right]^{1/\beta}$$

Nuevamente, podemos usar U en vez de $1 - U$ y el algoritmo para generar Weibulls es:

$$X_i = \theta\left[-\log(U_i)\right]^{1/\beta} \quad i = 1, 2, \cdots$$



Las gráficas de la izquierda son variables generadas a partir de uniformes. Las de la derecha usaron los generadores internos de R.

```
# Generar Exponenciales y Weibulls
set.seed=65656
n     <- 2000
lam   <- 2
rexpo <- -(1/lam)*log(runif(n))
rexpr <- rexp(n,rate=lam)
teta  <- 2
beta  <- 2
rweio <- teta*(-log(runif(n)))^(1/beta)
rweir <- rweibull(n, shape=beta, scale=teta )
par(mfrow=c(2,2),mar=c(3,3,3,3))
hist(rexpo, cex.axis=.8, cex.lab=.8, cex.main=.9, mgp=c(1.5,.5,0),col="cyan")
hist(rexpr, cex.axis=.8, cex.lab=.8, cex.main=.9, mgp=c(1.5,.5,0),col="cyan")
hist(rweio, cex.axis=.8, cex.lab=.8, cex.main=.9, mgp=c(1.5,.5,0),col="cyan")
hist(rweir, cex.axis=.8, cex.lab=.8, cex.main=.9, mgp=c(1.5,.5,0),col="cyan")
```

Generación de Normales. Método de Box y Muller. El uso del teorema de la probabilidad inversa es (en teoría) siempre posible de implementar pues la función de distribución siempre es monótona creciente (al menos para las variables continuas usuales) y por lo tanto siempre es posible invertirla. Sin embargo, para el caso de la distribución normal no es posible tener una forma cerrada para la inversa de su función de distribución, de modo que usamos otro procedimiento para la generación de normales.

La relación entre coordenadas cartesianas y polares es:

$$x = r\mathsf{Cos}(\theta) \qquad r^2 = x^2 + y^2$$
$$y = r\mathsf{Sen}(\theta) \qquad \theta = \mathsf{ArcTan}(y/x)$$

La clave del método de Box y Muller es observar que si $X \sim N(0,1)$ y $Y \sim N(0,1)$, X y Y independientes, entonces

$$r^2 \sim \mathsf{Exp}(\lambda = 1/2) \quad \text{y} \quad \theta \sim \mathsf{Unif}(0, 2\pi)$$

y como ya sabemos generar exponenciales y uniformes, pues ya está. El método de Box y Muller para generar variables normales estándar es:

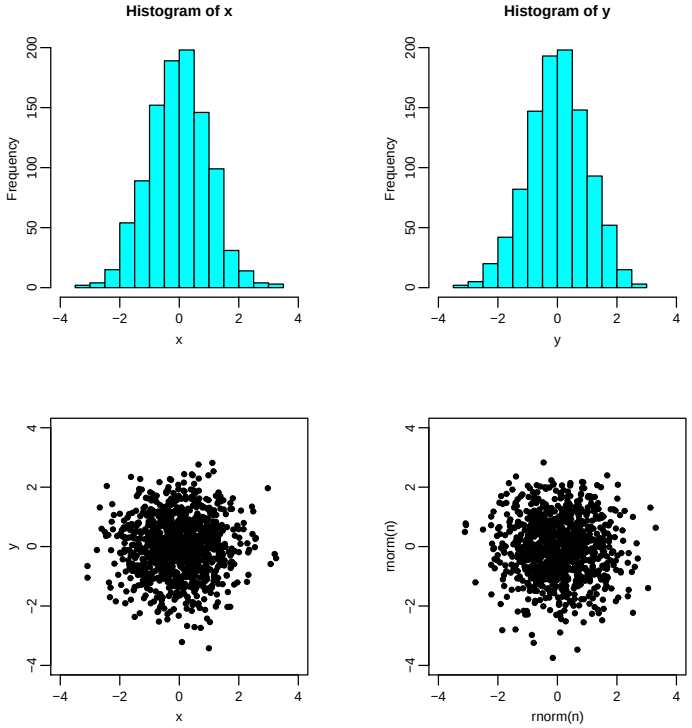
- Generar uniformes (0,1), $(U_{i1}, U_{i2}),\ i = 1, 2, \cdots, n$
- Las normales generadas son

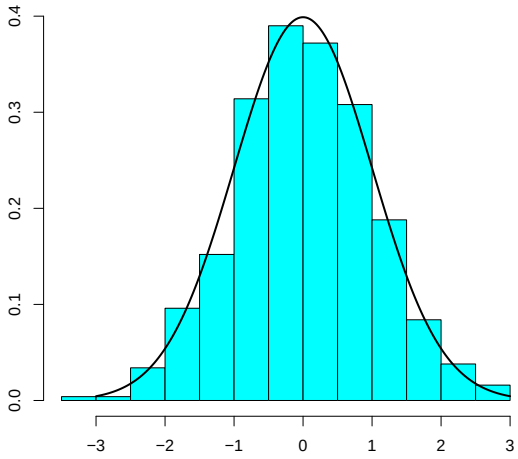
$$X_i = \sqrt{-2\mathsf{log}(U_{i1})}\ \mathsf{Cos}(2\pi U_{i2})$$
$$Y_i = \sqrt{-2\mathsf{log}(U_{i1})}\ \mathsf{Sen}(2\pi U_{i2})$$

Una implementación en R es como sigue:

```
# Metodo de Box y Muller
set.seed(73744)
n  <- 1000
u1 <- runif(n)
u2 <- runif(n)
x  <- sqrt(-2*log(u1))*cos(2*pi*u2)
y  <- sqrt(-2*log(u1))*sin(2*pi*u2)
r  <- c(-4,4)
par(mfrow=c(2,2),mar=c(3,3,3,3))
hist(x, cex.axis=.8, cex.lab=.8,col="cyan",cex.main=.9, xlim=r, mgp=c(1.5,.5,0))
hist(y, cex.axis=.8, cex.lab=.8,col="cyan",cex.main=.9, xlim=r, mgp=c(1.5,.5,0))
plot(x,y, pch=20, cex.axis=.8,cex.lab=.8, xlim=r, ylim=r, mgp=c(1.5,.5,0))
plot(rnorm(n),rnorm(n), pch=20,cex.axis=.8, cex.lab=.8, xlim=r,ylim=r, mgp=c(1.5,.5,0))
```

```
# Ahora usando el metodo de la inversa
n  <- 1000
u  <- runif(n)
z  <- qnorm(u)
hist(z,prob=T,col="cyan",ylim=c(0,.41),main="",xlab="",ylab="")
zz <- seq(-3,3,length=200)
lines(zz,dnorm(zz),lwd=2)
```





Aproximación Estocástica: Algoritmo Robbins-Monro. En muchas aplicaciones hay un interés central en encontrar el umbral (o nivel) de una variable, x, que causa un cierto efecto. Por ejemplo, en el área de Confiabilidad es de interés saber el nivel de stress que hace que cierta componente falle con una probabilidad de, digamos, 95%.

El problema se puede formular como sigue: Sea $y(x)$ una variable binaria (ocurre o no ocurre una falla) tal que $P(y(x) = 1) = M(x)$. El objetivo es encontrar el valor de x, llamémosle θ, tal que $M(\theta) = \alpha$ (por ejemplo, α puede ser 0.95). Ese valor de θ es lo que diríamos que es el "umbral" de x.

Usualmente $M(x)$ es una función de distribución obtenida bajo el supuesto de que x es la variable latente que subyace a la variable binaria y; en este caso (en el que $M(x)$ es una función de distribución), lo que se andaría buscando es el cuantil α de la variable x. Una cosa importante es que la función M **puede ser desconocida** (aunque suponemos que si podemos observar y para diferentes valores de x).

El algoritmo Robbins-Monro está diseñado para encontrar θ que sea solución de la ecuación $M(x) = \alpha$. Produce una secuencia $x_1, x_2, x_3, \cdots$, con la propiedad de que converge a θ. El algoritmo es como sigue:

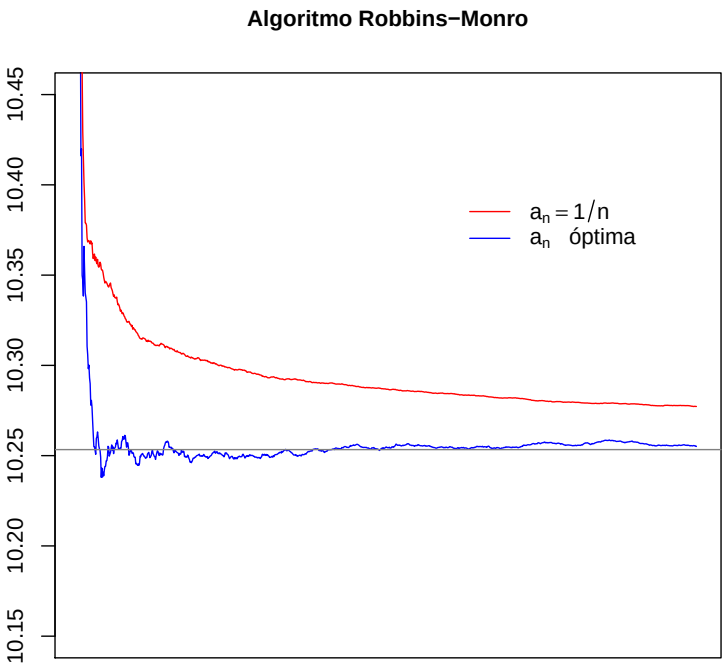
- Escoger x_1, un valor inicial
- Hacer $x_{n+1} = x_n - a_n[M(x_n) - \alpha], \qquad n = 1, 2, \cdots$

donde las constantes a_n son tales que

$$\sum a_n = \infty \quad \text{y} \quad \sum a_n^2 < \infty$$

(por ejemplo, $a_n = 1/n$).

Ejemplificamos el procedimiento, suponiendo que $M(x)$ es la función de distribución de una Normal con media $\mu = 10$ y desviación estándar $\sigma = 1$. Suponemos además $\alpha = 0.6$ (en otras palabras, lo que buscamos es el cuantil del 60% de una Normal) (por supuesto, el algoritmo no debería ser usado para este problema pues la respuesta es directa: $\theta =$ `qnorm(0.6,10,1)`!). Mostramos el comportamiento del algoritmo en la siguiente gráfica:



```
set.seed(919)
mu   <- 10
sig  <- 1
alf  <- .6
tet  <- qnorm(alf,mean=mu,sd=sig)
M    <- 100000
x    <- rep(0,M)
xo   <- rep(0,M)
x[1] <- rnorm(1,mean=mu,sd=sig)
xo[1]<- x[1]
Mp   <- dnorm(tet,mean=mu,sd=sig)
for(n in 2:M){
 p    <- pnorm(x[n-1],mean=mu,sd=sig)
 po   <- pnorm(xo[n-1],mean=mu,sd=sig)
 y    <- rbinom(1,size=1, prob=p)
 yo   <- rbinom(1,size=1, prob=po)
 x[n] <- x[n-1] - (y-alf)/(n-1)
 xo[n]<- xo[n-1] - (yo-alf)/((n-1)*Mp) }

zz <- c(1,seq(100,M,by=100))    # (solo una muestra para grafica)
plot((1:n)[zz],xo[zz],type="l",ylim=c(10.15,10.45),xlab="",cex.main=1,
 ylab="",xaxt="n",mgp=c(2,1,0),col="blue",main="Algoritmo Robbins-Monro")
lines((1:n)[zz],x[zz],col="red")
abline(h=tet,col=gray(.5))
legend(60000,10.4,lty=1, legend=c(expression(a[n] == 1/n),
 expression(paste(a[n],"   ptima"))),col=c("red","blue"),bty="n")
```

La línea en rojo muestra el resultado del algoritmo usando $a_n = 1/n$. Hay resultados teóricos que nos dicen que las a_n's óptimas son de la forma $a_n = 1/(nM'(\theta))$ y como para este problema sencillo podemos conocer ese programa óptimo, pues lo mostramos en azul. En la práctica esto no se conoce; más aún, note que aquí usamos $\alpha = 0.6$ lo cual no es un cuantil muy grande, sin embargo, en Confiabilidad, los cuantiles de interés son precisamente los grandes y se sabe que el Robbins-Monro no tiene buenas propiedades de velocidad de convergencia para cuantiles extremos.

(Un artículo que modifica el Robbins-Monro para cuantiles extremos es Joseph, V.R. (2004) *Biometrika*, Vol.91, No.2, pp 461-470).

Valores Propios: Método de Potencias. Supongamos que A es una matriz simétrica $n \times n$ y que tiene un valor propio dominante; más aún, por simplicidad en la discusión, supongamos además que es positiva definida; esto es, suponemos que sus valores propios se pueden ordenar como

$$\lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_1 > 0$$

En esta sección comentaremos sobre el Método de Potencias para obtener el máximo valor propio y un correspondiente vector propio.

Como A es positiva definida, los vectores propios $v_1, v_2, \cdots, v_n$ correspondientes a las λ_i's forman una base de $\mathbb{R}^n$, así, si x_0 es un vector cualquiera, entonces

$$x_0 = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n$$

multipicando repetidas veces ambos lados de esta expresión por A, puede verse que

$$\frac{1}{\lambda_1^k} A^k x_0 = \alpha_1 v_1 + \sum_{j=2}^{n} \alpha_j \left(\frac{\lambda_j}{\lambda_1}\right)^k v_j$$

ahora, como $(\lambda_j/\lambda_1) < 1$, entonces, para k grande

$$\frac{1}{\lambda_1^k} A^k x_0 \approx \alpha_1 v_1$$

en otras palabras, para k grande,

$$A^k \left(\frac{x_0}{\alpha_1 \lambda_1^k} \right) \approx v_1 = \text{un vector propio correspondiente a } \lambda_1$$

o sea que $A^k x_0^{(k)}$ es aproximadamente un vector propio asociado a λ_1. Ahora bien, tal como tenemos escrito a $x_0^{(k)}$, no se puede calcular como $\left(\frac{x_0}{\alpha_1 \lambda_1^k}\right)$, pues depende de λ_1, sin embargo, en la literatura sobre el tema puede verse que, en cada iteración, una normalización adecuada para x_0 (que no dependa de λ_1), puede obtenerse como en el siguiente algoritmo:

Método de potencias

1. Empezar en $x_0 \in \mathbb{R}^n$
2. Actualizar $x_k = A x_{k-1}$
3. Normalizar $x_k = x_k / ||x_k||_\infty$
4. Iterar 2 y 3

La sucesión $x_0, x_1, x_2, \cdots$ converge a un vector propio, digamos v, asociado al valor propio más grande de A. Una vez que tenemos v, entonces,

$$Av = \lambda_1 v \quad \Rightarrow \quad v'Av = \lambda_1 v'v$$

y, por lo tanto $\lambda_1 = (v'Av)/(v'v)$. El siguiente código implementa este método para calcular el máximo valor propio de A, así como un vector propio asociado.

```
#################
# Eigenvalor dominante
n <- 10
A <- matrix( runif(n^2), n,n)
A <- t(A)%*%A
x <- runif(n)
for(i in 1:100){
 x <- as.vector(A%*%x)
 x <- x/max(abs(x))}
lambda <- sum(x*A%*%x)/sum(x*x)
lambda
x
out <- eigen(A)     # comparar con eigen()
out$values[1]
(out$vectors[,1])/(max(abs(out$vectors[,1])))
```

Prácticas Sesión 5

1. **Raíz Cuadrada.** Escriba un programa en R para calcular la raíz cuadrada de un número real, c. Use el siguiente algoritmo

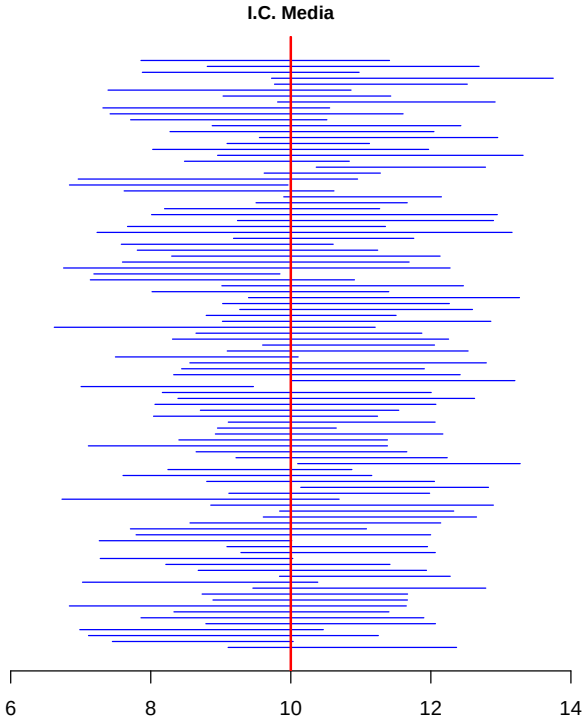
   - Empieze en un punto arbitrario, r.
   - Reemplaze r por el promedio entre r y c/r.
   - Repita el paso anterior hasta que se satisfaga algún criterio de convergencia.

   Este algoritmo es el método de Newton para encontrar una raíz de la función $g(x) = x^2 - c$. ¿Porqué?

2. **Intervalos de Confianza.** Suponga que $x_1, \cdots, x_n$, forman una muestra aleatoria de una distribución $N(\mu, \sigma^2)$. Un intervalo del $(1-\alpha) \times 100\%$ de confianza para μ es

$$\left(\bar{x} - t_{n-1,\alpha/2} \frac{s}{\sqrt{n}} \, , \; \bar{x} + t_{n-1,\alpha/2} \frac{s}{\sqrt{n}} \right)$$

   (a) Queremos ilustrar la interpretación frecuentista de estos intervalos usando un pequeño ejercicio de simulación. Simule 100 muestras aleatorias de tamaño $n = 11$, tomadas de una $N(\mu = 10, \sigma^2 = 9)$. Para cada una de estas muestras aleatorias, calcule los intervalos del 90% de confianza para μ. Muestre los intervalos en una gráfica semejante a la siguiente. ¿Qué interpretación se sigue de esta gráfica?.

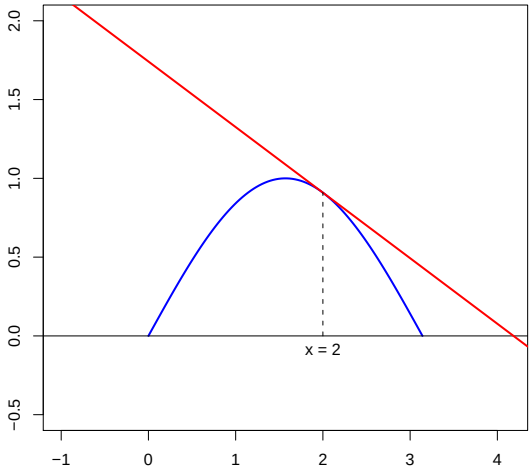


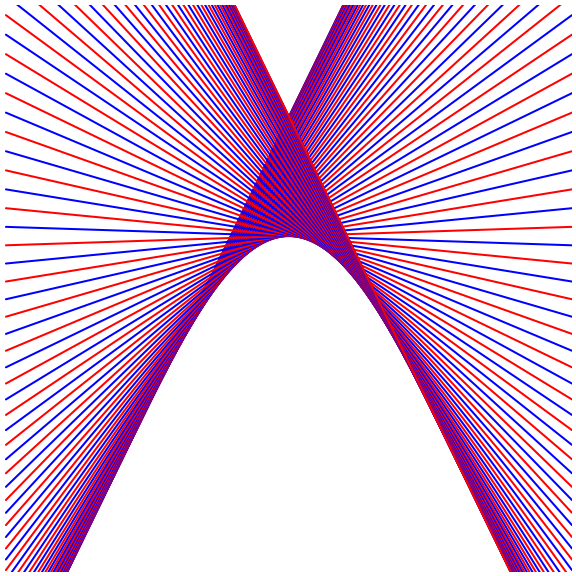
   (b) Construya una gráfica como la anterior, pero mostrando intervalos de confianza para la varianza σ^2. ¿Qué comportamientos observa diferentes con respecto a los intervalos para la media?.

3. **Gráficas.** Enseguida, a la izquierda, tenemos una gráfica de la función $y = \text{Sen}(x)$ y la recta tangente a la curva en el punto $x = 2$. Sabemos que la ecuación de la recta que pasa por el punto (x_0, y_0) y tiene pendiente m está dada por

$$\frac{y - y_0}{x - x_0} = m \quad \Rightarrow \quad \frac{y - \text{Sen}(x_0)}{x - x_0} = \text{Cos}(x_0) \quad \Rightarrow \quad y = \text{Sen}(x_0) - x_0\text{Cos}(x_0) + x\text{Cos}(x_0)$$

de aquí que la recta tangente puede graficarse usando `abline(a=sin(x0)-x0*cos(x0), b=cos(x0))`. La línea punteada puede lograrse con `segments(x0,0,x0,sin(x0),lty=2)` y la información $x = 2$ con `text(x0,0,"x = 2",pos=1)`. Ahora imagine que grafica muchas tangentes $\cdots$ esa es la gráfica de la derecha. **Reproduzca ambas gráficas**. Note que, para la segunda gráfica, tendrá que suprimir el graficado de ejes, no poner el marco, usar parámetros adecuados en `par( mar=c( ) )` para que la gráfica ocupe más espacio y alternar los colores azul y rojo para las tangentes (por supuesto, tendrá que usar `for(i in 1:M){ abline( ) }`, para obtener todas las tangentes).





4. **Análisis Exploratorio de Datos.** Los datos del archivo `belsley.csv` contienen información de 50 países (información de la década de los 60's) sobre 5 variables. Parte del archivo es como sigue:

```
        PAIS      TA    POB15   POB75     IPD      CPIPD
1  Australia   11.43   29.35   2.87    2329.68    2.87
2    Austria   12.07   23.32   4.41    1507.99    3.93
...
49     Libia    8.89   43.69   2.07     123.58   16.71
50   Malasia    4.71   47.20   0.66     243.69    5.08
```

donde

`TA`	=	Tasa de ahorro promedio per cápita
`POB15`	=	Proporción de la población de menos de 15 años
`POB75`	=	Proporción de la población de más de 75 años
`IPD`	=	Ingreso personal disponible
`CPIPD`	=	Crecimiento porcentual del `IPD` durante la década de los 60's

De acuerdo a cierto modelo económico, se tendrá una mayor tasa de ahorro en individuos de edad madura, mientras que las personas jóvenes tenderán a tener tasas menores (probablemente anticipando mayores ingresos en etapas posteriores de sus vidas). Por otro lado, los individuos de edad avanzada tenderán a gastar los ahorros que acumularon durante su etapa de edad madura.

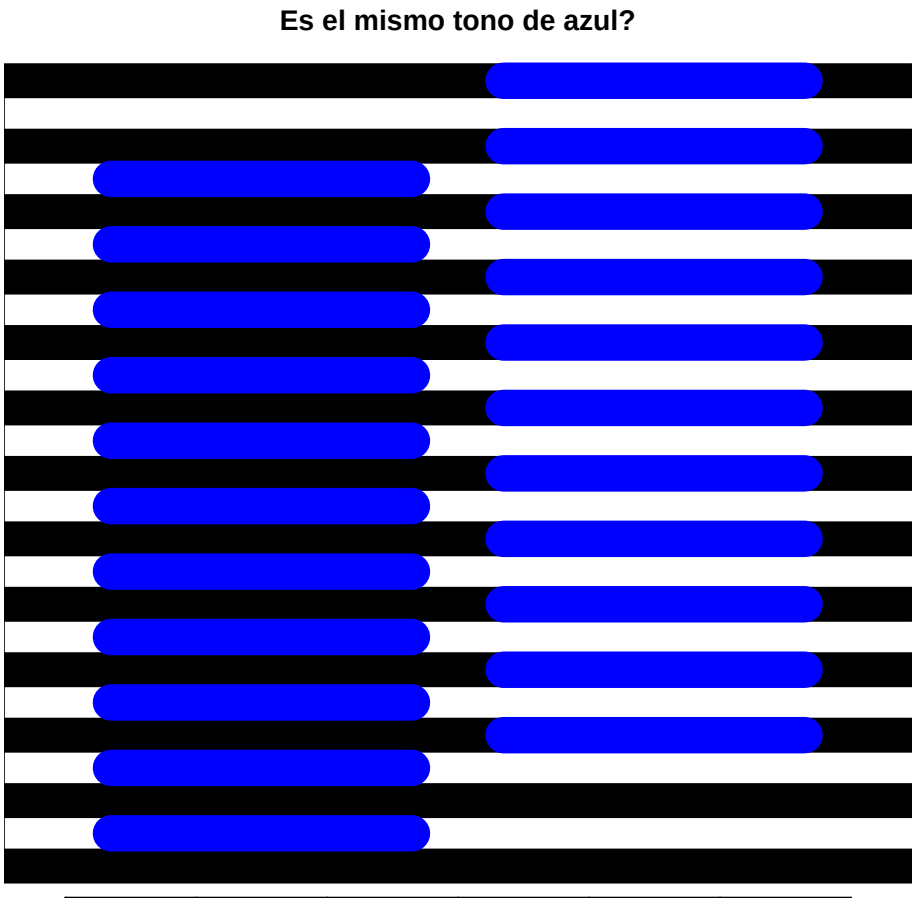
(a) Efectúe un análisis exploratorio de este conjunto de datos. ¿Hay evidencia de que el módelo en cuestión es válido?.

(b) Efectúe un agrupamiento de los datos en $K = 4$ grupos usando k-*means* (no usar funciones predefinidas de R como `kmeans()`). Comente sus resultados.

(c) Produzca una imágen bidimensional de estos datos usando Escalamiento Multidimensional (no usar funciones predefinidas de R como `cmdscale()`). Comente sus resultados.

5. **Sliders.**

(a) La gráfica

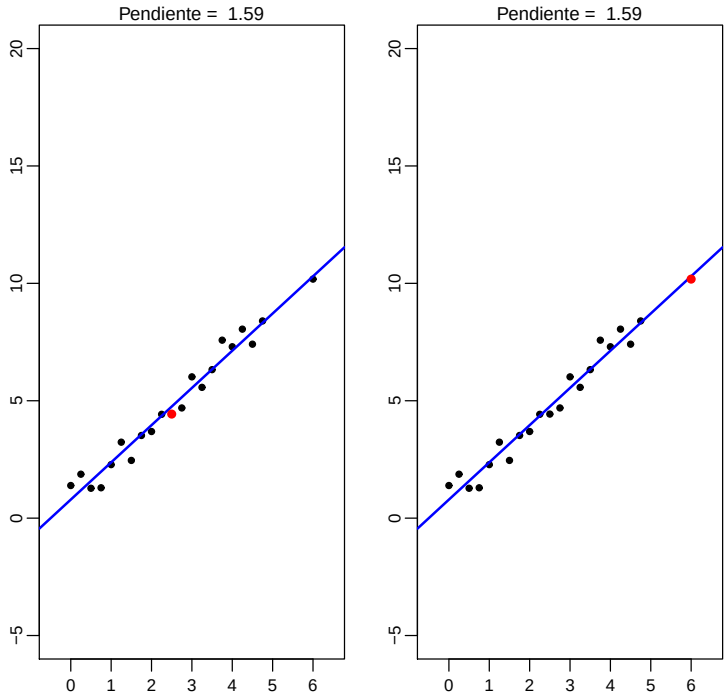


fue hecha con:

```
par(mar=c(0,0,2,0))
plot(0,0,type="n",xlim=c(-3,3),ylim=c(-3,3),asp=1,xlab="",
 ylab="",bty="n",main="Es el mismo tono de azul?")
abline(h=seq(-3,3,by=.5),lwd=23)
for(i in seq(-2.75,2.25,by=.5)){
 segments(-2.65,i,-0.35,i,col="blue",lwd=24)
 segments(0.35,i+.75,2.65,i+.75,col="blue",lwd=24)}
```

Construya un "slider", usando el paquete `tcltk`, para variar el color de las líneas horizontales negras, desde un nivel de gris, `gray(0)`, hasta `gray(1)`; esto es, desde completamente negro hasta completamente blanco, pasando por diferentes tonos de gris.

(b) La gráfica



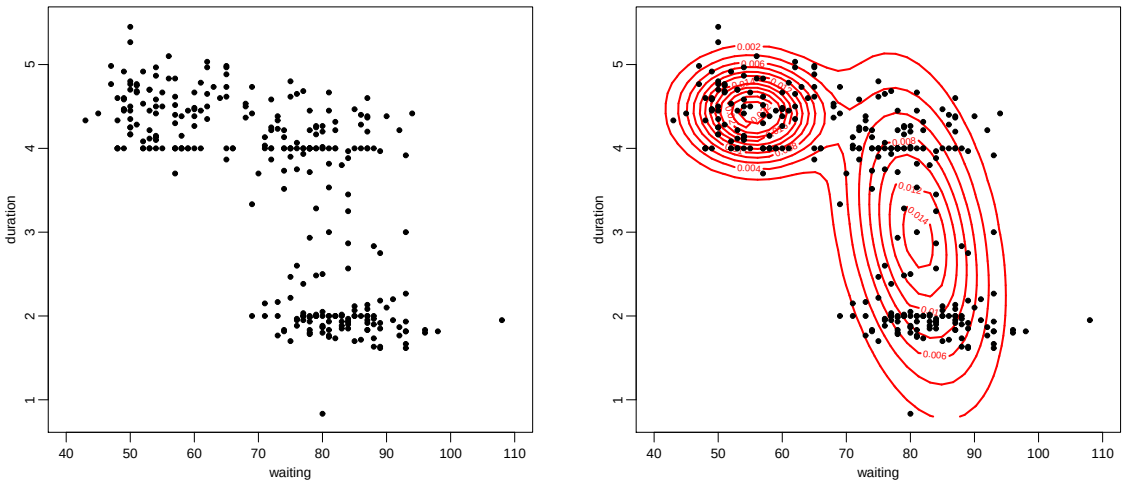
fue hecha con:

```
par(mfcol=c(1,2),mar=c(3,2,2,1))
algo <- 0
n    <- 21
x    <- seq(0,5,length=n)
x[n] <- 6
set.seed(7587)
y    <- 1+1.5*x+rnorm(n,mean=0,sd=.5)
y[11]<- y[11] + algo
plot(x,y,pch=20,xlim=c(-.5,6.5),ylim=c(-5,20),mgp=c(1.5,.5,0),
 cex.lab=.8,cex.axis=.8,xlab="",ylab="")
out  <- lm(y~x)
abline(out,lwd=2,col="blue")
points(x[11],y[11],pch=16,col="red")
mtext(paste("Pendiente = ",round(out$coef[2],2)),cex=.9)
set.seed(7587)
y    <- 1+1.5*x+rnorm(n,mean=0,sd=.5)
y[n] <- y[n] + algo
plot(x,y,pch=20,xlim=c(-.5,6.5),ylim=c(-5,20),mgp=c(1.5,.5,0),
 cex.lab=.8,cex.axis=.8,xlab="",ylab="")
out  <- lm(y~x)
abline(out,lwd=2,col="blue")
points(x[n],y[n],pch=16,col="red")
mtext(paste("Pendiente = ",round(out$coef[2],2)),cex=.9)
```

Construya un "slider" para variar la perturbación (en `algo`), desde -10 hasta 10. Comente sobre el efecto de tal perturbación en el modelo ajustado.

6. **Algoritmo EM.** En la gráfica de la izquierda tenemos los datos del géiser "Old Faithful". Representan 299 registros de erupciones del géiser. Cada punto indica el tiempo que dura la erupción y el tiempo que se tuvo que esperar entre esa erupción y la pasada:

```
library(MASS)
attach(geyser)
plot(waiting, duration, pch=20, mgp=c(1.5,.5,0), cex.lab=.8,
     cex.axis=.8, xlim=c(40,110), ylim=c(.8,5.5))
```



El objetivo de este problema es producir la gráfica de la derecha. Son las curvas de nivel de una mezcla de dos distribuciones gaussianas bivariadas.

Podemos conceptualizar la estructura de los datos, de la siguiente forma

$$(x_1, z_1), (x_2, z_2), \cdots, (x_n, z_n)$$

donde las x_i's son gaussianas bivariadas y las z_i's son variables indicadoras (0 o 1) que nos dicen a que grupo pertenecen las correspondientes x_i's; así, por ejemplo, si $z_1 = 0$ entonces x_1 viene de la primera distribución normal bivariada y si $z_1 = 1$ entonces viene de la segunda.

Supongamos que las densidades gausianas bivariadas son, respectivamente

$$f(x \mid \mu_1, \Sigma_1) = \frac{1}{2\pi|\Sigma_1|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right)$$

y

$$f(x \mid \mu_2, \Sigma_2) = \frac{1}{2\pi|\Sigma_2|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\right)$$

Ahora, **si las z_i's fueran conocidas**, entonces ya sabríamos cual x viene de cual población y entonces los estimadores de máxima verosimilitud de los parámetros μ_1, μ_2, Σ_1 y Σ_2 serían

$$\widehat{\mu}_1 = \frac{1}{n-\sum_{j=1}^n z_j}\sum_{i=1}^n (1-z_i)x_i \qquad \widehat{\Sigma}_1 = \frac{1}{n-\sum_{j=1}^n z_j}\sum_{i=1}^n (1-z_i)(x_i-\widehat{\mu}_1)(x_i-\widehat{\mu}_1)^T$$
$$\widehat{\mu}_2 = \frac{1}{\sum_{j=1}^n z_j}\sum_{i=1}^n z_i x_i \qquad \widehat{\Sigma}_2 = \frac{1}{\sum_{j=1}^n z_j}\sum_{i=1}^n z_i(x_i-\widehat{\mu}_2)(x_i-\widehat{\mu}_2)^T$$

En R, por ejemplo, podemos calcular $\widehat{\mu}_1$ y $\widehat{\Sigma}_1$ como sigue (z sería un vector de tamaño n, definido de antemano) (en el caso de que las z_i's fuesen conocidas):

```
X  <- as.matrix(geyser)
n  <- dim(X)[1]
m1 <- colSums((1-z)*X)/(n-sum(z))   # Estimacion de mu_1
aa <- t(X)-m1
S1 <- aa%*%((1-z)*t(aa))/(n-sum(z)) # Estimacion de Sigma_1
```

Ahora bien, en la práctica, usualmente **no conocemos** las z_i's; esto es, no sabemos a cual gaussiana pertenecen las observaciones. Por ejemplo, en los datos de erupciones se aprecian 3 agrupamientos y no es claro si el agrupamiento central pertenece al grupo de la izquierda o al de abajo (incluso podría hasta ser un tercer grupo por si mismo).

El modelo de mezcla de gaussianas que queremos estimar es:

$$h(x|p,\mu_1,\mu_2,\Sigma_1,\Sigma_2) = (1-p)f(x|\mu_1,\Sigma_1) + pf(x|\mu_2,\Sigma_2)$$

donde p es la probabilidad de que una observación pertenezca al grupo 2 y $1-p$, la probabilidad de que sea del grupo 1. Para estimar los parámetros, $p,\mu_1,\mu_2,\Sigma_1,\Sigma_2$, de este modelo usaremos el **Algoritmo EM**, el cual es un procedimiento iterativo para encontrar estimadores de máxima verosimilitud (pensado en particular para el caso de observaciones con datos perdidos). Como vimos, en el caso de tener las z_i's, el problema de estimación es fácil, pero como no las tenemos entonces el algoritmo las estima en un proceso iterativo:

param iniciales $\rightarrow$ z_i's $\rightarrow$ actualiza param $\rightarrow$ actualiza z_i's $\rightarrow$ actualiza param $\rightarrow$ actualiza z_i's $\cdots$

i. Arrancamos en ciertos valores iniciales $p^1,\mu_1^1,\mu_2^1,\Sigma_1^1,\Sigma_2^1$.

ii. Estimamos las z_i mediante la probabilidad de que la observación i provenga del grupo 2:

$$\widehat{z}_i = \frac{p^1 f(x_i \mid \mu_2^1,\Sigma_2^1)}{(1-p^1)f(x_i \mid \mu_1^1,\Sigma_1^1) + p^1 f(x_i \mid \mu_2^1,\Sigma_2^1)}, \qquad i = 1,2,\cdots,n$$

iii. Ya que tenemos estas $\widehat{z}_i$'s, actualizamos los parámetros para obtener $p^2,\mu_1^2,\mu_2^2,\Sigma_1^2,\Sigma_2^2$, mediante

$$p^2 = \frac{1}{n}\sum_{i=1}^{n}\widehat{z}_i$$

y $\mu_1^2,\mu_2^2,\Sigma_1^2,\Sigma_2^2$ se calculan con expresiones similares a las de la hoja anterior.

iv. Iteramos los pasos ii y iii hasta convergencia.

(a) Usando el Algoritmo EM encuentre los estimadores de máxima verosimilitud para los parámetros del modelo de mezclas con dos gaussianas, $p,\mu_1,\mu_2,\Sigma_1,\Sigma_2$.

(b) Escriba una función, con argumentos x,μ,Σ que calcule la densidad normal bivariada, digamos

f <- function(x,mu,Sig){ return($\frac{1}{2\pi|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$) }

y haga la gráfica de contornos del modelo mezcla. Por ejemplo, pueden usar algo como:

```
m  <- 40
xx <- seq(40,110,length=m)
yy <- seq(.8,5.5,length=m)
zz <- matrix(0,m,m)
for(i in 1:m){
 for(j in 1:m){
  x <- c(xx[i],yy[j])
  zz[i,j] <- (1-p)*f(x,m1,S1)+p*f(x,m2,S2)}}
# p,m1,m2,S1,S2 son las estimaciones obtenidas en el inciso anterior
contour(xx,yy,zz,xlab="waiting",ylab="duration",mgp=c(1.5,.5,0),
 cex.lab=.8,cex.axis=.8,xlim=c(40,110),ylim=c(.8,5.5),col="red",lwd=2)
points(X[,1],X[,2],pch=20)
```

7. Escriba un programa para generar números aleatorios usando el generador congruencial:

$$x_i = 35\, x_{i-1} \mod 2^{15}$$

Muestre que este generador no es bueno, observando que los puntos caen en hiperplanos de la forma

$$x_{i+3} - 9x_{i+2} + 27x_{i+1} - 27x_i = j$$

donde j es entero.

8. Cierta enzima rompe cadenas de DNA en dos partes (donde el punto de rompimiento se localiza aleatoriamente a lo largo de la cadena). Suponga que 24 cadenas idénticas son partidas cada una en dos partes. ¿Cuál es la longitud promedio de la cadena más corta? (esto es, si este experimento fuera replicado un gran número de veces) (usar simulación).

9. Un biólogo usa una enzima para partir una cadena de DNA en 10 partes (los puntos de rompimiento ocurren al azar). La molécula original tenía una longitud de 10,000 parejas base. Después de examinar las diferentes piezas, el biólogo encontró que la más pequeña tenía una longitud de sólo 10 parejas base. ¿Qué tan probable (o improbable) es que la cadena más pequeña tenga esta longitud? (esa longitud o algo incluso más pequeño) (usar simulación). ¿Hay razón para que el biólogo dude que el rompimiento de la cadena ocurra en puntos al azar?

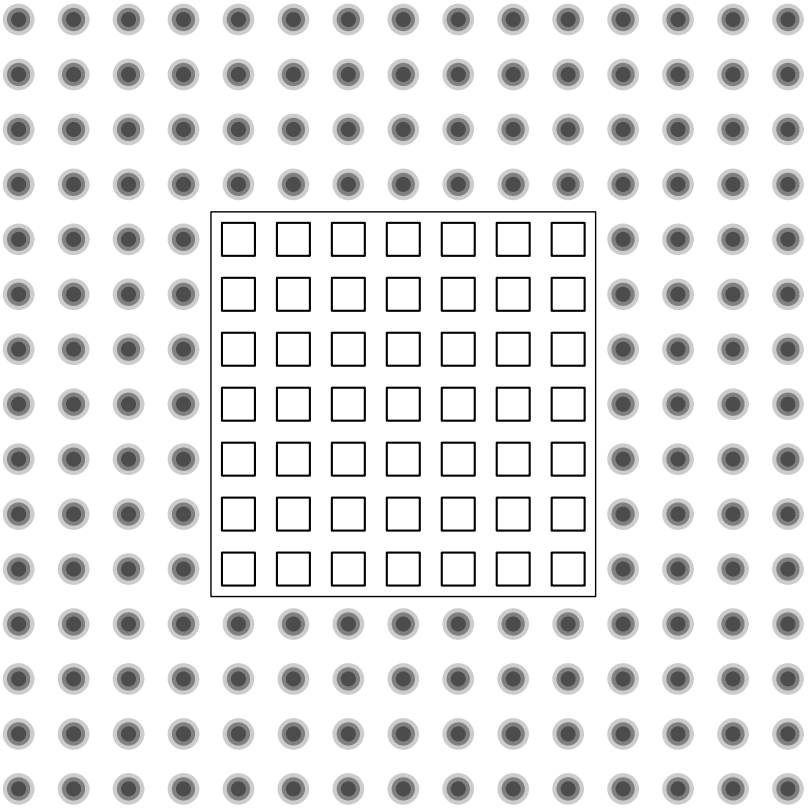
10. Considere las variables aleatorias independientes $X_1, X_2, X_3, \cdots$ tales que

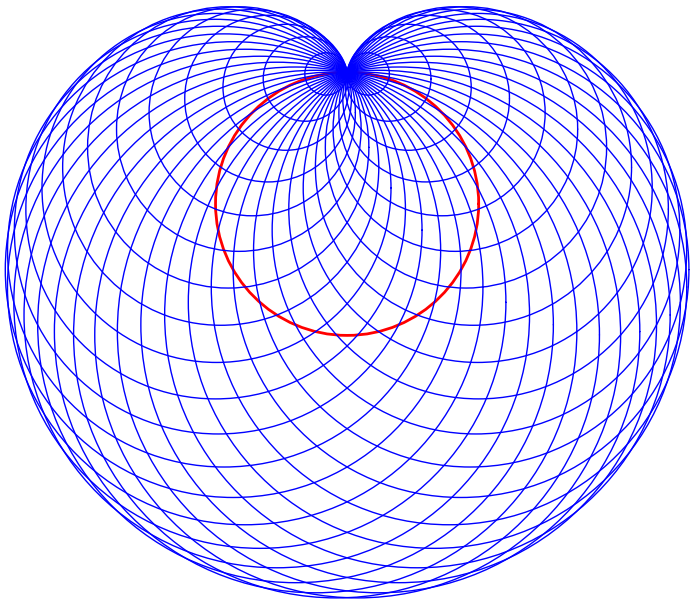
$$X_i = \begin{cases} -1 & \text{con probabilidad } 1/2 \\ 1 & \text{con probabilidad } 1/2 \end{cases}$$

Usando simulación, examine la convergencia de:

$$\sum_{n=1}^{\infty} \frac{X_n}{n}$$

Apéndice Sesión 5





Apéndice Sesión 5

```
#########################################################################
# Figura 1: Ilusion Optica
# ( tomado de http://www.michaelbach.de/ot/ )

par(mar=c(0,0,0,0))
n <- 15
plot(0,0,type="n",xlim=c(1,n),ylim=c(1,n),xaxt="n",yaxt="n",asp=1,
 xlab="",ylab="",bty="n",main="")
a  <- 1:n
b  <- expand.grid(a,a)
m  <- dim(b)[1]
symbols(b[,1],b[,2],circles=rep(.25,m),add=TRUE,inches=FALSE,
 bg=gray(.5),fg=gray(.8),lwd=3)
symbols(b[,1],b[,2],circles=rep(.15,m),add=TRUE,inches=FALSE,
 bg=gray(.3),fg=gray(.5))

a  <- 5:11
b  <- expand.grid(a,a)
m  <- dim(b)[1]
symbols(b[,1],b[,2],squares=rep(.6,m),add=TRUE,inches=FALSE,
 lwd=1.5,bg="white")
symbols(8,8,squares=7,add=TRUE,inches=FALSE)
#########################################################################

#########################################################################
# Figura 2: Generacion de una cardioide.
# Construccion tomada de: http://mathworld.wolfram.com/Cardioid.html
library(grid)
ori <- c(7.5,8.5)
r   <- 2.5
grid.circle(x=ori[1], y=ori[2], r=r, default.units="cm",
     gp=gpar(col="red", lwd=2) )
a   <- ori+c(0,r)
tet <- seq(pi/2,5*pi/2,length=40)
x   <- r*cos(tet)+ori[1]
y   <- r*sin(tet)+ori[2]
rad <- sqrt( (x-a[1])^2+(y-a[2])^2 )
grid.circle(x=x, y=y, r=rad, default.units="cm", gp=gpar(col="blue") )
#########################################################################
```