

Introducción al lenguaje R y su aplicación al análisis de datos

Curso introductorio

Débora Chan
Andrea Rey
Cristina Badano

Buenos Aires, Argentina, 2017

Índice

1	Introducción	1
2	Entorno R-Studio	3
3	Objetos y operaciones básicas	5
3.1	Operaciones básicas de números	5
3.2	Vectores	6
3.3	Matrices	10
3.4	Data-frame	12
3.5	Listas	13
3.6	Generación de números al azar	14
3.7	Ejercicios de revisión	14
4	Introducción al análisis de datos	17
4.1	Directorio de trabajo	17
4.2	Bases de datos	17
4.3	Cálculo de resúmenes estadísticos	21
4.4	Aplicación: Test de Chi-cuadrado	27
4.5	Aplicación: Paradoja de Simpson	29
4.6	Ejercicios de revisión	30
5	Gráficos	33
5.1	Gráfico de funciones reales	34
5.2	Gráficos en tres dimensiones o de campos escalares	39
5.3	Gráficos con movimientos	42
5.4	Gráficos estadísticos	42
5.5	Ejercicios de revisión	75
6	Funciones	77
6.1	Simulación	80
6.2	Haz de rectas	82
6.3	Integración	84
6.4	Distribuciones triangulares	88
6.5	Generación de variables aleatorias	89
6.6	Ejercicio de aplicación	91
6.7	Ejercicios de revisión	94
7	Aplicaciones didácticas	95
7.1	Estimación del número π	95
7.2	Estimadores de probabilidades o áreas	96
7.3	Iniciación en la distribución binomial	98
7.4	Estimación puntual	100
7.5	Nivel de significación de un test	109

7.6	Nivel asintótico	111
7.7	Transformaciones de datos por filas	112
7.8	Transformaciones de datos por columnas	117
7.9	Ejercicios de revisión	121
8	Apéndice	123

Capítulo 1

Introducción

R es un entorno especialmente diseñado para el **tratamiento de datos, cálculo y desarrollo gráfico**. Facilita el trabajo con vectores y matrices y ofrece diversas herramientas para el análisis de datos. Se trata de un lenguaje creado específicamente para la visualización y exploración de datos así como para su uso en modelización y programación estadística.

El lenguaje de programación R forma parte del **proyecto GNU** y puede verse como una implementación alternativa del lenguaje S, que a su vez tiene muchos elementos del lenguaje C desarrollado en AT&T Bell Laboratories.

Se presenta como software libre, donde el término software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.

Página: <http://cran.r-project.org/>

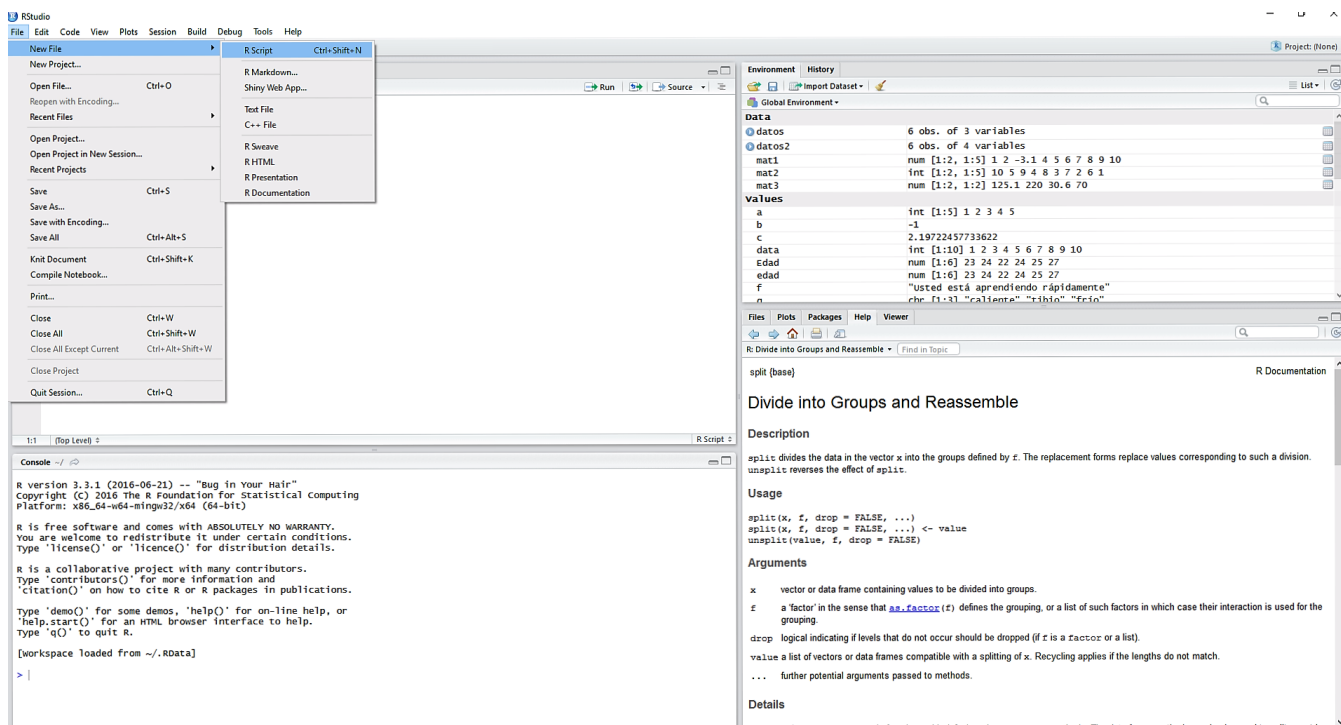
Documentación: el sitio de CRAN <http://cran.r-project.org/>

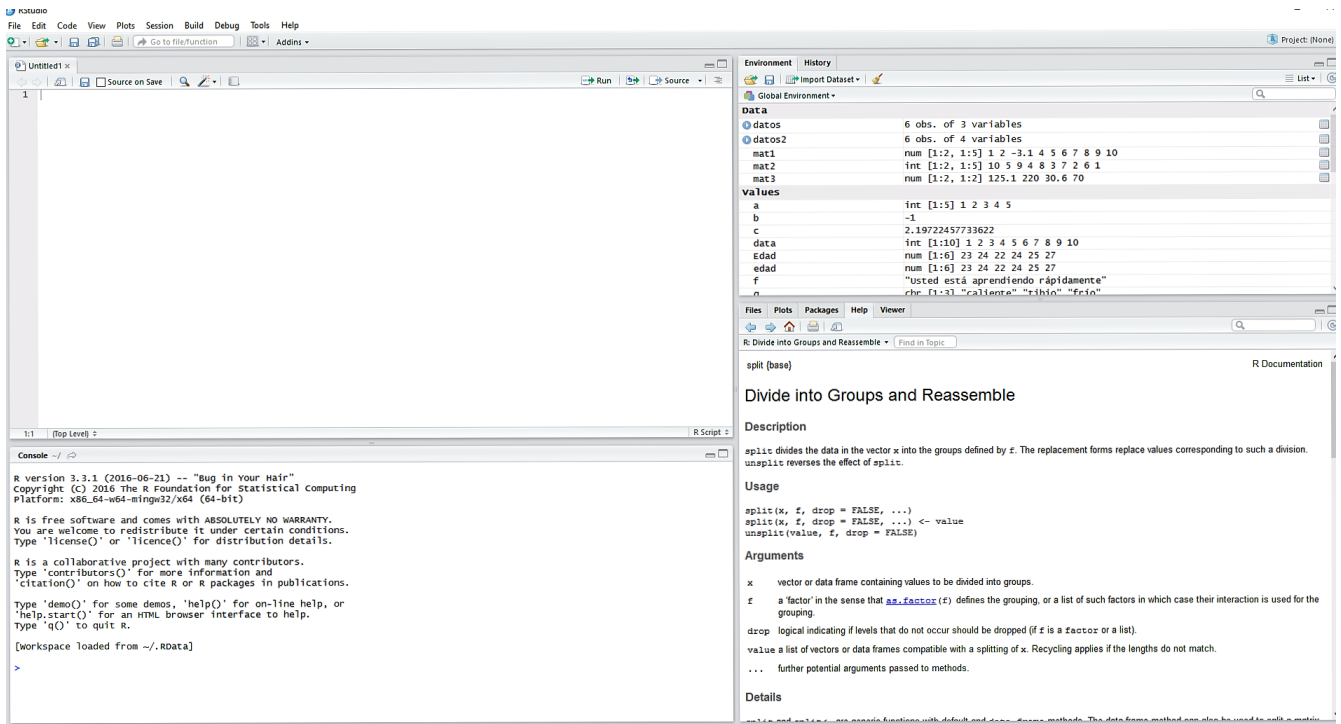
La instalación de R se realiza a través de la **CRAN** (Comprehensive R Archive Network). Además, R es un entorno en el que se han ido implementando diversas técnicas estadísticas. Algunas de ellas se encuentran en la base de R pero otras muchas están disponibles como paquetes (packages). Estos paquetes están disponibles en la web <http://cran.au.r-project.org/>.

Capítulo 2

Entorno R-Studio

R Studio es un entorno de trabajo que requiere la instalación previa de R. facilita la visualización de la tarea, dividiendo la pantalla en cuatro regiones.





En la parte izquierda superior se escriben los comandos del *script* (programa) que puede ser guardado para reproducirlo en otro momento.

En la parte derecha superior se guardan los objetos que se van definiendo y se encuentran disponibles en ese momento en la memoria.

En la región izquierda inferior se ve la ejecución de los comandos.

En la región derecha inferior se pueden apreciar: la ayuda, los gráficos, los paquetes instalados y los archivos usados recientemente.


Capítulo 3

Objetos y operaciones básicas

En el lenguaje R se puede trabajar con varias clases de objetos; algunos de ellos son estándar en cualquier lenguaje de programación y otros son objetos específicos de R, objetos pensados para ser manejados con propósitos estadísticos.

La mayoría de las expresiones en R se escriben directamente a continuación del *prompt* en la Consola de R.

3.1 Operaciones básicas de números

Para ejecutar cada línea de comando se mantiene el cursor en la línea y se clickea  en la parte superior derecha de la consola.

Cuando una línea se inicia con un numeral, esa línea no se ejecuta. Se puede usar este recurso para los títulos o ir anotaciones en los pasos del procedimiento. Todos los caracteres correspondientes a estos ?carteles? aparecen en verde.

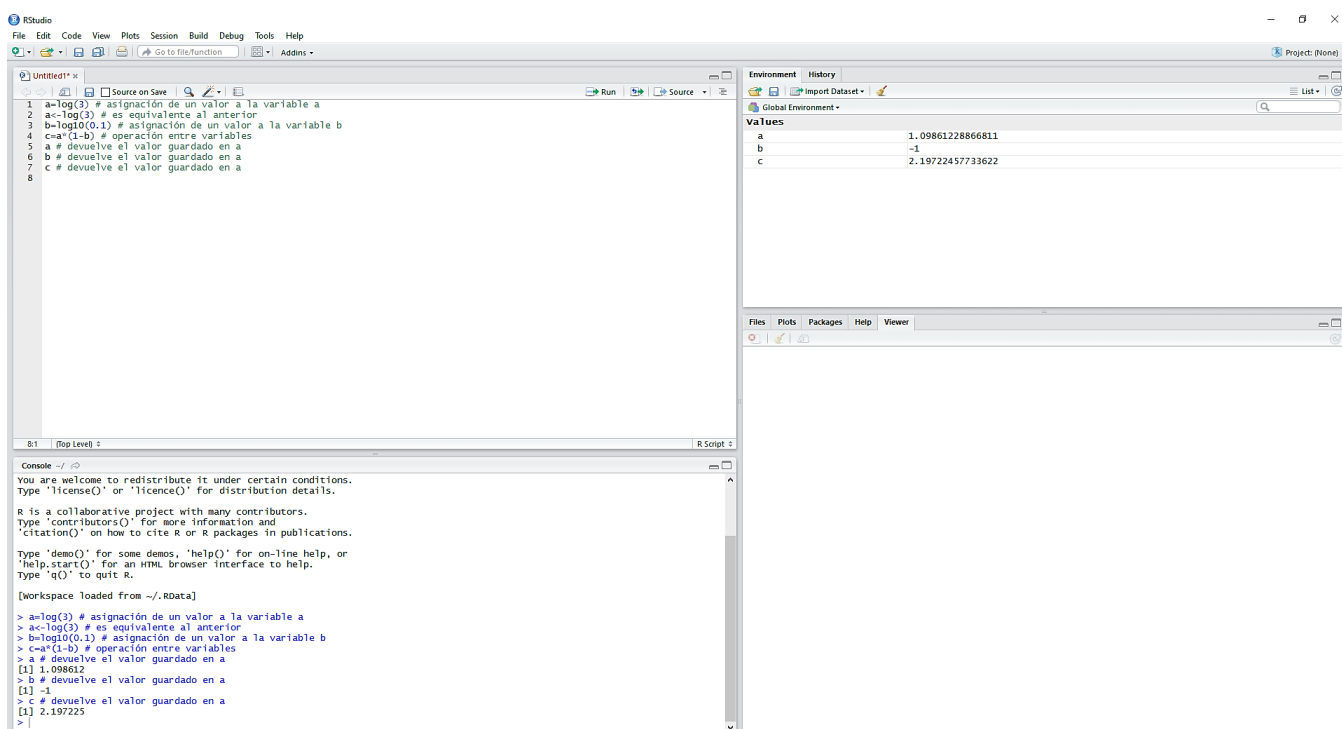
Operaciones numéricas y funciones trascendentes

```
2+6                # suma
3*4                # producto
-4/(5-3)           # cociente
4 ^ 0.2            # potencia
4 ** 0.2           # otra forma de potencia
sqrt(9)            # raíz cuadrada
cos(pi)            # funciones trigonométricas
exp(-2)            # función exponencial
factorial(5)       # devuelve el factorial de un número
```

Almacenamiento de números en variables y uso de datos almacenados

```
a=log(3)           # asignación de un valor a la variable a
a<-log(3)          # es equivalente al anterior
log(3)->a          # es equivalente al anterior
b=log10(0.1)       # asignación de un valor a la variable b
c=a*(1-b)          # operación entre variables
a                 # devuelve el valor guardado en a
b                 # devuelve el valor guardado en b
c                 # devuelve el valor guardado en c
```

Observar que estos datos disponibles en el *Environment* (región superior derecha).



3.2 Vectores

3.2.1 El operador :

Analizar qué ocurre al correr los siguientes comandos y probar con otros valores numéricos. Se recomienda hacer anotaciones de lo observado.

```

1:5      # números del 1 al 5 orden creciente
2:100    # números del 2 al 100 orden creciente
15:10    # números del 15 al 10 orden decreciente

```

¿Qué ocurre si la secuencia es muy larga?

¿Qué ocurre si al almacenar alguno de estos vectores se usa una variable que ya estaba utilizada, por ejemplo a?

```

a=1:5    # reemplaza lo guardado en la variable a
a        # devuelve el valor guardado en a

```

3.2.2 Funciones de concatenación

Se pueden distinguir varias funciones de este tipo.

La función **c** concatena (enlaza, une) objetos en un vector.

La función **seq** produce secuencias equiespaciadas.

La función **rep** reproduce un valor o secuencia de valores la cantidad de veces que se le indica.

```
x <- c(1,2,3,4,5,6,7,8) # devuelve los enteros del 1 a 8
x
x <- 1:8                # es equivalente al anterior
x
assign("x", 1:8)        # otra manera de asignar valores a un vector
x
x <- seq(1,8)            # es equivalente al anterior
x
x1=8:1                  # devuelve los números enteros del 8 al 1
x1
y=seq(1,4,0.2)          # devuelve los números de 1 a 4 espaciados en 0.2
y
z=rep(1,5)              # repite el 1 cinco veces
z
w=rep(1:3,4)            # repite del 1 al 3 cuatro veces
w
q=rep(c(1,3,7),2)       # repite la concatenación dos veces
q
```

3.2.3 Vectores lógicos

Se permiten entradas del tipo TRUE (verdadero) y FALSE (falso).

```
m <- c(T,F,F,T)        # concatena lo pedido
m
n <- rep(c(T,F),3)      # repite la concatenación tres veces
n
l=w>2                  # asigna una proposición a la variable l
w
l                       # asigna valor de verdad de la proposición
p<-c(T,F,T,F)
p
m & p                  # devuelve la conjunción entre componentes de vectores lógicos de
                        # igual longitud
m|p                    # devuelve la disyunción entre componentes de vectores lógicos de
                        # igual longitud
!m                     # devuelve la negación de cada componente
```

3.2.4 Acceso a las componentes de un vector

Esta acción se puede llevar a cabo de diversas maneras.

Se puede acceder por la posición.

```
w
w[5]                   # devuelve la quinta componente almacenada en el vector w
w[2:4]                 # devuelve las componentes de 2 a 4 del vector w
w[c(1,4,6)]            # devuelve las componentes 1, 4 y 6 del vector w
w[-c(1,4,6)]           # devuelve el vector w sin las componentes 1, 4 y 6
rep(c(T,T,F,F),3)
w[rep(c(T,T,F,F),3)]   # devuelve las componentes de w que corresponden con TRUE
```

Se puede acceder mediante la función **which**.

```

u=c(3,8,2,7,3,2,1)
u==3                # operación lógica que busca las posiciones de u que guardan
                    # el 3
u[u==3]<-4           # en esas posiciones asigna un 4
u
which(u>=4)          # devuelve las posiciones de u que tienen números mayores o
                    # iguales a 4
u[which(u>=4)]<-0    # asigna 0 a esas posiciones
u

```

Se puede acceder a través de nombres asignados.

```

h<-c(2,7,4)
names(h)              # consulta los nombres de las
                    # componentes del vector h

names(h)<-c("Alicia", "Pedro", "Lucas") # asigna nombres a las componentes
                    # de h
h                     # verifica la asignación
h["Alicia"]           # devuelve la componente según el
                    # nombre

```

3.2.5 Vectores de caracteres

Uniendo cadenas de caracteres se obtiene un vector de caracteres.

```

f<-"Usted está aprendiendo rápidamente" # asigna un vector de caracteres
                    # de longitud 1
f
g<-c("caliente", "tibio", "frío")        # asigna un vector de caracteres
                    # de longitud 3
g
h <- paste(c("a","b"), 2:5, sep="")       # genera pares ordenados
h
t=paste(c("a","b"), 2:5, sep=",")         # genera pares ordenados separados
                    # por comas
t

```

3.2.6 Longitud de un vector y modo de almacenamiento

```

length(w)            # devuelve la longitud del vector w
length(g)            # devuelve la longitud del vector g
length(c(w,g))       # devuelve la longitud del vector w concatenado con g
mode(w)              # devuelve el tipo de datos del vector w, en este caso numérico
mode(g)              # devuelve el tipo de datos del vector g, en este caso
                    # de caracteres
mode(m)              # devuelve el tipo de datos del vector m, en este caso lógico
storage.mode(y)       # equivalente a mode, en este caso el tipo es "double" que
                    # significa de doble precisión
storage.mode(w)       # en este caso el tipo es entero
storage.mode(g)
storage.mode(m)

```

También existen diferentes maneras de almacenar un valor numérico.

```

pi
round(pi, 3)      # redondea el valor de pi a 3 decimales
signif(pi, 2)     # devuelve el valor de pi con dos cifras significativas
abs(-8)           # devuelve el valor absoluto
trunc(-pi)        # devuelve el valor entero de pi sin su parte decimal
floor(pi)         # devuelve el entero inmediato anterior a pi
ceiling(pi)       # devuelve el entero inmediato posterior a pi

```

3.2.7 Ordenamiento de vectores

```

n=c(2,1,-8,3)
sort(n)           # ordena las componentes numéricas en orden creciente
g
g[c(2, 3, 1)]    # cambia el orden en las componentes
sort(g)           # ordena alfabéticamente las componentes del vector de
                  # caracteres
rev(g)            # invierte el orden de las componentes del vector
order(g)          # devuelve la posición de las componentes del vector en el
                  # orden alfabético
g[order(g)]       # equivalente a sort
q=rep(c(1,3,7),2)
q
unique(q)         # devuelve las cifras que aparecen omitiendo repeticiones
duplicated(q)     # indica cifras repetidas

```

3.2.8 Operaciones con vectores

Analizar el resultado de cada una de las siguientes operaciones.

```

v1<-rep(2,4)
v2<-2:5
-2*v2            # producto de un vector por un escalar
v2+3             # suma 3 a cada componente
v1+3*v2          # combinación lineal de vectores
v1/v2            # división componente a componente
v1*v2            # producto componente a componente
v1**v2           # potenciación componente a componente

```

```

v3=2*v1-3*v2
v3
min(v3)          # devuelve el valor mínimo del vector
max(v3)          # devuelve el valor máximo del vector
sum(v3)          # suma las componentes del vector
cumsum(v3)       # devuelve un vector que guarda en cada componente la suma de las
                  # anteriores más ésta
prod(v3)         # multiplica las componentes del vector
sum(v1*v2)       # producto escalar de dos vectores
v1**v2           # producto escalar de dos vectores

```

El espacio de trabajo - *Workspace* -: Todas las variables u “objetos” creados en R están guardados en lo que se llama el espacio de trabajo. Para ver qué variables están en el espacio de trabajo puede usarse la función `ls()` (esta función no necesita argumentos entre los paréntesis).

Observar cuáles son los objetos que se han creado.

```
ls() # lista todas las variables que se han creado en el espacio de trabajo
```

3.3 Matrices

3.3.1 Carga y odenamiento de valores en una matriz

```
data=1:10
matrix(data,nrow=2,ncol=5) # acomoda los datos por columna en una matriz
                             de nrow filas y ncol columnas
matrix(data,nrow=2,ncol=5,byrow=T) # acomoda los datos por fila en una matriz
                                     de nrow filas y ncol columnas
matrix(c(2,4,5,6,-8,11),nrow=2) # acomoda la concatenación por columnas de
                                  acuerdo a la cantidad de filas indicada
```

3.3.2 Instrucciones **rbind** y **cbind**

```
vec1=seq(2,5) # asigna valores a un vector
vec2=seq(-5,-2) # asigna valores a otro vector
cbind(vec1,vec2) # devuelve la matriz que tiene a estos vectores como columnas
rbind(vec1,vec2) # devuelve la matriz que tiene a estos vectores como filas
```

3.3.3 Nombres de filas y de columnas

```
mat1=matrix(data,nrow=2,ncol=5) # asigna valores a una matriz
mat1 # devuelve la matriz
colnames(mat1)<-c("A","B","C","D","E") # asigna nombres a las columnas de la matriz
mat1 # devuelve la matriz, ahora con nombres en
      sus columnas
rownames(mat1)<-c("2015","2016") # asigna nombres a las filas de la matriz
mat1 # devuelve la matriz, ahora con nombres en
      sus filas
```

3.3.4 Atributos de una matriz

```
dim(mat1) # devuelve la cantidad de filas y de columnas de la matriz
storage.mode(mat1) # devuelve el tipo de valores guardados en la matriz
```

3.3.5 Acceso a los elementos de una matriz

```
mat1[1,2] # devuelve el elemento de la fila 1 y la columna 2 de la
           matriz
mat1[1,3:5] # devuelve los elementos de la fila 1 y correspondientes
             a las columnas de 3 a 5
mat1[1,] # devuelve la fila 1 de la matriz
mat1[,2] # devuelve la columna 2 de la matriz
mat1[1,2]<-3.1 # asigna un valor dado en la fila 1 y la columna 2 de la
               matriz
mat1
storage.mode(mat1) # observar que cambió el modo de almacenamiento de la matriz
```

3.3.6 Operaciones con matrices

```

mat2<-matrix(seq(10,1),nrow=2,byrow=T) # asigna valores a una nueva matriz
mat2                                     # devuelve la matriz
t(mat2)                                 # devuelve la matriz traspuesta de la
                                        # matriz
mat1+mat2                               # suma de matrices
mat1-mat2                               # resta de matrices
mat2+3                                  # suma 3 a cada elemento de la matriz
3*mat2                                  # producto de matriz por escalar
mat1*mat2                               # producto elemento a elemento
sqrt(mat2)                             # raíz cuadrada de cada elemento de la
                                        # matriz
sqrt(mat1)                             # observar que cuando la operación no
                                        # está definida devuelve NaN (not a number)
sqrt(-9)                               # no está definido entonces devuelve NaN
sqrt(-9+0i)                            # lo trata como número complejo
exp(mat2)                               # exponencial a cada elemento de la
                                        # matriz
log10(mat2)                            # logaritmo a cada elemento de la
                                        # matriz
mat3=mat1%*%t(mat2)                    # asigna a una matriz el producto
                                        # matricial de dos matrices
mat3                                    # devuelve el resultado del producto
                                        # matricial
det(mat3)                              # devuelve el determinante de la matriz
solve(mat3)                            # devuelve la matriz inversa de la matriz
1/det(mat3)                            # devuelve el inverso del determinante
det(solve(mat3))                       # devuelve el determinante de la matriz
                                        # inversa
mat3*%solve(mat3)                      # devuelve el producto de una matriz por
crossprod(mat3,solve(mat3))            # devuelve el producto entre la traspuesta de
                                        # la primera matriz y la segunda matriz
                                        # su inversa; es decir, la matriz identidad
diag(mat3)                             # devuelve la diagonal principal de la
                                        # matriz
sum(diag(mat3))                        # devuelve la traza de la matriz
eigen(mat3)                            # devuelve los autovalores y los autovectores
                                        # de la matriz
eigen(mat3)$values                     # devuelve sólo los autovalores de la matriz
eigen(mat3)$vectors                    # devuelve sólo los autovectores de la matriz

```

3.3.7 Función **apply**

```
mat1
apply(mat1,1,sum)      # devuelve la suma de cada fila de la matriz
apply(mat1,2,sum)      # devuelve la suma de cada columna de la matriz
apply(mat1,2,min)      # devuelve el mínimo de cada columna de la matriz
apply(mat1,1,mean)     # devuelve la media de cada fila de la matriz
apply(mat1,1,median)   # devuelve la mediana de cada fila de la matriz
apply(mat1,1,var)      # devuelve la varianza de cada fila de la matriz
apply(mat1,1,sd)       # devuelve el desvío estándar de cada fila de la matriz
apply(mat1,1,summary)  # devuelve un resumen de cada fila de la matriz, incluyendo
                        # valor mínimo, primer cuartil, mediana, media, tercer cuartil
                        # y valor máximo
```

3.4 Data-frame

Un grupo de amigos está formado por:

- Ana de 23 años
- Luis de 24 años
- Pedro de 22 años
- Juan de 24 años
- Eva de 25 años
- Jorge de 27 años

Creamos los vectores correspondientes a nombre, edad y sexo. En la variable sexo usamos la siguiente codificación:

- M = mujer
- H = hombre

```
Nombre = c("Ana","Luis","Pedro","Juan","Eva","Jorge") # crea un vector con los
                                                         nombres
Edad = c(23,24,22,24,25,27)                             # crea un vector con las
                                                         edades correspondientes
Sexo = as.factor(c("F",rep("M",3),"F","M"))              # crea un vector como factor
                                                         con el sexo correspondiente
levels(Sexo)                                              # devuelve los grupos del
                                                         vector dado como factor
datos=data.frame(Nombre,Edad,Sexo)                       # arma un entorno de datos
datos                                                    # devuelve el entorno de
                                                         datos
mean(datos$Edad[datos$Sexo=="F"])                        # devuelve el promedio de la
                                                         edad de las mujeres
```

¿Qué diferencia hay con una matriz? El *dataframe* soporta que las columnas tengas distinto modo de almacenamiento, por ejemplo en el caso anterior tenemos una que es numérica y otra que es factor.

3.5 Listas

3.5.1 Generación de listas

```

milista<-list(T,3,"curso R",mat3)           # genera una
milista                                     # lista de objetos
names(milista)<-c("valor de verdad","marcas","presente","matriz 3") # devuelve la
milista                                     # lista creada
milista                                     # asigna nombres
                                           # a los elementos
                                           # de la lista
milista                                     # devuelve la
                                           # lista con
                                           # nombres
milista2=list("a"=3,"b"=mat3,"c"=vec1)      # genera otra
milista2                                   # lista
                                           # devuelve la
                                           # otra lista

```

Observar que en una lista se pueden almacenar distintos tipos de objetos, vectores, matrices, valores lógicos, dataframes, etc.

3.5.2 Acceso a elementos de una lista

```

milista$presente      # devuelve el elemento guardado según el nombre
milista[[2]]          # devuelve el elemento guardado según la posición
length(milista)       # devuelve la cantidad de elementos de la lista
milista$a<-7          # agrega una componente al final de la lista
milista               # devuelve la lista modificada
milista[[2]]<-mat1    # reasigna un valor de una componente según la posición
milista               # devuelve la lista modificada

```

3.5.3 Operación sobre elementos de una lista

```

listal=list("mat1"=mat1,"mat2"= mat2, "mat3"= mat3) # genera una lista
listal                                             # devuelve la lista
listal$mat1+listal$mat2                          # devuelve la suma entre el
                                                    # primer y el segundo elemento
                                                    # de la lista

```

3.5.4 Función **split**

```

split(datos, Sexo)
# particiona un entorno de datos a partir del factor Sexo
datos2=data.frame(datos,"Nación"=as.factor(c(rep("arg",3),rep("per",3))))
# agrega información al entorno de datos
datosmujeres=split(datos2,datos2$Sexo)[[1]]
# almacena los datos correspondientes a la partición por mujeres
split(datosmujeres,datosmujeres$Nación)
# particiona los nuevos datos por el factor Nación; es decir, se ha
# particionado un data frame por dos factores

```

3.6 Generación de números al azar

```

muestra.unif1=runif(100)           # genera una muestra uniforme en [0,1]
                                   # de 100 datos
muestra.unif1                      # devuelve la muestra generada
muestra.unif2=runif(200,min=2,max=5) # genera una muestra uniforme en [2,5]
                                   # de 200 datos
muestra.unif2                      # devuelve la muestra generada
muestra.norm.est=rnorm(30)         # genera una muestra normal estándar
                                   # de 30 datos
muestra.norm.est                  # devuelve la muestra generada
muestra.norm=rnorm(50,mean=10,sd=3) # genera una muestra normal (10,3)
                                   # de 50 datos
muestra.norm                      # devuelve la muestra generada
muestra.gamma=rgamma(40,rate=2, shape=3) # genera una muestra gamma (2,3)
                                   # de 40 datos
muestra.gamma                    # devuelve la muestra generada
muestra.f=rf(80,df1=5,df2=6)      # genera una muestra F de Snedekor (5,6)
                                   # de 80 datos
muestra.f                        # devuelve la muestra generada
muestra.exp=rexp(90,2)            # genera una muestra exponencial (2)
                                   # de 90 datos
muestra.exp                      # devuelve la muestra generada
muestra.chi=rchisq(70,df=4)       # genera una muestra chi cuadrado con
                                   # 4 grados de libertad de 70 datos
muestra.chi                      # devuelve la muestra generada

```

3.7 Ejercicios de revisión

1. (a) Construir el vector $v1$ con 10 componentes que sean una progresión aritmética de razón 3 y término inicial 4.
 (b) Construir el vector $v2$ con 10 componentes aleatorias de distribución normal, con media 3 y desvío 4.
 (c) Efectuar las siguientes operaciones:
 - i. la suma del doble de $v1$ y el triple de la raíz cuadrada de los elementos de $v2$.
 - ii. el producto escalar entre $v1$ y $v2$.
 - iii. la multiplicación entre $v1$ y $v2$ de modo tal que el resultado sea una matriz de 10×10 .
- (d) Armar una matriz que tenga por filas a $v1$ y a $v2$.
 (e) Ponerles nombre a las filas y las columnas de la matriz armada.
2. (a) Construir una matriz de 2×3 con valores enteros distintos.
 (b) Construir otra matriz de 2×3 con los valores absolutos de la primera.
 (c) Hallar la traspuesta de la primera matriz y multiplicarla por la segunda.
 (d) Mostrar con un ejemplo que la suma de una matriz cuadrada y su traspuesta es una matriz simétrica.
 (e) Verificar con un ejemplo que la traza de la suma es la suma de las trazas.
 (f) Verificar con un ejemplo que el determinante del producto de dos matrices cuadradas es igual al producto de los determinantes.
 (g) Verificar que el producto de los autovalores de una matriz cuadrada es igual al determinante de la misma.
 (h) Verificar que la suma de los autovalores de una matriz cuadrada es igual a la traza de la misma.
3. (a) Cargar el siguiente data.frame

Candidatos	Cordialidad	Presencia	Idioma	Juez
Mariana	80	90	70	A
Maia	80	90	60	A
Sabrina	90	60	50	A
Daniel	80	50	50	A
Alejandra	70	60	50	A
Carlos	90	85	60	A
Mariana	60	78	80	B
Maia	65	90	65	B
Sabrina	70	60	50	B
Daniela	70	58	40	B
Alejandra	55	70	65	B
Carlos	80	90	40	B

- (b) Particionar el data frame por juez.
- (c) Agregar a la base la columna sexo.
- (d) Particionar el data frame por sexo.
- (e) Hallar los puntajes medios de idioma, presencia y cordialidad por sexo.
- (f) Hallar los puntajes medianos de idioma, presencia y cordialidad por juez.

Capítulo 4

Introducción al análisis de datos

Antes de comenzar con este capítulo, observemos que cada vez que se cierra una sesión de trabajo en R, aparece un cuadro de diálogo donde se pregunta si se quiere guardar el área de trabajo. Al responder afirmativamente, los objetos creados estarán disponibles en la siguiente sesión. En caso contrario, se deberán volver a cargar los datos a fin de tenerlos disponibles en la próxima sesión.

4.1 Directorio de trabajo

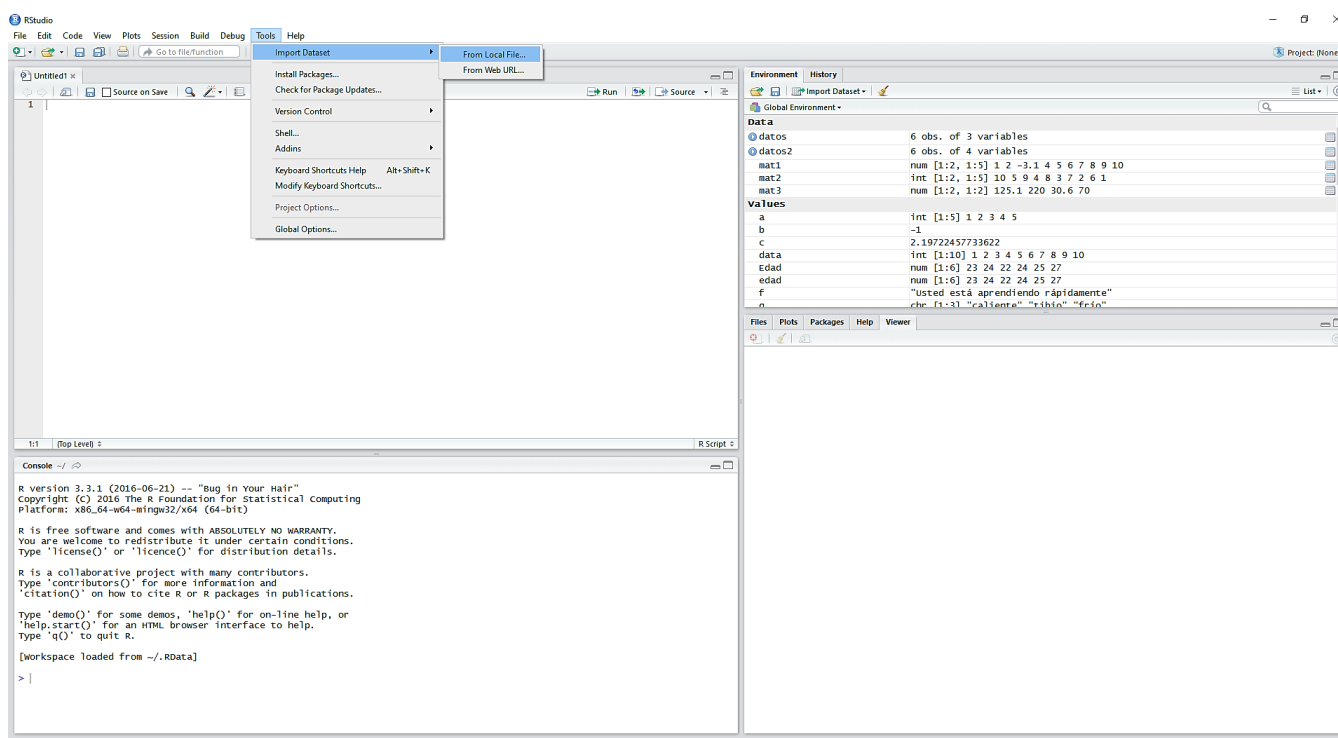
Cada vez que se inicia una nueva sesión de trabajo, es conveniente establecer el directorio en donde uno va a trabajar. Este procedimiento también debe realizarse al trabajar en documentos compartidos (por ejemplo, al utilizar *Dropbox*). Hay que tener en cuenta que al escribir la ruta correspondiente, las barras que indican los niveles cambian de sentido; es decir es vez de escribir \ hay que escribir /.

```
setwd("C:/Users/Usuario-PC/Curso R/Clases") # fija la ruta de la carpeta donde se
                                              # encuentran los archivos con los que se
                                              # va a trabajar y donde se guardarán las
                                              # salidas que se ejecuten
getwd()                                     # devuelve la ruta de la carpeta de
                                              # trabajo
```

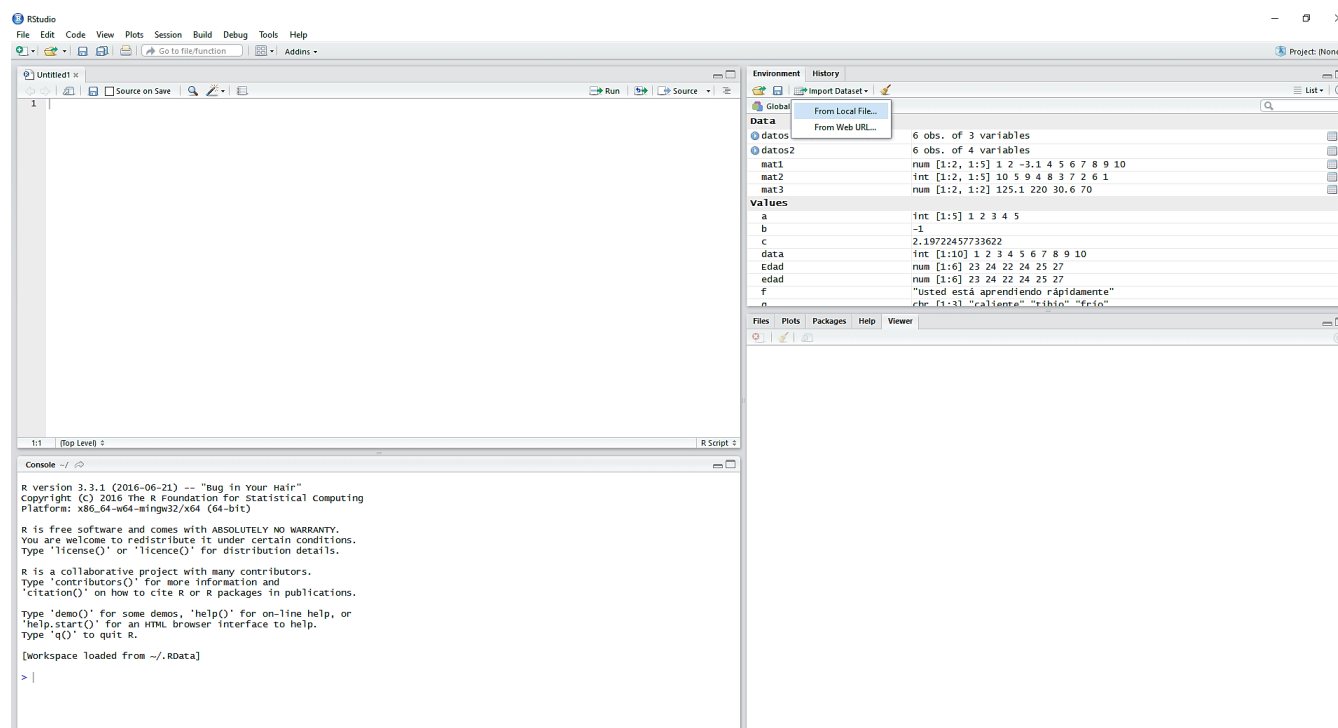
4.2 Bases de datos

4.2.1 Importación de bases de datos

Existen varias maneras de importar datos de una hoja de cálculo en Excel a R. La primera es guardar el archivo de datos con el formato CSV (delimitado por comas) e importarlo a R desde la barra de herramientas como se muestra en la siguiente imagen.



La segunda es desde la barra de herramientas de la ventana *Environment* como se indica a continuación.



También podemos hacer esto por línea de comando.

```
read.csv2("C:/Users/Usuario-PC/Dropbox/Curso R/Datos/IMCinfantil.csv")
# importa el archivo con la base de datos
IMCinfantil<-read.csv2("C:/Users/Usuario-PC/Dropbox/Curso R/Datos/IMCinfantil.csv")
# guarda la base de datos bajo un nombre
View(IMCinfantil)
# muestra la base de datos importada
head(IMCinfantil)
# muestra las seis primeras filas de datos y los nombres de las columnas
```

La base con la cual vamos a trabajar, llamada *IMCinfantil* (ver 8.1, 8.2, 8.3), corresponde a un estudio antropométrico y contiene 150 registros de niños para los cuales se observaron las variables descriptas a continuación.

PACIENTE: indica el número de orden del paciente en la base.

EDAD: indica la edad en años del paciente, por lo cual es una variable numérica con valores enteros positivos.

SEXO: indica el sexo del paciente, siendo una variable categórica con dos niveles de medición M (masculino) y F (femenino).

PESO: indica el peso del niño en kilogramos, siendo una variable continua con valores reales positivos redondeados a una cifra decimal.

TALLA: indica la estatura del niño en metros, siendo una variable continua con valores reales positivos redondeados a una cifra decimal.

IMC: indica el índice de masa corporal (IMC) y es la variable continua que resultada del cociente entre el peso y el cuadrado de la talla. Sus siglas en inglés son BMI de *Body Mass Index*.

PIMC: indica el percentil del índice de masa corporal del niño respecto de la distribución de IMC mundial correspondiente a esa edad.

CC: indica la circunferencia de cintura en centímetros, siendo una variable redondeada a valores enteros positivos.

CatPeso: indica la categoría de un niño respecto de su peso, siendo una variable categórica con los siguientes niveles

D (deficiente) cuando el PIMC es inferior a 5

N (normal) cuando el PIMC está entre 5 y 85

SO (sobrepeso) cuando el PIMC está entre 85 y 95

OB (obesidad) cuando el PIMC es superior a 95

En lo que sigue, nos proponemos estudiar algunas características de esta base con el propósito de analizar órdenes en R.

4.2.2 Llamada de datos

Existen dos alternativas de llamar a los datos guardados en cierta variable de una base: por su nombre o por su posición. Observemos que en nuestro caso particular, la variable EDAD se encuentra en la segunda columna de la base *IMCinfantil*.

```
IMCinfantil$EDAD # devuelve los datos de la variable indicada por su nombre
IMCinfantil[,2]  # devuelve los datos de la variable indicada por su posición
```

4.2.3 Análisis de distribuciones

Comenzamos analizando la distribución de las categorías de sexo.

```
table(IMCinfantil$SEXO) # devuelve las frecuencias
                        # absolutas de las categorías
                        # de la variable

100*table(IMCinfantil$SEXO)/length(IMCinfantil$SEXO) # calcula las frecuencias
                                                        # porcentuales

sal.sexo=rbind(table(IMCinfantil$SEXO),
100*table(IMCinfantil$SEXO)/length(IMCinfantil$SEXO)) # combina las dos frecuencias
                                                        # por fila en una salida

rownames(sal.sexo)=c("frec.abs", "frec.porc") # asigna nombre a las filas
                                                # de la salida

colnames(sal.sexo)=c("Femenino", "Masculino") # asigna nombre a las columnas
                                                # de la salida

sal.sexo # muestra la salida
round(sal.sexo, 2) # redondea los datos de la
                  # salida a dos dígitos decimales
```

Ahora analizamos la distribución de las categorías de peso.

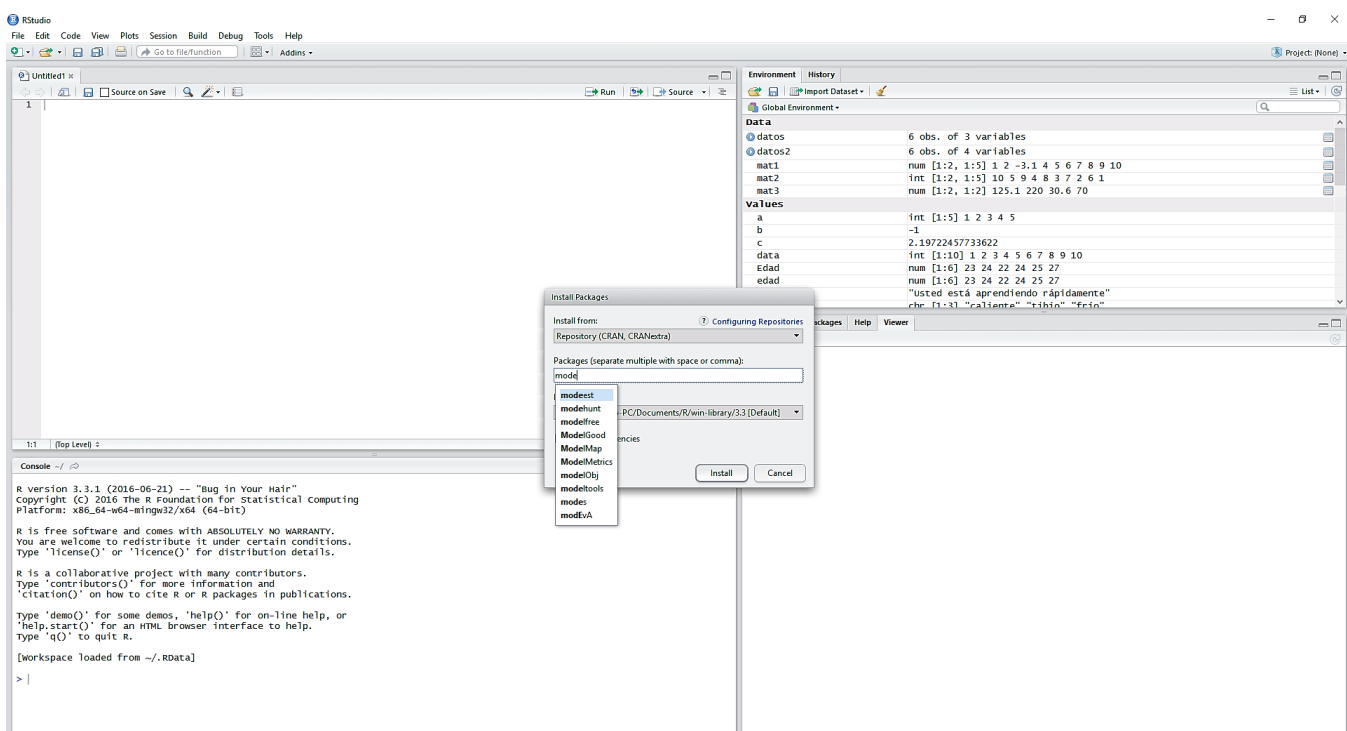
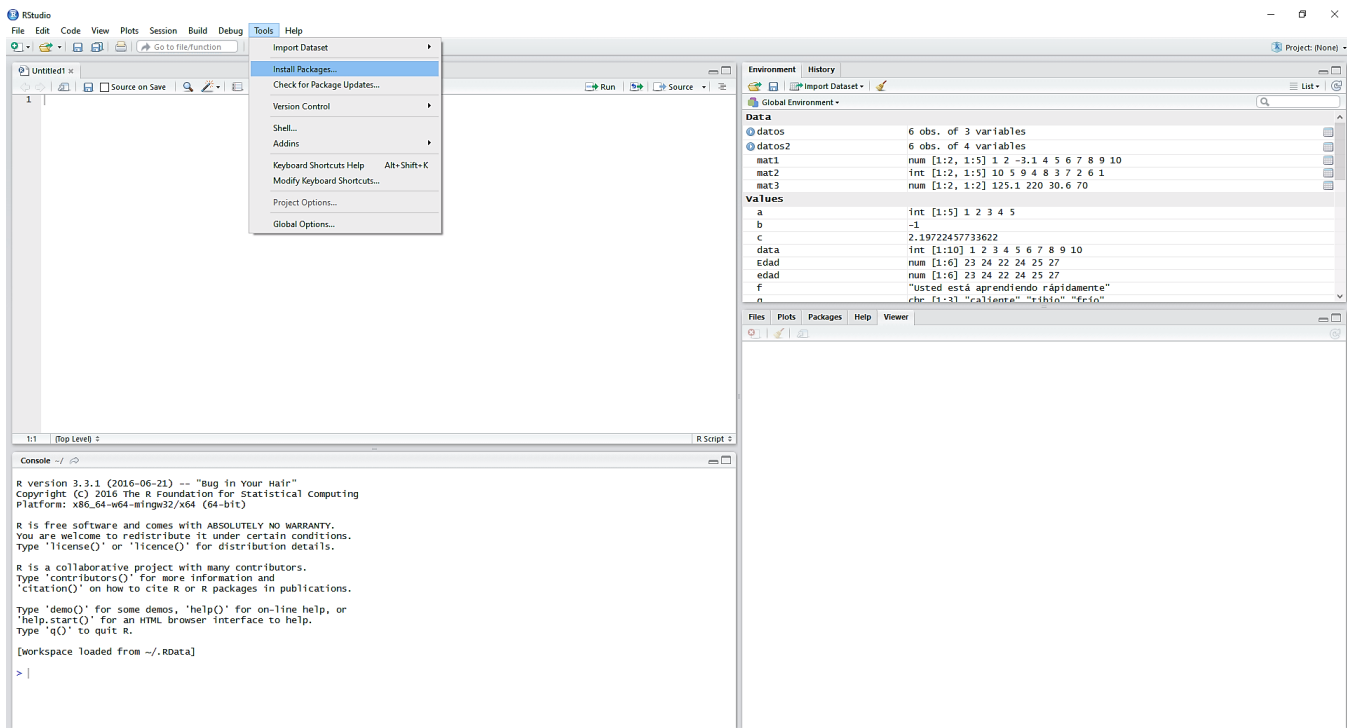
```
table(IMCinfantil$CatPeso)
100*table(IMCinfantil$CatPeso)/length(IMCinfantil$CatPeso)
sal.catpeso=rbind(table(IMCinfantil$CatPeso),
100*table(IMCinfantil$CatPeso)/length(IMCinfantil$CatPeso))
rownames(sal.catpeso)=c("frec.abs", "frec.porc")
levels(IMCinfantil$CatPeso)
# devuelve las categorías ordenadas alfabéticamente
colnames(sal.catpeso)=c("Deficiente", "Normal", "Obeso", "Con sobrepeso")
sal.catpeso
round(sal.catpeso, 2)
sal.catpeso[, c(1, 2, 4, 3)]
# muestra la salida con las columnas en el orden indicado
```

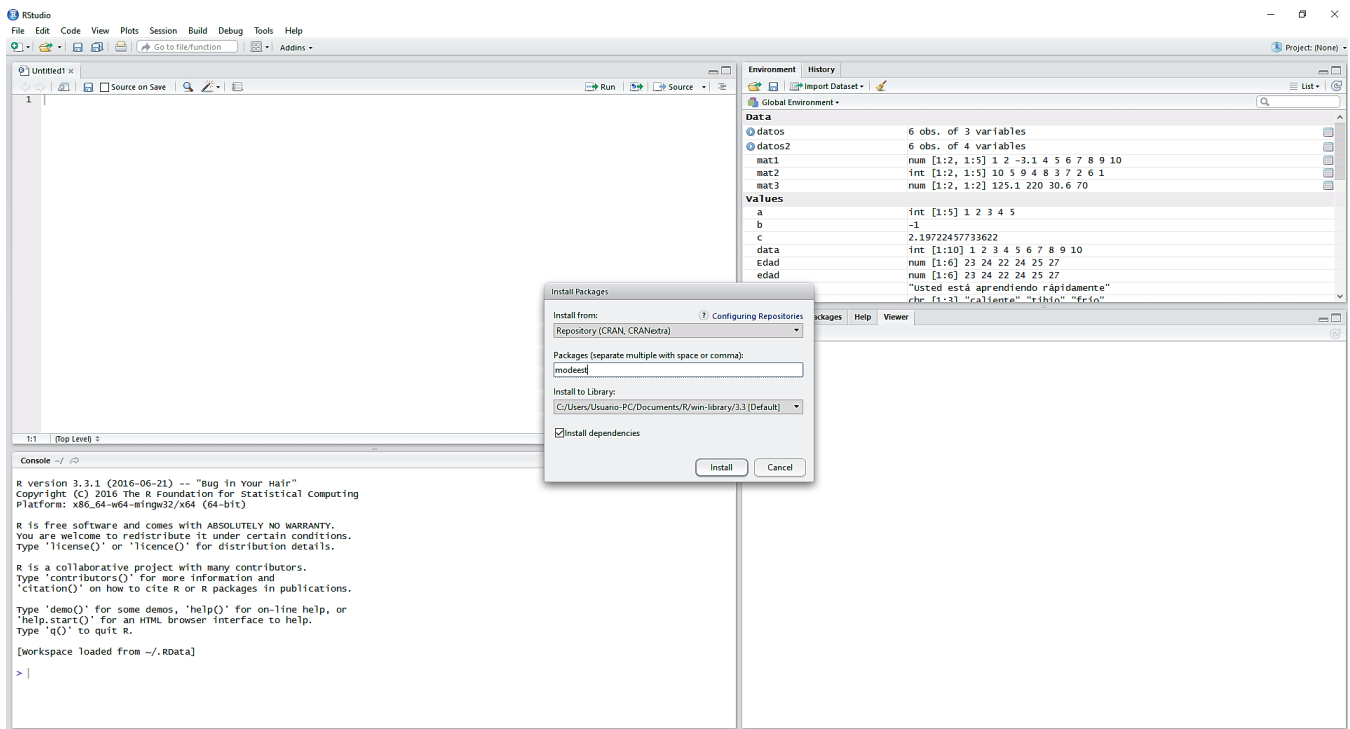
Finalmente, analizamos la **distribución conjunta** de las categorías de sexo y peso.

```
dist.conj=table(IMCinfantil$CatPeso, IMCinfantil$SEXO)
# devuelve la distribución conjunta
total=apply(dist.conj, 2, sum)
# calcula los totales de las categorías de la variable sexo
dist.porc=round(100*cbind(dist.conj[,1]/total[1], dist.conj[,2]/total[2]), 2)
# calcula la distribución porcentual por sexo
colnames(dist.porc)<-c("F(%)", "M(%)")
# asigna nombre a las columnas de la distribución porcentual
sal.conj=cbind(dist.conj, dist.porc)
# combina ambas distribuciones por columna
Totales=apply(sal.conj, 2, sum)
# calcula el total de cada columna
sal.fin=rbind(sal.conj, Totales)
# agrega una fila con los totales
sal.fin
# muestra la salida final
```


4.3 Cálculo de resúmenes estadísticos

Existe un paquete o librería en R llamado *Mode Estimation* el cual provee resúmenes estadísticos especiales como la moda que no está disponible en el R básico. Lo primero que se debe hacer es cargar este paquete desde la barra de herramientas buscándolo como `modeest`, para lo cual hay que seguir estos pasos.





De esta manera, se ha instalado esta librería en el equipo. Sin embargo, cada vez que se necesite que esté disponible en el lugar de trabajo, se deberá correr la siguiente instrucción

```
library(modeest) # llama a la librería
```

Utilizamos nuestra base *IMCinfantil* a modo de ejemplo.

```
imc.base=cbind(IMCinfantil[,c(2,4:6,8)]) # arma una base
                                         # seleccionando las
                                         # columnas indicadas

medias=round(apply(imc.base,2,mean),2)   # calcula las medias
medianas=round(apply(imc.base,2,median),2) # calcula las medianas
varianzas=round(apply(imc.base,2,var),2)  # calcula las varianzas
desv.standard=round(apply(imc.base,2,sd),2) # calcula los desvíos
                                         # estándar

resumenes=rbind(medias,medianas,varianzas,desv.standard) # junta en una tabla los
                                                         # resúmenes

resumenes # muestra la salida
```

4.3.1 Otros resúmenes estadísticos de tendencia central

```
mfv(IMCinfantil$EDAD) # calcula la moda de la edad
mfv(IMCinfantil$TALLA) # calcula la moda de la talla
Z= imc.base[,3]*100    # guarda los datos de la talla en centímetros
mean(Z)                # calcula la media
mean(Z, trim=0.1)      # calcula la media podada al 10% de cada lado
mean(Z, trim=0.5)      # calcula la media podada al 50% de cada lado
median(Z)               # calcula la mediana
```

¿Qué se puede observar a partir de los dos últimos comandos?

Observación: La media alfa podada generaliza a la media aritmética o promedio muestral y a la mediana, puesto que la primera corresponde a poda cero y la última a la máxima poda posible; es decir, del 50%.

$$\frac{\bar{X}}{0} \quad X_{trim} \quad \frac{\tilde{X}}{0.5}$$

▲

4.3.2 Coeficiente de variación

No hay un comando en R que calcule el coeficiente de variación. En un caso particular, se podrían ejecutar comandos sólo para ese caso. Sin embargo, sería conveniente tener una expresión de manera general aplicable a cualquier variable. La idea es construir una función (en el capítulo [] se trabajará con funciones de forma más detallada.)

Entonces, vamos a definir una función que calcule el coeficiente de variación de una variable de datos numéricos, digamos x .

```
c.var=function(x){
  100*sd(x)/mean(x)
}                                     # define la función de coeficiente de variación
c.var(Z)                             # aplica la función a la variable Z
```

4.3.3 Otros resúmenes estadísticos de posición y dispersión

```
quantile(Z, 0.75)
# calcula el cuantil 75
quantile(IMCinfantil$EDAD, probs = seq(0, 1, 0.2))
# calcula los cuantiles 0, 20, 40, 60, 80 y 100
quantile(IMCinfantil$TALLA, probs = seq(0, 1, 0.25))
# calcula los cuantiles 0, 25, 50, 75 y 100
DI.edad=quantile(IMCinfantil$EDAD,0.75)-quantile(IMCinfantil$EDAD,0.25)
# calcula la desviación intercuartil
DI.edad
DI.Z=quantile(Z,0.75)-quantile(Z,0.25)
DI.Z
```

4.3.4 Mediana de la desviación absoluta

El estadístico **MAD** (median absolute deviation) es la mediana de los valores absolutos de los desvíos respecto de una constante, que salvo que se indique lo contrario, se considera la mediana de la muestra, aunque puede elegirse otro valor.

```
mad(Z, constant = 1)               # calcula la mediana de los valores absolutos
                                   # de los desvíos
mad(Z, constant = 1.4826)          # multiplica lo anterior por la constante 1.4826
mad(Z)                             # toma por default la constante 1.4826
abs(mad(Z)-sd(Z))                 # cuantifica el apartamiento de normalidad
100*(abs(mad(Z)-sd(Z)))/sd(Z)     # calcula el porcentaje del apartamiento de normalidad
```

Cuando la distribución subyacente es normal, existe una relación entre el MAD y el desvío standard dada por

$$\frac{\text{MAD}(x)}{\sigma(x)} \approx \phi^{-1}\left(\frac{3}{4}\right) \quad (4.3.1)$$

Se puede deducir entonces que

$$\text{MAD}(x) \approx \sigma(x) \phi^{-1} \left(\frac{3}{4} \right) = 0.6745 \sigma(x) \quad (4.3.2)$$

o, equivalentemente

$$\sigma(x) \approx \frac{\text{MAD}(x)}{\phi^{-1} \left(\frac{3}{4} \right)} \approx 1.4826 \text{MAD}(x) \quad (4.3.3)$$

Por esta razón, salvo que se indique lo contrario, R multiplica a los desvíos por este valor (1.4826). La idea es que si la muestra resultara normal, el MAD y el desvío estándar deberían ser aproximadamente iguales. De esta forma tenemos una manera simple de analizar normalidad, sin necesidad de aplicar un test. El apartamiento de la normalidad puede cuantificarse mediante la distancia entre el desvío estándar y el MAD. Proponemos observar las siguientes salidas.

```
muestra.norm=rnorm(1000)      # genera 1000 datos con distribución normal
sd(muestra.norm)              # calcula el desvío estándar
mad(muestra.norm)             # calcula el MAD
muestra.chi=rchisq(1000,3)    # genera 1000 datos con distribución chi cuadrado
                               con 3 grados de libertad

sd(muestra.chi)
mad(muestra.chi)
```

En algunas ocasiones el MAD es calculado respecto de la media; es decir, es la mediana de los valores absolutos de las diferencias entre cada dato y la media. Sin embargo, R lo calcula respecto de la mediana por “default”.

4.3.5 Tratamiento de los NA

La sigla NA significa “not available” y representa una constante lógica de longitud uno la cual contiene el indicador de un valor que falta.

```
is.na(Z)                      # indica los valores que faltan
W<-Z
W[1]<-NA                      # asigna un valor perdido en la primera componente del vector W
is.na(W)
mean(W)                       # devuelve error
mean(W, na.rm=T)             # no considera los valores no disponibles
mean(na.omit(W))             # equivalente al anterior
median(W, na.rm=T)           # otra función que requiere excluir los valores no disponibles
sd(W, na.rm=T)               # otra función que requiere excluir los valores no disponibles
```

4.3.6 Resúmenes segmentados por dos categorías

Vamos a generar un archivo *Excel* que contenga los resúmenes estadísticos de una base particionada por alguna de sus categorías.

A partir de nuestra base *IMCinfantil*, comenzamos con un resumen estadístico para la sub-base de la categoría mujeres.

```

base.split=split(IMCinfantil[,c(2,4:6,8)],IMCinfantil$SEXO)
# arma una base numérica dividida según el sexo
mujeres=as.data.frame(base.split[[1]])
# guarda la primera subclase (F) como entorno de datos
varones=as.data.frame(base.split[[2]])
# guarda la segunda subclase (M) como entorno de datos
lapply(mujeres,"summary")
# devuelve una lista aplicando el resumen de todas las variables de la base
unlist(lapply(mujeres,"summary"))
# transforma la salida de lista a vector
matrix(unlist(lapply(mujeres,"summary")),nrow=5,ncol=6,byrow=T)
# acomoda los elementos del vector en una matriz de modo tal que cada variable ocupe
una fila
sumary.muj=matrix(unlist(lapply(mujeres,"summary")),nrow=5,ncol=6,byrow=T)
# guarda los resúmenes estadísticos
sd.mujeres=round(unlist(lapply(mujeres,"sd")),2)
# calcula el desvío estándar de las variables de interés redondeado a dos dígitos
decimales
salida.muj=cbind(sumary.muj[,4],sd.mujeres,sumary.muj[,c(3,2,5)])
# combina las columnas indicadas
colnames(salida.muj)<-c("MEDIA","SD","MEDIANA","Q1","Q3")
# pone nombre a las columnas
salida.muj
# devuelve la salida
write.table(salida.muj,"sal_muj.xls",col.names=NA, row.names=TRUE, sep="\t",quote=FALSE)
# exporta la salida en un archivo excel separado por tabulaciones

```

Ahora, realizaremos un resumen estadístico teniendo en cuenta la categoría dada por la variable peso.

```

### partición por categoría de peso

```

```

base.porpeso=split(IMCinfantil[,c(2,4:6,8)],IMCinfantil$CatPeso)
defic=as.data.frame(base.porpeso[[1]])
normal=as.data.frame(base.porpeso[[2]])
obeso=as.data.frame(base.porpeso[[3]])
sobrep=as.data.frame(base.porpeso[[4]])

```

```

### generación de salida para niños con peso deficiente

```

```

sumary.defic=matrix(unlist(lapply(defic,"summary")),nrow=5,ncol=6,byrow=T)
sd.defic=round(unlist(lapply(defic,"sd")),2)
cbind(sumary.defic[,4],sd.defic,sumary.defic[,c(3,2,5)])
salida.defic=cbind(sumary.defic[,4],sd.defic,sumary.defic[,c(3,2,5)])
colnames(salida.defic)<-c("MEDIA","SD","MEDIANA","Q1","Q3")
rownames(salida.defic)<-c("EDAD.D","PESO.D","TALLA.D","IMC.D","CC.D")
salida.defic

```

```
### generación de salida para niños con peso normal

sumary.normal=matrix(unlist(lapply(normal,"summary")),nrow=5,ncol=6,byrow=T)
sd.normal=round(unlist(lapply(normal,"sd")),2)
cbind(sumary.normal[,4],sd.normal,sumary.normal[,c(3,2,5)])
salida.normal=cbind(sumary.normal[,4],sd.normal,sumary.normal[,c(3,2,5)])
colnames(salida.normal)<-c("MEDIA","SD","MEDIANA","Q1","Q3")
rownames(salida.normal)<-c("EDAD.N","PESO.N","TALLA.N","IMC.N","CC.N")
salida.normal
```

```
### generación de salida para niños con sobrepeso

sumary.sobrep=matrix(unlist(lapply(sobrep,"summary")),nrow=5,ncol=6,byrow=T)
sd.sobrep=round(unlist(lapply(sobrep,"sd")),2)
cbind(sumary.sobrep[,4],sd.sobrep,sumary.sobrep[,c(3,2,5)])
salida.sobrep=cbind(sumary.sobrep[,4],sd.sobrep,sumary.sobrep[,c(3,2,5)])
colnames(salida.sobrep)<-c("MEDIA","SD","MEDIANA","Q1","Q3")
rownames(salida.sobrep)<-c("EDAD.S","PESO.S","TALLA.S","IMC.S","CC.S")
salida.sobrep
```

```
### generación de salida para niños obesos

sumary.obeso=matrix(unlist(lapply(obeso,"summary")),nrow=5,ncol=6,byrow=T)
sd.obeso=round(unlist(lapply(obeso,"sd")),2)
cbind(sumary.obeso[,4],sd.obeso,sumary.obeso[,c(3,2,5)])
salida.obeso=cbind(sumary.obeso[,4],sd.obeso,sumary.obeso[,c(3,2,5)])
colnames(salida.obeso)<-c("MEDIA","SD","MEDIANA","Q1","Q3")
rownames(salida.obeso)<-c("EDAD.O","PESO.O","TALLA.O","IMC.O","CC.O")
salida.obeso
```

```
### generación y exportación de una salida conjunta

salida_porpeso=rbind(salida.defic,salida.normal,salida.sobrep,salida.obeso)
write.table(salida_porpeso,"sal_porpeso.xls",col.names=NA, row.names=TRUE,
            sep="\t",quote=FALSE)
```

Finalmente, vamos a segmentar una base teniendo en cuenta dos factores diferentes, categoría de peso y sexo.

```
base.porpeso2=split(IMCinfantil[,c(2:6,8)],IMCinfantil$CatPeso)
# parte la base por categoría de peso
defic=as.data.frame(base.porpeso2[[1]])
# selecciona los niños de peso deficitario
muj.defic=split(defic,defic$SEXO)[[1]]
# parte la base de niños con peso deficitario por sexo y selecciona las mujeres
var.defic=split(defic,defic$SEXO)[[2]]
# parte la base de niños con peso deficitario por sexo y selecciona los varones
```

```

### Análogamente para el nivel "normal"

normal=as.data.frame(base.porpeso2[[2]])
muj.norm=split(normal,normal$SEXO)[[1]]
var.norm=split(normal,normal$SEXO)[[2]]

### Análogamente para el nivel "sobrepeso"

sobrepeso=as.data.frame(base.porpeso2[[4]])
muj.sobre=split(sobrepeso,sobrepeso$SEXO)[[1]]
var.sobre=split(sobrepeso,sobrepeso$SEXO)[[2]]

### Análogamente para el nivel "obeso"

obeso=as.data.frame(base.porpeso2[[3]])
muj.obes=split(obeso,obeso$SEXO)[[1]]
var.obes=split(obeso,obeso$SEXO)[[2]]

```

4.4 Aplicación: Test de Chi-cuadrado

Al momento de realizar un estudio de datos, en muchos casos resulta de interés:

- Comparar la distribución de una variable dicotómica en dos o más grupos.
- Comparar la distribución de una variable con más de dos categorías en dos o más grupos.
- Estudiar la asociación entre dos variables categóricas.
- Estudiar la asociación entre dos variables con dos o más categorías.

La forma general de presentar datos categóricos en estos casos es a través de **tablas de contingencia**. Se presentará una aproximación general para analizar este tipo de tablas, la cual está basada en la distribución χ^2 .

El **test Chi-cuadrado** es uno de los más usados para estudiar asociación entre variables. Mostraremos que las hipótesis que se pueden testear con el mismo, dependen de cómo fueron obtenidos los datos, es decir, dependen del diseño del estudio.

Supongamos que un equipo de médicos está analizando los datos obtenidos en la base *IMCinfantil*. Ellos están estudiando la hipótesis de que la categoría de peso es independiente del sexo. ¿Los ayudamos?

Primero construimos una tabla de contingencia

```

tabaux=table(IMCinfantil$SEXO,IMCinfantil$CatPeso)
tabaux
total.col=apply(tabaux,2,sum)
total.col
tabauxcol=rbind(tabaux,total.col)
tabauxcol
total.fil=apply(tabauxcol,1,sum)
total.fil
tab.cont=cbind(tabauxcol,total.fil)
tab.cont

```

Nos preguntamos si hay evidencia significativa a favor de la sospecha de los médicos a la vista de los resultados de la muestra.

En muchos casos, la información presentada en la tabla resulta más clara si se calculan los porcentajes por columna o por fila; es decir, se divide cada valor del cuerpo de la tabla por el total de su correspondiente columna (respectivamente fila) y se lo multiplica por 100.

$$\begin{aligned} \text{tab.porc.fil} &= \text{round}(100 * \text{tabaux} / \text{total.fil}[1:2], 2) \\ \text{cbind}(\text{tab.porc.fil}, \text{apply}(\text{tab.porc.fil}, 1, \text{sum})) \end{aligned} \quad (4.4.1)$$

Nos interesa saber si las observaciones realizadas tienen o no significación estadística; es decir, si lo observado es consecuencia del muestreo o bien obedece a alguna relación poblacional entre las variables.

En el siguiente cuadro se presenta un esquema general del test de independencia de Chi-cuadrado.

- Se toma una muestra aleatoria simple de una población.
- Se clasifica a cada individuo de la muestra respecto de dos variables categóricas, la primera con r categorías y la segunda con c categorías.
- Se formula la hipótesis nula de que las variables que definen las filas y las columnas son independientes. Es decir, las probabilidades conjuntas pueden ser calculadas como el producto de las probabilidades marginales.
- Se formula la hipótesis alternativa de que las variables no son independientes. Dicho de otra manera, las variables que definen las filas y las columnas mantienen algún tipo de relación.

Siguiendo con la línea de nuestro ejemplo, consideramos los siguientes conjuntos:

$$A = \{F, M\}$$

$$B = \{D, N, SO, O\}$$

Entonces las hipótesis de interés son:

$$\begin{cases} H_0 : P[(SEXO = a) \cap (CatPeso = b)] = P(SEXO = a) \cdot P(CatPeso = b) & \forall (a, b) \in A \times B \\ H_1 : \text{no se cumple alguna de las igualdades planteadas en } H_0 \end{cases} \quad (4.4.2)$$

Nos preguntamos ahora cuáles serían las frecuencias que hubiéramos observado si se verificara la hipótesis nula de independencia. Para ello, introducimos la siguiente notación:

e_{ij} es la frecuencia esperada en la casilla del elemento de la fila i y la columna j de la matriz asociada al conjunto $A \times B$. Por ejemplo, e_{12} es equivalente a la frecuencia esperada de una niña con peso normal.

De esta manera, podemos calcular

$$e_{ij} = n \cdot P(SEXO = i, CatPeso = j) = n \cdot P(SEXO = i) \cdot P(CatPeso = j)$$

donde n es el total de observaciones.

Sin embargo, estas probabilidades son desconocidas y se estiman a partir de la muestra. De este modo:

$$\hat{P}(SEXO = F) = \frac{\# \text{ mujeres}}{n}$$

$$\hat{P}(CatPeso = j) = \frac{\# (CatPeso = j)}{n}$$

$$\hat{P}[(SEXO = F) \cap (CatPeso = j)] = \hat{P}(SEXO = F) \cdot \hat{P}(CatPeso = j) \text{ aplicando la hipótesis de independencia.}$$

$$\hat{e}_{1j} = n \hat{P}[(SEXO = F) \cap (CatPeso = j)] = \frac{\# \text{ mujeres} \cdot \# (CatPeso = j)}{n}$$

donde $\#$ indica la cantidad de observaciones de acuerdo a la característica indicada.

Si las frecuencias observadas se parecen considerablemente a las estimadas bajo la hipótesis de independencia, entonces no hay motivo suficiente para suponer que hay asociación entre las variables. Por el contrario, si las frecuencias observadas difieren notablemente de las frecuencias estimadas, existe evidencia suficiente para no aceptar la hipótesis nula. Por tal motivo, necesitamos cuantificar de alguna manera la distancia entre las observaciones y las estimaciones bajo la suposición de independencia.

Se define un estadístico que testea la hipótesis nula comparando todas las frecuencias observadas en cada una de las celdas de la tabla con las frecuencias que se esperarían observar si la hipótesis nula fuera cierta.

El **estadístico del test** se define como

$$\chi^2 = \sum_{i=1}^r \sum_{j=1}^c \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

donde o_{ij} representa el valor observado en la celda correspondiente a la fila i y la columna j y e_{ij} es la frecuencia esperada para esa celda bajo la hipótesis nula.

Existe un comando en R que aplica el test de Chi-cuadrado.

```
library(MASS)           # librería requerida
esp=0                   # inicializa una variable
n=total.fil[3]          # asigna el total de filas a la variable
esp=total.fil[1:2]%*%t(total.col)/n # calcula las frecuencias esperadas
                               mediante producto matricial
chitabla=((esp-tabaux) ^ 2 )/esp # calcula el término del estadístico
                                $\chi^2$  de cada celda
chiest=sum(apply(chitabla,2,sum)) # calcula el estadístico  $\chi^2$ 
chisq.test(tabaux)      # aplica el test programado en R
```

Para poder hacer un test necesitamos conocer la distribución de muestreo del estadístico χ^2 bajo H_0 ; es decir, cuando se cumple la hipótesis nula que afirma que las dos variables son independientes.

Cuando el número de observaciones es suficientemente grande la distribución de muestreo del estadístico propuesto puede ser aproximada por la distribución χ^2 .

Una pregunta interesante es pensar qué significado tienen los grados de libertad de una distribución χ^2 . Veamos cómo se pueden interpretar los grados de libertad de una tabla. Cuando calculamos las frecuencias esperadas para una tabla, lo hacemos basándonos en los resultados obtenidos en las frecuencias marginales.

Las frecuencias esperadas deben satisfacer una serie de restricciones, como que la suma por fila o por columna debe coincidir con los correspondientes totales marginales. Esto implica que las frecuencias esperadas no son todas independientes. Por ejemplo, en una tabla de 2×2 , una vez que calculada una frecuencia esperada, el resto puede obtenerse por diferencia con la correspondiente frecuencia marginal. Por esta razón decimos que tenemos un único grado de libertad en una tabla de 2×2 . Extendiendo esta idea a una tabla de r filas y c columnas, los grados de libertad que corresponden son $(r - 1) \times (c - 1)$.

Al momento de tomar la decisión de rechazar H_0 o no, debemos calcular el p -valor del test. En este test en particular, debido a la distribución del estadístico bajo H_0 , se rechaza por cola derecha con lo cual el p -valor se calcula sobre una cola como sigue

$$p\text{-valor} = P(\chi_{obs}^2 > \chi_{(r-1)(c-1)}^2)$$

Concluimos esta sección esbozando algunos comentarios sobre el test Chi-cuadrado.

- El test Chi-cuadrado que hemos presentado es un método aproximado válido para muestras grandes. En general, se considera que la aproximación es válida cuando más del 80% de las celdas tienen frecuencias esperadas mayores que 5, y ninguna frecuencia esperada es menor que 1.
- Notar que sólo se hace referencia a frecuencias esperadas y no tienen importancia las frecuencias observadas.
- Las frecuencias esperadas muy pequeñas pueden contribuir enormemente al valor del estadístico χ^2 .
- El programa R, al aplicar este test, muestra una advertencia en rojo luego de los resultados en el caso de que los supuestos no se cumplan.

4.5 Aplicación: Paradoja de Simpson

La **paradoja de Simpson** (1951) o **efecto Yule-Simpson** es una paradoja en la cual una tendencia que aparece en varios grupos de datos desaparece cuando estos grupos se combinan, y en su lugar aparece la tendencia contraria para los datos agregados.

Esta situación se presenta con frecuencia en las ciencias sociales y en la estadística médica, resultando causa de confusión cuando se le asigna a la frecuencia de los datos una interpretación causal sin fundamento. De hecho, los médicos suelen expresar que existe una variable “confusora”.

Aunque relativamente desconocida por la mayoría de las personas, la paradoja de Simpson es bien conocida por los estadísticos y se describe en muchos libros introductorios de Estadística. Inclusive, muchos estadísticos sostienen que debería enseñarse acerca de estos resultados contrarios a la intuición a los estudiantes.

4.5.1 Ejemplo de Berkeley

Uno de los ejemplos más conocidos de la paradoja de Simpson ocurrió cuando se presentó una demanda contra la Universidad de California en Berkeley, por discriminación contra las mujeres que habían solicitado su ingreso al posgrado. Los resultados de las admisiones para el verano de 1973 mostraban que los hombres solicitantes tenían mayor posibilidad de ser elegidos que las mujeres, y que la diferencia era tal que no se creía posible que fuera debida al azar.

En la tabla 8.4 se encuentran los datos correspondientes a las solicitudes de ingreso a los cursos de posgrado de la Universidad de California del año 1973, agrupadas por sexo y por departamento. Se sugiere seguir el último ejercicio propuesto en este capítulo para comprender mejor esta paradoja.

4.6 Ejercicios de revisión

1. A partir de la base de datos *IMCinfantil*, construir las siguientes variables:

Y representa la talla en centímetros

Z representa la talla en centímetros más 2 cm asumiendo un error en el registro de datos

W representa la talla en centímetros con un error de carga en el primer valor, digamos 5000.

- (a) Verificar las siguientes propiedades de la media aritmética o promedio (mean).

- i. $\bar{Y} = 100 \cdot \overline{TALLA}$

- ii. $\bar{Z} = 100 \cdot \overline{TALLA} + 2$

- iii. La media es muy sensible a la presencia de un valor muy alejado del conjunto general.

- (b) Verificar las siguientes propiedades de la mediana (median).

- i. $\tilde{Y} = 100 \cdot \widetilde{TALLA}$

- ii. $\tilde{Z} = 100 \cdot \widetilde{TALLA} + 2$

- iii. La mediana no es muy sensible a la presencia de un valor muy alejado del conjunto general. Se dice que es una medida **robusta**. Sin embargo, la presencia un valor extremo puede cambiarla. Buscar un ejemplo en donde esto suceda y otro en donde no.

- (c) Verificar las siguientes propiedades de la varianza (var).

- i. $var(Y) = 100^2 \cdot var(TALLA)$

- ii. $var(Z) = 100^2 \cdot var(TALLA)$

- iii. La varianza es muy sensible a la presencia de valores extremos.

- (d) Verificar las siguientes propiedades del desvío estándar (sd).

- i. $sd(Y) = 100 \cdot sd(TALLA)$

- ii. $sd(Z) = 100 \cdot sd(TALLA)$

- iii. El desvío estándar es muy sensible a la presencia de valores extremos.

2. Calcular los valores mínimos y máximos de las edades y las tallas correspondientes a la base *IMCinfantil*. Comparar los resultados obtenidos con los cuantiles calculados previamente. ¿Se obtienen las mismas conclusiones en el caso discreto que continuo?

3. Consideremos la siguiente muestra: 2, 3, 7, 9, 15, 34.

- (a) Calcular la mediana.

- (b) Calcular los desvíos respecto de la mediana.
 - (c) Calcular los valores absolutos de los desvíos.
 - (d) Calcular la mediana de los valores obtenidos en el inciso anterior.
 - (e) Comparar lo obtenido aplicando la función `mad` con constante igual a 1.
4. Utilizando la muestra: 1, 2, 4, 6, 9; verificar que el MAD calculado por R es respecto de la mediana y no respecto de la media.
5. (a) Análogamente a lo realizado para `salida.muj`, generar una salida particionando la base por varones.
(b) Exportar en un archivo *Excel* la combinación de ambas.
6. Realizar los resúmenes estadísticos para cada una de las partes de la partición conjunta de la base *IMCinfantil* aplicando criterio de sexo y criterio de categoría de peso.
7. Considerando la salida de la tabla `tab.por.fil` ((4.4.1)) se pide lo siguiente.
- (a) Observar la salida e inferir alguna hipótesis.
 - (b) Construir una tabla similar tomando los porcentajes por columna.
 - (c) Observar la nueva salida e inferir alguna hipótesis.
8. Para las hipótesis del test planteado en (4.4.2), se pide lo siguiente.
- (a) Estimar las frecuencias esperadas bajo independencia.
 - (b) Calcular el estadístico χ^2 correspondiente.
 - (c) Comparar el resultado con el test de Chi-cuadrado disponible en R.
 - (d) Obtener una conclusión
9. Vamos a analizar la paradoja de Simpson utilizando los datos de la tabla 8.4.
- (a) Completar la siguiente tabla

Table 4.1: Grupo general

Sexo	Frecuencia absoluta de admitidos	Frecuencia absoluta de no admitidos
Femenino		
Masculino		

- i. Estudiar si hay asociación estadística significativa entre la admisión y el sexo, aplicando el test de Chi-cuadrado.
 - ii. Se define **chance** de ser admitido como el cociente entre los admitidos y los no admitidos. Calcular la chance de ser admitido para un hombre y la chance de ser admitida para una mujer.
 - iii. Calcule el cociente entre las chances de los hombres y las de las mujeres. Este cociente se denomina **Odds Ratio** y se lo designa con OR.
 - iv. Sacar una conclusión.
- (b) Completar las siguientes tablas.
- i. Replicar lo hecho para la tabla 4.1 para cada departamento; es decir, estudiar la presencia de asociación significativa entre sexo y admisión (ahora por departamento), aplicando el test de Chi-cuadrado.
 - ii. Estimar la chance de ser admitido siendo hombre en cada departamento por separado y la chance de ser admitido siendo mujer en el mismos departamento. Comparar esas chances mediante el OR (cociente de chances) utilizando siempre la chance de hombres como numerador.
 - iii. Concluir si se repite la tendencia del grupo general.

Table 4.2: Admisión porcentual por departamento

Departamento	Hombres		Mujeres	
	Solicitudes	Porcentaje de Admisiones	Solicitudes	Porcentaje de Admisiones
A				
B				
C				
D				
E				
F				

Table 4.3: Departamento A

Sexo	Frecuencia absoluta de admitidos	Frecuencia absoluta de no admitidos
Femenino		
Masculino		

Table 4.4: Departamento B

Sexo	Frecuencia absoluta de admitidos	Frecuencia absoluta de no admitidos
Femenino		
Masculino		

Table 4.5: Departamento C

Sexo	Frecuencia absoluta de admitidos	Frecuencia absoluta de no admitidos
Femenino		
Masculino		

Table 4.6: Departamento D

Sexo	Frecuencia absoluta de admitidos	Frecuencia absoluta de no admitidos
Femenino		
Masculino		

Table 4.7: Departamento E

Sexo	Frecuencia absoluta de admitidos	Frecuencia absoluta de no admitidos
Femenino		
Masculino		

Table 4.8: Departamento F

Sexo	Frecuencia absoluta de admitidos	Frecuencia absoluta de no admitidos
Femenino		
Masculino		

Capítulo 5

Gráficos

“Un gráfico puede valer más que mil palabras, pero puede tomar muchas palabras para hacerlo”
John Tukey

En la sección §4.3 se vio cómo cargar paquetes desde la barra del menú. Esta operación también puede realizarse mediante un comando. Mostramos esto aplicándolo a los paquetes que son necesarios para trabajar a lo largo de este capítulo.

```
install.packages("aplpack") # permite hacer caritas de Chernov
install.packages("corrplot") # permite personalizar colores y estilos de fuente
                             para gráficos
install.packages("ggplot2") # permite realizar gráficos con movimiento
install.packages("plotrix") # permite realizar gráficos de torta con volumen
install.packages("rgl")     # permite realizar gráficos en 3D

install.packages("tcltk")   # posee comandos de lenguaje de herramientas para la
                             creación de interfases gráficas
install.packages("tcltk2")  # posee comandos adicionales a tcltk

installed.packages()        # muestra los paquetes que están instalados en el
                             dispositivo
```

Como ya hemos indicado, el hecho de instalar un paquete no implica que esté disponible en cada sesión de trabajo. Existen dos comandos a tal fin: `library(nombre)` y `require(nombre)`. Ambos cargan el paquete especificado según lo ingresado en `nombre` y lo ubican en la lista de búsqueda. Se recomienda usar `library(nombre)` debido a que este comando devuelve un mensaje de error si el paquete especificado no existe. Sin embargo, dentro de una función, es conveniente utilizar `require(nombre)` puesto que devuelve un valor lógico y una advertencia. Ambas funciones corroboran y actualizan la lista actual pero ninguna vuelve a cargar un paquete ya existente. Notar que la instalación de un paquete se debe hacer una única vez en cada dispositivo pero la llamada al mismo se debe hacer cada vez que se inicia una sesión de trabajo.

```
library(grDevices) # Equipos gráficos y soporte para la base y la red de gráficos
library(tcltk)
library(aplpack)
library(corrplot)
library(ggplot2)
library(plotrix)
library(rgl)
library(tcltk2)
```

5.1 Gráfico de funciones reales

Comenzaremos graficando funciones simples para familiarizarnos con los comandos gráficos elementales, por lo cual vamos a graficar funciones conocidas.

```
### Gráfico de una recta y una parábola

x=seq(-3,3,0.5)
y=x^ 2
w=x+2
plot(x,y,col=2,type="p")

plot(x,y,col=2,type="o")
plot(x,y,col=2,type="l")

plot(x,y,col=2,lwd=2,type="l")

plot(x,y,col=2,lwd=2,type="l",ylim=c(-1,10))
abline(0,0)

segments(0,-1,0,10)

points(x,w,col=4,type="l",lwd=2)
text(locator(1),col=2,"parábola")

text(locator(1),col=4,"recta")
title("Gráfico de una recta y una parábola")
```

```
# asigna valores a la variable x
# define la función cuadrática
# define la función lineal
# grafica los puntos de la función y
# en rojo
# idem anterior y superpone una línea
# grafica la función y con una línea
# continua
# idem anterior pero con distinto
# grosor de línea
# cambia los límites del eje y
# dibuja una línea horizontal que pasa
# por el origen de coordenadas
# dibuja un segmento que une los puntos
# (0,-1) y (0,10) pasando por el origen
# superpone la función w en azul
# coloca un cartel en la posición
# indicada
# agrega un título al gráfico activo
```

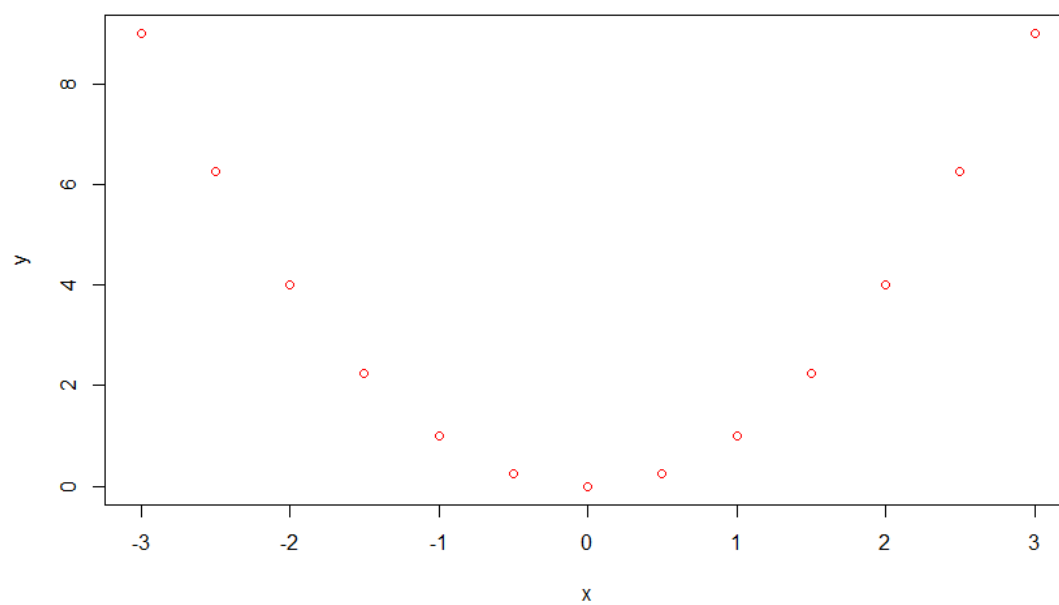


Figura 5.1: Ejemplo type="p"

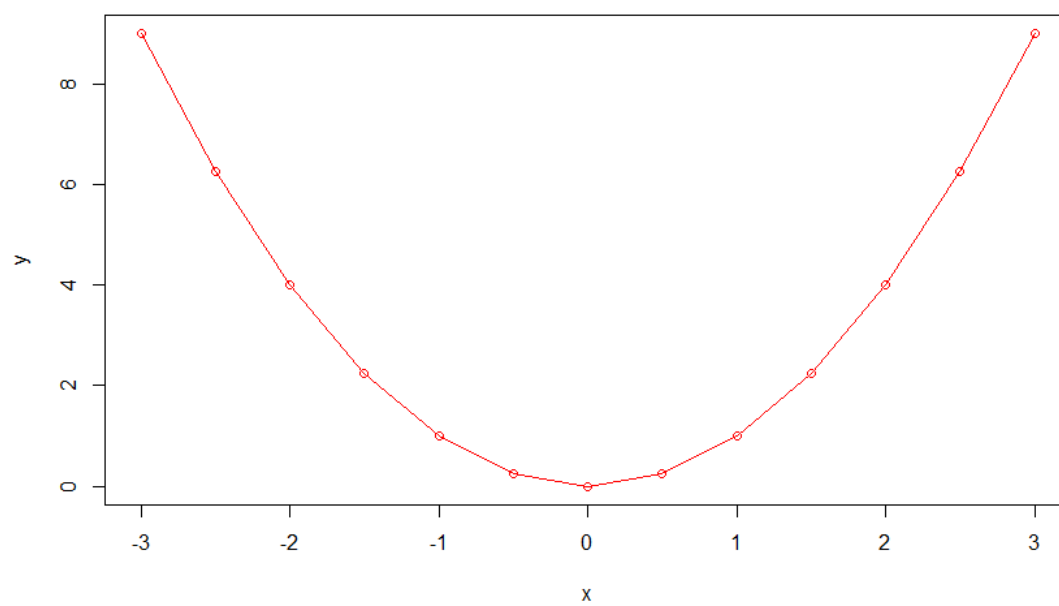


Figura 5.2: Ejemplo type="o"

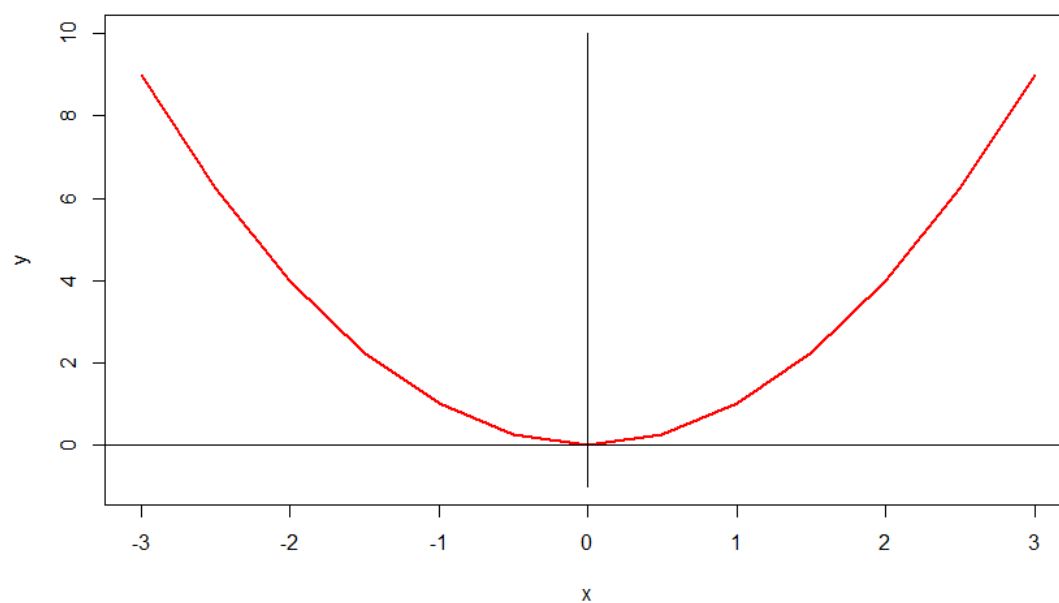


Figura 5.3: Ejemplo con ejes dibujados

Gráfico de una recta y una parábola

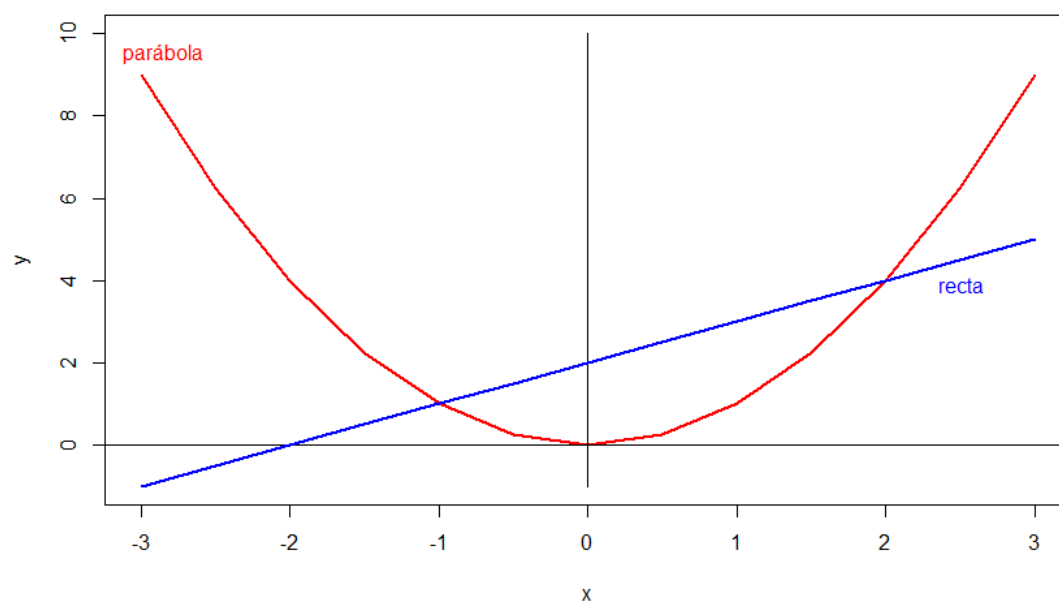


Figura 5.4: Gráfico final


```
### Gráfico de distintas fases

x=seq(-5,5,.1)
y=sin(x)
# define una curva senoidal
w=sin(x+pi/3)
# define una curva senoidal con el mismo período pero distinta fase
z=sin(x-pi/4)
plot(x,y,type="l",col="pink",axes=F)
# grafica la función y sin los ejes
plot(x,y,type="l",lwd=2.5,col="pink",axes=T)
# grafica la función y con los ejes
plot(x,y,type="l",lwd=2.5,col="pink",axes=T,xlab="eje x",ylab="eje y")
# pone nombre a los ejes
plot(x,y,type="l",lwd=2.5,col="pink",axes=T,xlab="eje x",ylab="eje y",
     main="Senoidales con diferentes fases")
# agrega título
points(x,w,type="l",lwd=2.5,col="green")
# superpone la función w
points(x,z,type="l",lwd=2.5,col="violet")
# uperpone la función z
a=expression(paste(alpha,"=",0))
# escribe letras griegas
b=expression(paste(alpha,"=",pi/3))
c=expression(paste(alpha,"=",pi/4))
legend.text=c(a,b,c)
legend("topright",legend.text,text.col=c("pink","green","violet"),text.width=0.8)
# coloca una leyenda
```

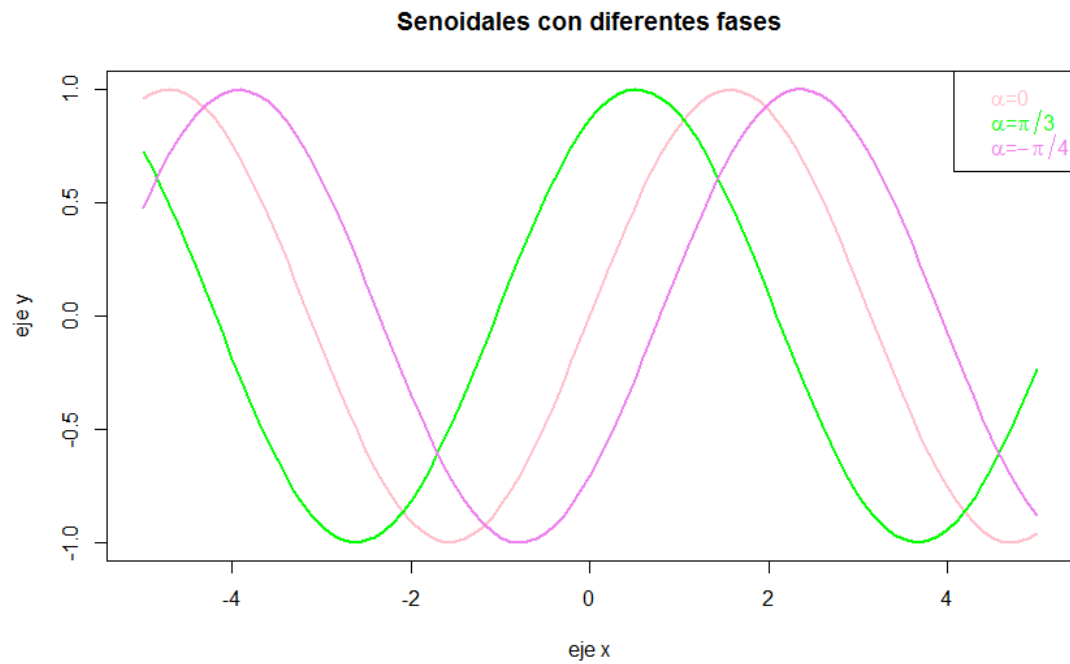


Figura 5.5: Ejemplo con leyendas

Todos los gráficos pueden requerir ampliaciones para poder ver con claridad las leyendas o formas de los mismos. Esto es posible mediante la opción *Zoom*, como se muestra en la siguiente imagen.

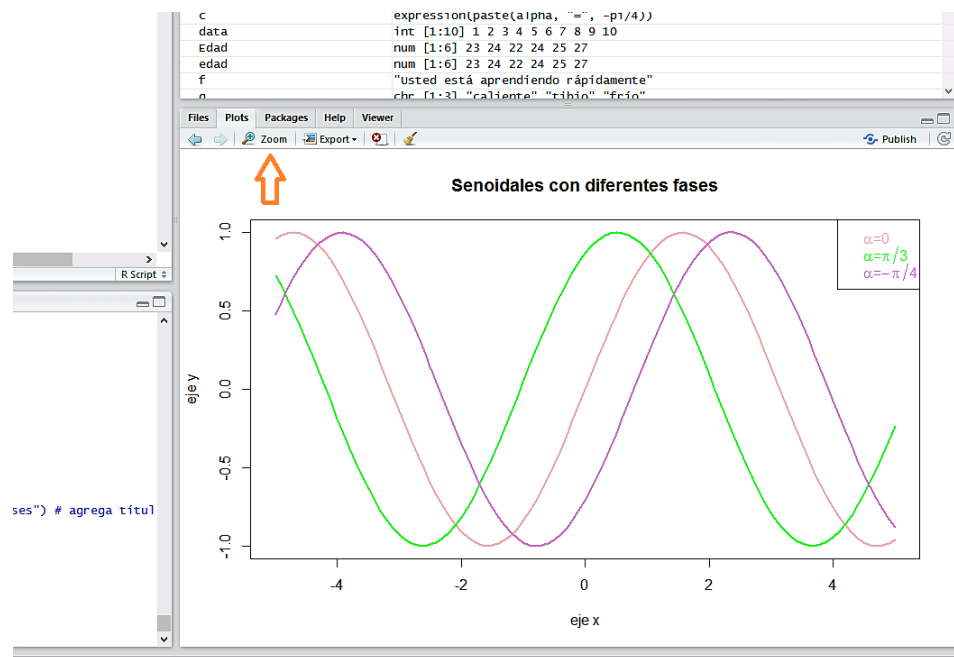


Figura 5.6: Opción de ampliación

```

### Gráfico de una parametrización

alfa=seq(0,2*pi,0.01)           # asigna valores al parámetro
x=cos(alfa)/(1+sin(alfa)**2)     # define la variable x en función
                                # del parámetro
y=cos(alfa)*sin(alfa)/(1+sin(alfa)**2) # define la variable y en función
                                # del parámetro
par(bg="thistle1")              # establece el color del fondo
plot(x,y,lwd=2,type="l",col="royalblue1") # grafica la curva
abline(0,0)
segments(0,-1,0,1)
rtx=seq(-1,1,0.1)*(x[472]-x[471])/0.01+x[471] # calcula una aproximación de la
rtty=seq(-1,1,0.1)*(y[472]-y[471])/0.01+y[471] # derivada cerca del origen
points(rtx,rtty,col="seagreen",type="l",lwd=2) # superpone la recta tangente
title("Lemniscata de Bernoulli")

```

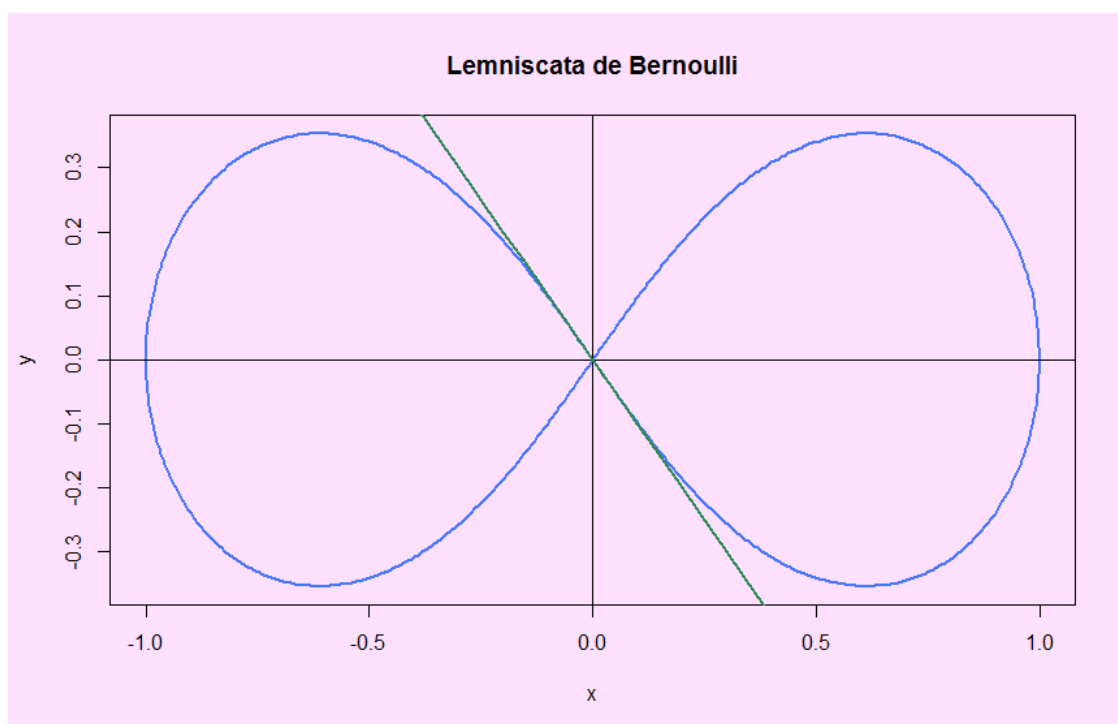


Figura 5.7: Gráfico de la lemniscata de Bernoulli

5.2 Gráficos en tres dimensiones o de campos escalares

A esta altura podemos observar la gran variedad de colores que ofrece R, para un listado de los mismos nos referimos a <http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf>.

```

par(bg="papayawhip")
fun1=function(x,y) {x^ 2 + y^ 2 }
# define un paraboloide circular
fun2=function(x,y) exp(-x^ 2 - y^ 2)
# define la función de la distribución normal bivariada con ro=0
fun3=function(x,y) x+y
# define un plano
x=seq(-3,3,0.1)
y=x
persp(x,y,outer(x,y,fun1), col="tomato3")
# grafica la función fun1
persp(x,y,outer(x,y,fun1), col="tomato3", phi = 30)
# cambia el ángulo de visión
persp(x,y,outer(x,y,fun1), col="tomato3", phi = 30, shade = 0.75)
# cammbia el sombreado
persp(x,y,outer(x,y,fun1), col="tomato3", phi = 30, shade = 0.75, xlab = "abscisas",
      ylab = "ordenadas", zlab ="cotas")
# pone nombre a los ejes
persp(x,y,outer(x,y,fun2),col="turquoise1", shade=0.25, xlab = "X", ylab = "Y",
      zlab ="Z")
# grafica la normal bivariada
persp(x,y,outer(x,y,fun3),phi=45, xlab = "X", shade = 0.25, ylab = "Y", zlab ="Z",
      col="palevioletred3")
# grafica el plano

```

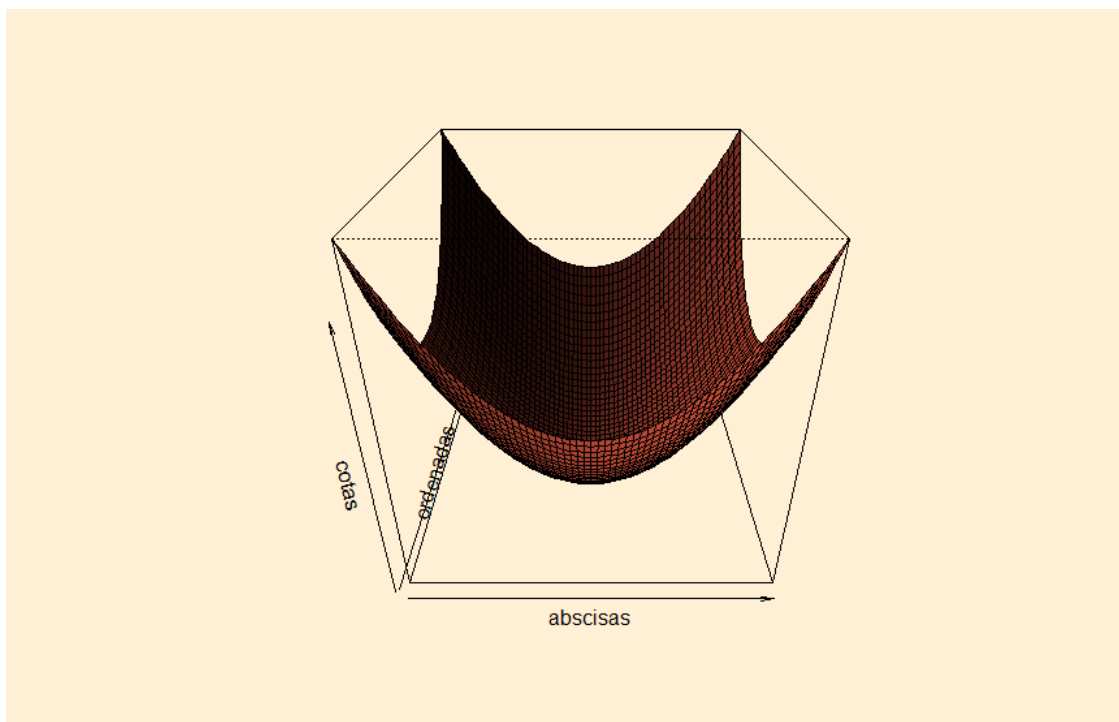


Figura 5.8: Gráfico de un paraboloide

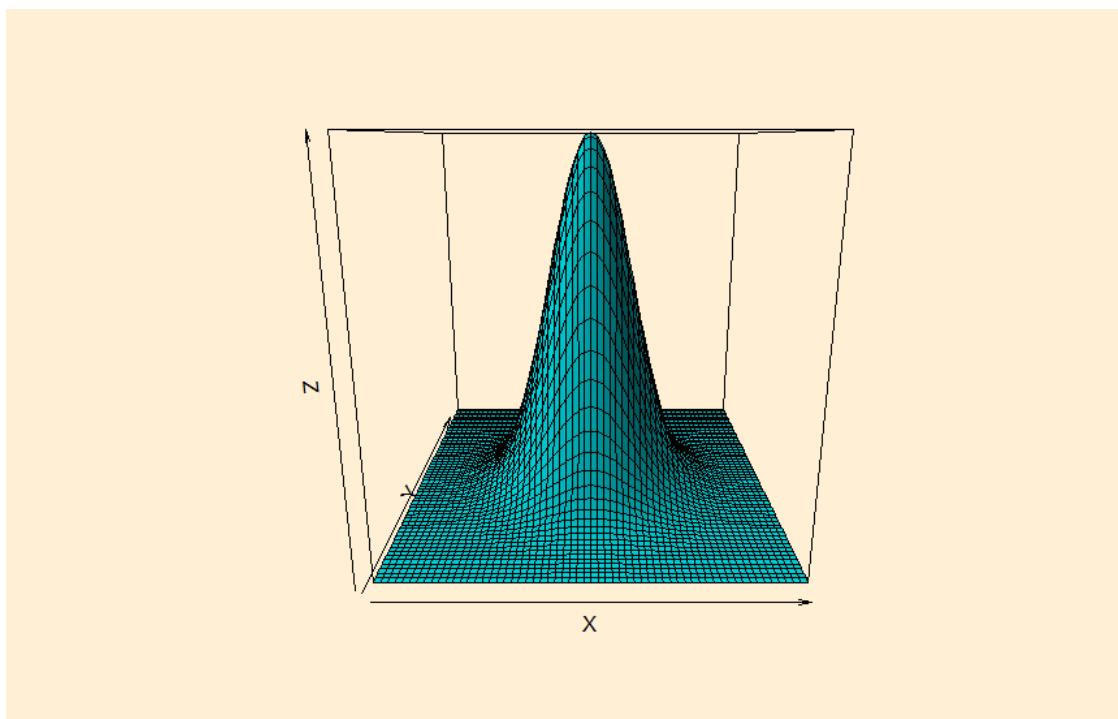


Figura 5.9: Gráfico de la normal bivariada con $\rho = 0$

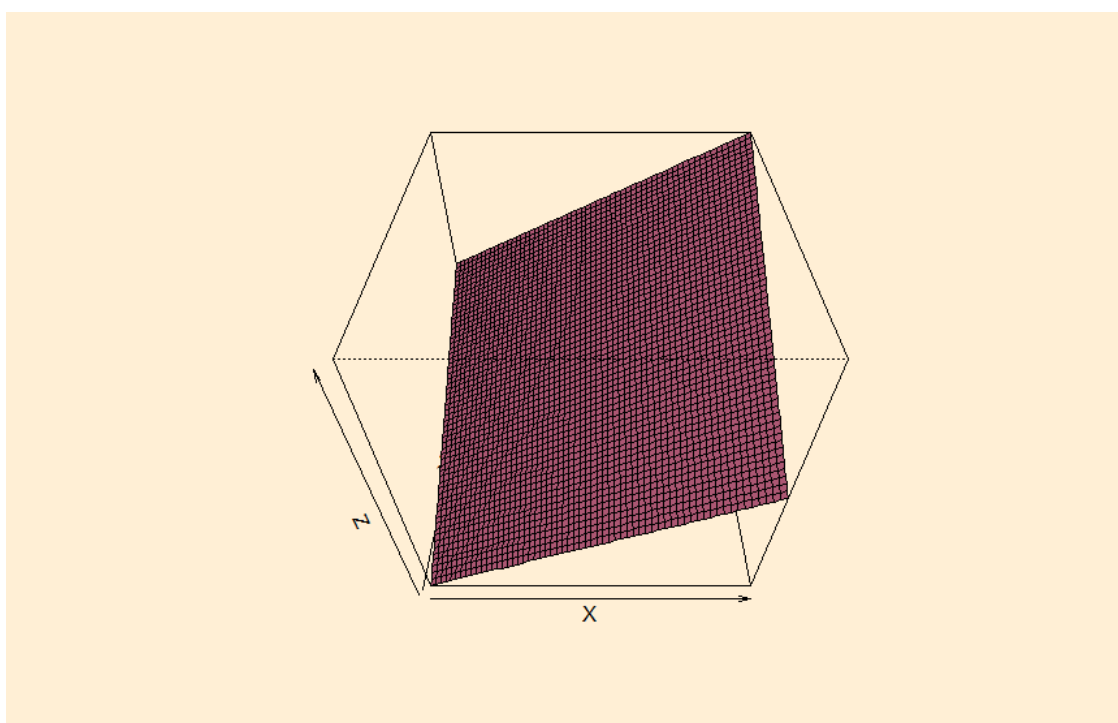


Figura 5.10: Gráfico de un plano

5.3 Gráficos con movimientos

Al realizar este tipo de gráficos, las reproducciones de los mismos podrán ser observadas en ventanas emergentes.

```
x=seq(-3,3,0.1)
y=x
z=x**2+y**2

plot3d(x,y,z,col="orangered")
play3d(spin3d(axis=c(1,0,0),rpm=7),duration=10)

play3d(spin3d(axis=c(0,1,0),rpm=7),duration=10)
play3d(spin3d(axis=c(0,0,1),rpm=7),duration=10)
```

definimos la tercera
coordenada como función
de las dos primeras
graficamos en 3D
rota sobre el eje x
indicando velocidad y duración
rota sobre el eje y
rota sobre el eje z

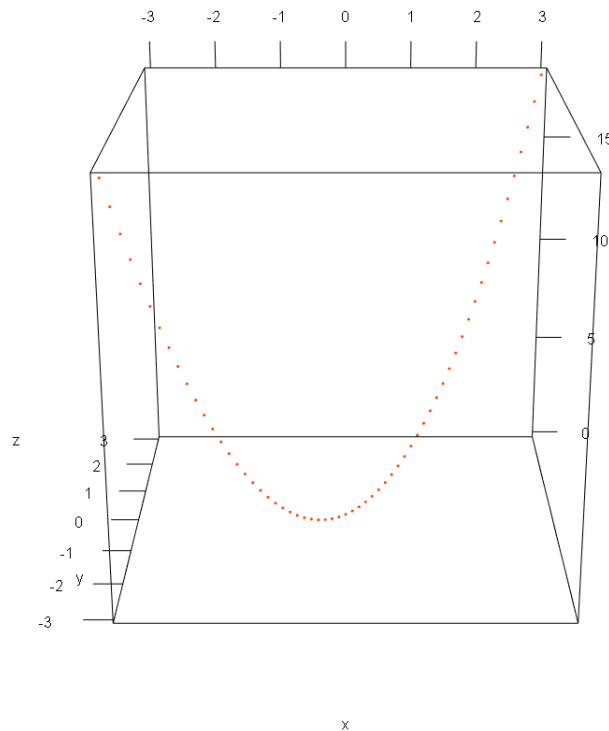


Figura 5.11: Gráfico en 3D al cual se le puede aplicar movimiento

5.4 Gráficos estadísticos

5.4.1 Gráficos circulares

Para este ejemplo volveremos a importar la base de datos *IMCinfantil* y la cargaremos en la base de la memoria activa, recordando las siguientes instrucciones.

```
IMCinfantil<-read.csv2("C:/Users/Usuario-PC/Dropbox/Curso R/Datos/IMCinfantil.csv")
attach(IMCinfantil)
```

```
frec.catpeso<-table(CatPeso)
# construye la distribución de frecuencias
pie(frec.catpeso)
# dibuja el diagrama circular
pie(frec.catpeso, col=rainbow(25))
# cambia la gama de colores
pie(frec.catpeso, col=rainbow(25), font=8)
# cambia el tipo de letra
pie(frec.catpeso, col=rainbow(25), font=8, cex=1.5)
# cambia el tamaño de letra
pie(frec.catpeso, col=rainbow(25), font=8, cex=1.5, radius=1)
# cambia el tamaño de la torta
pie(frec.catpeso, col=rainbow(25), font=8, cex=1.5, radius=1, border=F)
# quita el borde
pie(frec.catpeso, col=rainbow(25), font=8, cex=1.5, radius=1, border=F,
    main="Gráfico de Torta")
# pone nombre
```

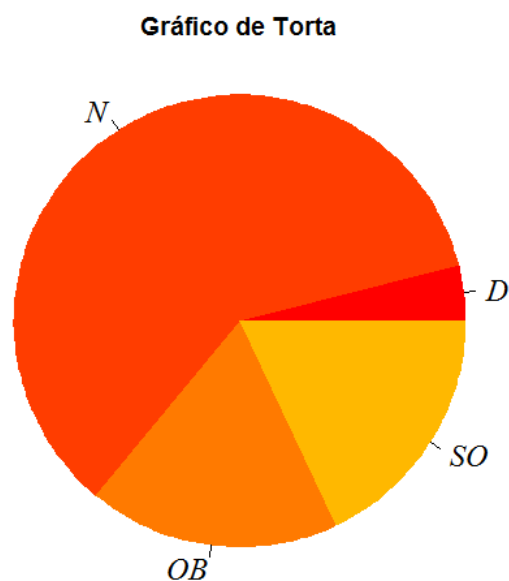


Figura 5.12: Gráfico circular de las categorías de peso de la base *IMCinfantil*

En el próximo ejemplo, veremos como etiquetar las categorías y como asignar distintas paletas de colores.

```
etiquetas<-c("Deficiente", "Normal", "Obeso", "Con sobrepeso")
# define etiquetas
pct<-round(frec.catpeso/sum(frec.catpeso)*100)
# calcula las frecuencias porcentuales
etiquetas<-paste(etiquetas,pct)
# agrega los porcentajes a las etiquetas
etiquetas<-paste(etiquetas,"%",sep="")
# agrega el símbolo % a los porcentajes
pie(frec.catpeso,labels =etiquetas,col=heat.colors(4,alpha=1))
# otra manera de asignar una paleta de colores
pie(frec.catpeso,labels =etiquetas,col=terrain.colors(4,alpha=1))
# otra manera de asignar una paleta de colores
pie(frec.catpeso,labels =etiquetas,col=topo.colors(4,alpha=1))
# otra manera de asignar una paleta de colores
pie(frec.catpeso,labels =etiquetas,col=cm.colors(4,alpha=1))
# otra manera de asignar una paleta de colores
pie(frec.catpeso,labels =etiquetas,col=cm.colors(4,alpha=1),
    main="Diagrama circular con etiquetas")
```

Diagrama circular con etiquetas

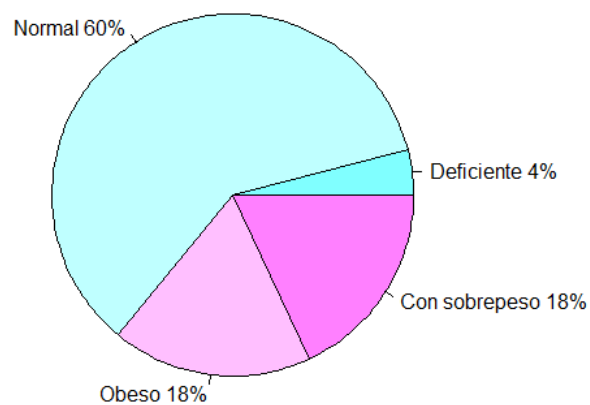


Figura 5.13: Gráfico circular con etiquetas según porcentajes

Finalmente, vamos a darle volumen y sombra a esta torta.


```
pie3D(frec.catpeso)
# grafica una torta con volumen
pie3D(frec.catpeso, labels=etiquetas)
pie3D(frec.catpeso, labels=etiquetas, explode=0.1)
# separa los sectores
pie3D(frec.catpeso, labels=etiquetas, explode=0.1, labelcex=0.9)
# cambia el tamaño de las etiquetas
pie3D(frec.catpeso, labels=etiquetas, explode=0.1, labelcex=0.9, radius=1.5)
pie3D(frec.catpeso, labels=etiquetas, explode=0.1, labelcex=0.9, radius=1.5, height=0.2)
# cambia el alto de la torta
pie3D(frec.catpeso, labels=etiquetas, explode=0.1, labelcex=0.9, radius=1.5, height=0.2,
      shade=0.6)
# sombrea
pie3D(frec.catpeso, labels=etiquetas, explode=0.1, labelcex=0.9, radius=1.5, height=0.2,
      shade=0.6, col=terrain.colors(4:8, alpha=1))
```

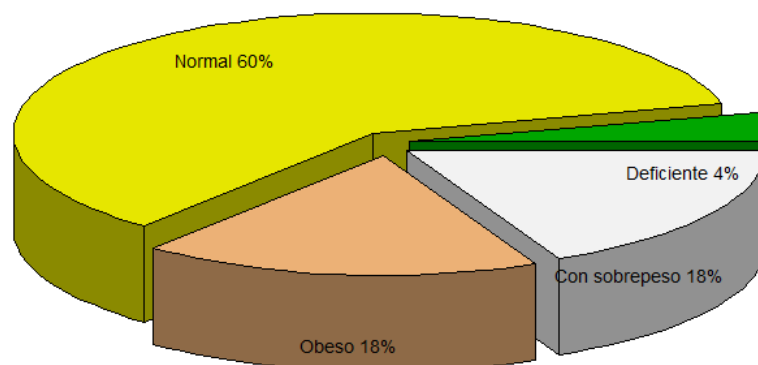


Figura 5.14: Ejemplo de un gráfico circular en 3D

5.4.2 Diagramas de barras

Los gráficos de barras son adecuados cuando queremos representar una variable categórica. Este tipo de gráficos también se adaptan para representar la distribución conjunta de dos variables categóricas. La posición de las barras puede ser tanto vertical como horizontal y las partes correspondientes a la segunda variable categórica de cada barra, pueden representarse en forma superpuesta o adyacente.

```

par(bg="mistyrose")
barplot(table(CatPeso),main="Categorías de Peso",col="mediumpurple1")
# hace un gráfico de barras simple
barplot(table(SEXO,CatPeso))
# hace un gráfico de barras superpuesto
barplot(table(SEXO,CatPeso)[,c(1,2,4,3)])
# cambia el orden de las barras
barplot(table(SEXO,CatPeso)[,c(1,2,4,3)],col=rainbow(11),
        main="Categorías de Peso según Sexo")
legend("topright",cex=1,title="Sexo",c("F","M"),fill=rainbow(11),horiz=T)
# asigna leyendas en posición horizontal
tabla<-table(SEXO,CatPeso)
barplot(tabla,main="Gráfico de barras",horiz=TRUE,col=c("olivedrab1","springgreen1"))
# hace un gráfico de barras horizontales
legend("topright",cex=1,title="Sexo",c("F","M"),fill=c("olivedrab1","springgreen1"),
      horiz=F)
# asigna leyendas en posición vertical
barplot(tabla,main="Gráfico de barras",beside=TRUE,col=c("tan1","mistyrose4"))
# hace un gráfico de barras adyacentes
legend("topleft",cex=1,title="Sexo",c("F","M"),fill=c("tan1","mistyrose4"),horiz=F)
# cambia la ubicación de las leyendas

```

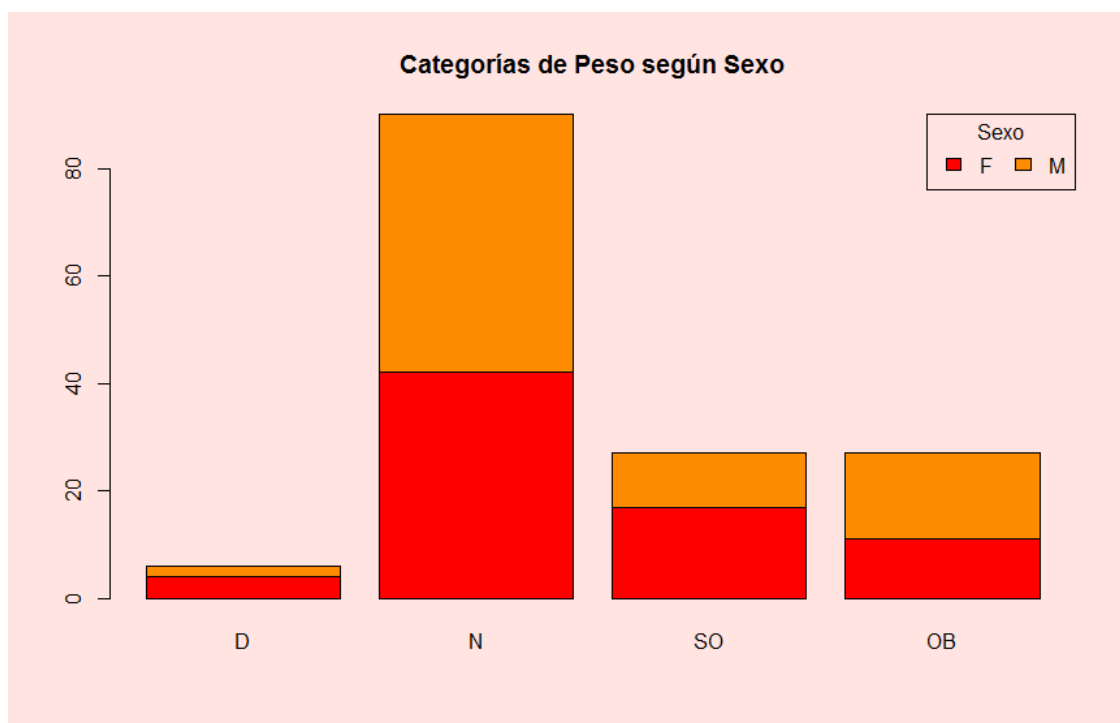
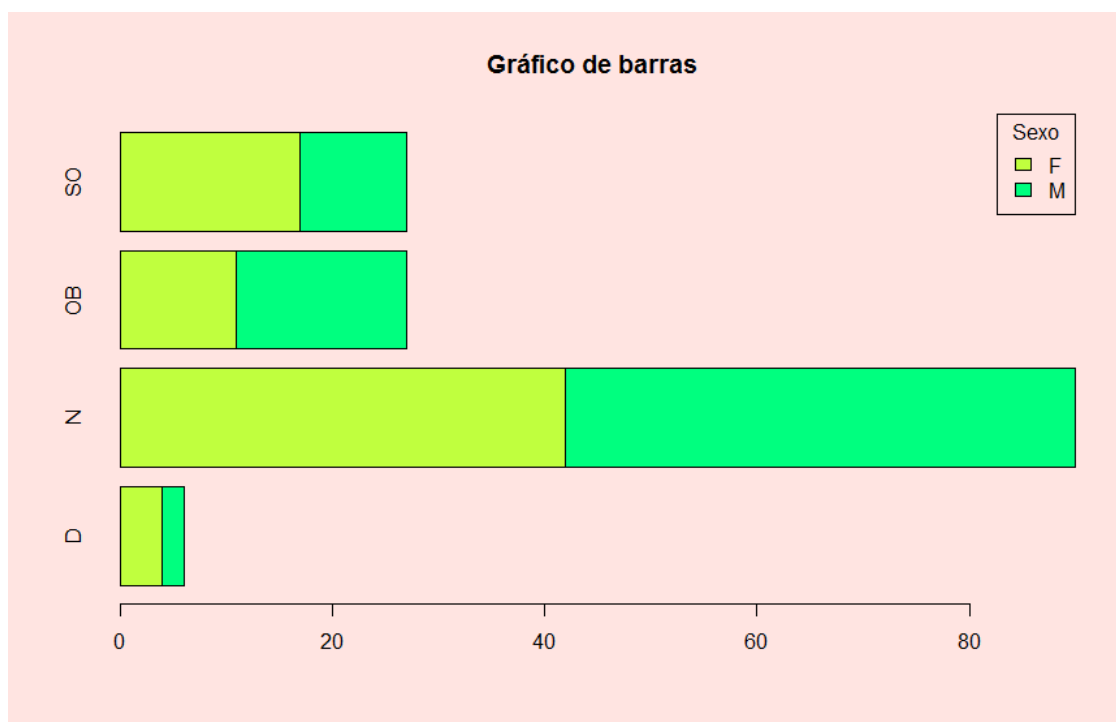
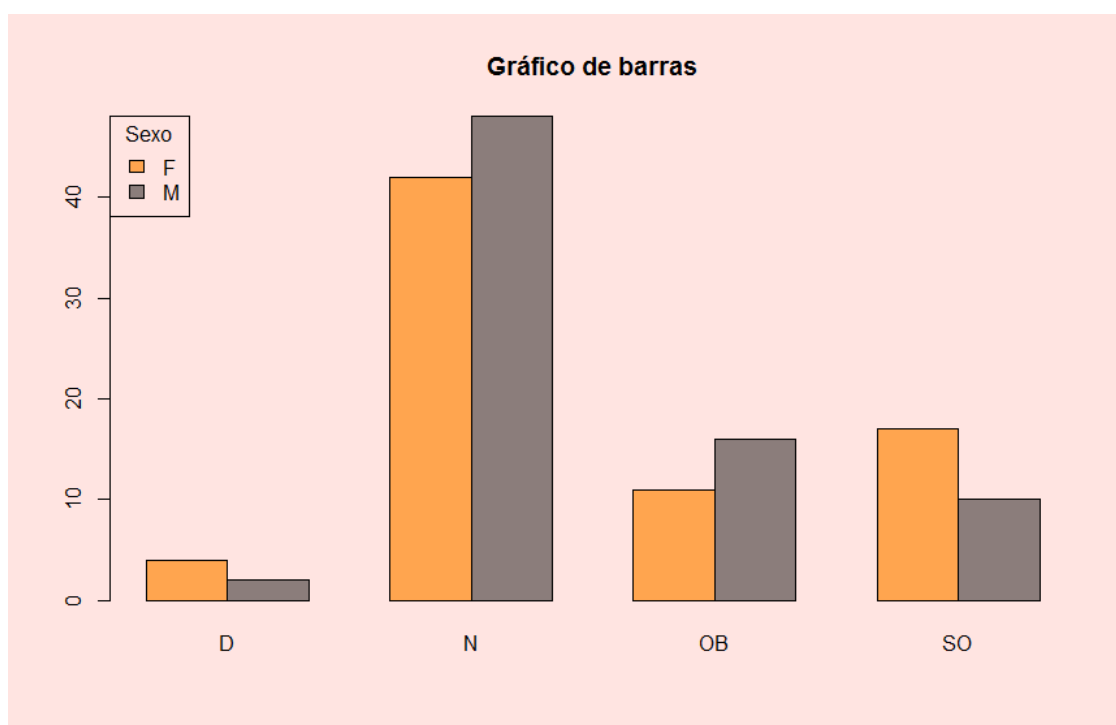


Figura 5.15: Gráfico de barras superpuestas

**Figura 5.16:** Gráfico de barras horizontales**Figura 5.17:** Gráfico de barras adyacentes

5.4.3 Gráficos de mosaicos

Una manera alternativa de representar la distribución conjunta de dos variables categóricas o una tabla de contingencia, es mediante los gráficos de mosaicos. La diferencia con los gráficos de barras es que, en el gráfico de mosaicos, la superficie de cada sector es proporcional a su frecuencia absoluta en la tabla, es por esto que pueden aparecer columnas de distinto grosor.

```
tabla2=table(EDAD,CatPeso)
par(bg="lightcyan")
mosaicplot(tabla2)
# hace un gráfico de mosaicos simple
mosaicplot(tabla2[,c(1,2,4,3)],col=terrain.colors(7:11),main="Gráfico de Mosaicos",
  ylab="Categoría de Peso",xlab="Edad",cex=0.8)
# este gráfico permite visualizar una tabla de contingencia
```

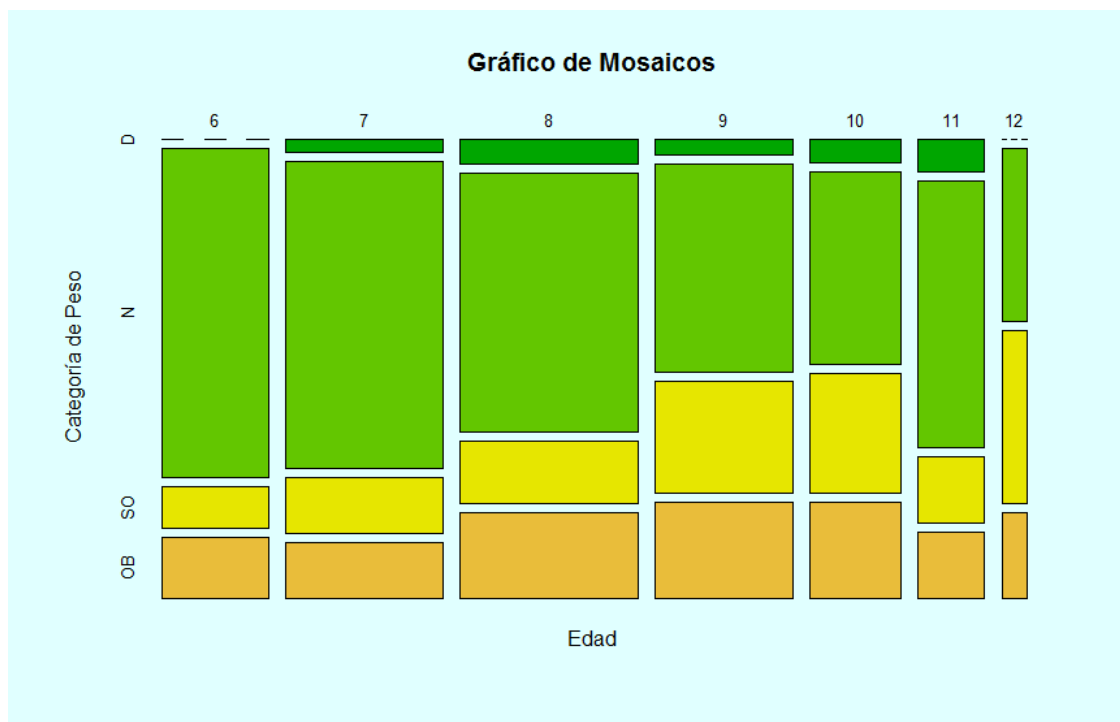


Figura 5.18: Gráfico de mosaicos

5.4.4 Gráficos de bastones

El patrón de comportamiento de una variable se denomina distribución. La distribución registra los valores que toma la variable de interés y cuán frecuentemente lo hace para cada uno de ellos. Se llama distribución de frecuencias a la asignación de la frecuencia observada (absoluta o relativa) para cada valor de la variable de interés y, en general, se presenta mediante una tabla.

Para representar una distribución de frecuencias o bien la función de cuantía de una variable aleatoria discreta se utiliza un gráfico de bastones.

```
Modelos<-2010:2016                                     # ingresa los
                                                         modelos de autos
Ventas<-c(2,3,7,4,9,0,5)                                # ingresa las
                                                         frecuencias de
                                                         ventas

par(bg="snow2")
plot(Modelos,Ventas)                                     # grafica
                                                         puntos
plot(Modelos,Ventas,type="h")                           # grafica
                                                         bastones
plot(Modelos,Ventas,type="h",lty="twodash")             # cambia estilo
plot(Modelos,Ventas,type="h",lty="twodash",lwd=4)       # cambia grosor
plot(Modelos,Ventas,type="h",lty="twodash",lwd=4,col=heat.colors(9)) # cambia color
title("Ventas mensuales de una Agencia Chevrolet")
```

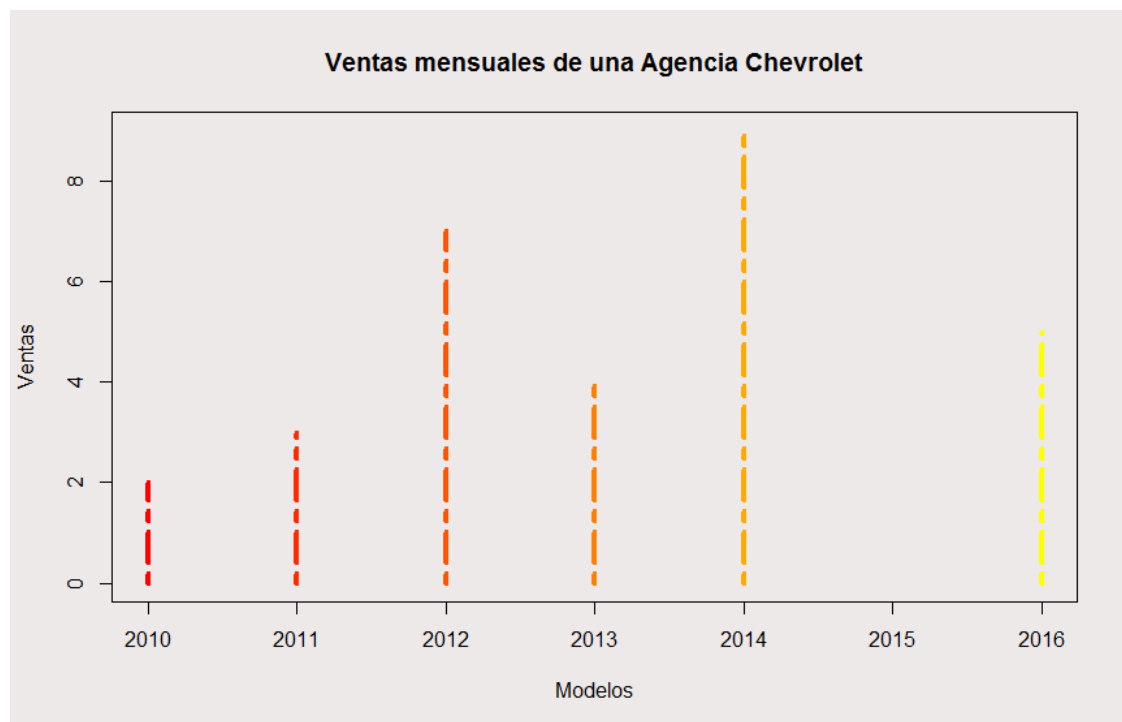


Figura 5.19: Gráfico de bastones

Otra forma de lograr este gráfico es agregando segmentos sobre un gráfico de puntos.

```
plot(Modelos,Ventas)
segments(2010,0,2010,2)                                # agrega un segmento del punto
                                                         (2010,0) al punto (2010,2)
                                                         # estilo rayado
segments(2011,0,2011,3,lwd=3,lty="dotted",col=2)       # estilo punteado
segments(2012,0,2012,7,lwd=3,lty="solid",col=3)         # estilo sólido
segments(2013,0,2013,4,lwd=3,lty="dotdash",col=4)       # alterna estilos
segments(2014,0,2014,9,lwd=3,lty="twodash",col=5)       # estilo doble rayado
segments(2016,0,2016,5,lwd=3,lty="longdash",col=6)      # estilo rayado largo
```

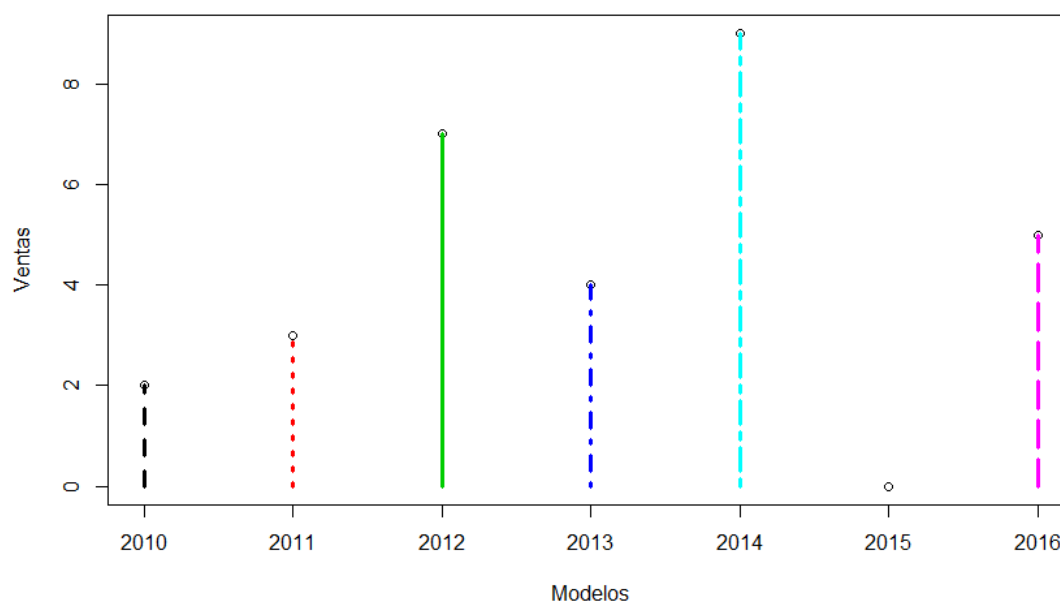


Figura 5.20: Estilos de bastones

5.4.5 Diagramas de tallo-hoja

El diagrama de tallo-hoja es similar a un histograma en donde se pueden apreciar los valores de las observaciones. Se denomina profundidad del tallo a la cantidad de hojas que cuelgan de él. Existe un parámetro de escala que permite controlar la profundidad de los tallos.

```
datos=PESO
stem(datos,scale=0.5) # da un histograma en el que se pueden apreciar los valores
stem(datos,scale=1)  # cambia la escala
```

The decimal point is 1 digit(s) to the right of the |

```
1 | 999
2 | 0000011111222222222233333344444444444444445556666666677777788888999
3 | 000001112222233333334444555556777888888888
4 | 0000001111233344566677899
5 | 0111222355789
6 | 00
7 |
8 | 1
```

Figura 5.21: Gráfico de tallo-hoja

```

The decimal point is 1 digit(s) to the right of the |
1 | 999
2 | 0000011111222222222333334444444444444444
2 | 555666666667777788888999
3 | 0000011122222333333334444
3 | 55556777888888888
4 | 0000001111233344
4 | 566677899
5 | 01112223
5 | 55789
6 | 00
6 |
7 |
7 |
8 | 1

```

Figura 5.22: Gráfico de tallo-hoja con diferente escala

5.4.6 Diagrama de dispersión en dos y tres variables

El objetivo de este tipo de gráficos es visualizar la distribución conjunta de dos o tres variables. En ocasiones resulta interesante ver si distintos grupos e un conjunto de datos tienen patrones de dispersión diferentes.

Para el siguiente ejemplo, utilizaremos los datos de la tabla ??.

```

avispas<-read.csv("C:/Users/Usuario-PC/Dropbox/Curso R/Datos/avispas.csv", sep=";")
attach(avispas)
plot(X1[Grupo=="Chaq.amar"],X2[Grupo=="Chaq.amar"])
# grafica un diagrama de dispersión
plot(X1[Grupo=="Chaq.amar"],X2[Grupo=="Chaq.amar"],pch=8)
# cambia estilo de puntos
plot(X1[Grupo=="Chaq.amar"],X2[Grupo=="Chaq.amar"],pch=8,col=3)
# cambia el color
plot(X1[Grupo=="Chaq.amar"],X2[Grupo=="Chaq.amar"],pch=8,col=3,
     xlab="Longitud de la antena",ylab="Longitud del ala")
# pone nombre a los ejes
plot(X1[Grupo=="Chaq.amar"],X2[Grupo=="Chaq.amar"],pch=8,col=3,
     xlab="Longitud de la antena",ylab="Longitud del ala",ylim=c(1.5,2.2),xlim=c(1,1.7))
# cambia escala a los ejes
points(X1[Grupo=="Neg.peq"],X2[Grupo=="Neg.peq"])
# agrega la otra categoría al gráfico de dispersión
points(X1[Grupo=="Neg.peq"],X2[Grupo=="Neg.peq"],pch=13,col="orange")
title("Diagrama de dispersión por especie")

```

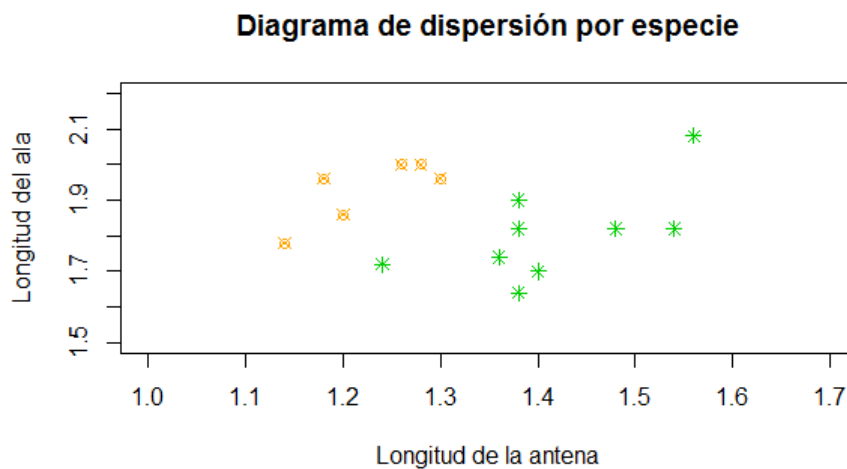


Figura 5.23: Gráfico de dispersión de dos variables

Para más opciones sobre pch se sugiere visitar <http://www.endmemo.com/program/R/pchsymbols.php>. En el ejemplo que sigue volveremos a usar los datos de la tabla 8.1 y sus continuaciones.

```
attach(IMCinfantil)
base.niños=data.frame(EDAD,PESO,TALLA,IMC,CC)
# arma una sub-base con las variables numéricas de IMCinfantil
pairs(base.niños)
# representa todos los diagramas de dispersión de a pares
pairs(base.niños,col=rainbow(dim(base.niños)[2]))
# cambia color
```

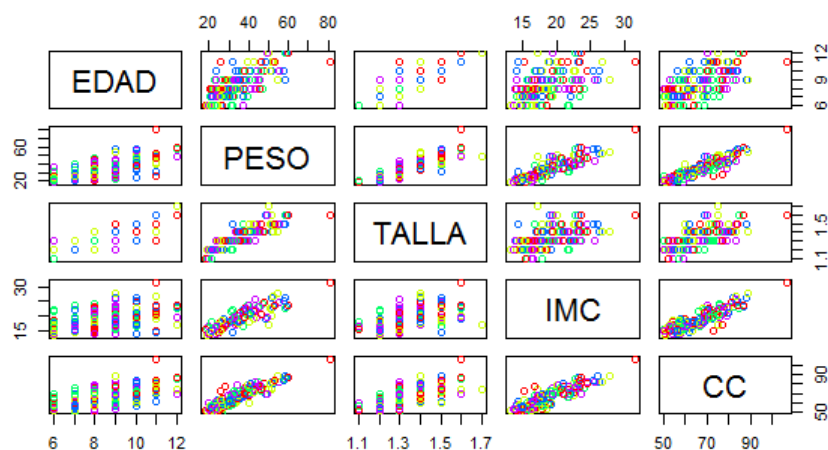


Figura 5.24: Gráfico de dispersión de dos variables

5.4.7 Gráfico de series de tiempo

Una serie temporal o cronológica es una secuencia de observaciones numéricas, medidas en determinados momentos y ordenadas cronológicamente. Los datos pueden estar espaciados a intervalos iguales o desiguales. En este caso, utilizaremos series de tiempo con observaciones equiespaciadas.

```
par(bg="darksalmon")
observaciones=runif(60,2,5)
# genera 60 observaciones de una uniforme en el intervalo (2,5)
ts.plot(observaciones,col="darkred",main="Serie de Tiempo")
# graficamos una serie de tiempo
```

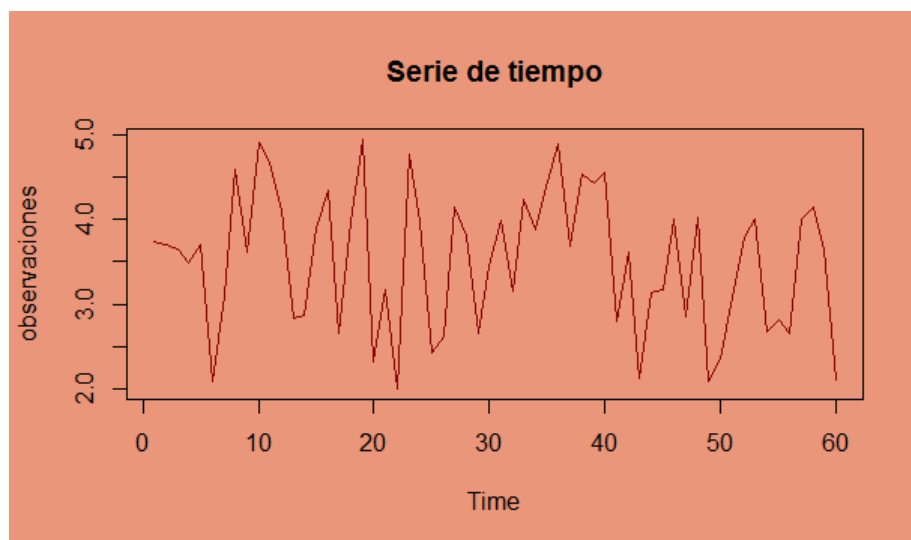


Figura 5.25: Gráfico de una serie de tiempo

En el siguiente ejemplo graficaremos varias series de tiempo en simultáneo, utilizando los datos de la tablas 8.6 y 8.7.

```
par(bg="linen")
empleos<-read.csv("C:/Users/Usuario-PC/Dropbox/Curso R/Datos/empleos.csv", sep=";")
emp=data.frame(empleos)
attach(emp)
com=ts(Comercio)
serv= ts(Servicios)
prod=ts(Producción)
ts.plot(com,serv,prod,lwd=2,col=c("sienna1","palegreen4","lightslateblue"),
        main="Evolución del empleo")
text(locator(1),"Comercio",col="sienna1")
text(locator(1),"Servicio",col="palegreen4")
text(locator(1),"Producción",col="lightslateblue")
```

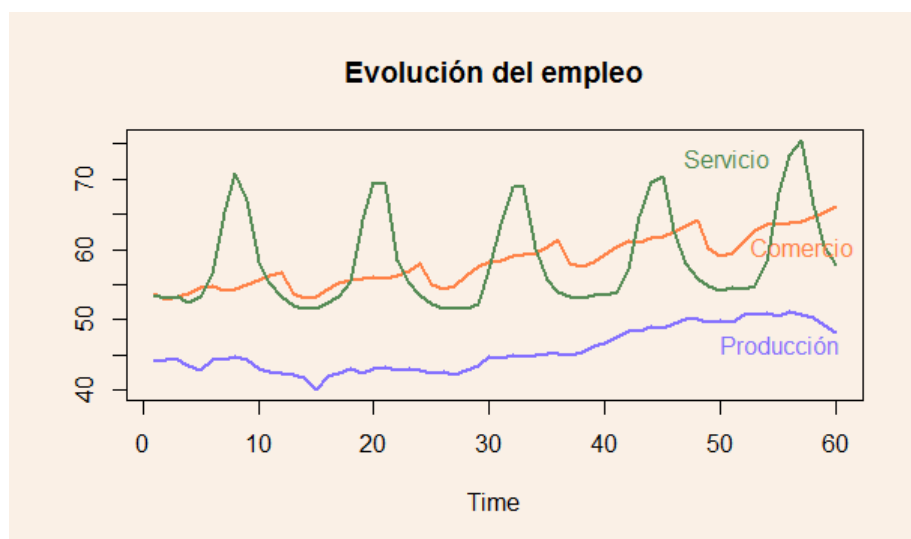


Figura 5.26: Gráfico de series de tiempo en simultáneo

5.4.8 Histogramas

Un histograma es un gráfico de barras donde la superficie de cada barra es proporcional a la frecuencia del correspondiente intervalo de la variable.

Su utilidad principal consiste en brindar una “primera aproximación visual” de la distribución de la población, de modo tal que permite apreciar la presencia de alguna tendencia. Además, en un histograma, también se pueden apreciar la simetría, la dispersión y el rango de la variable.

Si un histograma es de densidad, el área de todos los rectángulos es unitaria.

Este tipo de gráfico resulta de gran utilidad cuando se desea comparar el comportamiento de una misma variable en dos o más sub-poblaciones.

Existen diferentes sugerencias relativas al ancho de las clases; es decir, a las bases de los rectángulos. Vamos a ejemplificar con dos de ellas.

```
attach(IMCinfantil)

par(bg="oldlace")
hist(PESO)
# grafica el histograma de los pesos de todos los niños
hist(PESO,col="maroon1")
# rellena las barras con color
hist(PESO,col="maroon1",density=18)
# rellena las barras con rayas
hist(PESO,col="maroon1",density=18,angle=70)
# cambia la inclinación del rayado
hist(PESO,col="maroon1",density=18,border="blueviolet")
# cambia el color de los bordes
hist(PESO,col="maroon1",density=18,border="blueviolet",main="Histograma",
      ylab="Frecuencia")

R=quantile(PESO,0.75)-quantile(PESO,0.25)
# calcula el rango intercuartil
n=length(PESO)
# guarda la cantidad de observaciones
h.FD=2*R*n^(-1/3)
# sugerencia de Freedman-Diaconis para el ancho de clase
h.Scott=3.39*sd(PESO)*n^(-1/3)
# sugerencia de Scott para el ancho de clase
primero=floor(min(PESO))-1
# guarda primer valor de la grilla
ultimo=ceiling(max(PESO))+3
# guarda último valor de la grilla
grilla.FD=seq(primero,ultimo,h.FD)
# defino primer valor de la grilla de Freedman-Diaconis
grilla.Scott=seq(primero,ultimo,h.Scott)
# defino primer valor de la grilla de Scott

hist(PESO,breaks=grilla.FD)
# cambia el ancho de las columnas
hist(PESO,breaks=grilla.FD,col=2:8,main="Histograma de Freedman- Diaconis",
      ylab="Frecuencia")
hist(PESO,breaks=grilla.Scott,col=22:28,main="Histograma de Scott",ylab="Frecuencia")
```

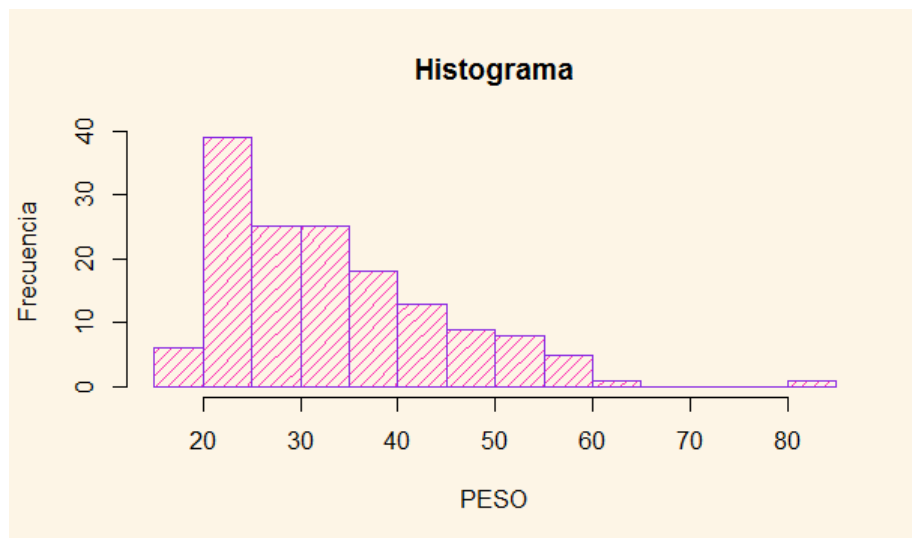


Figura 5.27: Gráfico de un histograma

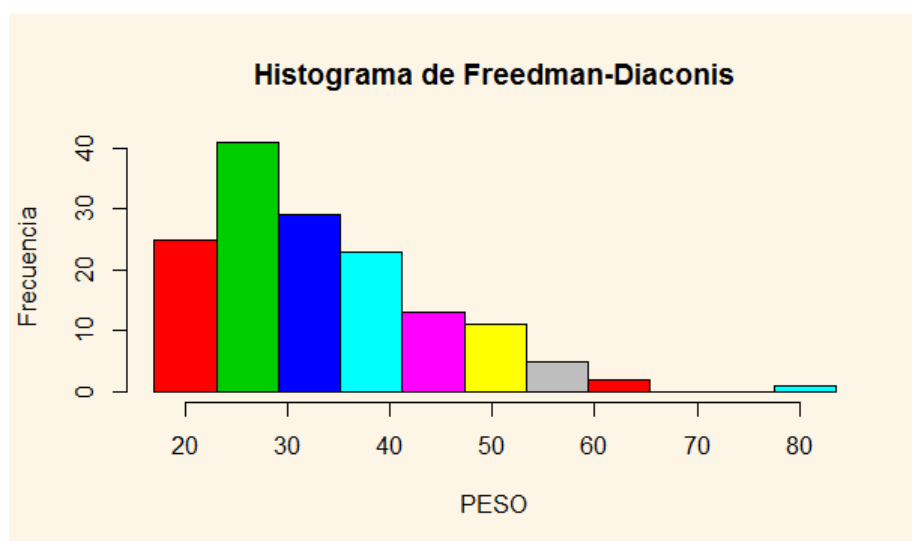


Figura 5.28: Gráfico de un histograma aplicando la sugerencia de Freedma-Diaconis

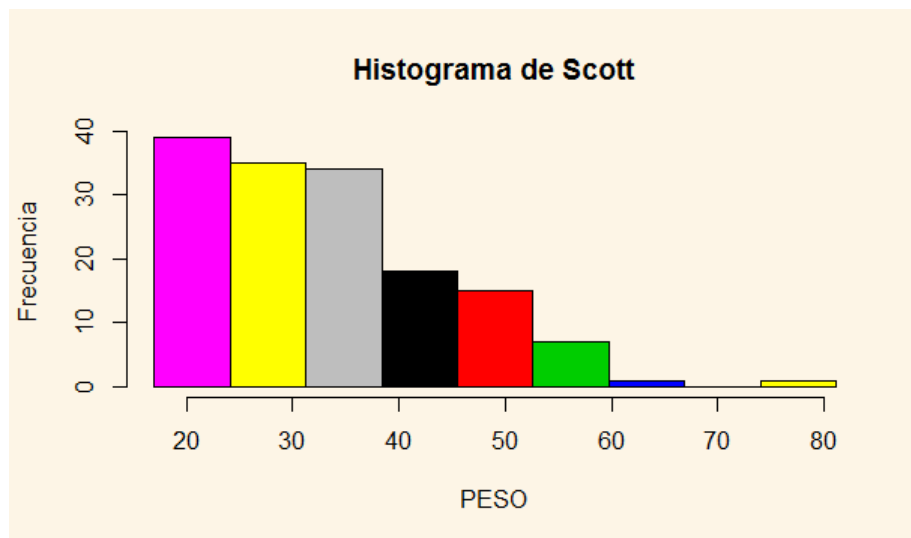


Figura 5.29: Gráfico de un histograma aplicando la sugerencia de Scott

5.4.9 Polígonos de frecuencia

```
a=length(grilla.FD)
pto.medio=rep(0,a-1)
# inicia un vector
for (i in 1:length(grilla.FD)-1)
{pto.medio[i]=(grilla.FD[i]+grilla.FD[i+1])/2}
# calcula los puntos medios de los intervalos
alt.dens=hist(PESO,breaks=grilla.FD,plot=F)$counts
# calcula la altura correspondiente a cada punto medio
par(bg="blanchedalmond")
hist(PESO,breaks=grilla.FD,col=heat.colors(a-1,alpha=1),main="Polígono de
frecuencia usando Freedman-Diaconis",ylab="Frecuencia")
points(pto.medio,alt.dens,type="l",lwd=2)
# superpone el polígono de frecuencias al histograma

b=length(grilla.Scott)
pto.medio=rep(0,b-1)
for (i in 1:length(grilla.Scott)-1)
{pto.medio[i]=(grilla.Scott[i]+grilla.Scott[i+1])/2}
alt.dens=hist(PESO,breaks=grilla.Scott,plot=F)$counts
par(bg="blanchedalmond")
hist(PESO,breaks=grilla.Scott,col=heat.colors(b-1,alpha=1),main="Polígono de
frecuencia usando Scott",ylab="Frecuencia")
points(pto.medio,alt.dens,type="l",lwd=2)
```

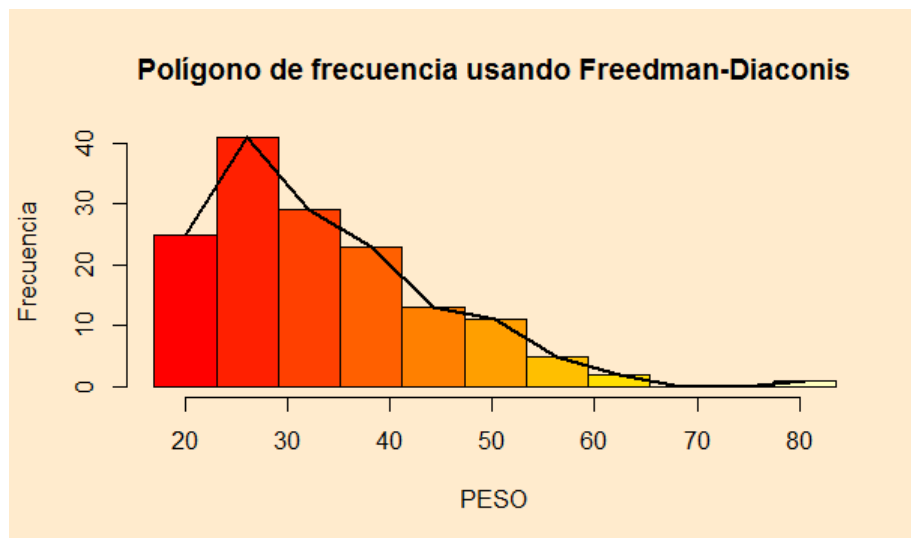


Figura 5.30: Ejemplo de un polígono de frecuencias aplicando la sugerencia de Freedman-Diaconis

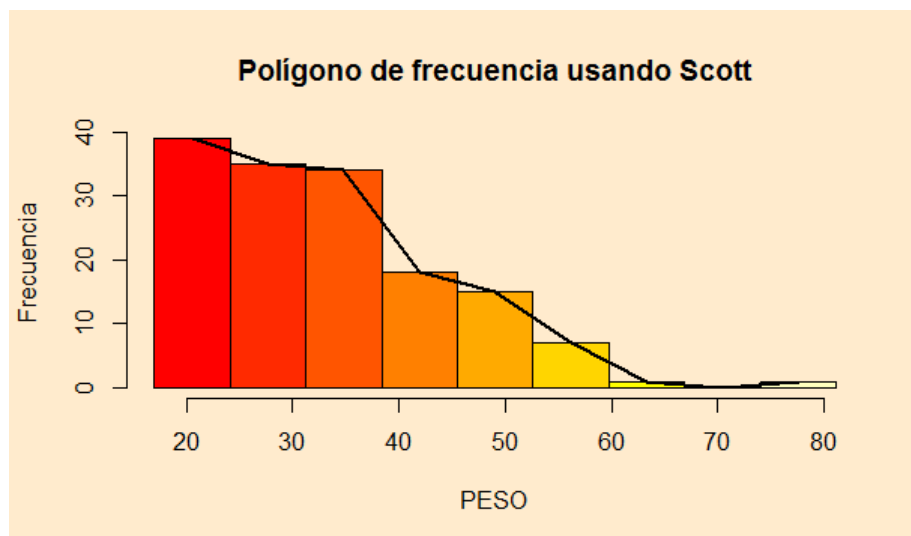


Figura 5.31: Ejemplo de un polígono de frecuencias aplicando la sugerencia de Scott

5.4.10 Curva de estimación de densidades

En un histograma de áreas, el área total es igual a 1. Para lograr esto, la altura del rectángulo correspondiente a cada intervalo se corresponde con la frecuencia relativa observada para dicho intervalo dividida por la amplitud de la clase (longitud de la base del rectángulo). Si la variable representada es continua, reduciendo la longitud de los intervalos de clase, pero manteniendo el área igual a 1, a medida que aumenta la cantidad de intervalos se obtiene cada vez un histograma más suave. La curva límite que une los puntos medios de las bases superiores de estos rectángulos se denomina curva de densidad de la variable.

```
par(bg="white")
dens=density(PESO)
# Kernel density estimation, es una manera no paramétrica de estimar la función de
# densidad de una variable aleatoria
plot(dens,main="Densidad de Peso",xlab="Peso",ylab="Densidad")
# grafica la estimación de la densidad de la variable PESO
polygon(dens,lwd=2,col="khaki1",border="khaki4",main="Densidad de Peso")
# cambia colores de relleno y borde

hist(PESO,col=cm.colors(8,alpha=1),probability=T,breaks=grilla.Scott,main="Suavizado
normal",ylab="Densidad")
# histograma de densidad
xfit=seq(min(PESO),max(PESO),length=40)
# arma una grilla de valores de datos
yfit=dnorm(xfit,mean=mean(PESO),sd=sd(PESO))
# realiza un suavizado normal de datos
lines(xfit,yfit,col="dodgerblue",lwd=2)
# superpone el suavizado al histograma
```

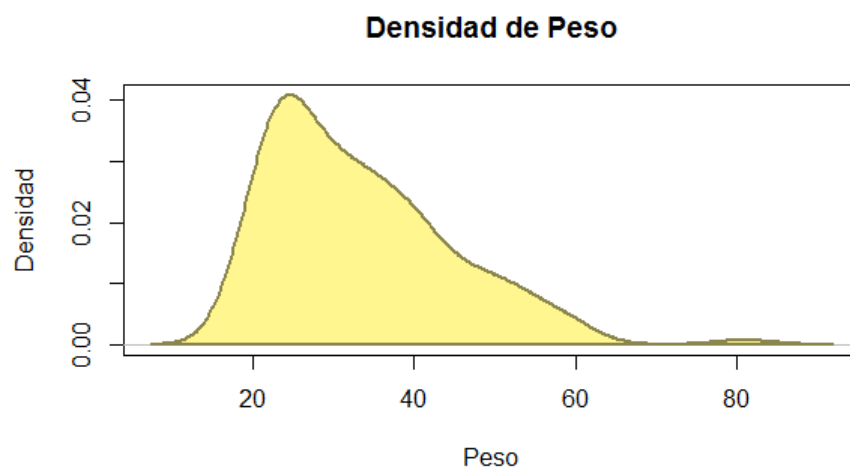


Figura 5.32: Gráfico de la estimación de una función de densidad

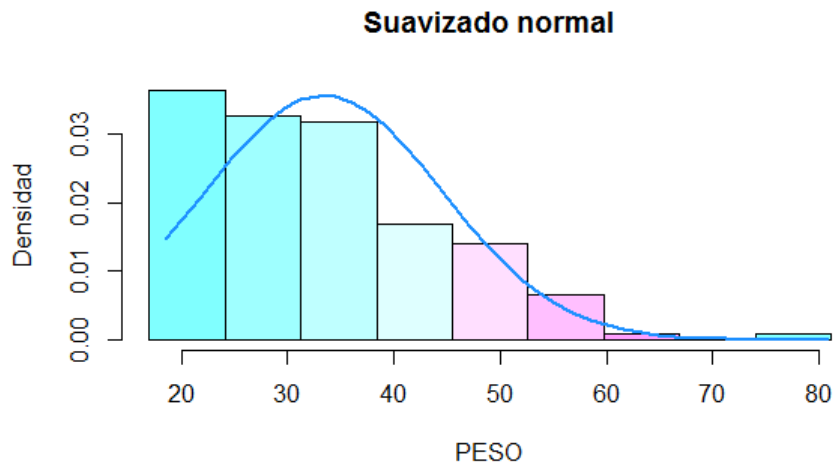


Figura 5.33: Superposición de un suavizado normal a un histograma

5.4.11 Función de distribución empírica

En el caso de variables numéricas, la función de distribución empírica de un conjunto de observaciones indica, para cada valor observado, la proporción de observaciones iguales o menores a él. Es importante destacar que, si bien una variable puede ser continua, la distribución empírica asociada a la misma es similar a la de una variable discreta, puesto que corresponde a una muestra finita.

```
attach(avispas)
par(mfrow=c(1,2))
# dividimos el área de gráficos en dos columnas
plot.ecdf(X1[Grupo=="Chaq.amar"],col="magenta",main="FDE Chaq.amar",ylab="F(x)")
# dibuja la función de distribución empírica
plot.ecdf(X1[Grupo=="Neg.peq"],col="chartreuse1",main="FDE Neg.peq",
  ylab="F(x)")

par(mfrow=c(1,1))
# unifica la pantalla de gráficos
n=length(X1)
plot(stepfun(1:(n-1),sort(X1)),main="Función escalonada")
# otra manera de definir y graficar la función acumulada
plot(stepfun(1:(n-1),sort(X1)),main="Función escalonada",col="coral",lwd=2,
  ylab="F(x)")
```

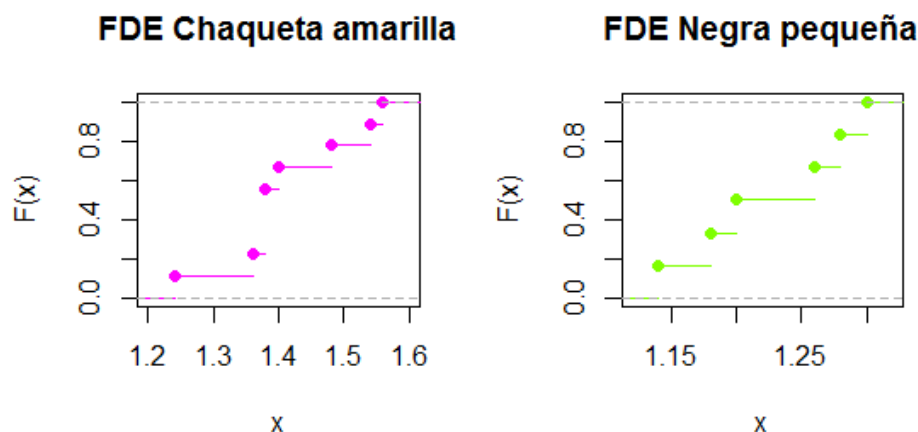



Figura 5.34: Comparación de dos funciones acumuladas empíricas

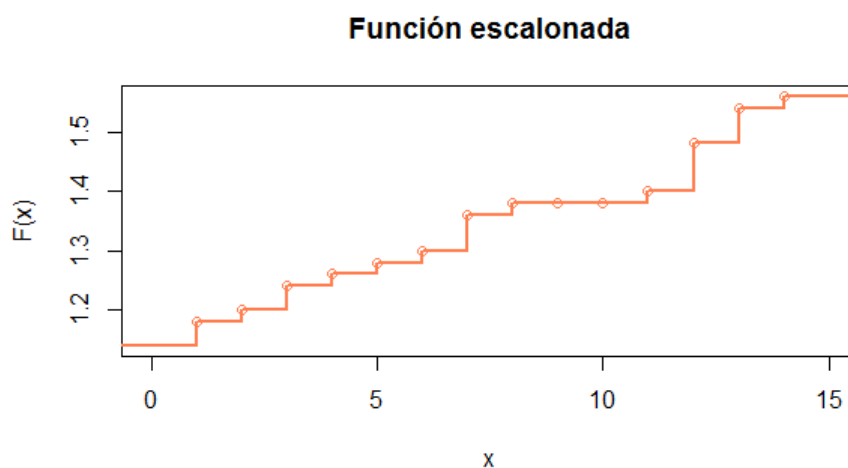


Figura 5.35: Gráfico de una función escalonada

5.4.12 Comparación de distribuciones

En Estadística es de esencial interés comparar el comportamiento de una variable en distintas sub-poblaciones así como también analizar si el modelo teórico ajusta al conjunto de datos que han sido observados. En especial, es importante saber si un conjunto de observaciones puede suponerse como proveniente de una distribución normal. Existen diferentes modos de realizar estas comparaciones. En este apartado veremos modos gráficos.

Un gráfico Cuantil-Cuantil permite observar qué tan próximas son nuestras observaciones de una determinada distribución teórica o bien comparar las distribuciones de dos conjuntos de datos observados.

En muchos casos resulta de interés comparar las observaciones con los cuantiles de la distribución normal. en este caso el gráfico recibe el nombre de gráfico de probabilidad normal. Se grafican los datos ordenados versus los cuantiles gaussianos. Cuanto más cercanos a una distribución normal estén los datos, más próximos a una recta aparecerán en el gráfico de probabilidad normal.

```

y=rchisq(200,df=5)
# genera una muestra aleatoria Chi cuadrado con 5 grados de libertad de tamaño 200
qqnorm(y)
# compara los cuantiles de esta muestra con los de una normal
qqnorm(y,col="mediumorchid1",pch=18,main="Gráfico cuantil-cuantil con una normal",
      xlab="Cuantiles teóricos",ylab="Cuantiles muestrales")
qqline(y,col="navy",lwd=1)
# agrega una línea bajo el supuesto de normalidad

attach(IMCinfantil)
cc.muj=CC[SEXO=="F"]
# guarda los datos de circunferencia de cintura de las mujeres
cc.var=CC[SEXO=="M"]
# guarda los datos de circunferencia de cintura de los varones
qqplot(cc.muj,cc.var)
# produce un gráfico cuantil-cualtil que si resulta una línea recta las
distribuciones coinciden
qqplot(cc.muj,cc.var,col=8:9,pch=18,main="Comparación de distribuciones por sexo",
      xlab="Circunferencia de cintura en mujeres",ylab="Circunferencia de cintura
      en varones")
media=mean(c(cc.muj,cc.var))
desvio=sd(c(cc.muj,cc.var))
qqline(c(cc.muj,cc.var), distribution=function(p) qnorm(p,mean=media,sd=desvio),
      col="chocolatel")
# agrega la línea en el caso en que las distribuciones fueran iguales

```

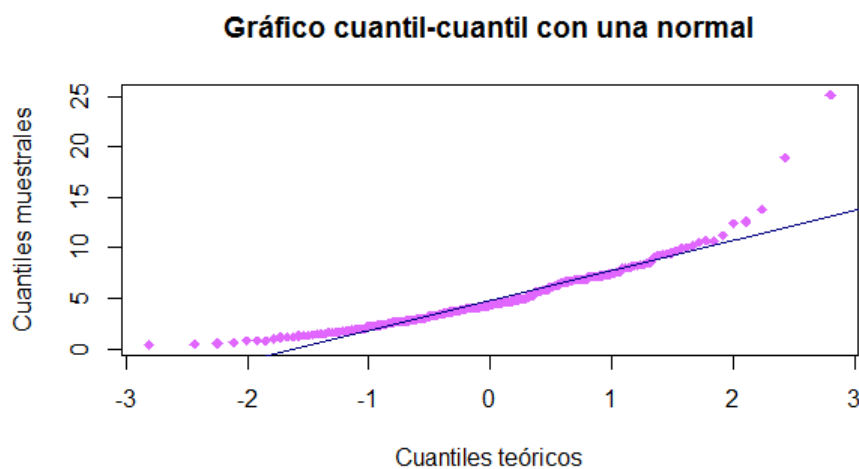


Figura 5.36: Gráfico cuantil-cuantil contra una distribución normal

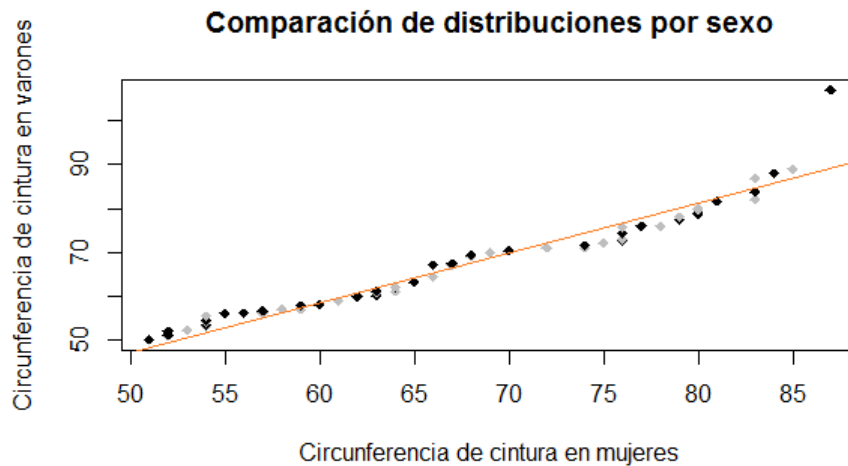


Figura 5.37: Gráfico comparativo entre dos distribuciones

5.4.13 Gráfico de caja y bigote o Boxplot

Este tipo de gráfico fue propuesto por Turkey en 1977 y está basado en las medidas de posición. El mismo es de fácil lectura e interpretación, a la hora de representar datos numéricos, apreciar características importantes de la distribución y comparar distintas distribuciones.

Podemos describir la manera de realizar un gráfico boxplot de la siguiente manera.

- Se dibuja una **caja** (*box* en inglés) cuyos extremos se ubican en los cuartiles primero y tercero. Dentro de ella, se dibuja un segmento que corresponde a la mediana o segundo cuartil.
- A partir de cada extremo se dibuja un segmento, llamado **bigote**, hasta el dato que se encuentra más alejado cuya posición está a lo sumo a 1.5 veces el desvío intercuartil (DI) del extremo de la caja.
- Se marca con el símbolo \circ aquellos datos que están a más de 1.5 DI de cada extremo de la caja. Se denominan **outliers moderados** a aquellos datos que se encuentran entre 1.5 y 3 DI del borde de la caja. Se denominan **outliers severos** a los que están a más de 3 DI del borde de la caja.
- La detección de observaciones atípicas es importante pues las mismas pueden determinar o influenciar fuertemente en los resultados de un análisis estadístico clásico. Esto ocurre debido a que muchas de las técnicas que habitualmente se usan suelen ser muy sensibles a la presencia de esta clase de datos.
- Los outliers deben ser inspeccionados cuidadosamente, por ejemplo, si no hay evidencia de error y su valor es posible no deben ser eliminados. Éstos pueden estar alertando anomalías en un tratamiento o alguna patología.

A partir de un boxplot podemos apreciar los siguientes aspectos sobre la distribución de un conjunto de datos:

- posición
- dispersión
- asimetría
- longitud de los bigotes
- puntos anómalos o outliers

Los *boxplots* son especialmente útiles para comparar varios conjuntos de datos, pues nos dan una rápida impresión visual de sus características.

Ejemplo: Supongamos que en una muestra de tamaño $n = 13$, se tienen los siguientes datos

14 18 24 26 35 39 43 45 56 62 68 92 198

Si se desea observar qué tipo de distribución presenta esta muestra; es decir, si es simétrica o asimétrica, se deben observar las distancias entre los cuartiles primero y tercero y la ubicación de la mediana dentro de la caja, así como el tamaño de los bigotes.

Se observa claramente que el valor 198 está alejado del grupo de valores restantes, por lo que 198 es un valor atípico (*outlier*). La pregunta que sigue de manera natural es si se trata de un *outlier* salvaje.

```
muestra=c(14,18,24,26,35,39,43,45,56,62,68,92,198)
Md=median(muestra)
summary(muestra)
Q1=quantile(muestra,0.25)
Q3=quantile(muestra,0.75)
DI=Q3-Q1
Q3+1.5*DI
Q1-1.5*DI
Q3+3*DI
Q1-3*DI
```

Como $198 > 170 = Q_3 + 3DI$, estamos en presencia de un *outlier* severo.

Ahora, realizaremos gráficos utilizando R.

```
attach(IMCinfantil)
par(mfrow=c(1,2),oma=c(0,0,2,0))
# personaliza el espacio de gráfico
boxplot(PESO)
# realiza un boxplot básico
boxplot(PESO,horizontal=T)
# realiza un boxplot horizontal
mtext("Gráficos de cajas básicos", outer = TRUE, cex = 1.5)
# pone un título para ambos gráficos
par(mfrow=c(1,1),col.main="aquamarine4",adj=0)
# cambia el color y la posición del título
boxplot(PESO,horizontal=T,boxcol=2)
# colorea el borde de la caja
boxplot(PESO,horizontal=T,col=3)
# colorea el interior de la caja
par(mfrow=c(1,1),col.main="aquamarine4",adj=0)
# cambia el color y la posición del título
boxplot(PESO,horizontal=T,col="antiquewhite",boxcol="antiquewhite4",
        main="Distribución del Peso")
```

Gráficos de cajas básicos

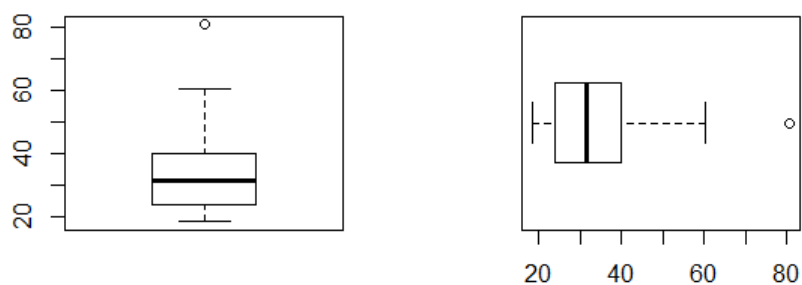


Figura 5.38: Ejemplos de gráficos de cajas básicos

Distribución del Peso

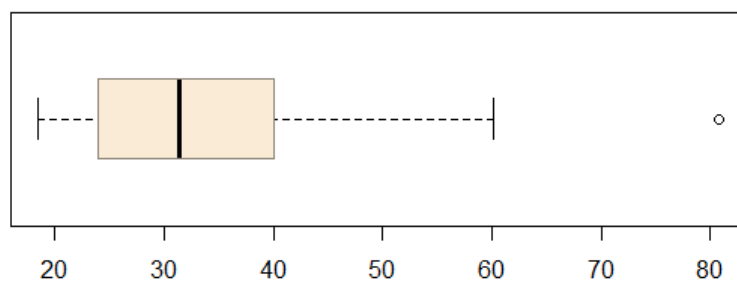


Figura 5.39: Ejemplo de un gráfico de cajas coloreado

También se pueden realizar gráficos de cajas paralelos según los niveles de una categoría.

```

par(col.main="aquamarine3",adj=0.5)
boxplot(CC~CatPeso)
# hace un boxplot para cada categoría de peso
boxplot(split(CC,CatPeso))
# idem anterior
boxplot(CC~CatPeso,horizontal=T)
# grafica horizontalmente
IMCinfantil$CatPeso<-ordered(IMCinfantil$CatPeso,levels=c("D","N","SO","OB"))
# cambia el orden de las cajas
with(IMCinfantil,boxplot(CC~CatPeso))
# hace el boxplot con el orden cambiado
with(IMCinfantil,boxplot(CC~CatPeso,boxcol=topo.colors(5),
  col=terrain.colors(5),
  main="Circunferencia de cintura según peso"))
par(col.main="black")
boxplot(PESO~SEXO*CatPeso,data=IMCinfantil)
# otra manera de relaizar un gráfico de cajas
boxplot(PESO~SEXO*CatPeso,data=IMCinfantil,notch=T)
# cambia el estilo de las cajas
boxplot(PESO~SEXO*CatPeso,data=IMCinfantil,notch=T,
  col=(c("gold","darkgreen")),main="Pesos por categoría y sexo",
  cex.axis=0.7, xlab="Categorías")

```

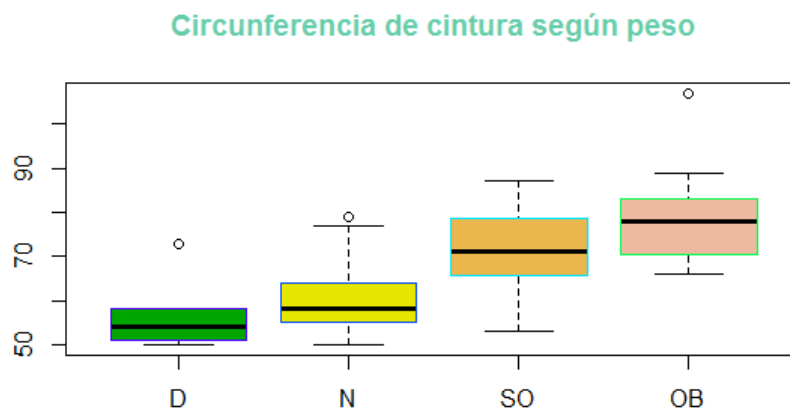


Figura 5.40: Ejemplo de gráficos de cajas en paralelo

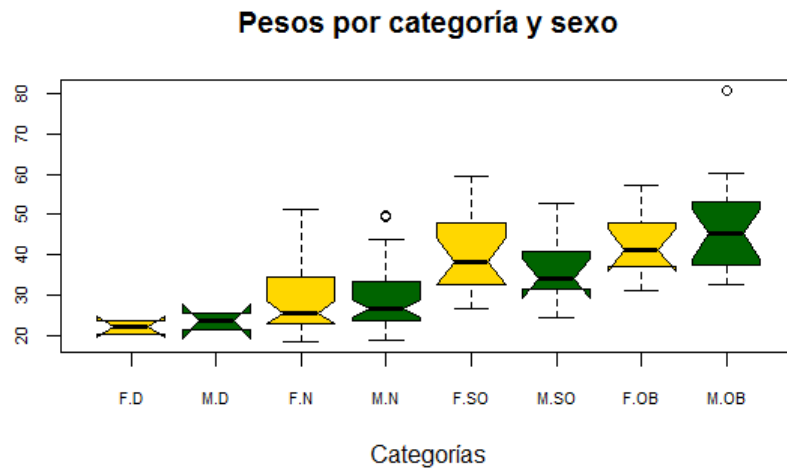


Figura 5.41: Ejemplo de gráficos de cajas en paralelo con otro estilo

5.4.14 Gráficos de correlación

Cuando se quiere explorar la información disponible en una gran base de datos es de particular interés estudiar la asociación estadística entre las variables cuantitativas de dicha base. Para este subconjunto de variables de la base, se dispone de una matriz que compacta la información respecto de la presencia de **correlación** (variación conjunta) entre pares de variables.

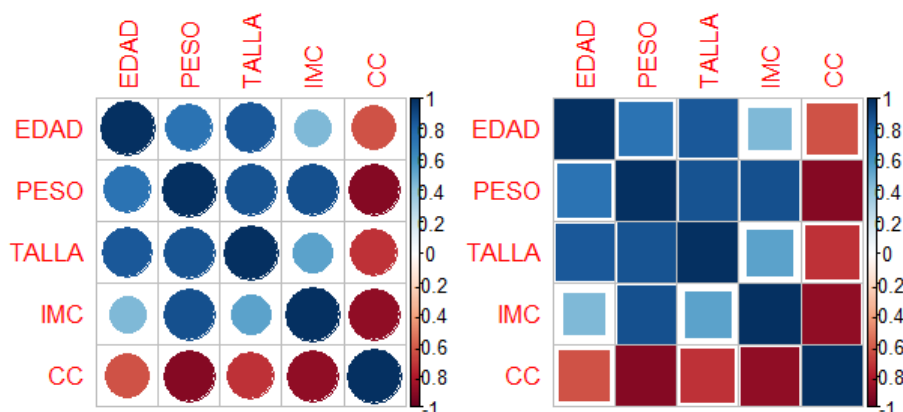
La **matriz de correlación** es una matriz cuadrada que tiene 1 en su diagonal, correspondiendo a la correlación de cada una de las variables consigo misma. Además, es simétrica puesto que la correlación entre X e Y es la misma que la correlación entre Y y X . Por esta razón, en ocasiones sólo se informa el bloque triangular superior o inferior de la misma.

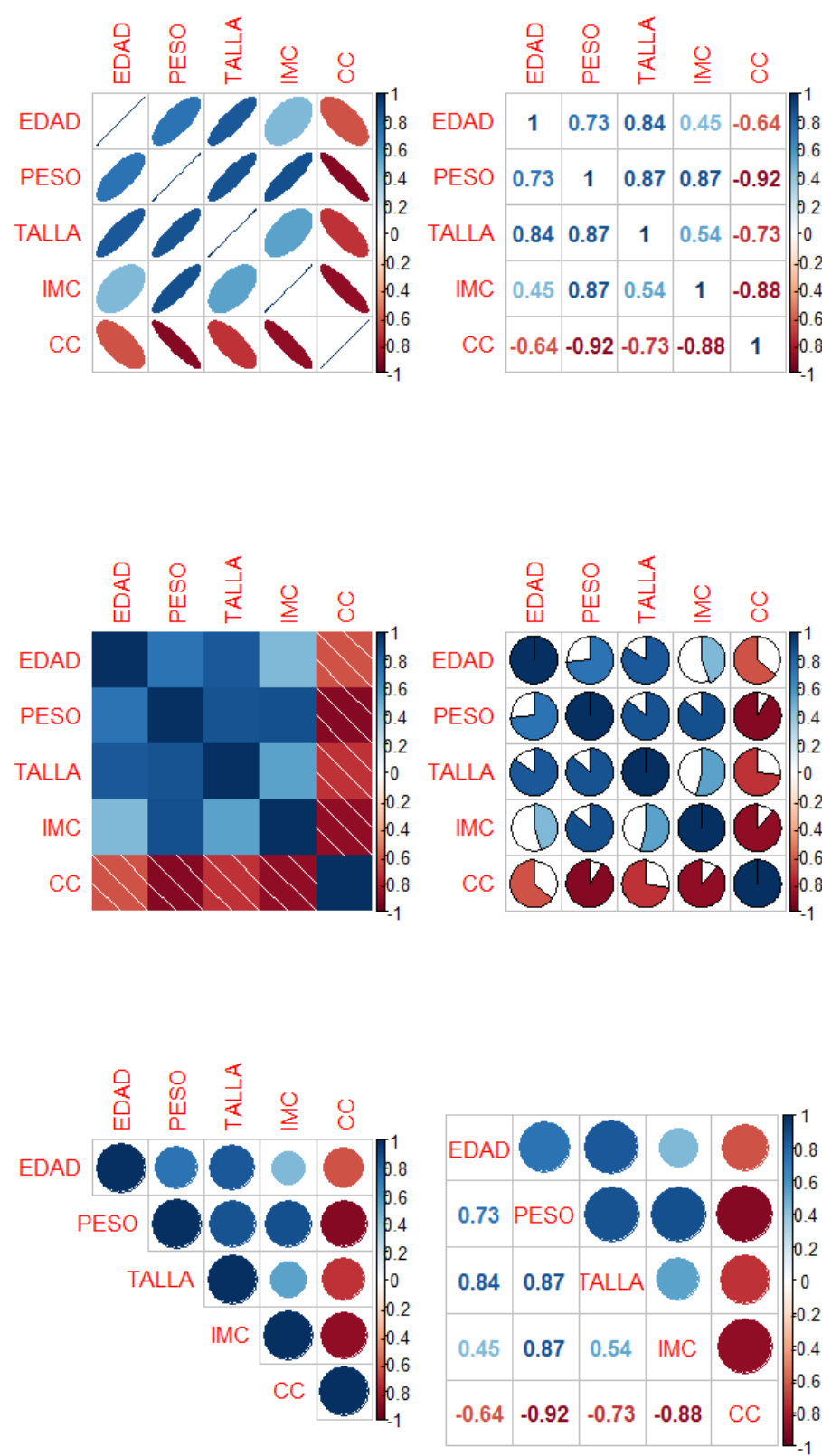
```

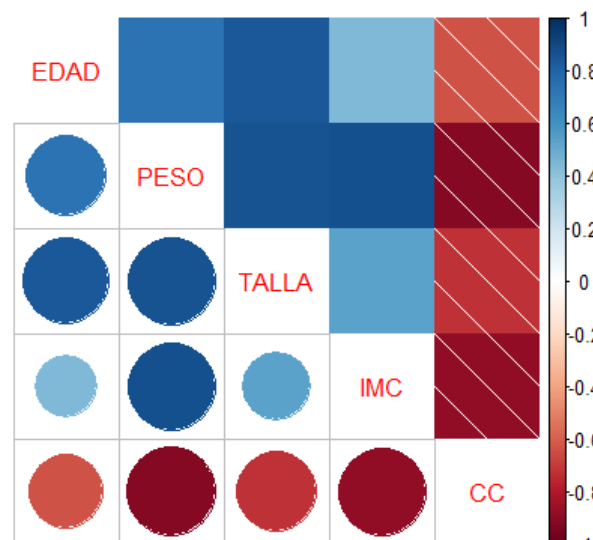
attach(IMCinfantil)
base.niños=data.frame(EDAD,PESO,TALLA,IMC,CC)
# arma una sub-base con las variables numéricas de IMCinfantil
base.niños$CC=max(base.niños$CC)-base.niños$CC
# cambiamos una variable para que correlacione en forma negativa con las restantes
M=cor(base.niños)
# calcula la matriz de correlación de las variables de la base
M
corrplot(M,method="circle")
# representa la matriz de correlaciones mediante círculos
corrplot(M,method="square")
# representa la matriz de correlaciones mediante cuadrados
corrplot(M,method="ellipse")
# representa la matriz de correlaciones mediante elipses
corrplot(M,method="number")
# representa la matriz de correlaciones mediante números
corrplot(M,method="shade")
# representa la matriz de correlaciones mediante sombreandos
corrplot(M,method="pie")
# representa la matriz de correlaciones mediante gráficos de torta
corrplot(M,type="upper")
# representa sólo la parte superior de la matriz de correlación
corrplot(M,type="lower")
# representa sólo la parte inferior de la matriz de correlación
corrplot(M,method="ellipse",type="upper")
# permite combinaciones de estilos
corrplot.mixed(M)
# representa la matriz de correlación combinando círculos y números
corrplot.mixed(M,lower="circle",upper="shade")
# permite combinaciones de estilos por bloques

```

A continuación presentamos distintos modelos de matrices de correlación confeccionadas con R.







5.4.15 Gráficos de curvas de nivel

Mostramos cómo graficar las curvas de nivel de una superficie cónica.

```
x=y=seq(-4*pi,4*pi,len=27)
r=sqrt(outer(x^2,y^2,"+"))
filled.contour(exp(-0.1*r),axes=FALSE)
# grafica las curvas de nivel del cono dado por la función r
filled.contour(exp(-0.1*r),frame.plot=FALSE,plot.axes=)
# pone referencias de colores
```

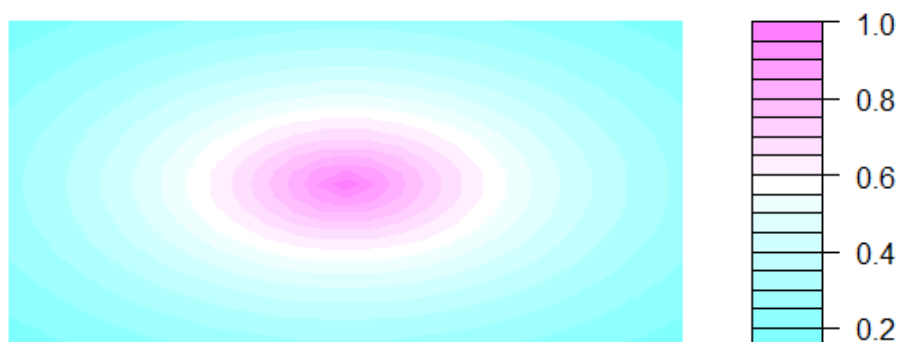


Figura 5.42: Gráfico de curvas de nivel de un cono

5.4.16 Caras de Chernoff

Las caras de Chernoff, que fueron creadas por Herman Chernoff, permiten visualizar datos multivariados con la forma de un rostro humano. Los rasgos individuales, tales como ojos, orejas, boca, cabello y nariz representan los valores de las variables a través de su forma, tamaño, ubicación y orientación. La idea subyacente de este tipo de representación es que los seres humanos estamos familiarizados con la observación de rostros y percibimos parecidos y diferencias sin dificultad.

Las caras de Chernoff se encargan de manejar cada variable de manera diferente, debido a que las características de los rostros varían en importancia percibida.

```
par(col.main="violet")
# cambia el color de los textos
galle=read.csv("C:/Users/Usuario-PC/Dropbox/Curso R/Datos/galletitas.csv", sep=";")
galle.salad=galle[c(1:3,7,15:17),]
# agrupa las galletitas saladas
galle.dulce=galle[c(4:6,8:14),]
# agrupa las galletitas dulces
faces(galle.salad[,2:6])
# hace un gráfico con las caras de Chernoff
faces(galle.salad[,2:6],nrow.plot=3)
# ajusta el alto de las caras
faces(galle.salad[,2:6],ncol.plot=4)
# acomoda la cantidad de caras por fila
faces(galle.salad[,2:6],face.type=0)
# grafica las caras sin color
faces(galle.salad[,2:6],face.type=2)
# cambia el estilo de cara
faces(galle.salad[,2:6],labels=galle.salad[,1])
# etiqueta las caras
title("Caritas de Chernoff saladas")
# ponemos título

faces(galle.dulce[,2:6],nrow.plot=3,ncol.plot=4,face.type=2,labels=galle.dulce[,1])
title("Galletitas Dulces")
```

Caritas de Chernoff saladas

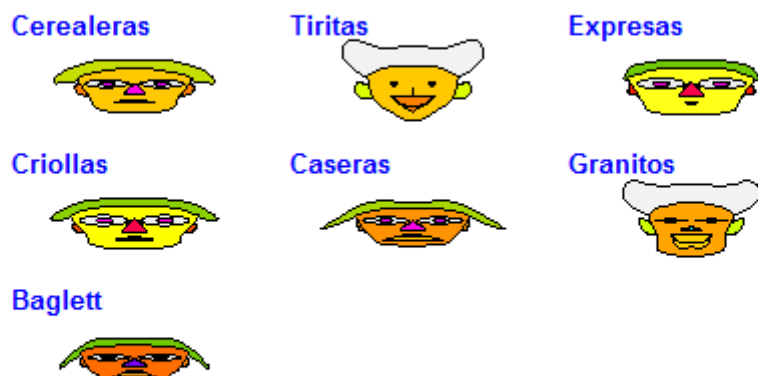


Figura 5.43: Gráfico de caras de Chernoff

Galletitas Dulces

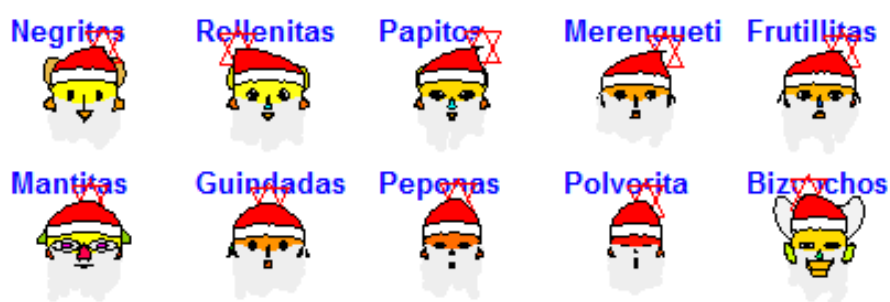


Figura 5.44: Gráfico de caras de Chernoff con diferente estilo

5.4.17 Gráfico de estrellas

Supongamos que se dispone de un conjunto de datos multivariados ordenados de manera matricial, de modo que las filas corresponden a las observaciones y las columnas a cada variable. Representando en el plano, se pueden construir un círculo por cada observación multivariada de un radio prefijado, con tantos rayos como columnas igualmente espaciados desde el centro. Las longitudes de estos rayos son proporcionales a los valores de las variables en cada observación. Los extremos de los rayos pueden conectarse con segmentos rectos formando una estrella.

```
par(col.main="black",adj=0.5)
row.names(galle.salad)=galle.salad[,1]
# coloca etiquetas al gráfico
stars(galle.salad[,2:6])
# hace un gráfico de estrellas
stars(galle.salad[,2:6],full=T)
# dibuja con volumen
stars(galle.salad[,2:6],full=F)
# dibuja en perspectiva
stars(galle.salad[,2:6],radius=F)
# omite aristas
stars(galle.salad[,2:6],axes=T)
# dibuja los ejes
stars(galle.salad[,2:6],frame.plot=T)
# recuadra el gráfico
stars(galle.salad[,2:6],draw.segments=T)
# cambia el estilo
stars(galle.salad[,2:6],col.lines=rainbow(15))
# cambia el color a las líneas
stars(galle.salad[,2:6],cex=0.8,flip.labels=T)
# cambia la posición de las etiquetas
stars(galle.salad[,2:6],cex=0.8,flip.labels=F,len=0.8)
# cambia el tamaño de las estrellas
stars(galle.salad[,2:6],cex=0.8,flip.labels=F,len=0.8,col.stars=terrain.colors(7))
# colorea los interiores de las estrellas
stars(galle.salad[,2:6],cex=0.8,flip.labels=F,len=0.8,col.stars=terrain.colors(7),
      ncol=4,frame.plot=T,main="Galletitas saladas")

row.names(galle.dulce)=galle.dulce[,1]
stars(galle.dulce[,2:6],full=T,draw.segments=T,cex=0.9,len=0.8,ncol=4,frame.plot=T,
      main="Galletitas dulces")
```

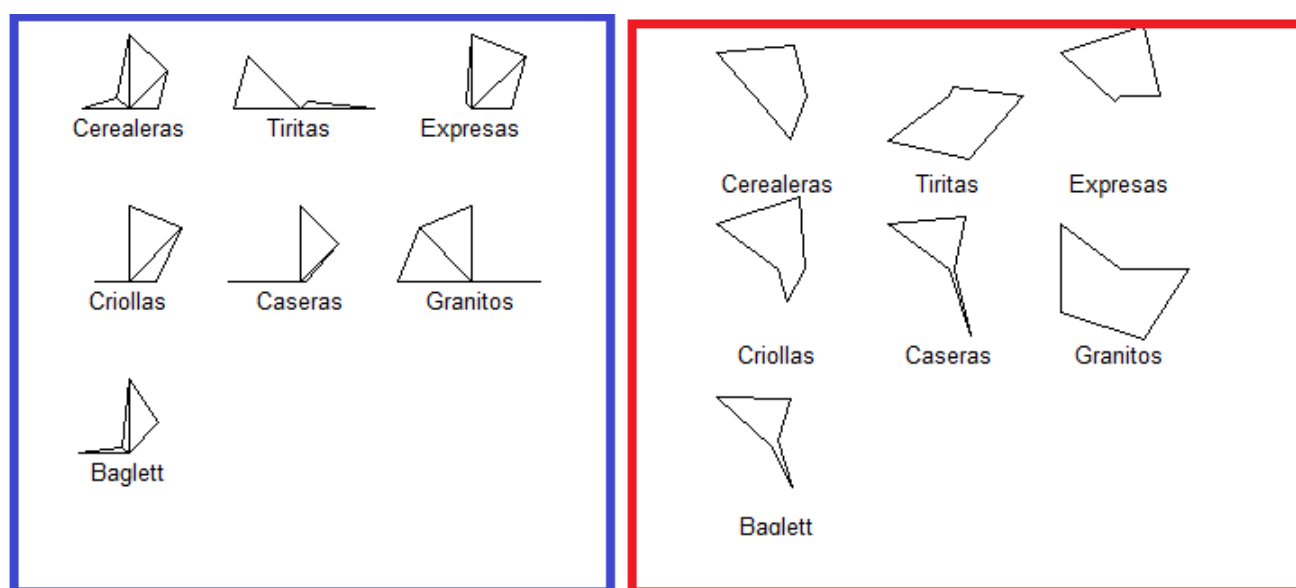


Figura 5.45: Distintos estilos para gráficos de estrellas

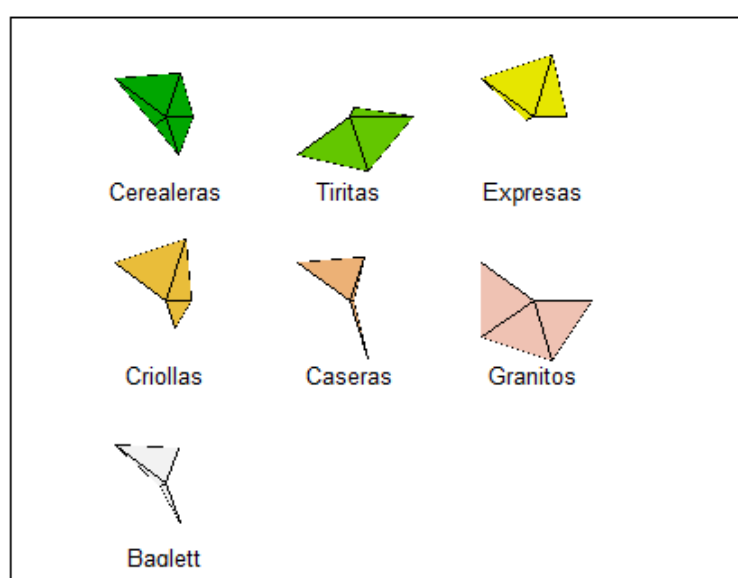


Figura 5.46: Gráfico de estrellas

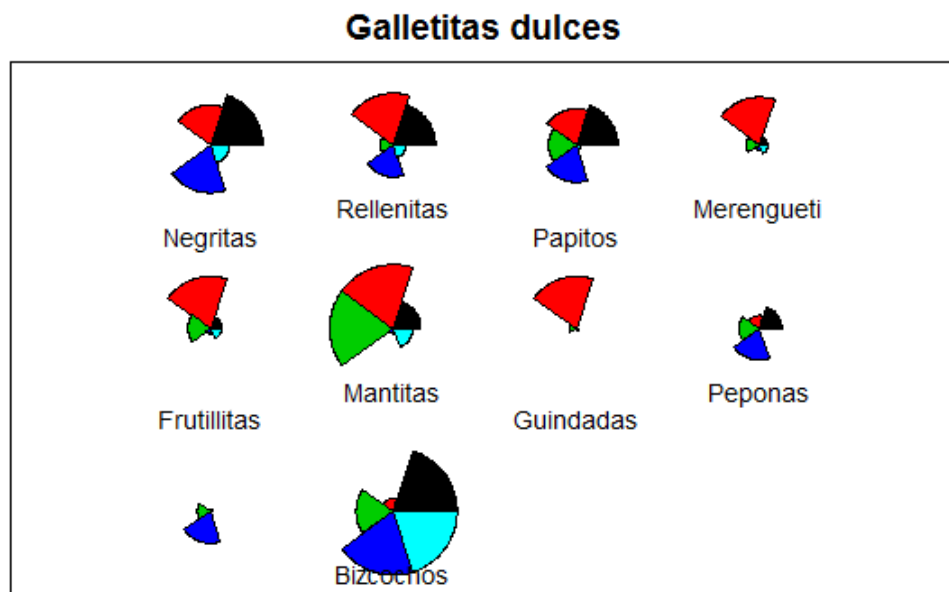


Figura 5.47: Gráfico de estrellas con diferente estilo

5.5 Ejercicios de revisión

La base de datos *sitiosweb* (ver Tablas 8.9 y 8.10) consta de las siguientes variables:

Id: corresponde a la identificación de cada individuo .

Nac: corresponde al país de la nacionalidad, que puede ser ARGENTINA, BRASIL o CANADÁ.

Edad: corresponde a la edad en años.

Estatura: corresponde a la estatura en centímetros.

Sitio: corresponde al sitio de preferencia indicado por el individuo, que puede ser Chat, Correo, Buscador, Programas, Música, Deportes u Otros.

Tiempo: corresponde al tiempo promedio de uso de este sitio por día y medido en minutos.

Temp: corresponde a la temperatura exterior, medida en grados Celsius.

Autos: corresponde a la cantidad aproximada de autos que estacionan por día en la manzana donde vive el individuo.

Cigarrillos: corresponde a la cantidad de cigarrillos consumidos por el individuo mientras visita su sitio web preferido.

1. Para los datos de la base *sitiosweb* se piden los siguientes informes gráficos y se sugiere cambiar estilos, colores y tamaños.
 - (a) Un gráfico de torta para la variable nacionalidad.
 - (b) Un gráfico de barras adyacentes para las variables nacionalidad y sexo.
 - (c) Un gráfico de mosaico para las variables sitio web preferido y sexo.
 - (d) Un gráfico de bastones para la variable sitio web preferido.

- (e) Un histograma con polígono de frecuencias para la variable estatura.
- (f) Un gráfico de cajas del tiempo promedio de uso del sitio preferido
 - i. por sexo.
 - ii. por sitio preferido.
- (g) Un gráfico de correlación múltiple de todas las variables numéricas.
- (h) Un gráfico con las caras de Chernoff para las primeras 20 filas de todas las variables numéricas.

Capítulo 6

Funciones

La propuesta de este capítulo está orientada hacia una aplicación didáctica integradora. Como herramientas para esta propuesta consideramos:

- Cálculo de esperanza y varianza de variables aleatorias discretas.
- Simulación de experimentos aleatorios sencillos.
- Integración aproximada con diversos recursos.
- Aplicación de la Ley de los grandes números.
- Cálculo de probabilidades para variables aleatorias continuas.
- Definición de distribuciones triangulares simétricas y asimétricas.
- Generación de variables aleatorias.

Para ello se necesitan las siguientes librerías.

```
library(stats)      # paquete con funciones estadísticas
library(stats4)     # paquete que provee un acercamiento a métodos y clases en un
                    # lenguaje funcional según la versión 4 de S
require(graphics)
```

Comenzamos definiendo una función que calcula la esperanza de una variable aleatoria discreta con soporte finito.

```
esperanza=function(x, probs) { # asigna nombre a la función e indica las variables
                                # de la misma
  esp=sum(x*probs)              # es el cuerpo de la función que va entre llaves
  return(esp)
}
rec.x=seq(1, 6)                 # guarda el recorrido de la tirada de un dado
prob.x=rep(1, 6)/6              # guarda la probabilidad
esperanza(rec.x, prob.x)        # aplica la función para calcular la esperanza
```

Ahora le agregamos el gráfico de la función de probabilidad puntual y la esperanza.

```

graf.esp=function(x,probs){
  esp=esperanza(x,probs)
  plot(x,probs,type="h",lwd=2,col=terrain.colors(8),xlab="Recorrido",
  ylab="Probabilidad")
  # grafica la función de probabilidad
  points(esp,0.1,pch=25,col="red")
  # ubica el valor esperado
  title("Probabilidad puntual y valor esperado")
}
graf.esp(rec.x,prob.x)
# aplica la función a la tirada de un dado

```

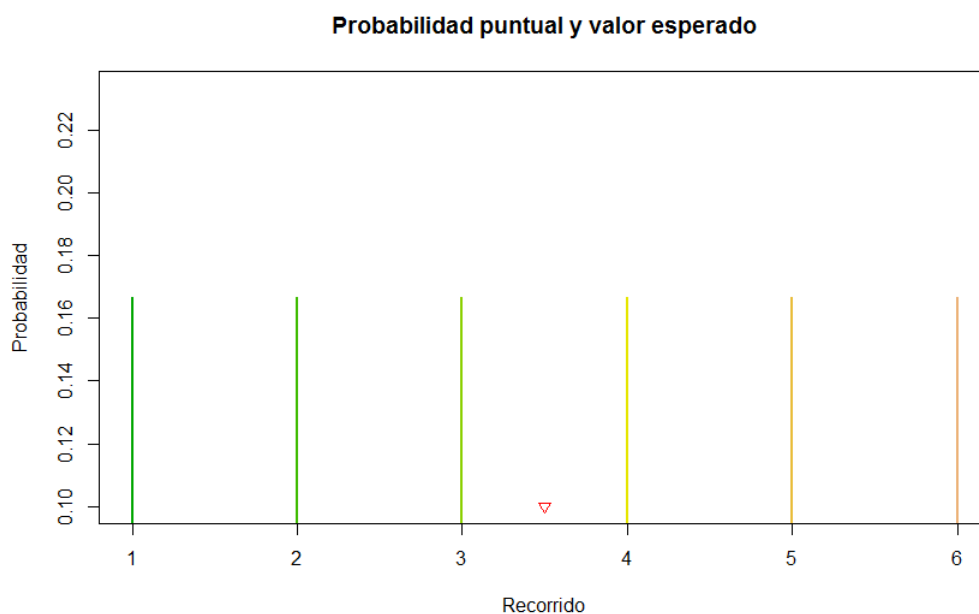


Figura 6.1: Ejemplo de una distribución simétrica y su esperanza

Observemos lo que sucede cuando la función de probabilidad puntual varía en sus características de simetría.

```

rec=c(5,10,20,50)
prob=c(0.1,0.2,0.4,0.3)
graf.esp(rec,prob)

```

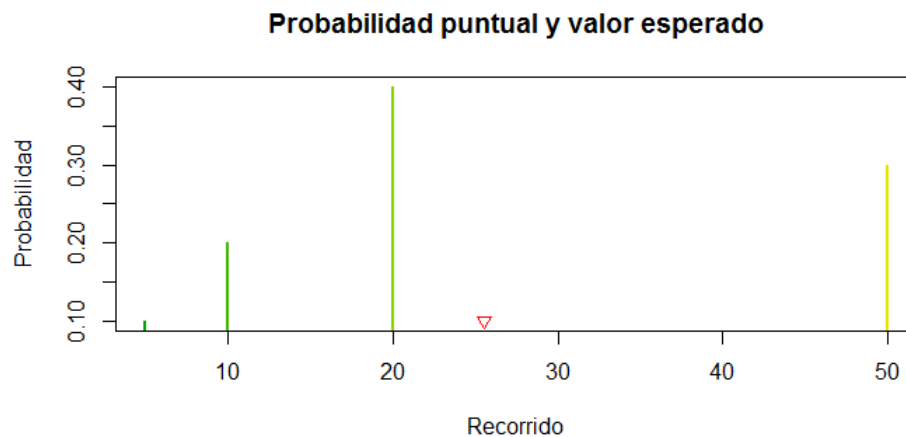


Figura 6.2: Ejemplo de una distribución asimétrica y su esperanza

Calculamos la varianza de la distribución discreta.

```
espyvar=function(x, probs) {
  esp=sum(x*probs)
  esp2=sum(x*x*probs)
  vari=esp2-esp*esp
  return(list("Esperanza = "=esp, "Varianza = "=vari))
}

espyvar(rec, prob)
```

Agregamos a la función anterior la orden para que grafique la función de distribución.

```
espvardist=function(x, probs) {
  plot(stepfun(x, cumsum(c(0, probs))), col=2:6,
  main="Distribución acumulada", ylab="F(x)")
  return(espyvar(x, probs))
}

espvardist(rec, prob)
```

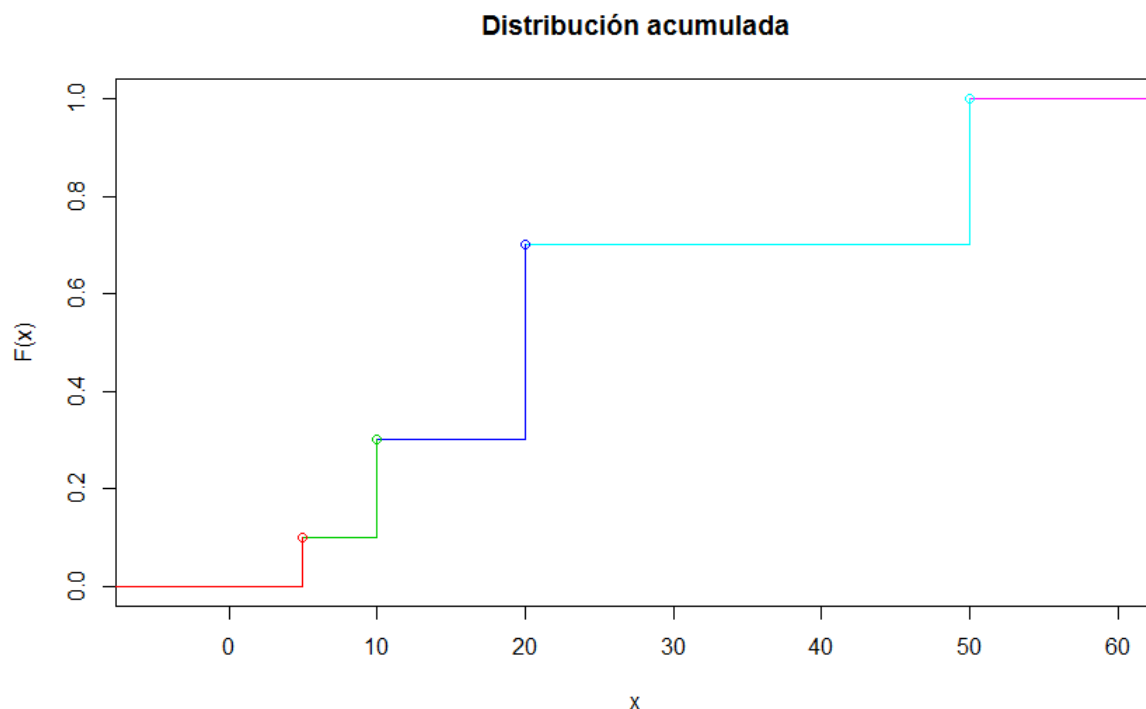


Figura 6.3: Ejemplo del gráfico de una distribución acumulada

En ciertos casos, puede resultar interesante hallar relaciones entre la función de distribución y la función de probabilidad puntual.

6.1 Simulación

El objetivo de las siguientes simulaciones es analizar la teoría frecuencial de la probabilidad.

En el lenguaje R, la sentencia `for` repite un proceso para un rango especificado de una variable, el mismo se indica luego del `for` y va entre paréntesis; es decir, `for (var in seq)` donde `var` indica el nombre de la variable y `seq` una expresión que se evaluará al vector.

Simulemos la tirada de una moneda.

```
exitos=0
tiradas=0
prob.est=0
tiradas=rbinom(10000,1,0.5)

for(i in 1:1000){
  exitos[i]=sum(tiradas[1:(i*10)])
  ensayos=seq(10,10000,10)
  prob.est[i]= exitos[i]/ensayos[i]
}

plot(ensayos,prob.est,type="l",col=3,lwd=1,main="Tendencia de la frecuencia
relativa",xlab="Ensayos",ylab="Probabilidad estimada")
abline(0.5,0,col=2,lwd=1)
```

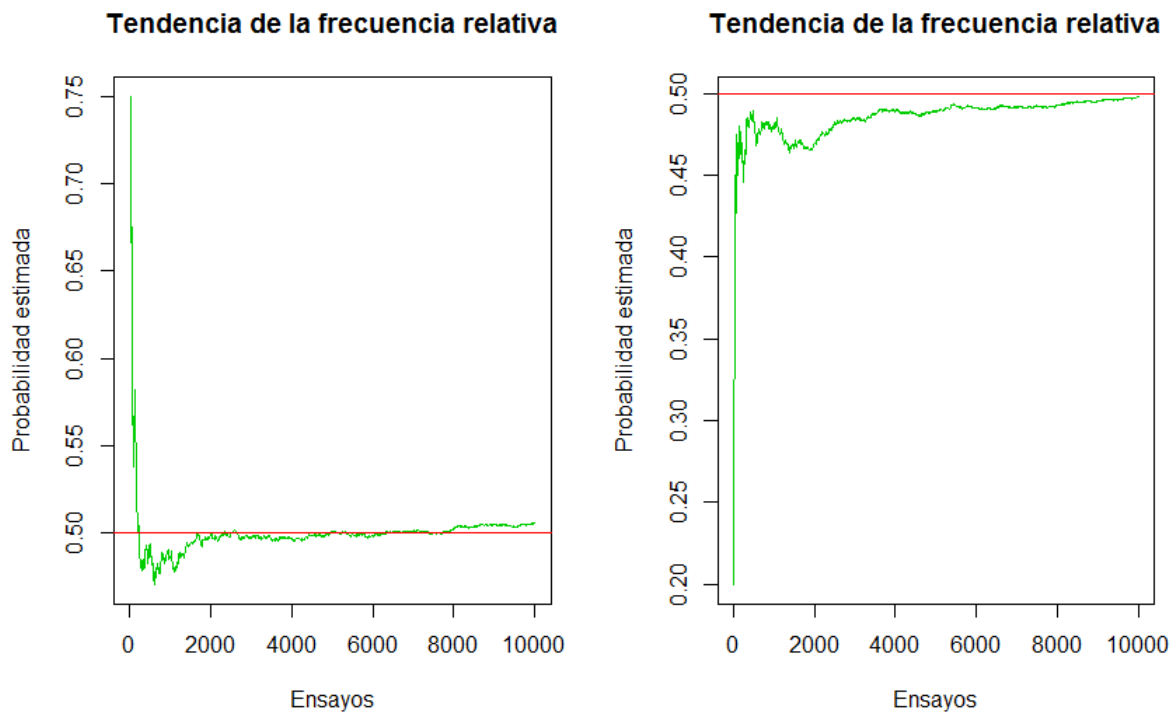


Figura 6.4: Ejemplos de simulación para la tirada de una moneda

Simulemos la tirada de un dado lo que origina una variable aleatoria uniforme discreta. El objetivo de esta aplicación es estudiar el comportamiento de una simulación en cuanto a la convergencia de la distribución empírica a la distribución teórica.

```

tiro.dado=0
prob.est=0
resultado=rep(0,6)
dado=1:6

simuladado=function(n) {
  tiro.dado=sample(1:6,n,replace=TRUE)
  for(i in 1:6){
    resultado[i]=sum(tiro.dado==i)
  }
  return(barplot(resultado,lwd=4,col="gold",ylim=c(0,200),
    names.arg=1:6, main="Simulación dado"))
}

simuladado(1000)

```

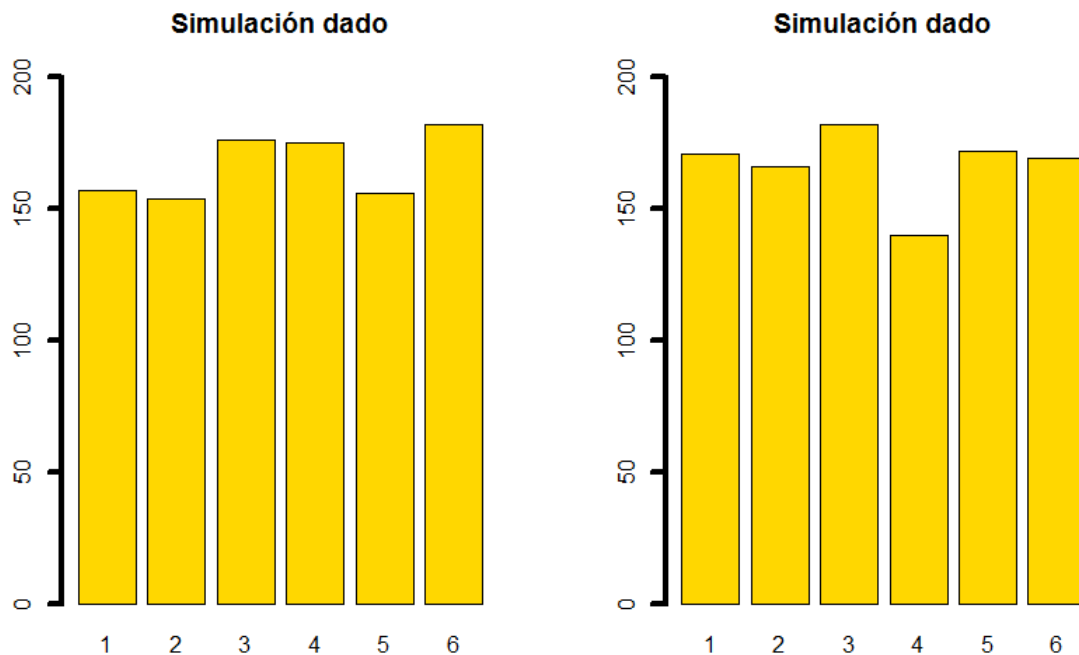


Figura 6.5: Ejemplos de simulación para la tirada de un dado

6.2 Haz de rectas

Mostramos otra aplicación de la sentencia `for` para graficar haces de rectas. Comenzamos por el haz de rectas que pasa por el origen de coordenadas del plano.

```
m=seq(-50,50,5)                                # define las distintas pendientes
x=seq(-1,1,0.1)                                  # asigna valores a x
y=x*(m[1])                                         # devuelve los valores de y en
                                                    # función de x
plot(x,y,type="l",ylim=c(-4,4))                  # grafica la primera recta

for (i in 2:length(m)) {
  y=x*m[i]
  points(x,y,type="l",col=i)
}                                                  # grafica el haz de rectas por
                                                    # el origen

abline(v=0)                                       # grafica la recta vertical
title("Haz de rectas por el origen de coordenadas")
```

Ahora, desplazamos el haz para que pase por el punto $(1, -2)$.

```
x=seq(-2,2,0.1)
yd=(x-1)*m[1]-2
plot(x,yd,type="l",xlim=c(-0.5,2),ylim=c(-4,4),ylab="y")

for (i in 2:length(m)){
  y=(x-1)*m[i]-2
  points(x,y,type="l",col=i)
}

abline(v=1)                                     # grafica la recta vertical
abline(0,0)                                     # grafica el eje x
text(locator(1),col="orange","eje x")
abline(v=0)                                     # grafica el eje y
text(locator(1),col="orange","eje y")
title("Haz de rectas por el punto (1,-2)")
```

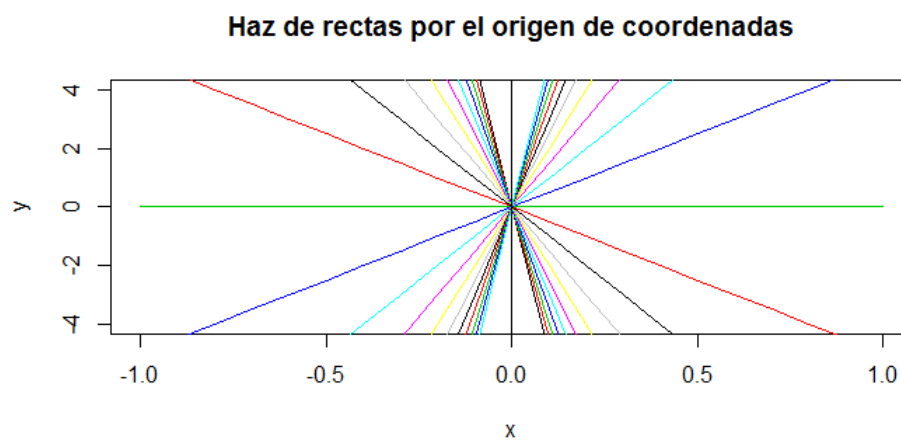


Figura 6.6: Gráfico de un haz de rectas por el origen

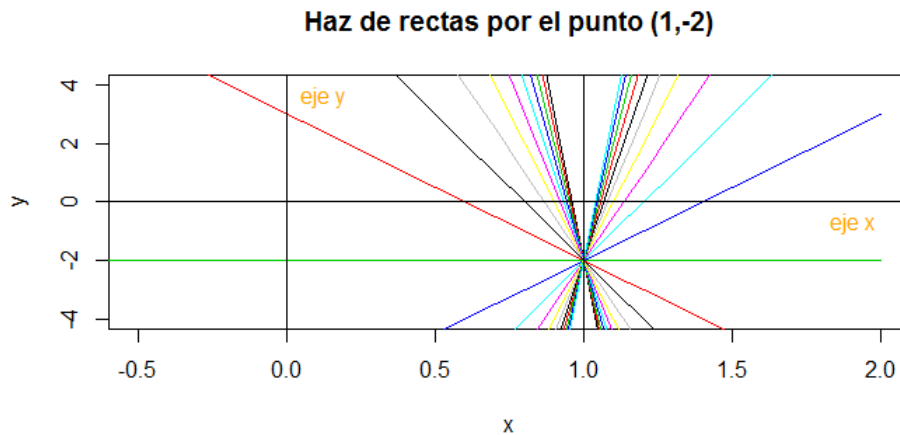


Figura 6.7: Gráfico de un haz de rectas desplazado

6.3 Integración

Para el cálculo de probabilidades de distribuciones continuas es necesario recurrir a la integración numérica. Analizamos diferentes estrategias y las comparamos con la herramienta de integración de R.

Integrales de Riemman

Sea $\pi = \{x_0, x_1, x_2, \dots, x_n\}$ una partición del intervalo cerrado $[a, b]$ y f una función acotada definida en ese intervalo. Se definen

- La *suma inferior de Riemman* de f respecto de la partición π como

$$I(f, \pi) = \sum_{j=1}^n d_j (x_j - x_{j-1})$$

siendo d_j el ínfimo de $f(x)$ en el intervalo $[x_{j-1}, x_j]$.

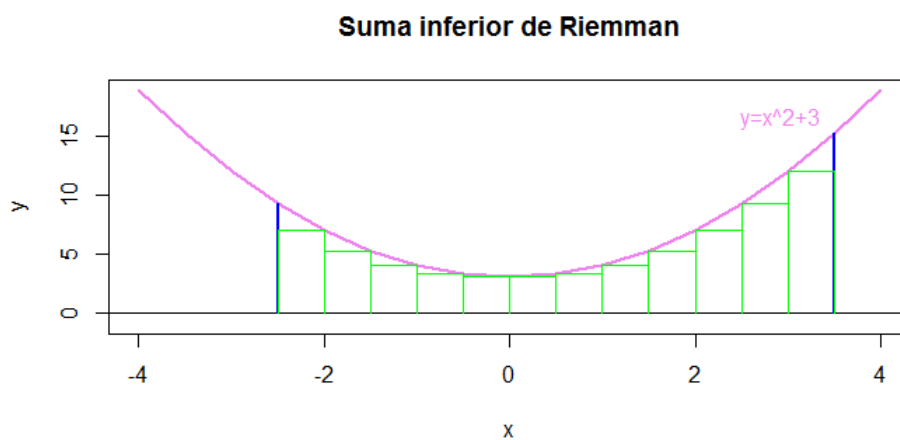


Figura 6.8: Ejemplo gráfico para una suma inferior de Riemman

- La *suma superior de Riemman* de f respecto de la partición π como

$$S(f, \pi) = \sum_{j=1}^n c_j (x_j - x_{j-1})$$

siendo c_j el supremo de $f(x)$ en el intervalo $[x_{j-1}, x_j]$.

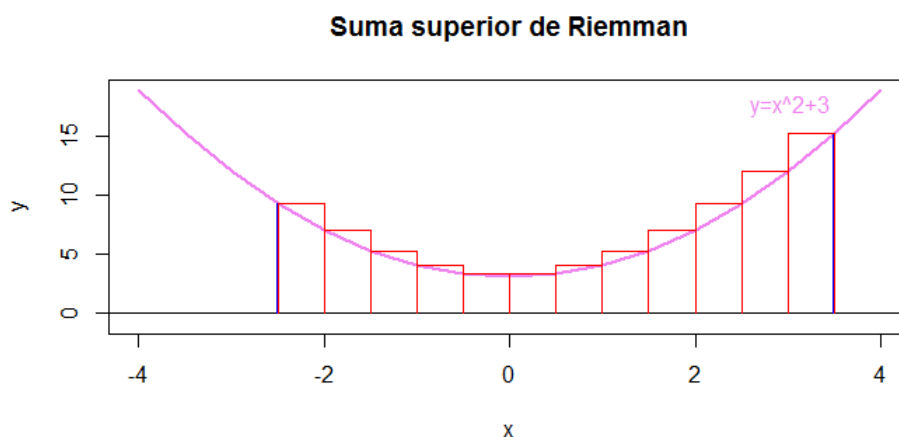


Figura 6.9: Ejemplo gráfico para una suma inferior de Riemman

Para los códigos de los gráficos, ver §8.

Calculamos sumas inferiores.

```
minimo=0

suma.inf=function(a,b,fun,paso){
  vec=seq(a,b,paso)
  n=length(vec)-1
  for (i in 1:n){
    minimo[i]=min(fun(seq(vec[i],vec[i+1],paso)))
  }
  suma.inf=sum(minimo*paso)
  return(suma.inf)
}

fun1=function(x) {x^ 2}

suma.inf(2,5,fun1,0.01)
suma.inf(2,5,fun1,0.001)
suma.inf(2,5,fun1,0.0001)
```

Calculamos sumas superiores.

```

maximo=0

suma.sup=function(a,b,fun,paso){
  vec=seq(a,b,paso)
  n=length(vec)-1
  for (i in 1:n){
    maximo[i]=max(fun(seq(vec[i],vec[i+1],paso)))
  }
  suma.sup=sum(maximo*paso)
  return(suma.sup)
}

suma.sup(2,5,fun1,0.01)
suma.sup(2,5,fun1,0.001)
suma.sup(2,5,fun1,0.0001)

```

Regla de trapecios para integración

La regla de trapecios es un método para integrar numéricamente, esta denominación se debe a que el área descrita por la integral definida se aproxima mediante una suma de áreas de trapecios. Se aproxima la función dividiendo el intervalo $[a, b]$ en n intervalos de igual longitud y formando trapecios por encima de cada intervalo. Consideramos

Δx el ancho de cada uno de los intervalitos en los que se divide en intervalo $[a, b]$

x_i los extremos de los intervalitos

Entonces la fórmula de este método es

$$\int_a^b f(x) dx = \frac{\Delta x}{2} [f(x_0) + 2f(x_1) + \dots + 2f(x_{n-1}) + f(x_n)]$$

Regla de trapecios

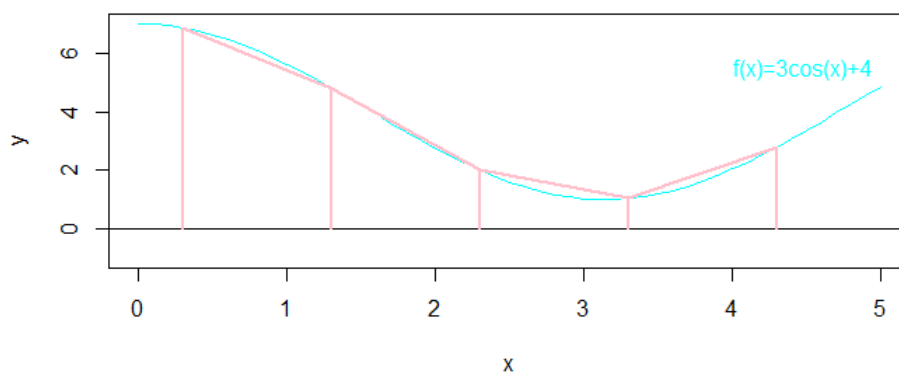


Figura 6.10: Ejemplo gráfico para la regla de trapecios

```

suma.trapecio=function(a,b,fun,paso) {
    vec=seq(a,b,paso)
    imag=fun(vec)
    suma.trap=sum(imag*paso)-(fun(a)+fun(b))/2*paso
    return(suma.trap)
}

suma.trapecio(2,5,fun1,0.1)
suma.trapecio(2,5,fun1,0.01)
suma.trapecio(2,5,fun1,0.001)

```

Para el código del gráfico, ver §8.

Integrales por la Ley de los grandes números

El método de Monte Carlo es un procedimiento general para seleccionar muestras aleatorias de una población utilizando números aleatorios. Este método se utiliza para calcular numéricamente expresiones matemáticas complejas y difíciles de evaluar con exactitud, o que no pueden resolverse analíticamente. Muchos son los ejemplos de aplicación de este método, en este caso vamos a tratar el cálculo de integrales definidas.

A tal fin, recordemos algunos conceptos.

La esperanza de una función de una variable aleatoria. Sea X una variable aleatoria con función de densidad (probabilidad) f y sea $g : \mathbb{R} \rightarrow \mathbb{R}$ una función escalar. La transformación $Y = g(X)$ define una nueva variable aleatoria, cuyo valor esperado se define como

$$E(Y) = \int_{-\infty}^{\infty} g(x)f(x) dx$$

La Ley Fuerte de los Grandes Números. Si $X_1, X_2, \dots, X_n, \dots$ es una sucesión de variables aleatorias independientes idénticamente distribuidas (v.a.i.i.d.), tales que $E(X_i) = \mu$, entonces:

$$\lim_{n \rightarrow +\infty} \bar{X}_n = \lim_{n \rightarrow +\infty} \frac{1}{n} \sum_{i=1}^n X_i \xrightarrow{ctp} \mu$$

Supongamos ahora que $\beta = \int_0^1 g(x) dx$ y $X \sim U(0, 1)$, entonces $\beta = E(g(X))$. Luego,

$$\frac{1}{n} \sum_{i=1}^n g(X_i) \xrightarrow{ctp} \beta$$

Si $\delta = \int_a^b g(x) dx$ y $X \sim U(a, b)$, entonces $\int_a^b g(x)f(x) dx = \frac{1}{b-a} \int_a^b g(x) dx = E(g(X))$. Luego,

$$\frac{b-a}{n} \sum_{i=1}^n g(X_i) \xrightarrow{ctp} \delta$$

```

suma.lgn=function(a,b,fun,n) {
    vec=runif(n,a,b)
    imag=fun(vec)
    suma.lgn=mean(imag)*(b-a)
    return(suma.lgn)
}

suma.lgn(2,5,fun1,1000)
suma.lgn(2,5,fun1,10000)
suma.lgn(2,5,fun1,100000)

```

Para integrar, R utiliza el paquete `QUADPACK` que es una librería de *FORTRAN77* utilizada para aproximar numéricamente integrales mediante el método de la cuadratura de Piessens, deDoncker-Kapenga, Ueberhuber y Kahaner (ver http://people.sc.fsu.edu/~jburkardt/f77_src/quadpack/quadpack.html).

```
integrate(fun1, lower=2, upper=5)           # calcula la integral definida

integrando=function(x) {1/(1+x^ 2)}
integrate(integrando, lower=0, upper=Inf)   # calcula integrales impropias
```

6.4 Distribuciones triangulares

La sentencia `if` marca un condicional, la condición a satisfacerse se escribe entre paréntesis. Para indicar lo que hacer para la condición contraria, se debe poner `else`. Podemos utilizar esto para definir funciones de densidad partidas.

Distribución triangular simétrica

```
triang=function(x){
  if (x<0|x>2) {triang=0}
  else if (x>=0 & x<1) {triang=x}
  else {triang=2-x}
  return(triang)
}

ej1=seq(-1, 3, 0.1)
ej2=Vectorize(triang)(ej1)
# vectoriza una función escalar
plot(ej1, ej2, type="l", col="purple", lwd=2, main="Distribución triangular simétrica",
     xlab="x", ylab="f(x) ")
abline(0, 0)
```

Distribucion triangular asimétrica

```
triang.asim=function(x){
  if (x<5|x>8) {triang.asim=0}
  else if (x>=5 & x<7) {triang.asim=(x-5)/3}
  else {triang.asim=-2*(x-8)/3}
  return(triang.asim)
}

ej1=seq(4, 9, 0.1)
ej2=Vectorize(triang.asim)(ej1)
plot(ej1, ej2, type="l", col="lightblue", lwd=2, xlab="x", ylab="f(x) ")
title("Distribución triangular asimétrica")
abline(0, 0)
```

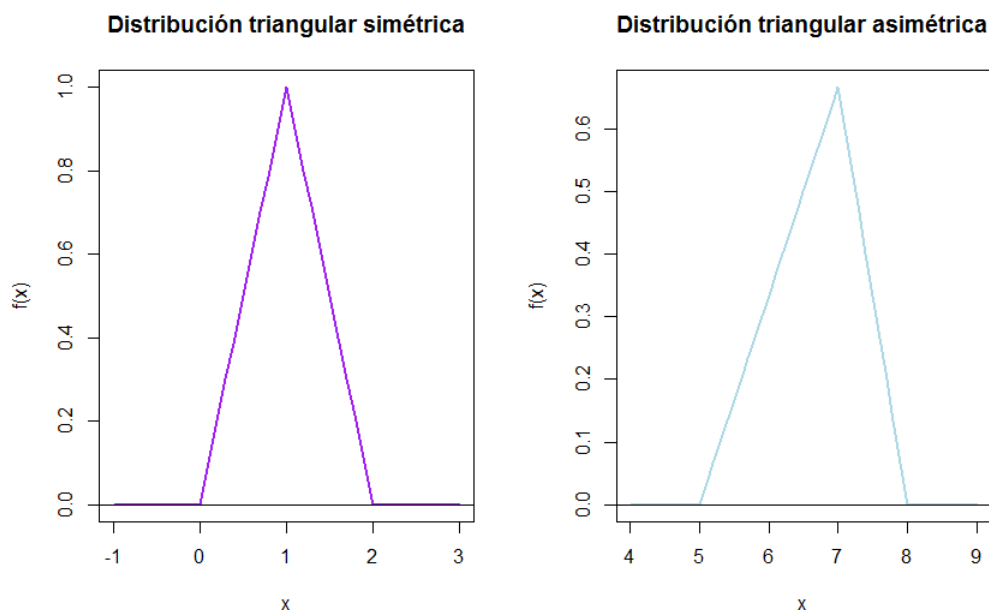


Figura 6.11: Ejemplos de funciones triangulares

Supongamos ahora que X es una variable aleatoria con distribución triangular simétrica en el intervalo $[0, 2]$.

```
integrate(Vectorize(triang), 0, 1)
# calcula  $P(0 < X < 1)$ 
integrate(Vectorize(triang), 0, 1) [[1]]
# devuelve sólo valor del área aproximada, sin el error de estimación
integrate(Vectorize(triang), 0, 2) [[1]]
# calcula  $P(0 < X < 2)$  verifica que la función de densidad de probabilidad está bien
# definida
integrate(Vectorize(triang), 0, 3) [[1]]
# calcula  $P(0 < X < 2)$ 

integrate(Vectorize(triang.asim), 5, 6) [[1]]
# calcula  $P(5 < X < 6)$  siendo  $X \sim \text{Triang.asim}(5, 8, \text{max} = 7)$ 
```

6.5 Generación de variables aleatorias

Los siguientes ejemplos pretenden mostrar la convergencia del estimador media muestral para la media poblacional en diferentes distribuciones y también ilustrar el Teorma Central del Límite.

Comparamos con el caso de una normal.

```
ffnor=function(n,a,b) {mean(rnorm(n,mean=a,sd=b))}
# calcula la media de una muestra normal de tamaño n
res=replicate(1000,ffnor(100,0,1))
# replica 1000 veces la función ffnor que genera medias muestrales de muestras de
tamaño 100
hist(res,col=topo.colors(15),freq=FALSE,probability=TRUE,main="Medias estimadas de
una normal")
# grafica el histograma correspondiente a las medias muestrales
mean(res)
# calcula la media de las 1000 medias muestrales
points(density(res,bw="nrd0",adjust=1,kernel="gaussian"),type="l",col="tomato",lwd=2)
points(mean(res),0.1,pch=16,col="red")
title("Medias estimadas de una normal")
```

Comparamos con el caso de una uniforme (simétrica).

```
ffuni=function(n,a,b) {mean(runif(n,a,b))}
# genera la media de una muestra uniforme en [a,b] de tamaño n
res=replicate(1000,ffuni(100,0,1))
# replica 1000 veces la función ffuni que genera medias muestrales de muestras de
tamaño 100
hist(res,col=topo.colors(15),freq=FALSE,probability=TRUE,main="Medias estimadas de
una uniforme")
# grafica el histograma correspondiente a las medias muestrales
mean(res)
# calcula la media de las 1000 medias muestrales
points(density(res,bw="nrd0",adjust=1,kernel="gaussian"),type="l",col="tomato",lwd=2)
points(mean(res),0.1,col="red",pch=16,cex=1.5)
```

Comparamos con el caso de una exponencial (asimétrica).

```
ffexp=function(n,alfa) {mean(rexp(n,alfa))}
# genera la media de una muestra exponencial de parámetro alfa de tamaño n
res=replicate(1000,ffexp(100,0.02))
# replica 1000 veces la función ffexp que genera medias muestrales de muestras de
tamaño 100
hist(res,col=terrain.colors(15),freq=FALSE,probability=TRUE,main="Medias estimadas de
una exponencial")
# grafica el histograma correspondiente a las medias muestrales
mean(res)
# calcula la media de las 1000 medias muestrales
points(density(res,bw="nrd0",adjust=1,kernel="gaussian"),type="l",col="tomato",lwd=2)
points(mean(res),0.001,col="red",pch=16,cex=1.5)
```

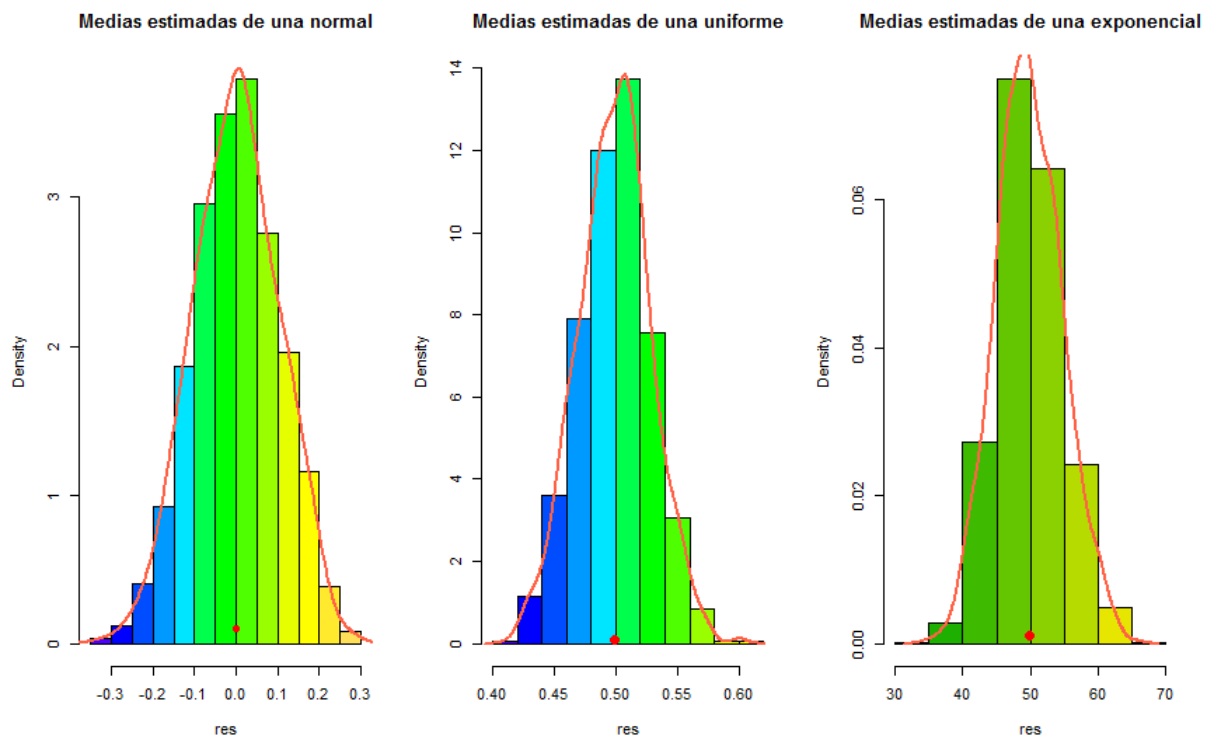


Figura 6.12: Ejemplos de comparaciones de medias estimadas

6.6 Ejercicio de aplicación

Viajando a Monte Carlo

En el área de finanzas, los **métodos de Monte Carlo** son usualmente utilizados para evaluar inversiones en un proyecto de una unidad de negocios o un nivel corporativo, o también para analizar derivaciones financieras. Éstos pueden ser utilizados para modelar cronogramas de proyectos en los cuales las simulaciones agregan estimaciones para “el peor caso”, “el mejor caso”, y las duraciones más probables de cada tarea para determinar los resultados del proyecto en su totalidad. Los métodos de Monte Carlo también son usados en el análisis de opciones de precios y estándares de riesgo.

En el siguiente problema, aplicaremos métodos de Monte Carlo para evaluar la propuesta de lanzamiento de un nuevo producto.

La rentabilidad de un producto puede simplificarse a través de la siguiente ecuación

$$\text{beneficio} = (\text{unidades vendidas} \times (\text{precio unitario} - \text{costo unitario})) - \text{costos fijos}$$

Nuestro modelo se basa en las siguientes hipótesis:

- El número de unidades vendidas y el precio unitario pueden variar dependiendo del **clima económico** (grado de riesgo asociado a las inversiones en los mercados de productos financieros).
- El costo unitario puede variar dependiendo, entre otras cosas, de los proveedores y del costo laboral.
- Los costos fijos son conocidos y no presentan ninguna variabilidad.

Nuestros pronosticadores de ventas han estimado el siguiente escenario para la expectativa del tiempo de vida del producto.

Nuestro Departamento de Planeamiento de Producción estima que el costo unitario variará entre \$6 y \$9 con un valor más probable de \$7. Esto nos invita fuertemente a aplicar la distribución triangular.

Clima económico	Probabilidad	Unidades vendidas	Precio unitario
Cálido	15%	135	5.5
Moderado	20%	105	7.0
Bajo	15%	70	8.5
Recesivo	50%	40	11.5

Los costos fijos se saben de \$25.

Nuestro trabajo consiste en construir un modelo para determinar la probabilidad de avanzar con la producción de este producto para que sea rentable. Deberíamos correr la simulación para un mínimo de 10000 intentos.

La tarea se basa en escribir una función que devuelva una matriz de dos columnas conteniendo la ganancia (o pérdida) en incrementos de \$10000 y la probabilidad expresada en porcentaje y redondeando a dos cifras decimales.

```

caso=1:10000
clim=runif(10000,0,1)
# genera 10000 datos uniformes en [0,1]
unit=runif(10000,0,1)
clima=1*(clim>0.15)+1*(clim>0.35)+1*(clim>0.5)+1
# asignamos el clima según la tabla
table(clima)/10000
# verifica la asignación asigna según la tabla

unid.vend=0
precio.uni=0
costo.uni=0
# inicializa las variables

for(i in 1:10000){
  if(clima[i]=="1") {unid.vend[i]=135}
  else if (clima[i]=="2") {unid.vend[i]=105}
  else if (clima[i]=="3") {unid.vend[i]=70}
  else {unid.vend[i]=40}
}
asigna unidades vendidas según el clima

for(i in 1:10000){
  if(clima[i]=="1") {precio.uni[i]=5.5}
  else if (clima[i]=="2") {precio.uni[i]=7}
  else if (clima[i]=="3") {precio.uni[i]=8.5}
  else {precio.uni[i]=11.5}
}
# asigna precio por unidad según el clima

```



```

triang.asim2=function(x){
  if (x<6|x>9) {triang.asim2=0}
  else if (x<7) {triang.asim2=2*(x-6)/3}
  else {triang.asim2=-1*(x-9)/3}
  return(triang.asim2)
}
# define la función triangular correspondiente

triang.asim2(6.1)
triang.asim2(7)
triang.asim2(9)
# evalua la función triangular en algunos valores

# Verifiquemos que se trata de una función de densidad de probabilidad

p1=integrate(Vectorize(triang.asim2),6,6.5)[[1]]
p2=integrate(Vectorize(triang.asim2),6.5,7)[[1]]
p3=integrate(Vectorize(triang.asim2),7,7.5)[[1]]
p4=integrate(Vectorize(triang.asim2),7.5,8)[[1]]
p5=integrate(Vectorize(triang.asim2),8,8.5)[[1]]
p6=integrate(Vectorize(triang.asim2),8.5,9)[[1]]
proba=c(p1,p2,p3,p4,p5,p6)
prob.acum=cumsum(proba)

costo.uni=0
for(i in 1:10000){
  costo.uni[i]=6.25+0.5*(1*(unit[i]>prob.acum[1])+1*(unit[i]>prob.acum[2])+
    1*(unit[i]>prob.acum[3])+ 1*(unit[i]>prob.acum[4])+1*(unit[i]>prob.acum[5]))
}

base=data.frame(caso,clima,precio.uni,unid.vend,costo.uni)
# arma una base de datos
head(base)
# muestra el encabezado
base$benef=unid.vend*(precio.uni-costo.uni)-25
pr1=sum(base$benef<=-200)
pr2=sum(base$benef<=-100)-sum(base$benef<=-200)
pr3=sum(base$benef<=0)-sum(base$benef<=-100)
pr4=sum(base$benef<=100)-sum(base$benef<=0)
pr5=sum(base$benef<=200)-sum(base$benef<=100)
pr6=sum(base$benef<=300)-sum(base$benef<=200)
pr7=sum(base$benef<=400)-sum(base$benef<=300)
pr8=sum(base$benef>400)

Probabilidad_estimada=c(pr1,pr2,pr3,pr4,pr5,pr6,pr7,pr8)/10000

Beneficio=c("-300 a -200","-200 a -100","-100 a 0","0 a 100","100 a 200",
  "200 a 300","300 a 400",">400")
salida.final=data.frame(Beneficio,Probabilidad_estimada)
salida.final

```

6.7 Ejercicios de revisión

1. Supongamos que un dado está cargado de manera que la probabilidad de obtener un número par es el doble de la probabilidad de obtener un número impar. Se pide lo siguiente:
 - (a) Hallar la función de probabilidad y graficarla.
 - (b) Hallar la función de distribución y graficarla.
 - (c) Calcular la esperanza.
 - (d) Calcular la varianza.
2. Aplicar la función que calcula la varianza a la variable discreta definida con distribución asimétrica.
3. Teniendo en cuenta la generación de variables aleatorias normales vistas en la sección §6.5.
 - (a) Observar que a medida que aumenta el tamaño de la muestra, la media muestral se aproxima cada vez más a la media poblacional.
 - (b) Estudiar el comportamiento variando los parámetros de las distribuciones consideradas, los tamaños muestrales y las cantidades de replicaciones.

Capítulo 7

Aplicaciones didácticas

En este capítulo mostraremos algunos ejemplos de diversas áreas que se pueden incluir en las clases utilizando el lenguaje R como herramienta. Estas aplicaciones tienen por objetivo mostrar distintos alcances de este programa y pretender ser motivadoras a la hora de despertar el interés de los alumnos.

7.1 Estimación del número π

Con el fin de dar una aproximación del número π , vamos a elegir un número de repeticiones que denotamos por n y “dispararemos” n veces en forma aleatoria en un cuadrado de superficie 4. Dicho formalmente, generamos datos uniformes en el intervalo $[-1, 1]$. Luego, colorearemos los disparos que disten menos de 1 del centro para calcular la proporción de disparos dentro del círculo unitario.

```
n=100000
x1=runif(n,-1,1)
# genera n datos con distribución uniforme en [-1,1]
x2=runif(n,-1,1)
color=0
# inicializa la variable

for (i in 1:n){
  if (x1[i]^2+x2[i]^2<1) {color[i]="brown1"}
  else {color[i]="darkseagreen1"}
}
# colorea según los puntos estén dentro del círculo unitario o no

windows(width=6.5,height=6.5,rescale="fit")
# abre una ventana con ciertas características para mantener la escala
plot(x1,x2,col=color)
# grafica los puntos
title(expression(paste("Aproximando ", pi)))

nro.pi=4*sum(color=="brown1")/n
# calcula una aproximación de pi
nro.pi
# muestra el valor almacenado
```

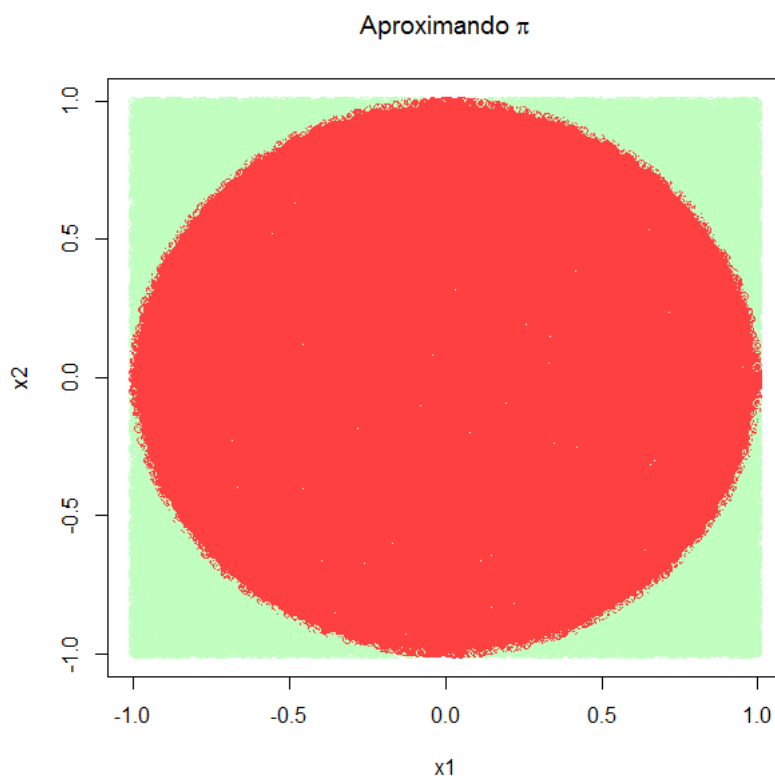


Figura 7.1: Ejemplo gráfico de cómo aproximar el número π

7.2 Estimadores de probabilidades o áreas

En esta actividad vamos a seleccionar la cantidad de ensayos para estimar, digamos n . Luego definiremos, a través de distintos colores, los sectores que simbolizarán el blanco, la zona más cercana y la región de fallo, para lo cual consideraremos la distancia al centro. Graficaremos los disparos generando muestras uniformes. Con estos datos, estimaremos las probabilidades de acertar con el tiro en cada uno de los sectores definidos previamente. Finalmente, generaremos una salida con tales probabilidades estimadas.

```

options(scipen=6)
# escribe números grandes
library(ggplot2)

n=100000
x1=runif(n,-1,1)
x2=runif(n,-1,1)

color=0
for (i in 1:n){
  if (x1[i]^2+x2[i]^2<1/3) {color[i]="orchid2"}
  else if (x1[i]^2+x2[i]^2<1) {color[i]="chartreuse"}
  else {color[i]="cadetblue1"}
}

windows(width=6.5, height=6.5, rescale="fit")
plot(x1,x2,col=color,xlab="",ylab="",main=paste("Tiro al blanco, n=",n), axes=F)

prob.blanco=sum(color=="orchid2")/n
prob.cerca=sum(color=="chartreuse")/n
prob.fallo=sum(color=="cadetblue1")/n

salida=list("Probabilidad de tiro al blanco"=prob.blanco,"Probabilidad de tiro
cerca del blanco"=prob.cerca,"Probabilidad de tiro fallido"=prob.fallo)
salida

```

Tiro al blanco, n= 1000

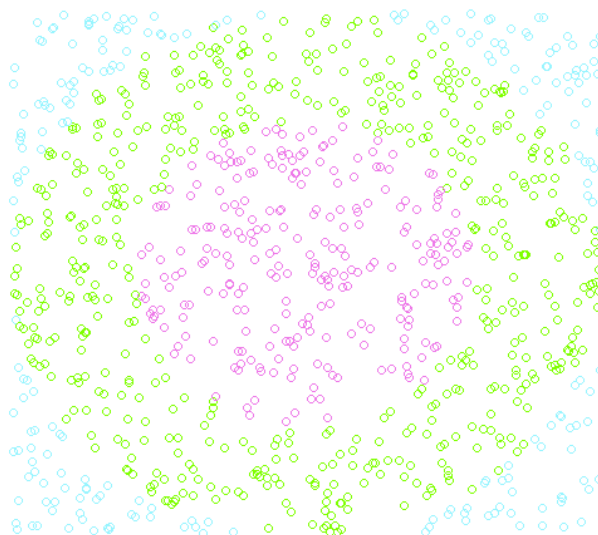


Figura 7.2: Gráfico de un tiro al blanco

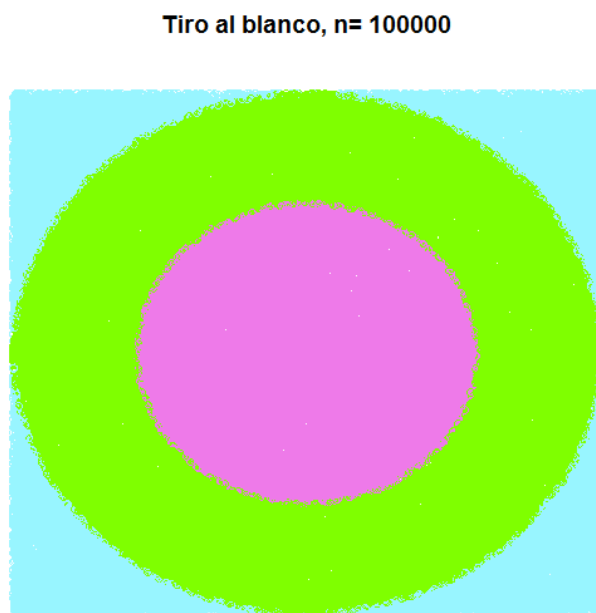


Figura 7.3: Gráfico de un tiro al blanco con más disparos

7.3 Iniciación en la distribución binomial

Supongamos que tenemos un especie de embudo colocado en forma vertical con el orificio más pequeño hacia arriba. Además, pensemos que está particionado donde cada terminal se identifica con un nodo. Un experimento aleatorio puede consistir en arrojar una bolita por la boca de este embudo y observar en qué nodo cayó. Si les preguntáramos a nuestros estudiantes alumnos cuál sería la probabilidad de que una bolita caiga en cada uno de los nodos, es probable que respondan que todos los nodos son equiprobables. Para que comprendan que en general esto no es verdad, se puede tomar como recurso didáctico es construir un **aparato de Galton** simulado, similar al que se muestra en el dibujo.

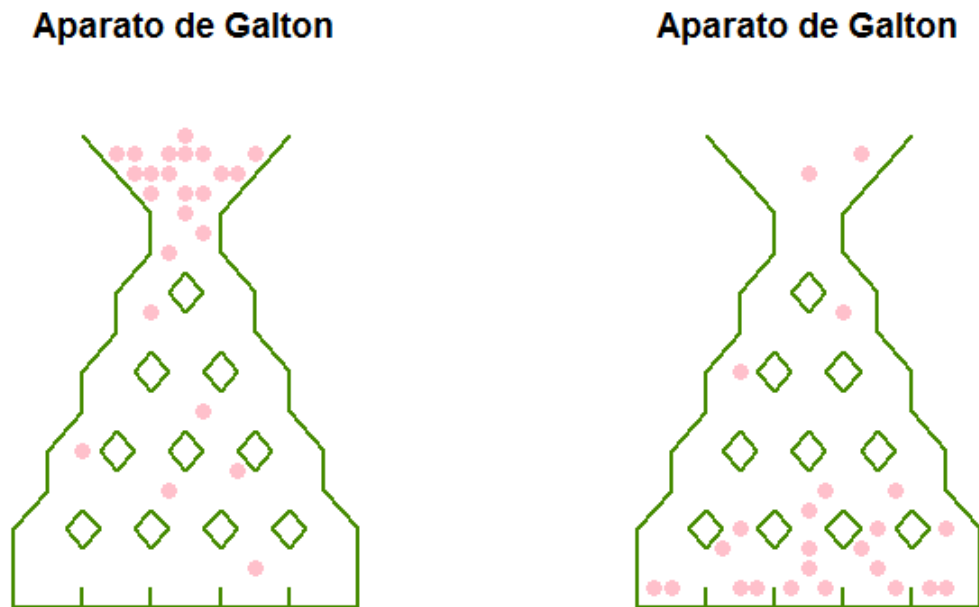


Figura 7.4: Ejemplo de un Quincunx o aparato de Galton

Para el código del gráfico, ver §8.

```

n=100000
p=1/3
# fija la probabilidad de éxito que en este caso es la de caer del lado izquierdo
x1=1*(runif(n,-1,1)<p)
# genera la decisión de la primera bifurcación
x2=1*(runif(n,-1,1)<p)
# genera la decisión de la segunda bifurcación en caso de haber caído por la izquierda

nodo=0
# define el nodo donde caerá cada bolita
for (i in 1:n){
  if(x1[i]< p) {nodo[i]=1*(x2[i]>p)+1}
  else {nodo[i]=2*(x1[i]>p)+1*(x2[i]>p)+1}
}
nodo
# toma los valores 1,2,3,4 que indican la posición del casillero final

sum(nodo=="1")/n
# calcula la probabilidad de que la bolilla caiga en el primer casillero
sum(nodo=="2")/n
# calcula la probabilidad de que la bolilla caiga en el segundo casillero
sum(nodo=="3")/n
# calcula la probabilidad de que la bolilla caiga en el tercer casillero
sum(nodo=="4")/n
# calcula la probabilidad de que la bolilla caiga en el cuarto casillero

```

7.4 Estimación puntual

Existen diversas estrategias para estimar un parámetro en una distribución dada. En este trabajo, vamos a comparar dos de ellas que son la estimación por momentos y la estimación por máxima verosimilitud. El **estimador de momentos** se construye igualando los momentos teóricos con los momentos empíricos. Dado que esto puede hacerse empleando momentos de distintos órdenes, resulta claro que este estimador depende de los órdenes elegidos. Por otra parte, el **estimador de máxima verosimilitud** es el que proviene de maximizar la probabilidad de extraer la muestra que hemos obtenido. El mismo se encuentra maximizando la **función de verosimilitud** que se define como la función de probabilidad conjunta pensada como función solamente de los parámetros y considerando constantes los datos muestrales. En muchas ocasiones, las expresiones correspondientes a estos estimadores coinciden, por lo cual no tiene sentido compararlos. Sin embargo, en las otras, resulta de interés analizar cuál es mejor y en qué sentido aventaja al otro. Con el fin de ilustrar estas situaciones, hemos seleccionamos cuatro distribuciones de las cuales, en tres de ellas las expresiones de los estimadores de momentos y máxima verosimilitud no coinciden.

En la siguiente tabla presentamos las expresiones correspondientes a estos estimadores para distintas distribuciones.

Table 7.1: Ejemplos de estimadores

Distribución	Parámetro	Estimador de momentos	Estimador de máxima verosimilitud
Exponencial	α	$\frac{1}{\bar{X}}$	$\frac{1}{\bar{X}}$
Normal	σ^2	$\frac{1}{n} \sum_{i=1}^n X_i^2 - \left(\frac{1}{n} \sum_{i=1}^n X_i\right)^2$	$\frac{1}{n} \sum_{i=1}^n (X_i - \mu)^2$
Uniforme en $[0, \theta]$	θ	$2\bar{X}$	$\max_{1 \leq i \leq n} \{X_i\}$
Exponencial desplazada	β	$\bar{X} - 1$	$\min_{1 \leq i \leq n} \{X_i\}$

Recordamos que la función de densidad de probabilidad para la distribución exponencial desplazada está dada por $f(x) = e^{-(x-\beta)}$ para $x > \beta$.

Como puede observarse en la tabla 7.1, el estimador de momentos es el mismo que el de máxima verosimilitud en el caso de la distribución exponencial.

Definimos **sesgo** de un estimador como la diferencia entre el valor esperado del mismo y el parámetro a estimar. Simbólicamente:

$$B(\hat{\theta}) = E(\hat{\theta}) - \theta$$

Dos de las propiedades más deseables para un estimador puntual son el insesgamiento y la consistencia.

- **Insesgamiento.** Se dice que un estimador es **insesgado** cuando su sesgo es nulo o bien su valor esperado coincide con el del parámetro a estimar.
- **Consistencia.** Se dice que un estimador es **consistente** cuando tiende en probabilidad al parámetro estimado. Para estimadores insesgados basta pedir que su varianza tienda a 0 a medida que crece el tamaño muestral.

A continuación, compararemos el cumplimiento de estas propiedades en los pares de estimadores puntuales propuestos, aplicando simulaciones de Montecarlo.

7.4.1 Varianza de una distribución normal

```
par(mfrow=c(2,1))
media=8
# fija el valor de la media
varianza=2
# fija el valor de la varianza

# media=5
# varianza=1

# inicialización de variables
mues.norm=0
mom.sig.norm=0
mv.sig.norm=0

# cálculo de los estimadores para muestras de tamaños 10,20,...,1000
for (i in 1:100){
  mues.norm=rnorm(10*i,media,varianza)
  mom.sig.norm[i]=mean(mues.norm^2)-mean(mues.norm)^2
  mv.sig.norm[i]=mean((mues.norm-media)^2)
}

# SESGO: gráfico de la tendencia de ambos estimadores al aumentar el tamaño muestral
x=10*(1:100)

plot(x,mom.sig.norm,type="l",col="lightblue1",lwd=2, xlab="tamaño muestral",
     ylab="estimación", main=paste("Varianza de una distribución normal con media ",
     media," y varianza ",varianza))
points(x,mv.sig.norm,type="l",lty="dotted",col="hotpink",lwd=2)
abline(varianza*2,0,col="tan3")
```

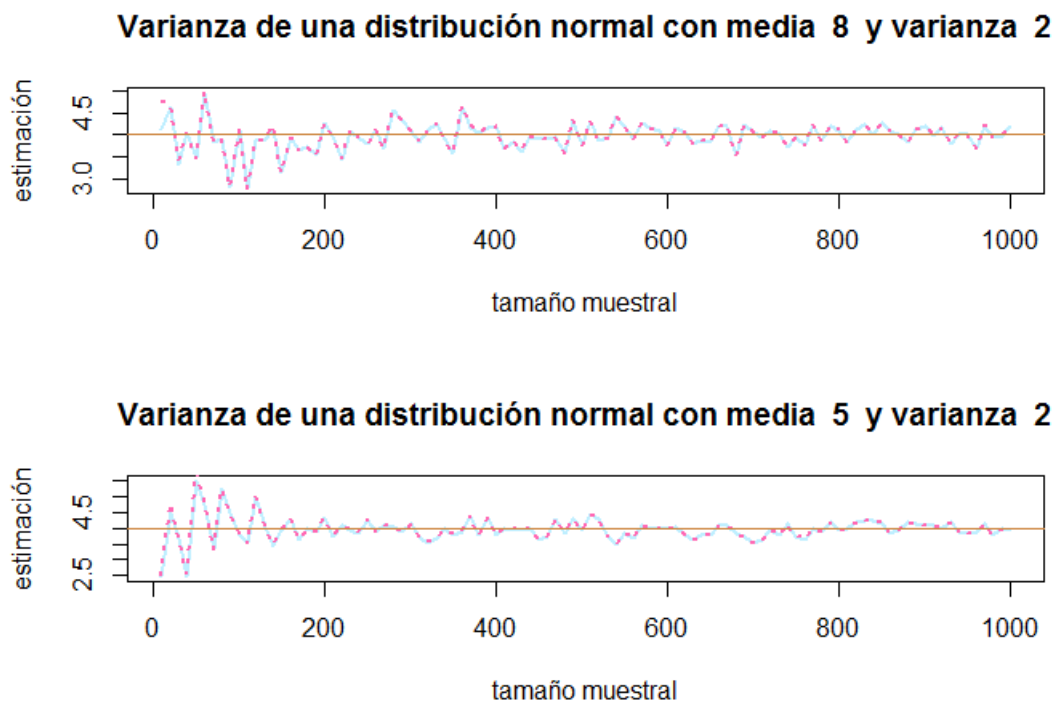


Figura 7.5: Ejemplos de estimación de la varianza de una distribución normal

En la figura 7.5 se aprecia que el comportamiento de ambos estimadores respecto del sesgo es similar para todos los tamaños muestrales.

7.4.2 Varianza-ECM

Las dos propiedades deseables para un estimador puntual que fueron citadas anteriormente, pueden combinarse aplicando el **criterio del error cuadrático medio**.

El error cuadrático medio de un estimador se define como la esperanza del cuadrado de la diferencia entre el estimador y el parámetro. Simbólicamente:

$$ECM(\hat{\theta}) = E \left[(\hat{\theta}) - \theta \right]^2$$

Se puede demostrar que el ECM es la suma del cuadrado del sesgo del estimador y la varianza del mismo. En el caso de estimadores insesgados, el error cuadrático medio coincide con la varianza del estimador.

```

mat.mom.var.nor=matrix(0,nrow=500,ncol=100)
mat.mv.var.nor=matrix(0,nrow=500,ncol=100)

for(i in 1:500){
  for(j in 1:100){
    muestra=rnorm(j+2,media,varianza)
    mat.mom.var.nor[i,j]=mean(muestra^2)-(mean(muestra))^2
    mat.mv.var.nor[i,j]=mean((muestra-media)^2)
  }
}

variab.mom=apply(mat.mom.var.nor,2,var)
variab.mv=apply(mat.mv.var.nor,2,var)

x=(1:100)
plot(x,variab.mom,type="l",col="violetred",lwd=2,xlab="tamaño muestral",
     ylab="estimación",main=paste("Estimadores de la varianza con media ",media," y
     varianza ",varianza))
text(locator(1),col="violetred","Estimador de momentos")
points(x,variab.mv,type="l",col="seagreen2",lwd=2)
text(locator(1),col="seagreen2","Estimador de máxima verosimilitud")

```

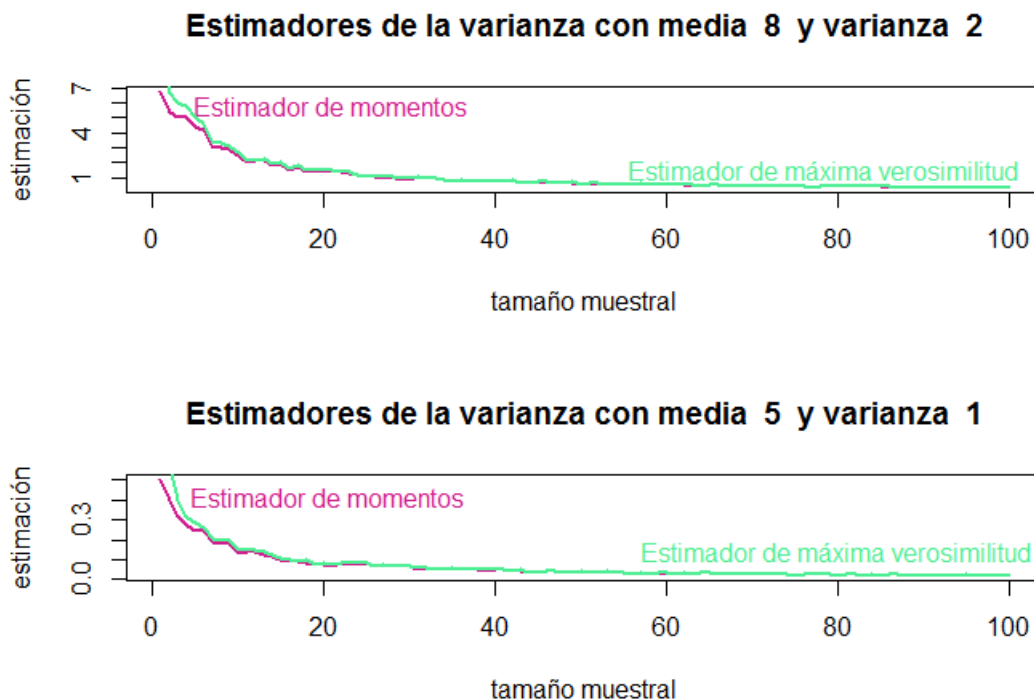


Figura 7.6: Comparación de distintos estimadores para la varianza de una distribución normal

En la figura 7.6 se aprecia que para valores muestrales pequeños el ECM del estimador de momentos aventaja al del de máxima verosimilitud, pero que para muestras grandes se comportan en forma similar.

7.4.3 Límite superior de una distribución uniforme

```
tita=3
# tita=1

# inicialización de las variables
mom.tita.unif=0
mv.tita.unif=0
# cálculo de los estimadores para muestras de tamaños 10,20,...,1000
for (i in 1:100){
  mues.unif=runif(10*i,0,tita)
  mom.tita.unif[i]=2*mean(mues.unif)
  mv.tita.unif[i]=max(mues.unif)
}

# SESGO: gráfico de la tendencia de ambos estimadores cuando el tamaño de la
# muestra aumenta
x=10*(1:100)
plot(x,mom.tita.unif,type="l",col="chartreuse",lwd=2,xlab="tamaño muestral",
     ylab="estimación",main=paste("Máximo uniforme con parámetro ",tita))
points(x,mv.tita.unif,type="l",col="chocolate1",lwd=2)
text(locator(1),col="chartreuse","Estimador de momentos")
text(locator(1),col="chocolate1","Estimador de máxima verosimilitud")
abline(tita,0,col="darkmagenta",lwd=1)
```

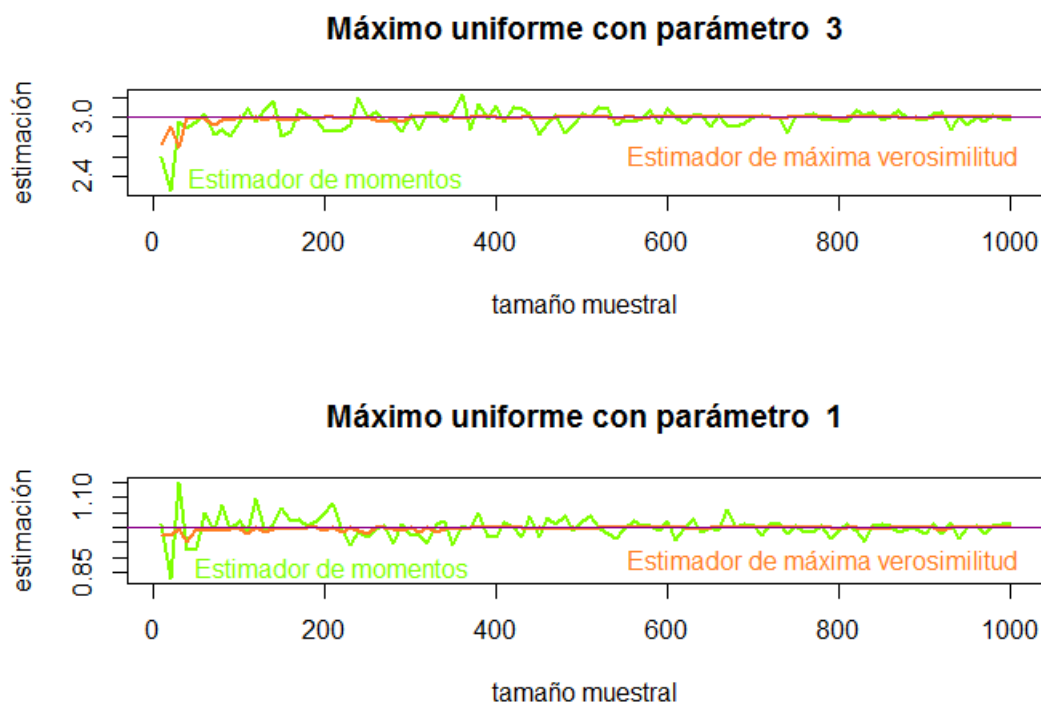


Figura 7.7: Comparación de estimadores para el límite superior de una distribución uniforme

En la Figura 7.7 se puede apreciar que, para todos los tamaños muestrales, el estimador de máxima verosimilitud tiene menor sesgo que el de momentos y converge más rápidamente al verdadero valor.

```
mat.mom.unif=matrix(0,nrow=500,ncol=100)
mat.mv.unif=matrix(0,nrow=500,ncol=100)

for(i in 1:500){
  for(j in 1:100){
    muestra=runif(j+2,0,tita)
    mat.mom.unif[i,j]=2*mean(muestra)
    mat.mv.unif[i,j]=max(muestra)
  }
}

variab.mom=apply(mat.mom.unif,2,var)
variab.mv=apply(mat.mv.unif,2,var)

x=(1:100)

plot(x,variab.mom,type="l",col="mediumspringgreen",lwd=2,xlab="tamaño muestral",
     ylab="estimación",main=paste("Estimadores de la varianza con parámetro ",tita))
text(locator(1),col="mediumspringgreen","Estimador de momentos")
points(x,variab.mv ,type="l",col="indianred1",lwd=2)
text(locator(1),col="indianred1","Estimador de máxima verosimilitud")
```

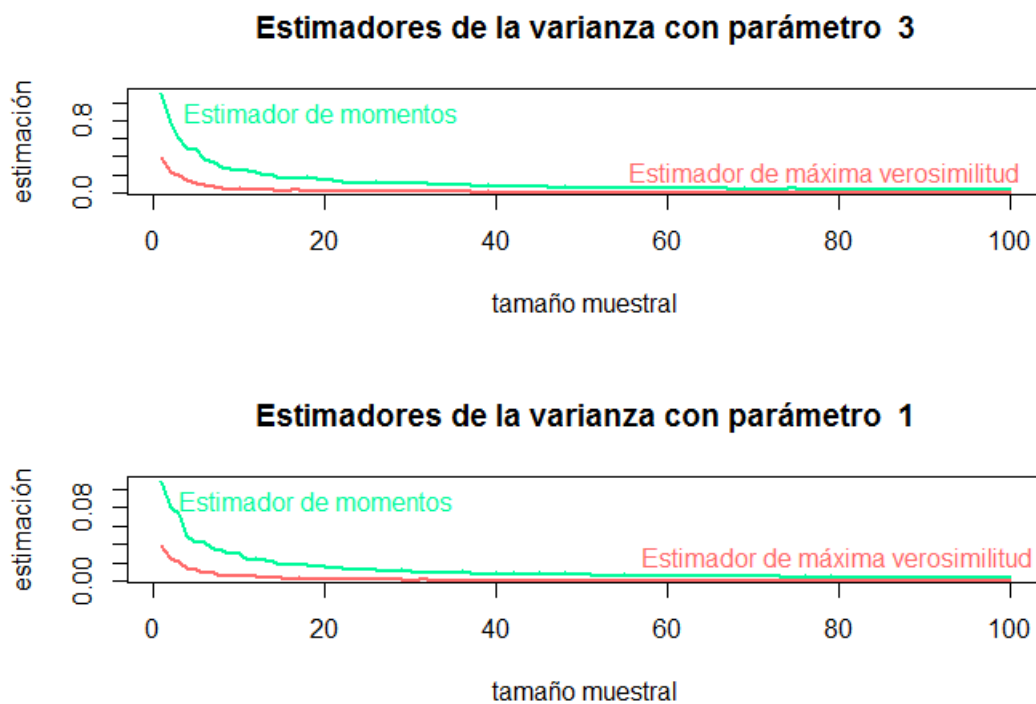


Figura 7.8: Comparación de estimador para el caso uniforme

En la figura 7.8 se puede observar que el estimador de máxima verosimilitud logra menor variabilidad que el estimador de momentos para todos los tamaños muestrales.

7.4.4 Estimación por intervalos

Se denomina intervalo de confianza, para un parámetro desconocido, a un intervalo construido a partir de una muestra, que se desea contenga al verdadero valor del parámetro. Esto no siempre ocurrirá.

La proporción de veces que el intervalo cubre el verdadero valor del parámetro recibe el nombre de **nivel de confianza** y se designa habitualmente $1 - \alpha$.

Para la construcción de un determinado intervalo de confianza es necesario definir las variables aleatorias que permitirán, en cada muestra, calcular el límite superior e inferior del intervalo.

Lo primero que haremos en esta sección es escribir un código para generar una muestra normal con el fin de comparar las longitudes de los intervalos de confianza en los casos en que la varianza resulta conocida y en aquellos en los cuales la varianza es desconocida.

```
mu=10
# fija la media de una normal
sigma=3
# fija la varianza de una normal
n=30
# fija el tamaño muestral
nivel=0.95
# fija el nivel de confianza

m.norm=rnorm(n,mu,sigma)
# genera una muestra
med.mues=mean(m.norm)
# calcula la media muestral
des.mues=sd(m.norm)
# calcula el desvío muestral

l.inf.vc=med.mues-qnorm(nivel+(1-nivel)/2)*sigma/sqrt(n)
# calcula el límite inferior del intervalo de confianza con varianza conocida
l.sup.vc=med.mues+qnorm(nivel+(1-nivel)/2)*sigma/sqrt(n)
# calcula el límite superior del intervalo de confianza con varianza conocida
l.inf.vd=med.mues-qt(nivel+(1-nivel)/2,n-1)*des.mues/sqrt(n)
# calcula el límite inferior del intervalo de confianza con varianza desconocida
l.sup.vd=med.mues+qt(nivel+(1-nivel)/2,n-1)*des.mues/sqrt(n)
# calcula el límite superior del intervalo de confianza con varianza desconocida
salida=rbind(c(l.inf.vc,l.sup.vc),c(l.inf.vd,l.sup.vd))
rownames(salida)=c("Varianza conocida","Varianza desconocida")
colnames(salida)=c("Límite inferior","Límite superior")
round(salida,2)
```

Frecuentemente se confunde el nivel de confianza exacto con el asintótico. Para ilustrar estos conceptos vamos a diseñar una simulación de n intervalos y vamos a calcular el porcentaje de cobertura del parámetro de interés en cada caso. Es decir, generaremos varias replicaciones con el objeto de entender el significado del nivel de confianza.

```
par(mfrow=c(2,1))
mu=10
sigma=3
n=30
nivel=0.95
l.inf.vc=0
l.sup.vc=0
repli=1000
med.mues=0
niv.de.conf=function(n,mu,sigma,nivel,repli){
  m.norm=matrix(0,nrow=n,ncol=repli)
  for (i in 1:repli){
    m.norm[,i]=rnorm(n,mu,sigma)
    med.mues[i]=mean(m.norm[,i])
    l.inf.vc[i]=med.mues[i]-qnorm(nivel+(1-nivel)/2)*sigma/sqrt(n)
    l.sup.vc[i]=med.mues[i]+qnorm(nivel+(1-nivel)/2)*sigma/sqrt(n)
  }
  return(list(l.inf.vc,l.sup.vc))
}

sal.int=niv.de.conf(30,10,3,0.95,100)
liminf=unlist(sal.int[[1]])
limsup=unlist(sal.int[[2]])
col=0
for(i in 1:repli){
  ifelse((liminf[i]<mu & limsup[i]>mu),col[i]<-8,col[i]<-2)
}

cobertura=1-sum(col=="2")/length(col)
# estima el nivel de cobertura de los intervalos con respecto al parámetro
cobertura
plot(0,0,xlim=c(0,100),ylim=c(mu-sigma,mu+sigma),xlab="",ylab="")
for(i in 1:repli){
  segments(i,sal.int[[1]][i],i,sal.int[[2]][i],col=col[i],lwd=2)
  abline(mu,0,col=3,lwd=2)
}

title(paste("Nivel de confianza con cobertura=",cobertura),col.main="royalblue")
```

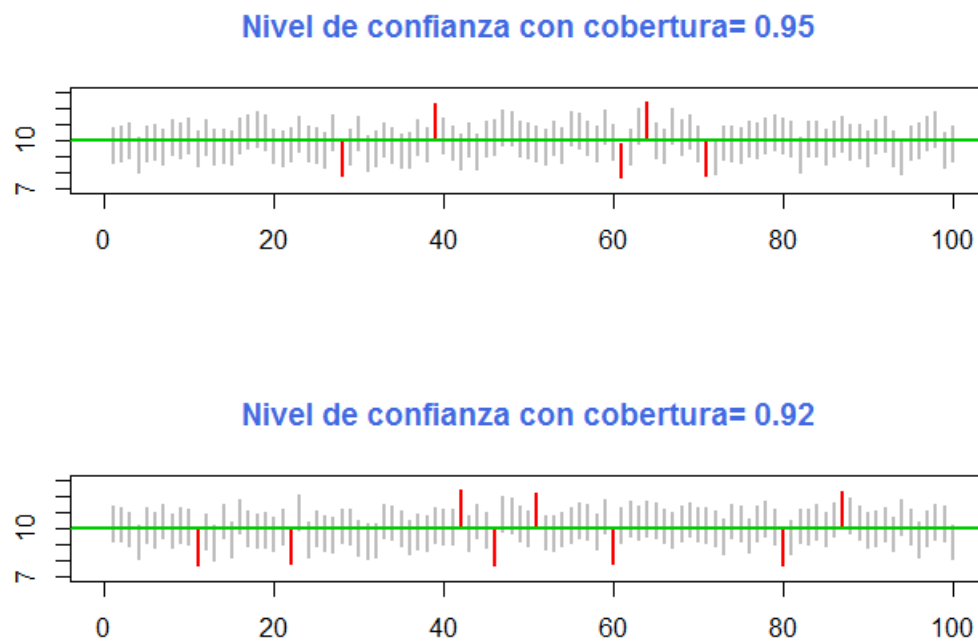



Figura 7.9: Ejemplos de niveles de coberturas en intervalos de confianza

En la figura 7.9 se aprecia que el nivel de confianza varía en cada simulación y se aproxima al nivel fijado en forma teórica.

7.5 Nivel de significación de un test

7.5.1 Test para la diferencia de medias de poblaciones normales

Suponiendo las varianzas conocidas

Vamos a generar un código que permita diseñar una prueba o test.

```

n1=20
n2=24
mu1=20
mu2=25
sigma1=2
sigma2=4
decis=0

dif.medias=function(n1,n2,mu1,mu2,sigma1,sigma2,val.obs1,val.obs2){
  xraya1=mean(val.obs1)
  xraya2=mean(val.obs2)
  estad=((xraya1-xraya2)-(mu1-mu2))/sqrt((sigma1^2)/n1+(sigma2^2)/n2)
  pvalor=2*(1-pnorm(abs(estad)))
  decido=1*(pvalor<0.05)
  decis=ifelse(decido=="1","rechazar la hipótesis nula","no rechazar la
    hipótesis nula")
  return(list(c("p-valor"=round(pvalor,3),"Se toma la decisión de
    "=decis)))
}

obs1=rnorm(n1,mu1,sigma1)
obs2=rnorm(n2,mu2,sigma2)

dif.medias(20,24,20,25,2,4,obs1,obs2)

```

Ahora replicaremos el test para controlar el nivel. Observar que en el lenguaje R, se pueden poner varias sentencias en un mismo renglón separándolas con punto y coma.

```

n1=20; n2=24; mu1=20; mu2=25; sigma1=2; sigma2=4
sal=0; pvalue=0; estad=0; conclu=0; pvalor=0; decido=0
xraya1=0; xraya2=0
obs1=matrix(0,nrow=n1,ncol=200)
obs2=matrix(0,nrow=n2,ncol=200)

for (i in 1:200){
  obs1[,i]=rnorm(n1,mu1,sigma1)
  obs2[,i]=rnorm(n2,mu2,sigma2)
  xraya1[i]=mean(obs1[,i])
  xraya2[i]=mean(obs2[,i])
  estad[i]=((xraya1[i]-xraya2[i])-(mu1-mu2))/sqrt((sigma1^2)/n1+(sigma2^2)/n2)
  pvalor[i]=2*(1-pnorm(abs(estad[i])))
  decido[i]=1*(pvalor[i]<0.05)
}

salida.test=round(data.frame(pvalor,decido),2)
salida.test
nivel_signif_observ=apply(salida.test,2,mean)[2]
print(paste("El nivel de significación de la observación es del ",
  nivel_signif_observ*100,"%"))

```

Suponiendo las varianzas desconocidas

```
dif.medias.t=function(n1,n2,mu1,mu2,val.obs1,val.obs2){
  xraya1=mean(val.obs1)
  xraya2=mean(val.obs2)
  sigma1=sd(val.obs1)
  sigma2=sd(val.obs2)
  des.amalg=sqrt(((n1-1)*sigma1^2+(n2-1)*sigma2^2)/(n1+n2-2))
  estad=((xraya1-xraya2)-(mu1-mu2))/(des.amalg*sqrt(1/n1+1/n2))
  pvalor=2*(1-pt(abs(estad),n1+n2-2))
  decido=1*(pvalor<0.05)
  decis=ifelse(decido=="1","rechazar la hipótesis nula","no rechazar
    la hipótesis nula")
  return(list(c("p-valor"=round(pvalor,3),"Se toma la decisión de "
    =decis)))
}

obs1=rnorm(15,10,2)
obs2=rnorm(20,7,3)
dif.medias.t(15,20,10,7,obs1,obs2)
```

7.6 Nivel asintótico**7.6.1 Test para el parámetro p de la distribución binomial o para la proporción poblacional**

```
dif.proporc=function(n1,n2,p1,p2,x1,x2){
  estad=(x1/n1-x2/n2-(p1-p2))/sqrt(p1*(1-p1)/n1+p2*(1-p2)/n2)
  pvalor=2*(1-pnorm(abs(estad)))
  decido=1*(pvalor<0.05)
  decis=ifelse(decido=="1","rechazar la hipótesis nula","no rechazar la
    hipótesis nula")
  return(list(c("p-valor"=round(pvalor,3),"Se toma la decisión de "
    =decis)))
}

dif.proporc(35,38,0.7,0.65,sum(rbinom(35,1,0.7)),sum(rbinom(38,1,0.65)))
```

Nuevamente, replicamos el test para controlar el nivel.

```

n1=30; n2=44; p1=0.3; p2=0.25
sal=0; pvalue=0; estad=0; conclu=0; pvalor=0; decido=0
x1=sum(rbinom(n1,1,p1)); x2=sum(rbinom(n2,1,p2))
obs1=matrix(0,nrow=n1,ncol=200)
obs2=matrix(0,nrow=n2,ncol=200)

for (i in 1:200){
  obs1[,i]=rbinom(n1,1,p1)
  obs2[,i]=rbinom(n2,1,p2)
  x1[i]=sum(obs1[,i])
  x2[i]=sum(obs2[,i])
  estad[i]=(x1[i]/n1-x2[i]/n2-(p1-p2))/sqrt(p1*(1-p1)/n1+p2*(1-p2)/n2)
  pvalor[i]=2*(1-pnorm(abs(estad[i])))
  decido[i]=1*(pvalor[i]<0.05)
}

salida.test=round(data.frame(pvalor,decido),2)
salida.test
nivel_signif_observ=apply(salida.test,2, mean)[2]
print(paste("El nivel de significación de la observación es del ",
  nivel_signif_observ*100,"%"))

```

7.7 Transformaciones de datos por filas

En esta sección vamos a trabajar con un archivo que guarda la información de la Tabla 8.11.

```

recep=read.csv("C:/Users/Usuario-PC/Dropbox/CursoR/Datos/recepcionistas.csv", sep=";")
recep
attach(recep)

```

Como primera tarea vamos a realizar gráficos de cajas para visualizar diferencias entre las puntuaciones de los distintos jueces.

```

boxplot(split(Cordialidad,Juez),horizontal=T,col=c("seagreen1","salmon"),
  main="Puntaje de cordialidad según juez")
boxplot(split(Presencia,Juez),horizontal=T,col=c("seagreen1","salmon"),
  main="Puntaje de presencia según juez")
boxplot(split(Idioma,Juez),horizontal=T,col=c("seagreen1","salmon"),
  main="Puntaje de idioma según juez")

```

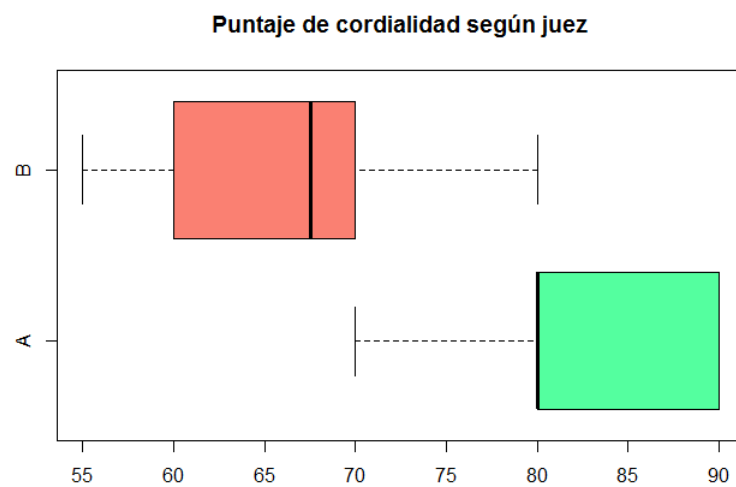


Figura 7.10: Boxplot comparando cordialidad

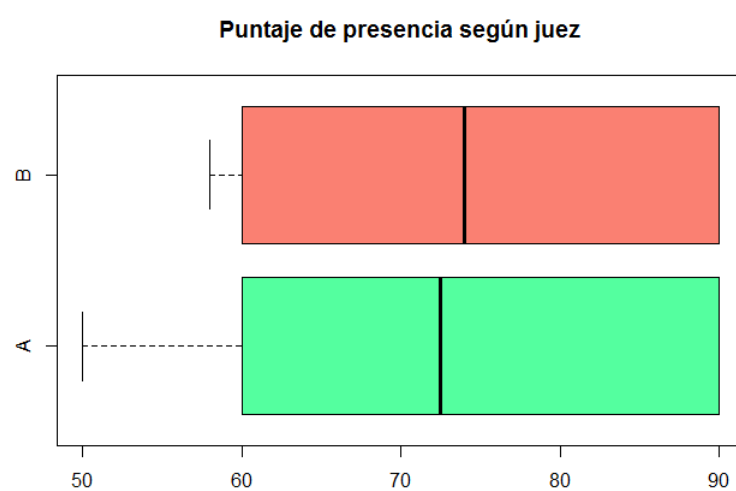


Figura 7.11: Boxplot comparando presencia

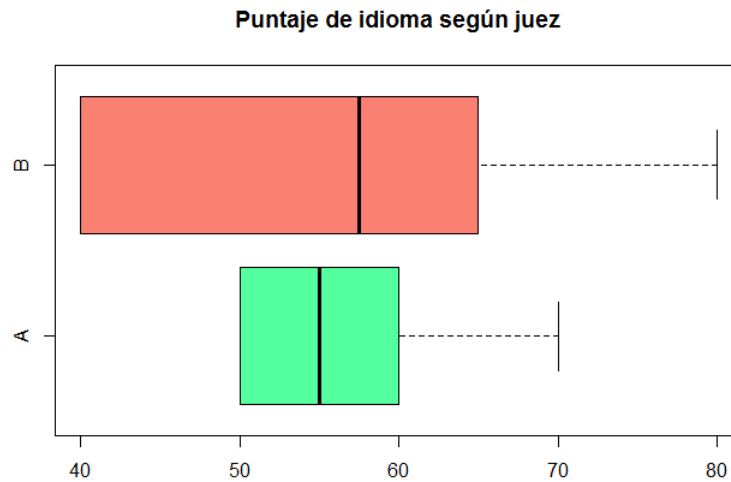


Figura 7.12: Boxplot comparando idioma

Ahora procederemos a realizar una transformación de datos por fila.

```
medias=apply(recep[,2:4],1,mean)
rangos=apply(recep[,2:4],1,max)-apply(recep[,2:4],1,min)

categ=dim(recep)[2]

med=cbind(rep(medias,categ))
rang=cbind(rep(rangos,categ))

rec.trans=(recep[,2:4]-med)/rang
# transforma los datos para obtener media 0 y desvío 1
```

Nuevamente, visualizamos diferencias entre los jueces pero con los datos transformados.

```
boxplot(split(rec.trans$Cordialidad,Juez),horizontal=T,
  col=c("royalblue","navajowhite"),main="Puntaje de cordialidad según juez")
boxplot(split(rec.trans$Presencia,Juez),horizontal=T,
  col=c("royalblue","navajowhite"),main="Puntaje de presencia según juez")
boxplot(split(rec.trans$Idioma,Juez),horizontal=T,
  col=c("royalblue","navajowhite"),main="Puntaje de idioma según juez")
```

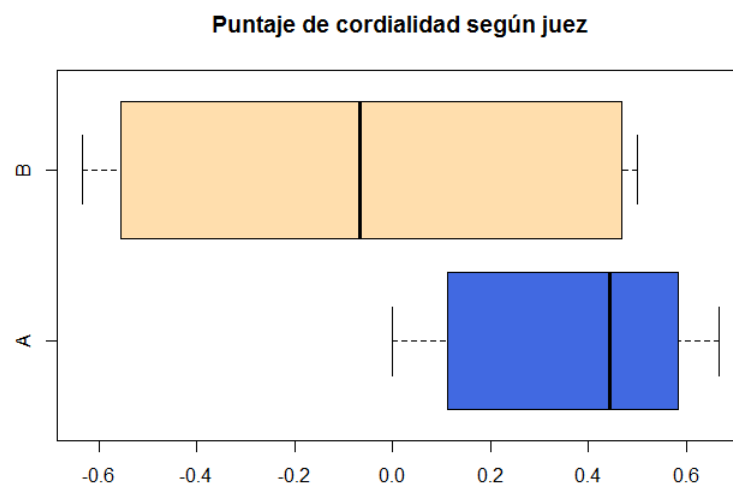


Figura 7.13: Boxplot comparando cordialidad con datos estandarizados

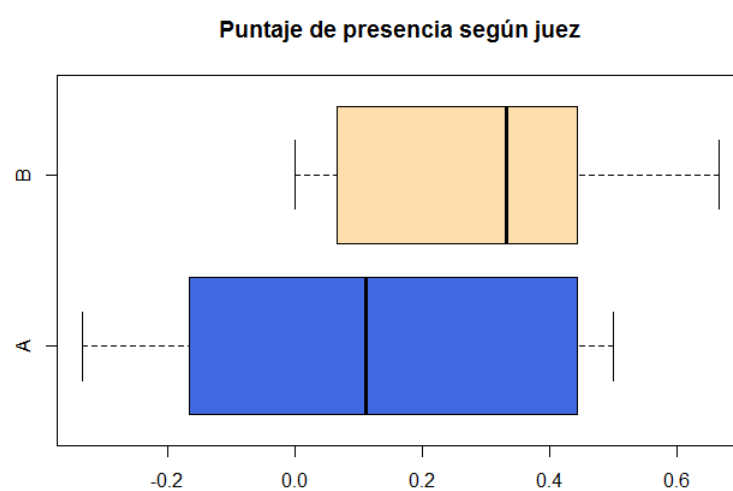


Figura 7.14: Boxplot comparando presencia con datos estandarizados

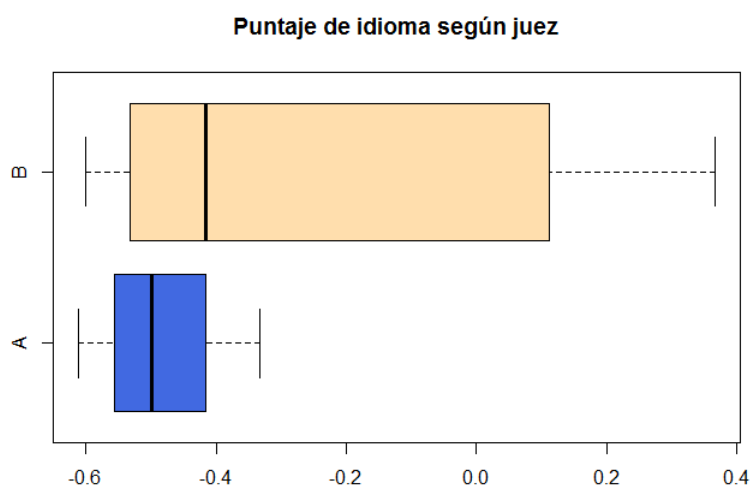


Figura 7.15: Boxplot comparando idioma con datos estandarizados

Comparamos las puntuaciones de manera gráfica.

```
plot(1:12,rec.trans$Cordialidad,type="o",col="red1",lwd=2,xlab="Candidatas",
     ylim=c(-1,1),ylab="Puntuación estandarizada",xlim=c(1,12))
points(1:12,rec.trans$Presencia,type="o",col="olivedrab1",lwd=2)
points(1:12,rec.trans$Idioma,type="o",col="turquoise1",lwd=2)
title("Comparación de perfiles")
legend.text=c("Cordialidad","Presencia","Idioma")
legend(10,1,legend.text,text.col=c("red1","olivedrab1","turquoise1"),cex=0.7,
      text.width=1.5)
```

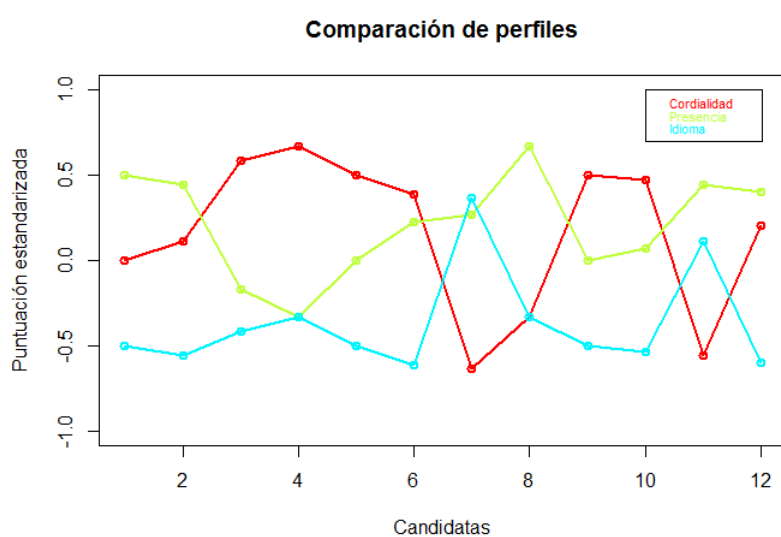


Figura 7.16: Comparación de perfiles con datos estandarizados

7.8 Transformaciones de datos por columnas

El primer objetivo de esta sección es mostrar un procedimiento para hacer que las variables sean comparables. Utilizaremos los datos que se muestran en la Tabla 8.8.

```
galle=read.csv("C:/Users/Usuario-PC/Dropbox/CursoR/Datos/galletitas.csv",header=TRUE,
  sep=";")
gallet=galle[,2:6]

# Cálculo de media y desvío por columna
medias=apply(gallet,2,mean)
desvios=apply(gallet,2,sd)

marcas=dim(gallet)[1]
variab=dim(gallet)[2]

# Conversión en variables comparables
med=matrix(rep(medias,marcas),byrow=T,nrow=marcas)
des=matrix(rep(desvios,marcas),byrow=T,nrow=marcas)

gall.tran=(gallet-med)/des

# Verificación de la transformación
round(apply(gall.tran,2,mean),3)
round(apply(gall.tran,2,sd),3)

attach(gall.tran)
head(gall.tran)

nombres=c("Calorías","Carbohidratos","Proteínas","Grasas","Sodio")
boxplot(Valor.energetico..kcalorias.100g., Carbohidratos..g.100g., Proteinas..g.100g.,
  Grasas.totales..g.100g., Sodio..mg.100g.,col=terrain.colors(8),names=nombres,
  cex.axis=0.6, ylab="",main="Valores nutricionales")
```

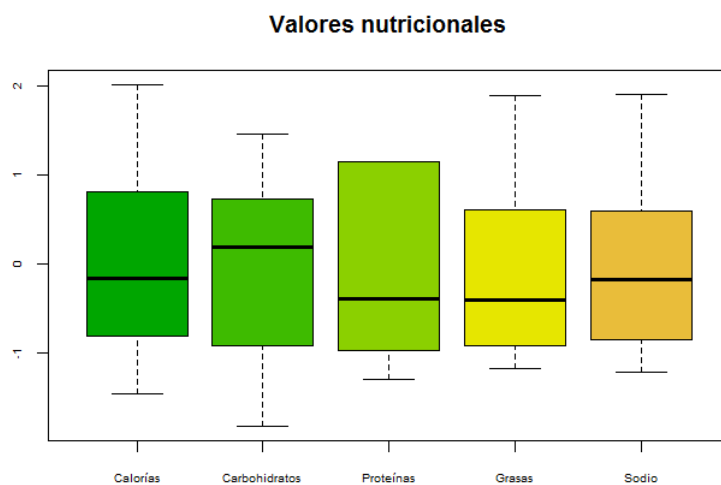


Figura 7.17: Boxplot comparando valores nutricionales de galletitas

El segundo objetivo es mostrar un procedimiento que permita simetrizar un conjunto de datos.

```

datos=c(1,4,16,36,143,196,257,288)
asim=mean(datos)-median(datos)
asim
raiz.datos=sqrt(datos)                                # transforma los datos mediante la raíz
                                                         cuadrada

asimraiz=mean(raiz.datos)-median(raiz.datos)
asimraiz
median(datos)                                           # calcula la mediana de los datos
sqrt(median(datos))                                    # calcula la raíz cuadrada de la mediana
                                                         de los datos
median(raiz.datos)                                     # calcula la mediana de los datos
                                                         transformados

```

No coinciden la raíz de la mediana y la mediana de la raíz.

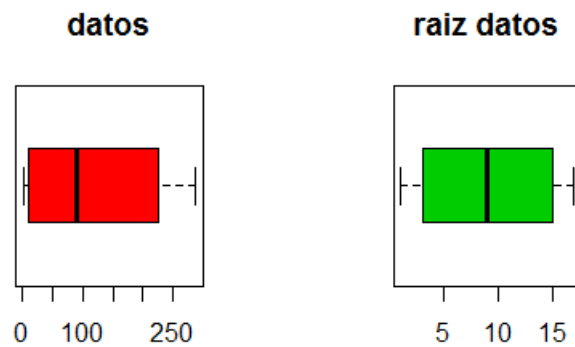


Figura 7.18: Transformaciones para simetrizar

En la Figura 7.18 se aprecia que la transformación de los datos elegida logra simetrizarlos.

Nuestro tercer y último objetivo consiste en mostrar una familia de transformaciones que se utiliza para normalizar un conjunto de datos: Familia de Transformaciones de Box & Cox. Para verificar la normalización de los datos se aplica una prueba de normalidad o bondad de ajuste a la distribución normal. En este caso, vamos a aplicar la prueba de Shapiro que requiere de la instalación de la librería MASS. Los datos elegidos siguen las Tablas 8.9 y 8.10.

```
library(MASS)

internet=read.csv("C:/Users/Usuario-PC/Dropbox/CursoR/Datos/sitiosweb.csv", sep=";")
attach(internet)
names(internet)

est=shapiro.test(Estatura)[[2]]
tpo= shapiro.test(Tiempo)[[2]]
aut=shapiro.test(Autos)[[2]]
cig=shapiro.test(Cigarrillos)[[2]]

sal=round(c(est,tpo,aut,cig),3)
sal

cart=0
for(i in 1:4){
  if (sal[i]<0.05) {cart[i]="No normal"}
  else {cart[i]="Normal"}
}

salida=rbind(sal, cart)
colnames(salida)=c("Estatura", "Tiempo de uso", "Autos en la manzana",
  "Cigarrillos consumidos")
salida=data.frame(salida)
salida

par(mfrow=c(2,2))

boxcox(Estatura~1,plotit=T)
esta=Estatura**(-2)
estat=shapiro.test(esta)[[2]]

boxcox(Tiempo~1,plotit=T)
tpo=Estatura**(1/2)
tiemp=shapiro.test(tpo)[[2]]

boxcox(Autos~1,plotit=T)
autos=Autos**(0.8)
autt=shapiro.test(autos)[[2]]

boxcox((Cigarrillos+1)~1,plotit=T)
ciga=log(Cigarrillos+1)
cig=shapiro.test(ciga)[[2]]

trans=round(c(estat,tiemp,autt,cig),3)
cartel=0
for(i in 1:4){
  if (trans[i]<0.05) {cartel[i]="No normal"}
  else {cartel[i]="Normal"}
}

salida.final=rbind(sal, cart, trans, cartel)
colnames(salida.final)=c("Estatura", "Tiempo de uso", "Autos en la manzana",
  "Cigarrillos consumidos")
salida.final=data.frame(salida.final)
salida.final
```

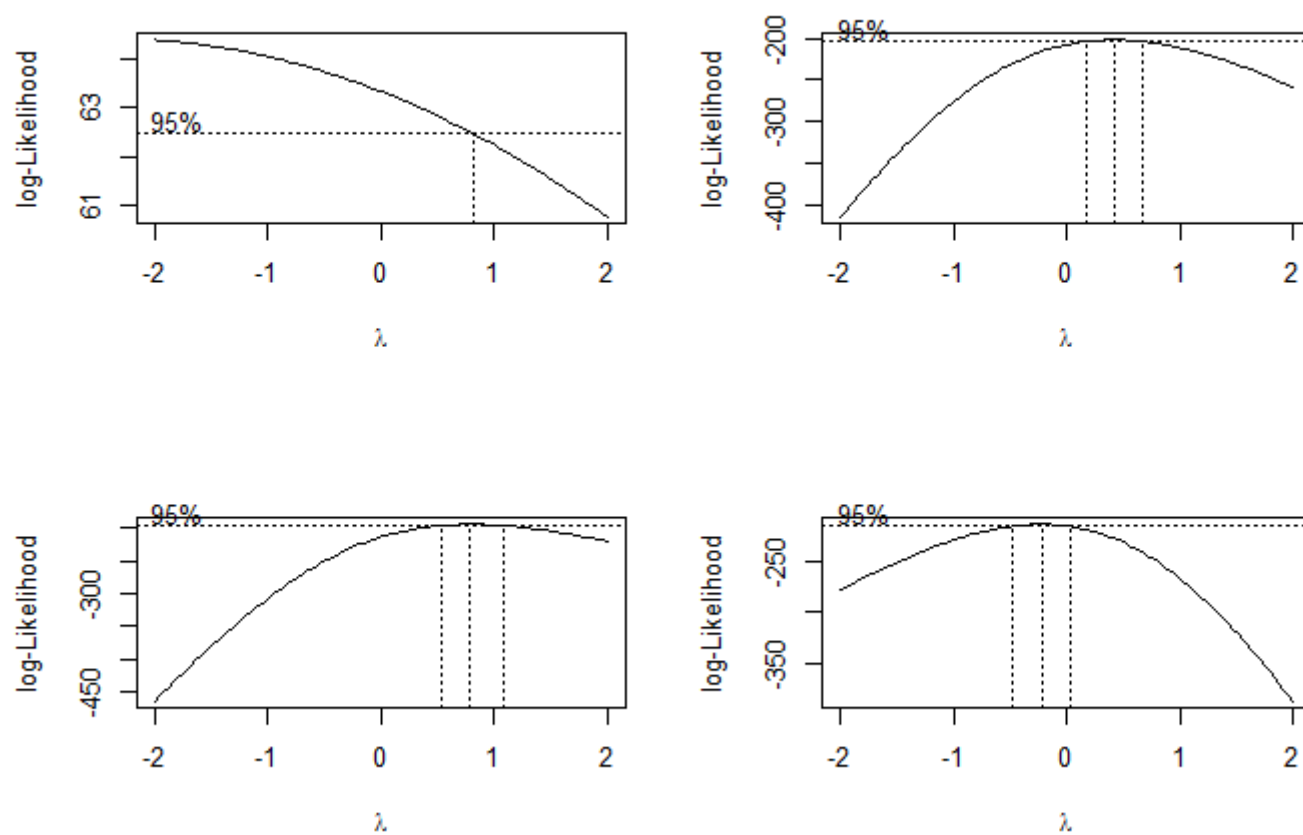


Figura 7.19: Gráficos de distintas pruebas de Shapiro

7.9 Ejercicios de revisión

1. Consideremos la siguiente imagen

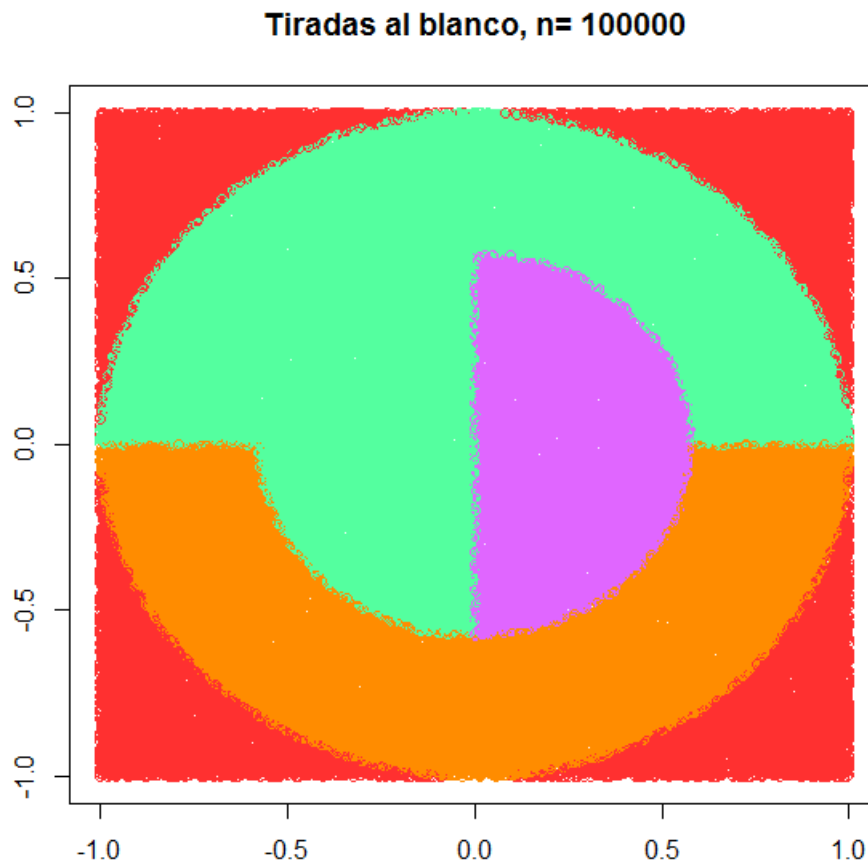


Figura 7.20: Ejemplo de tiro al blanco

Generar un código en R que produzca esta imagen y estimar las probabilidades de cada región dada por los diferentes colores, presentándolas en una salida con formato de tabla.

2. Siguiendo la idea que se mostró en este capítulo de la construcción de un aparato de Galton, se pide:
 - (a) extender el modelo pero agregándole un nivel más.
 - (b) reprogramar el aparato utilizando una probabilidad de 0.25.
3. Repetir el procedimiento realizado para comparar los estimadores de momentos y de máxima verosimilitud en el caso de una variable aleatoria con distribución exponencial desplazada.
4. Replicar el código que evalúa el nivel de cobertura de intervalos de confianza en los siguientes casos:
 - (a) para varianza desconocida.
 - (b) para muestras de distinto tamaño.
 - (c) para muestras no normales.

Capítulo 8

Apéndice

En este apartado incluiremos las tablas y algunos códigos de ejecución utilizadas en las secciones previas.

Table 8.1: Índice de masa corporal infantil

PACIENTE	EDAD	SEXO	PESO	TALLA	IMC	PIMC	CC	CatPeso
1	7	M	24.4	1.2	16.94	7.97	54	N
2	7	M	23.6	1.2	16.39	72.72	52	N
3	8	M	47.0	1.4	23.98	97.08	76	OB
4	7	F	24.0	1.2	16.67	83.88	63	N
5	7	M	23.9	1.2	16.60	45.85	56	N
6	10	M	41.0	1.4	20.92	87.33	78	SO
7	7	M	32.9	1.3	19.47	96.57	69	OB
8	7	M	22.4	1.2	15.56	32.88	52	N
9	7	M	28.7	1.3	16.98	80.77	60	N
10	9	M	31.4	1.3	18.58	92.72	69	SO
11	9	M	28.9	1.3	17.10	55.54	60	N
12	11	F	51.2	1.6	20.00	77.77	75	N
13	7	M	26.2	1.3	15.50	70.70	50	N
14	9	M	58.5	1.5	26.00	98.69	88	OB
15	9	F	23.7	1.3	14.02	3.25	58	D
16	11	M	25.5	1.3	15.09	2.07	73	D
17	12	M	49.7	1.7	17.20	38.08	75	N
18	7	M	39.6	1.3	23.43	98.75	76	OB
19	11	M	42.5	1.5	18.89	80.60	72	N
20	6	F	21.6	1.2	15.00	39.97	52	N
21	8	F	38.0	1.3	22.49	96.07	76	OB
22	8	F	26.6	1.2	18.47	71.06	54	N
23	7	F	20.4	1.2	14.17	3.44	52	D
24	10	F	23.7	1.3	14.02	2.02	56	D
25	7	M	21.4	1.2	14.86	56.86	56	N
26	8	M	45.7	1.4	23.32	98.99	78	OB
27	10	F	51.3	1.5	22.80	90.84	76	SO
28	7	F	28.0	1.3	16.57	57.50	57	N
29	9	F	26.9	1.3	15.92	44.77	57	N
30	10	M	43.9	1.5	19.51	84.89	76	N
31	7	F	25.7	1.2	17.85	75.49	54	N
32	8	M	28.2	1.3	16.69	62.17	56	N
33	8	F	36.8	1.3	21.78	94.25	74	SO
34	8	M	27.3	1.3	16.15	54.80	62	N
35	9	M	33.8	1.4	17.24	63.42	61	N
36	6	M	23.5	1.2	16.32	65.13	55	N
37	8	M	33.1	1.4	16.89	78.94	60	N
38	10	M	37.5	1.5	16.67	45.07	67	N
39	10	F	33.2	1.4	16.94	61.20	62	N
40	6	F	36.8	1.3	21.78	98.79	66	OB
41	8	M	29.2	1.3	17.28	86.86	56	SO
42	8	F	30.6	1.3	18.11	76.31	63	N
43	8	M	28.1	1.3	16.63	82.21	61	N
44	7	F	23.5	1.2	16.32	54.40	54	N
45	6	F	19.8	1.1	16.36	82.54	59	N
46	6	F	23.6	1.2	16.39	82.74	58	N
47	7	M	26.0	1.3	15.38	59.30	56	N
48	9	F	37.6	1.3	22.25	95.25	72	OB
49	6	M	25.5	1.2	17.71	83.68	57	N
50	9	M	40.0	1.3	23.67	95.78	70	OB

Table 8.2: Índice de masa corporal infantil (cont.)

PACIENTE	EDAD	SEXO	PESO	TALLA	IMC	PIMC	CC	CatPeso
51	11	M	80.8	1.6	31.56	99.27	107	OB
52	9	F	30.0	1.3	17.75	65.11	64	N
53	9	M	44.9	1.4	22.91	97.95	82	OB
54	10	M	51.6	1.4	26.33	97.67	82	OB
55	8	M	27.0	1.3	15.98	47.45	57	N
56	9	F	40.0	1.5	17.78	75.68	67	N
57	8	M	32.3	1.3	19.11	89.60	59	SO
58	7	M	24.5	1.2	17.01	84.40	57	N
59	11	F	48.5	1.6	18.95	74.03	69	N
60	9	F	43.5	1.3	25.74	97.80	83	OB
61	11	F	42.6	1.4	21.73	83.56	74	N
62	8	F	38.0	1.3	22.49	94.23	68	SO
63	6	F	23.9	1.2	16.60	76.79	56	N
64	11	F	52.0	1.5	23.11	91.36	80	SO
65	10	F	55.2	1.5	24.53	96.76	83	OB
66	10	F	48.0	1.4	24.49	94.85	81	SO
67	10	M	50.5	1.5	22.44	96.80	80	OB
68	6	M	19.0	1.1	15.70	33.15	52	N
69	11	M	32.0	1.5	14.22	5.94	62	N
70	9	M	33.5	1.4	17.09	71.52	67	N
71	8	M	38.0	1.3	22.49	95.43	71	OB
72	7	M	24.5	1.3	14.50	23.17	58	N
73	6	F	31.5	1.2	21.88	98.56	69	OB
74	6	M	21.2	1.2	14.72	26.76	57	N
75	6	F	19.9	1.1	16.45	79.81	54	N
76	6	F	26.7	1.2	18.54	91.55	64	SO
77	8	F	23.0	1.2	15.97	52.64	57	N
78	10	M	46.3	1.5	20.58	89.75	72	SO
79	9	F	38.2	1.4	19.49	89.74	68	SO
80	8	M	21.2	1.2	14.72	14.42	51	N
81	8	M	24.2	1.2	16.81	65.43	53	N
82	6	M	24.3	1.2	16.88	87.31	53	SO
83	7	F	30.0	1.3	17.75	85.98	57	SO
84	7	M	37.3	1.3	22.07	99.23	70	OB
85	12	M	49.4	1.6	19.30	69.60	70	N
86	8	F	24.7	1.3	14.62	20.54	52	N
87	10	M	35.0	1.4	17.86	50.19	63	N
88	8	M	22.3	1.2	15.49	41.62	56	N
89	8	F	35.5	1.4	18.11	83.59	65	N
90	9	F	38.4	1.3	22.72	93.29	76	SO
91	7	F	32.2	1.3	19.05	87.54	60	SO
92	10	F	40.3	1.5	17.91	45.88	60	N
93	10	M	40.5	1.4	20.09	84.74	71	N
94	12	F	59.5	1.6	23.24	89.87	87	SO
95	8	F	43.0	1.4	21.94	95.89	78	OB
96	8	F	20.3	1.2	14.10	4.44	51	D
97	7	F	22.0	1.2	15.28	14.73	51	N
98	12	F	58.0	1.6	22.66	88.63	85	SO
99	9	M	37.5	1.4	19.13	87.54	67	SO
100	8	F	41.3	1.4	21.07	95.33	70	OB

Table 8.3: Índice de masa corporal infantil (cont.)

PACIENTE	EDAD	SEXO	PESO	TALLA	IMC	PIMC	CC	CatPeso
101	6	F	18.5	1.1	15.29	22.20	52	N
102	6	M	19.1	1.2	13.26	13.58	51	N
103	7	F	32.5	1.3	19.23	86.11	68	SO
104	10	F	34.6	1.4	17.65	58.26	64	N
105	8	F	29.6	1.3	17.51	83.18	68	N
106	8	M	33.8	1.4	17.24	72.06	64	N
107	11	M	42.3	1.5	18.80	70.26	71	N
108	6	F	31.0	1.2	21.53	97.22	66	OB
109	7	F	23.4	1.2	16.25	71.20	58	N
110	8	M	21.5	1.2	14.93	4.28	50	D
111	8	M	21.3	1.2	14.79	28.29	53	N
112	11	F	39.5	1.4	20.15	80.18	79	N
113	9	M	31.5	1.3	18.64	76.20	58	N
114	8	M	30.3	1.3	17.93	74.22	61	N
115	7	M	22.3	1.2	15.49	39.01	57	N
116	12	M	60.2	1.6	23.52	95.27	87	OB
117	6	M	23.6	1.2	16.39	54.15	56	N
118	9	M	22.6	1.2	15.69	14.67	54	N
119	7	M	27.5	1.2	19.10	84.44	58	N
120	8	M	34.2	1.3	20.24	95.49	79	OB
121	11	F	46.5	1.5	20.67	79.36	74	N
122	9	M	54.8	1.4	27.96	99.17	89	OB
123	6	F	22.6	1.2	15.69	78.53	59	N
124	7	F	30.3	1.3	17.93	89.92	63	SO
125	6	F	19.7	1.2	13.68	34.06	54	N
126	9	M	35.4	1.3	20.95	89.83	69	SO
127	6	F	20.8	1.2	14.44	60.66	57	N
128	9	F	40.2	1.4	20.51	92.80	76	SO
129	10	F	57.3	1.5	25.47	97.86	84	OB
130	9	F	45.5	1.4	23.21	94.55	79	SO
131	9	F	28.4	1.3	16.80	32.34	77	N
132	8	F	34.6	1.4	17.65	81.33	67	N
133	9	F	26.4	1.3	15.62	37.21	59	N
134	9	F	35.0	1.4	17.86	82.71	69	N
135	7	F	23.7	1.2	16.46	81.57	61	N
136	8	M	26.4	1.3	15.62	42.47	56	N
137	11	F	52.4	1.4	26.73	96.67	83	OB
138	7	M	32.6	1.3	19.29	95.54	73	OB
139	10	F	41.1	1.4	20.97	85.96	80	SO
140	8	F	23.3	1.3	13.79	10.60	54	N
141	10	M	38.2	1.4	19.49	80.36	70	N
142	7	F	25.5	1.3	15.09	57.96	63	N
143	8	M	33.1	1.3	19.59	93.70	71	SO
144	7	M	21.6	1.2	15.00	13.63	56	N
145	9	M	32.9	1.3	19.47	80.67	69	N
146	11	M	52.9	1.5	23.51	93.45	84	SO
147	8	M	26.5	1.2	18.40	79.69	57	N
148	6	F	21.5	1.1	17.77	72.12	57	N
149	7	F	21.5	1.2	14.93	10.05	55	N
150	9	F	23.6	1.2	16.39	24.31	53	N

Table 8.4: Ejemplo de Berkeley

DEPARTAMENTO	HOMBRES ADMITIDOS	HOMBRES NO ADMITIDOS	MUJERES ADMITIDAS	MUJERES NO ADMITIDAS
A	512	315	89	19
B	353	207	17	8
C	120	205	202	395
D	138	279	131	244
E	53	138	94	299
F	16	256	24	317

Table 8.5: Géneros de avispas

X1	X2	Grupo
1.38	1.64	Chaq.amarilla
1.40	1.70	Chaq.amarilla
1.24	1.72	Chaq.amarilla
1.36	1.74	Chaq.amarilla
1.38	1.82	Chaq.amarilla
1.48	1.82	Chaq.amarilla
1.54	1.82	Chaq.amarilla
1.38	1.90	Chaq.amarilla
1.56	2.08	Chaq.amarilla
1.14	1.78	Neg.pequeña
1.20	1.86	Neg.pequeña
1.18	1.96	Neg.pequeña
1.30	1.96	Neg.pequeña
1.26	2.00	Neg.pequeña
1.28	2.00	Neg.pequeña

Table 8.6: Evolución mensual del empleo por sectores

Meses	Comercio	Servicios	Producción
1	53.7	53.5	44.2
2	52.8	53.0	44.3
3	53.2	53.2	44.4
4	53.8	52.5	43.4
5	54.5	53.4	42.8
6	54.7	56.5	44.3
7	54.2	65.3	44.4
8	54.3	70.7	44.8
9	55.0	66.9	44.4
10	55.7	58.2	43.1
11	56.2	55.3	42.6
12	56.8	53.4	42.4
13	53.7	52.1	42.2
14	53.0	51.5	41.8
15	53.3	51.5	40.1
16	54.3	52.4	42.0
17	55.3	53.3	42.4
18	55.7	55.5	43.1
19	55.8	64.2	42.4
20	56.0	69.6	43.1
21	55.8	69.3	43.2
22	56.3	58.5	42.8
23	57.0	55.3	43.0
24	58.0	53.6	42.8
25	55.0	52.3	42.5
26	54.3	51.5	42.6
27	54.8	51.7	42.3
28	56.2	51.5	42.9
29	57.5	52.2	43.6
30	58.3	57.1	44.7
31	58.5	63.6	44.5
32	59.0	68.8	45.0
33	59.2	68.9	44.8
34	59.5	60.1	44.9
35	60.3	55.6	45.2
36	61.3	53.9	45.2
37	58.0	53.3	45.0
38	57.5	53.1	45.5
39	58.2	53.5	46.2
40	59.2	53.5	46.8
41	60.3	53.9	47.5
42	61.2	57.1	48.3
43	61.0	64.7	48.3
44	61.7	69.4	49.1
45	61.8	70.3	48.9
46	62.5	62.6	49.4
47	63.3	57.9	50.0
48	64.2	55.8	50.0
49	60.2	54.8	49.6
50	59.0	54.2	49.9

Table 8.7: Evolución mensual del empleo por sectores (cont.)

Meses	Comercio	Servicios	Producción
51	59.5	54.6	49.6
52	61.2	54.3	50.7
53	62.7	54.8	50.7
54	63.5	58.1	50.9
55	63.5	68.1	50.5
56	63.8	73.3	51.2
57	64.0	75.5	50.7
58	64.5	66.4	50.3
59	65.3	60.5	49.2
60	66.0	57.7	48.1

Table 8.8: Información nutricional de galletitas

Marca	Valor energetico (kcalorias/100g)	Carbohidratos (g/100g)	Proteinas (g/100g)	Grasas totales (g/100g)	Sodio (mg/100g)
Cerealitas	439	65.0	11.0	15	574.00
Fajitas	466	57.0	10.0	22	828.00
Express	445	69.0	11.0	14	12.00
Oreo	478	67.0	5.6	21	363.00
Melba	464	70.0	6.3	18	263.00
Pepitos	463	66.0	7.1	19	136.00
Criollitas	438	69.0	11.0	13	431.00
Merengadas	418	69.0	6.3	13	201.00
Sonrisas	423	70.0	6.8	13	241.00
Maná	444	73.0	9.0	13	375.00
Guinditas	407	70.0	6.0	12	106.70
Pepas	437	60.0	6.7	18	76.67
Polvorón	410	56.7	6.3	18	66.70
Bizcochos	493	60.0	7.6	24	1066.00
Hogareñas	424	65.0	11.0	13	892.00
Granix	462	55.0	11.0	22	931.00
Bagley	421	63.0	11.0	14	624.00

Table 8.9: Características de individuos que utilizan internet

Id	Nac	Edad	Sexo	Estatura	Sitio	Tiempo	Temp	Autos	Cigarrillos
1	ARGENTINA	17	M	155	Correo	105	9	111	6
2	ARGENTINA	19	F	189	Buscador	106	18	121	6
3	ARGENTINA	28	F	157	Correo	106	6	169	4
4	ARGENTINA	47	F	155	Deportes	229	10	89	9
5	CANADA	27	M	161	Otros	33	-3	114	0
6	ARGENTINA	17	F	181	Chat	89	16	89	4
7	ARGENTINA	54	F	166	Otros	185	18	115	1
8	ARGENTINA	11	M	169	Correo	220	15	65	23
9	ARGENTINA	27	M	157	Musica	210	19	107	0
10	ARGENTINA	40	F	156	Programas	196	15	12	3
11	CANADA	31	M	161	Correo	42	-2	133	2
12	ARGENTINA	35	F	169	Musica	82	15	96	0
13	ARGENTINA	18	M	175	Correo	123	5	148	0
14	ARGENTINA	58	F	178	Otros	23	18	161	0
15	ARGENTINA	36	M	154	Musica	83	15	165	4
16	BRASIL	30	M	167	Musica	136	24	40	12
17	BRASIL	58	F	168	Otros	139	24	151	2
18	ARGENTINA	31	F	177	Programas	177	6	151	0
19	ARGENTINA	27	F	173	Buscador	266	12	199	5
20	CANADA	20	M	180	Buscador	330	3	43	1
21	ARGENTINA	24	F	181	Musica	15	8	18	0
22	BRASIL	34	F	166	Musica	170	12	19	0
23	ARGENTINA	40	F	161	Otros	39	11	56	0
24	ARGENTINA	29	F	152	Buscador	97	19	194	3
25	CANADA	33	F	162	Otros	72	21	199	1
26	ARGENTINA	21	M	182	Deportes	55	14	165	0
27	BRASIL	42	F	165	Deportes	117	20	19	8
28	BRASIL	33	F	168	Correo	250	3	80	13
29	CANADA	52	M	165	Otros	110	-7	139	0
30	BRASIL	19	M	167	Programas	111	23	184	0
31	BRASIL	23	F	162	Correo	154	20	47	4
32	BRASIL	53	F	160	Otros	186	18	152	8
33	BRASIL	54	F	159	Otros	163	17	171	2
34	CANADA	30	M	172	Musica	34	9	35	2
35	BRASIL	59	F	160	Otros	41	16	21	0
36	BRASIL	33	F	164	Programas	56	23	143	0
37	ARGENTINA	24	M	180	Programas	277	9	129	13
38	BRASIL	51	M	160	Otros	267	24	195	4
39	BRASIL	22	F	173	Otros	74	0	153	4
40	ARGENTINA	44	M	157	Correo	266	19	190	3
41	ARGENTINA	29	F	181	Buscador	49	8	110	2
42	ARGENTINA	53	M	167	Otros	139	8	7	0
43	ARGENTINA	24	M	173	Chat	93	12	149	5
44	ARGENTINA	47	F	189	Deportes	91	14	153	0
45	CANADA	31	F	156	Musica	118	9	191	6
46	ARGENTINA	57	M	165	Otros	180	12	175	0
47	BRASIL	11	F	162	Correo	83	17	10	0
48	CANADA	43	M	152	Chat	140	15	74	0
49	CANADA	16	M	153	Correo	135	5	142	2
50	ARGENTINA	22	M	164	Programas	114	7	75	8

Table 8.10: Características de individuos que utilizan internet (cont.)

Id	Nac	Edad	Sexo	Estatura	Sitio	Tiempo	Temp	Autos	Cigarrillos
51	ARGENTINA	59	F	159	Otros	28	7	186	1
52	ARGENTINA	46	M	154	Otros	44	5	112	2
53	CANADA	42	M	161	Buscador	29	-3	32	0
54	ARGENTINA	57	F	164	Otros	260	-3	23	3
55	ARGENTINA	44	F	168	Programas	107	9	50	2
56	ARGENTINA	23	F	191	Buscador	304	8	67	4
57	CANADA	27	F	170	Buscador	31	-5	74	1
58	BRASIL	35	M	180	Musica	254	34	74	0
59	ARGENTINA	55	F	163	Otros	74	13	17	2
60	ARGENTINA	49	M	170	Deportes	81	-5	9	1
61	BRASIL	50	M	164	Otros	260	33	129	7
62	ARGENTINA	15	M	150	Correo	71	-9	184	3
63	BRASIL	17	F	168	Chat	148	21	90	6
64	CANADA	51	F	173	Otros	121	0	76	0
65	ARGENTINA	28	M	164	Musica	320	9	125	0
66	ARGENTINA	42	M	169	Deportes	37	20	9	2
67	CANADA	36	F	165	Otros	250	0	126	0
68	BRASIL	29	F	175	Deportes	152	35	86	3
69	CANADA	31	F	186	Musica	-10	23	61	9
70	BRASIL	57	M	164	Otros	183	32	137	8
71	BRASIL	33	M	172	Musica	107	14	100	1
72	ARGENTINA	20	F	167	Buscador	94	-9	60	0
73	ARGENTINA	27	F	181	Musica	135	17	44	2
74	ARGENTINA	28	F	169	Correo	16	10	83	0
75	CANADA	52	F	168	Otros	141	-5	191	3
76	ARGENTINA	59	M	166	Otros	37	19	157	0
77	ARGENTINA	56	F	169	Otros	157	29	70	1
78	CANADA	15	M	149	Correo	72	8	98	5
79	CANADA	20	M	163	Correo	128	9	88	0
80	ARGENTINA	54	M	161	Otros	189	8	110	6
81	BRASIL	59	F	161	Otros	189	31	183	14
82	CANADA	59	M	173	Otros	271	-3	50	0
83	BRASIL	39	M	163	Musica	244	-2	193	14
84	BRASIL	16	M	177	Correo	23	8	190	0
85	ARGENTINA	44	F	157	Correo	13	27	147	1
86	ARGENTINA	27	M	171	Programas	29	13	69	1
87	ARGENTINA	36	F	184	Correo	115	28	182	0
88	BRASIL	33	F	164	Programas	30	16	136	1
89	CANADA	37	F	167	Musica	261	-3	89	3
90	ARGENTINA	23	F	181	Buscador	110	5	106	1
91	BRASIL	36	F	164	Chat	81	21	12	5
92	CANADA	20	F	174	Otros	78	17	70	8
93	ARGENTINA	21	M	162	Programas	37	27	177	1
94	ARGENTINA	14	M	166	Correo	152	15	29	4
95	ARGENTINA	51	M	164	Otros	83	6	157	3
96	CANADA	40	F	166	Otros	172	7	127	1
97	ARGENTINA	21	F	161	Buscador	39	14	161	2
98	ARGENTINA	59	F	168	Otros	136	4	137	2
99	BRASIL	56	F	171	Otros	194	16	56	0
100	ARGENTINA	23	M	170	Correo	37	9	11	1

Table 8.11: Puntuación de candidatas al puesto de recepcionista

Candidata	Cordialidad	Presencia	Idioma	Juez
Mariana	80	90	70	A
Maia	80	90	60	A
Sabrina	90	60	50	A
Daniela	80	50	50	A
Alejandra	70	60	50	A
Carla	90	85	60	A
Mariana	60	78	80	B
Maia	65	90	65	B
Sabrina	70	60	50	B
Daniela	70	58	40	B
Alejandra	55	70	65	B
Carla	80	90	40	B

Presentamos el código para la generación de los gráficos correspondientes a las sumas de Riemman.

```
x=seq(-4,4,0.5)
y=x**2+3
plot(x,y,type="l",ylim=c(-1,19),col="violet",lwd=2)
text(locator(1),col="violet","y=x^2+3")
abline(0,0)
part=seq(-2.5,3.5,0.5)

n=length(part)-1
segments(part[1],0,part[1],y[4],col="blue",lwd=2)
segments(part[n+1],0,part[n+1],y[length(y)-1],col="blue",lwd=2)

inf=0

for (i in 1:n){
  inf[i]=min(y[i+3],y[i+4])
  segments(part[i],0,part[i],inf[i],col="green")
  segments(part[i+1],0,part[i+1],inf[i],col="green")
  segments(part[i],inf[i],part[i+1],inf[i],col="green")
}

title("Suma inferior de Riemman")

sup=0

for (i in 1:n){
  sup[i]=max(y[i+3],y[i+4])
  segments(part[i],0,part[i],sup[i],col="red")
  segments(part[i+1],0,part[i+1],sup[i],col="red")
  segments(part[i],sup[i],part[i+1],sup[i],col="red")
}

title("Suma superior de Riemman")
```

Presentamos el código para la generación del gráfico correspondiente a la regla de trapecios.


```
x=seq(0,5,0.1)
y=3*cos(x)+4
plot(x,y,type="l",ylim=c(-1,7),lwd=1,col="cyan")
text(locator(1),col="cyan","f(x)=3cos(x)+4")
abline(0,0)

part=seq(0.3,4.3,1)
n=length(part)

for (i in 1:n){
  segments(part[i],0,part[i],y[4+(i-1)*10],col="pink",lwd=2)
  segments(part[i+1],0,part[i+1],y[4+i*10],col="pink",lwd=2)
  segments(part[i],y[4+(i-1)*10],part[i+1],y[4+i*10],col="pink",lwd=2)
}

title("Regla de trapecios")
```

Presentamos el código para la generación del gráfico correspondiente al aparato de Galton.

```

par(mfrow=c(1,2))

x=c(4,9,14,13,11,8,9,11,10,8,10,11,7,8,9,12,13,6,7,9,10,11,14,10)
y=c(8,6,2,7,10,15,18,19,20,21,21,21,22,22,22,22,22,23,23,23,23,23,24)
plot(x,y,type="p",pch=19,cex=1.3,col="pink",xlim=c(-1,21),ylim=c(-1,25),axes=F,
     xlab="",ylab="")

segments(0,0,0,4,col="chartreuse4",lwd=2)
segments(0,4,2,6,col="chartreuse4",lwd=2)
segments(2,6,2,8,col="chartreuse4",lwd=2)
segments(2,8,4,10,col="chartreuse4",lwd=2)
segments(4,10,4,12,col="chartreuse4",lwd=2)
segments(4,12,6,14,col="chartreuse4",lwd=2)
segments(6,14,6,16,col="chartreuse4",lwd=2)
segments(6,16,8,18,col="chartreuse4",lwd=2)
segments(8,18,8,20,col="chartreuse4",lwd=2)
segments(8,20,4,24,col="chartreuse4",lwd=2)

segments(20,0,20,4,col="chartreuse4",lwd=2)
segments(20,4,18,6,col="chartreuse4",lwd=2)
segments(18,6,18,8,col="chartreuse4",lwd=2)
segments(18,8,16,10,col="chartreuse4",lwd=2)
segments(16,10,16,12,col="chartreuse4",lwd=2)
segments(16,12,14,14,col="chartreuse4",lwd=2)
segments(14,14,14,16,col="chartreuse4",lwd=2)
segments(14,16,12,18,col="chartreuse4",lwd=2)
segments(12,18,12,20,col="chartreuse4",lwd=2)
segments(12,20,16,24,col="chartreuse4",lwd=2)

segments(4,3,3,4,col="chartreuse4",lwd=2)
segments(3,4,4,5,col="chartreuse4",lwd=2)
segments(4,5,5,4,col="chartreuse4",lwd=2)
segments(5,4,4,3,col="chartreuse4",lwd=2)

segments(8,3,7,4,col="chartreuse4",lwd=2)
segments(7,4,8,5,col="chartreuse4",lwd=2)
segments(8,5,9,4,col="chartreuse4",lwd=2)
segments(9,4,8,3,col="chartreuse4",lwd=2)

segments(12,3,11,4,col="chartreuse4",lwd=2)
segments(11,4,12,5,col="chartreuse4",lwd=2)
segments(12,5,13,4,col="chartreuse4",lwd=2)
segments(13,4,12,3,col="chartreuse4",lwd=2)

segments(16,3,15,4,col="chartreuse4",lwd=2)
segments(15,4,16,5,col="chartreuse4",lwd=2)
segments(16,5,17,4,col="chartreuse4",lwd=2)
segments(17,4,16,3,col="chartreuse4",lwd=2)

segments(6,7,5,8,col="chartreuse4",lwd=2)
segments(5,8,6,9,col="chartreuse4",lwd=2)
segments(6,9,7,8,col="chartreuse4",lwd=2)
segments(7,8,6,7,col="chartreuse4",lwd=2)

segments(10,7,9,8,col="chartreuse4",lwd=2)
segments(9,8,10,9,col="chartreuse4",lwd=2)
segments(10,9,11,8,col="chartreuse4",lwd=2)
segments(11,8,10,7,col="chartreuse4",lwd=2)

```

```

segments(14,7,13,8,col="chartreuse4",lwd=2)
segments(13,8,14,9,col="chartreuse4",lwd=2)
segments(14,9,15,8,col="chartreuse4",lwd=2)
segments(15,8,14,7,col="chartreuse4",lwd=2)

segments(8,11,7,12,col="chartreuse4",lwd=2)
segments(7,12,8,13,col="chartreuse4",lwd=2)
segments(8,13,9,12,col="chartreuse4",lwd=2)
segments(9,12,8,11,col="chartreuse4",lwd=2)

segments(12,11,11,12,col="chartreuse4",lwd=2)
segments(11,12,12,13,col="chartreuse4",lwd=2)
segments(12,13,13,12,col="chartreuse4",lwd=2)
segments(13,12,12,11,col="chartreuse4",lwd=2)

segments(10,15,9,16,col="chartreuse4",lwd=2)
segments(9,16,10,17,col="chartreuse4",lwd=2)
segments(10,17,11,16,col="chartreuse4",lwd=2)
segments(11,16,10,15,col="chartreuse4",lwd=2)

segments(0,0,20,0,col="chartreuse4",lwd=2)
segments(4,0,4,1,col="chartreuse4",lwd=2)
segments(8,0,8,1,col="chartreuse4",lwd=2)
segments(12,0,12,1,col="chartreuse4",lwd=2)
segments(16,0,16,1,col="chartreuse4",lwd=2)

title("Aparato de Galton")

x=c(9,11,15,1,2,6,7,10,14,17,18,5,10,13,6,14,18,10,11,15,6,12,10,13)
y=c(1,1,1,1,1,1,1,2,2,1,1,3,3,3,4,4,4,5,6,6,12,15,22,23)
plot(x,y,type="p",pch=19,cex=1.3,col="pink",xlim=c(-1,21),ylim=c(-1,25),axes=F,
     xlab="",ylab="")

segments(0,0,0,4,col="chartreuse4",lwd=2)
segments(0,4,2,6,col="chartreuse4",lwd=2)
segments(2,6,2,8,col="chartreuse4",lwd=2)
segments(2,8,4,10,col="chartreuse4",lwd=2)
segments(4,10,4,12,col="chartreuse4",lwd=2)
segments(4,12,6,14,col="chartreuse4",lwd=2)
segments(6,14,6,16,col="chartreuse4",lwd=2)
segments(6,16,8,18,col="chartreuse4",lwd=2)
segments(8,18,8,20,col="chartreuse4",lwd=2)
segments(8,20,4,24,col="chartreuse4",lwd=2)

segments(20,0,20,4,col="chartreuse4",lwd=2)
segments(20,4,18,6,col="chartreuse4",lwd=2)
segments(18,6,18,8,col="chartreuse4",lwd=2)
segments(18,8,16,10,col="chartreuse4",lwd=2)
segments(16,10,16,12,col="chartreuse4",lwd=2)
segments(16,12,14,14,col="chartreuse4",lwd=2)
segments(14,14,14,16,col="chartreuse4",lwd=2)
segments(14,16,12,18,col="chartreuse4",lwd=2)
segments(12,18,12,20,col="chartreuse4",lwd=2)
segments(12,20,16,24,col="chartreuse4",lwd=2)

```

```

segments(4,3,3,4,col="chartreuse4",lwd=2)
segments(3,4,4,5,col="chartreuse4",lwd=2)
segments(4,5,5,4,col="chartreuse4",lwd=2)
segments(5,4,4,3,col="chartreuse4",lwd=2)

segments(8,3,7,4,col="chartreuse4",lwd=2)
segments(7,4,8,5,col="chartreuse4",lwd=2)
segments(8,5,9,4,col="chartreuse4",lwd=2)
segments(9,4,8,3,col="chartreuse4",lwd=2)

segments(12,3,11,4,col="chartreuse4",lwd=2)
segments(11,4,12,5,col="chartreuse4",lwd=2)
segments(12,5,13,4,col="chartreuse4",lwd=2)
segments(13,4,12,3,col="chartreuse4",lwd=2)

segments(16,3,15,4,col="chartreuse4",lwd=2)
segments(15,4,16,5,col="chartreuse4",lwd=2)
segments(16,5,17,4,col="chartreuse4",lwd=2)
segments(17,4,16,3,col="chartreuse4",lwd=2)

segments(6,7,5,8,col="chartreuse4",lwd=2)
segments(5,8,6,9,col="chartreuse4",lwd=2)
segments(6,9,7,8,col="chartreuse4",lwd=2)
segments(7,8,6,7,col="chartreuse4",lwd=2)

segments(10,7,9,8,col="chartreuse4",lwd=2)
segments(9,8,10,9,col="chartreuse4",lwd=2)
segments(10,9,11,8,col="chartreuse4",lwd=2)
segments(11,8,10,7,col="chartreuse4",lwd=2)

segments(14,7,13,8,col="chartreuse4",lwd=2)
segments(13,8,14,9,col="chartreuse4",lwd=2)
segments(14,9,15,8,col="chartreuse4",lwd=2)
segments(15,8,14,7,col="chartreuse4",lwd=2)

segments(8,11,7,12,col="chartreuse4",lwd=2)
segments(7,12,8,13,col="chartreuse4",lwd=2)
segments(8,13,9,12,col="chartreuse4",lwd=2)
segments(9,12,8,11,col="chartreuse4",lwd=2)

segments(12,11,11,12,col="chartreuse4",lwd=2)
segments(11,12,12,13,col="chartreuse4",lwd=2)
segments(12,13,13,12,col="chartreuse4",lwd=2)
segments(13,12,12,11,col="chartreuse4",lwd=2)

segments(10,15,9,16,col="chartreuse4",lwd=2)
segments(9,16,10,17,col="chartreuse4",lwd=2)
segments(10,17,11,16,col="chartreuse4",lwd=2)
segments(11,16,10,15,col="chartreuse4",lwd=2)

segments(0,0,20,0,col="chartreuse4",lwd=2)
segments(4,0,4,1,col="chartreuse4",lwd=2)
segments(8,0,8,1,col="chartreuse4",lwd=2)
segments(12,0,12,1,col="chartreuse4",lwd=2)
segments(16,0,16,1,col="chartreuse4",lwd=2)

title("Aparato de Galton")

```