R en cursos básicos de estadística

Jorge Ortiz Pinilla
Profesor especial
Departamento de Estadística
Universidad Nacional

Universidad Nacional de Colombia - Sede Bogotá
Departamento de Estadística
Marzo de 2007

Índice general

Prefacio

Diversos programas, en particular las hojas electrónicas como *Excel*, se convierten en herramientas de enseñanza que aprovechan exitosamente los profesores para "efectuar a mano" los cálculos necesarios para los procedimientos de análisis. Sin embargo, tarde o temprano, este entrenamiento se abandona por la complejidad misma del "trabajo manual" y se tiene la necesidad de iniciar una nueva etapa de aprendizaje del uso de herramientas más apropiadas para el trabajo profesional.

Los programas específicos de análisis estadístico, como R, incluyen instrucciones de muy alto nivel que permiten también descomponer un procedimiento en etapas que equivalen al mismo trabajo manual que se desarrolla con las hojas electrónicas, así que el método de enseñanza no se afecta de manera negativa. Por el contrario, el estudiante entra a manejar una herramienta que le permitirá llegar mucho más lejos en relación con su potencial de análisis de datos.

El objetivo que busqué al escribir estas notas fue ofrecer un documento que sirviera para acompañar a los estudiantes y a los profesores de cursos básicos de estadística durante el desarrollo de sus actividades de cálculo estadístico e iniciarlos en el uso de una herramienta poderosa y al alcance de todos por su carácter de *software libre*.

Me siento en la obligación de hacer claridad sobre lo siguiente: no es un manual de R. La mejor documentación sobre este programa se encuentra en la página oficial, www.R-project.org. Tampoco es un documento para aprender estadística. Por eso no incluí ni desarrollos teóricos, ni conceptuales, ni metodológicos. Simplemente, en este momento, lo invito a leer de nuevo el párrafo anterior.

Capítulo 1

Datos y funciones

1.1. Formas de almacenamiento

1.1.1. Variables

R diferencia entre mayúsculas y minúsculas, tanto para variables, como para funciones propias de sus librerías o definidas por el usuario: N es una variable diferente de n y V_x no es v_x, ni v_X, ni V_X. $\sin(x)$ está definida en el lenguaje. El usuario puede definir $Sin(x)$ y esta definición no tiene nada que ver con la anterior. Las instrucciones terminan con *punto y coma* (;) o con salto de línea. Puede haber varias en una línea, pero, en este caso deben separarse por *punto y coma*.

Las variables se definen cuando el usuario lo considere necesario y puede cambiar su valor e incluso el tipo de dato cuando quiera. La última definición y el último valor asignado son los que valen para las instrucciones siguientes, mientras se mantenga el ámbito en el cual fueron creadas.

Operaciones aritméticas Se indican con los signos utilizados en los programas tradicionales: `+, -, *, /`. Además, en R, x `%%` y calcula x mod y y x `%/%` y, la división entera.

	Instrucciones en R		
Asignación de valores	`x = 5`		
	`y = 7`		
Despliegue de expresiones	`x + y`	--→	12
Asignación de expresiones	`a = x + y`		
	`a`	--→	12
	`b = x - y`		
	`b`	--→	-2
	`c = x * y`		
	`c`	--→	35
	`d = y / x`		
	`d`	--→	1.4
Resto de división	`e = y %% x`		
	`e`	--→	2
División entera	`f = y %/% x`		
	`f`	--→	1

Funciones Los ejemplos siguientes ilustran la forma como se utilizan algunas de las funciones más comunes:

```
x = 5
y = 7
a = sin(x) + cos(y) - tan(x * y)
a                                    --→  −0.6788367
sinh(x) + cosh(y) - tanh(x * y)      --→  621.5202
b = log(x) * log10(x) - exp(-y)
b                                    --→  1.124037
c = x**y + sqrt(x) + x**2 - y**(-2)
c                                    --→  78152.22
sign(x) + abs(-y)                    --→  8
a = asin(x) + acos(y) - atan(x * y)
a                                    --→  0.9823228
```

Operaciones y comparaciones lógicas Cuando es necesario, las variables lógicas se convierten a enteros. Para las comparaciones en-

tre expresiones se utilizan los símbolos $==, !=, <, <=, >, >=$ y se diferencia el operador de asignación ($=$) del de comparación ($==$).

	Instrucciones en R		
Asignación	`x = TRUE`		
	`y = TRUE`		
	`z = FALSE`		
Comparaciones	`u = (6 == 6)`		
	`u`	--→	$TRUE$
	`m = 9`		
	`u = ( m == 9)`		
	`u`	--→	$TRUE$
	`u = ( m != 9)`		
	`u`	--→	$FALSE$
	`u = ( m < 9)`		
	`u`	--→	$FALSE$
¡Cuidado!	`v = (m = 9)`		
	`v`	--→	9
Negación	`!x`	--→	$FALSE$
	`!z`	--→	$TRUE$
OR en expresiones	`x | z`	--→	$TRUE$
OR, condición en ciclos	`x || z`	--→	$TRUE$
AND en expresiones	`x & z`	--→	$FALSE$
AND en ciclos	`x && z`	--→	$FALSE$
XOR, uno TRUE y otro FALSE	`xor(x, y)`	--→	$FALSE$
	`xor(x, z)`	--→	$TRUE$
TRUE es 1	`a = x + y`		
	`a`	--→	2
	`b = x - y`		
	`b`	--→	0
FALSE es 0	`c = z + 0`		
	`c`	--→	0
Si $k \neq 0$, es TRUE	`7 & T`	--→	$TRUE$

1.1.2. Vectores

Creación de vectores

```
> x = c(4, 5, 2, 4, 8)
> x                          ----->          [1] 4 5 2 4 8

> y = 1:8
> y                          ----->    [1] 1 2 3 4 5 6 7 8

> z = seq(1, 20, 4)
> z                          ----->     [1]  1  5  9 13 17

> w = rep(7, 9)
> w                          ----->  [1] 7 7 7 7 7 7 7 7 7

> w1 = rep(c(7, 8, 9), c(4, 2, 5))
> w1                  ----->   [1] 7 7 7 7 8 8 9 9 9 9 9
```

Operaciones entre elementos

```
> x = c(5, 7)
> y = c(1, 2)
> x + y                       ----->              [1] 6 9

> a = x + y
> a                           ----->              [1] 6 9

> b = x - y
> b                           ----->              [1] 4 5

> c = x * y
> c                           ----->            [1]  5 14

> d = x/y
> d                           ----->          [1] 5.0 3.5
```

Operaciones y funciones vectoriales

```
> x = c(5, 7)
> y = c(1, 2)
> a = x %*% y
> a                          ----->                 [,1]
                                                 [1,]   19

> b = sum(x)
> b                          ----->               [1] 12

> c = min(x)
> c                          ----->               [1] 5

> d = max(x)
> d                          ----->               [1] 7

> m = mean(x)
> m                          ----->               [1] 6

> s = sd(x)
> s                          ----->         [1] 1.414214

> v = var(x)
> v                          ----->               [1] 2
```

Nota Todas las funciones escalares aplicadas sobre objetos vectoriales dan como resultado un vector de la misma dimensión que el argumento, con los elementos transformados por la función escalar aplicada.

```
> Sx = sin(x)
> Sx                 ----->   [1] -0.9589243  0.6569866

> Ly = log(y)
> Ly                 ----->     [1] 0.0000000 0.6931472
```

1.1.3. Matrices

Creación de matrices Las matrices pueden crearse mediante la instrucción `matrix`: el primer argumento es un vector con los elementos de la matriz, el segundo es el número de filas. Los elementos se organizan por columnas a menos que se use la opción `byrow=T`.

```
> a = matrix(c(2, 4, 1, 7), nrow = 2)
> a                  ----->                  [,1] [,2]
                                        [1,]    2    1
                                        [2,]    4    7

> b = matrix(c(2, 4, 1, 7), nrow = 2, byrow = T)
> b                  ----->                  [,1] [,2]
                                        [1,]    2    4
                                        [2,]    1    7
```

También se pueden crear a partir de vectores de la misma dimensión. `rbind` los pega considerándolos como filas y `cbind` los pega considerándolos como columnas:

```
> x = 1:5
> y = seq(1, 20, 4)
> z = rbind(x, y)
> z                  ----->   [,1] [,2] [,3] [,4] [,5]
                         x    1    2    3    4    5
                         y    1    5    9   13   17

> u = cbind(x, y)
> u                  ----->                       x  y
                                              [1,] 1  1
                                              [2,] 2  5
                                              [3,] 3  9
                                              [4,] 4 13
                                              [5,] 5 17
```

Operaciones y funciones matriciales

```
> tu = t(u)
> tu                  ----->  [,1] [,2] [,3] [,4] [,5]
                            x    1    2    3    4    5
                            y    1    5    9   13   17

> ztu = z + t(u)
> ztu                 ----->  [,1] [,2] [,3] [,4] [,5]
                            x    2    4    6    8   10
                            y    2   10   18   26   34

> zu = z %*% u
> zu                  ----->                   x   y
                                             x  55 175
                                             y 175 565

> u - 2 * sin(t(ztu))  ----->            x          y
                          [1,] -0.8185949 -0.8185949
                          [2,]  3.5136050  6.0880422
                          [3,]  3.5588310 10.5019745
                          [4,]  2.0212835 11.4748831
                          [5,]  6.0880422 15.9418346

> ai = solve(a)
> a %*% ai             ----->       [,1]          [,2]
                       [1,] 1.000000e+00 -2.775558e-17
                       [2,] 1.110223e-16  1.000000e+00

> avp = eigen(a)
> avp$values          ----->     [1] 7.701562 1.298438

> avp$vectors         ----->           [,1]       [,2]
                            [1,] -0.1727535 -0.8186307
                            [2,] -0.9849651  0.5743203
```

1.1.4. Algunas funciones estadísticas

Además de las funciones vectoriales ya vistas, como `min(x)`, `max(x)`, `mean(x)`, `sd(x)`, `var(x)`, presentamos las que permiten generar números pseudoaleatorios, probabilidades, cuantiles y funciones de densidad.

Para generar números pseudoaleatorios

n es la cantidad de números que se quiere generar.

Normal	rnorm(n, mean=0, sd=1)
Exponential	rexp(n, rate=1)
Poisson	rpois(n, lambda)
Cauchy	rcauchy(n, location=0, scale=1)
Student	rt(n, df)
F	rf(n, df1, df2)
χ^2	rchisq(n, df)
Binomial	rbinom(n, size, prob)
Uniform	runif(n, min=0, max=1)

Para calcular la función de distribución

x es un valor o un vector de valores para los cuales se quiere calcular la función de probabilidad acumulada

Normal	pnorm(x, mean=0, sd=1)
Exponential	pexp(x, rate=1)
Poisson	ppois(x, lambda)
Cauchy	pcauchy(x, location=0, scale=1)
Student	pt(x, df)
F	pf(x, df1, df2)
χ^2	pchisq(x, df)
Binomial	pbinom(x, size, prob)
Uniform	punif(x, min=0, max=1)

Para calcular los cuantiles

p es un valor o un vector de valores de probabilidad para los cuales se quiere calcular los cuantiles de la distribución.

Normal	qnorm(p, mean=0, sd=1)
Exponential	qexp(p, rate=1)
Poisson	qpois(p, lambda)
Cauchy	qcauchy(p, location=0, scale=1)
Student	qt(p, df)
F	qf(p, df1, df2)
χ^2	qchisq(p, df)
Binomial	qbinom(p, size, prob)
Uniform	qunif(p, min=0, max=1)

Para calcular la función de densidad

x es un valor o un vector de valores para los cuales se quiere calcular la función de densidad

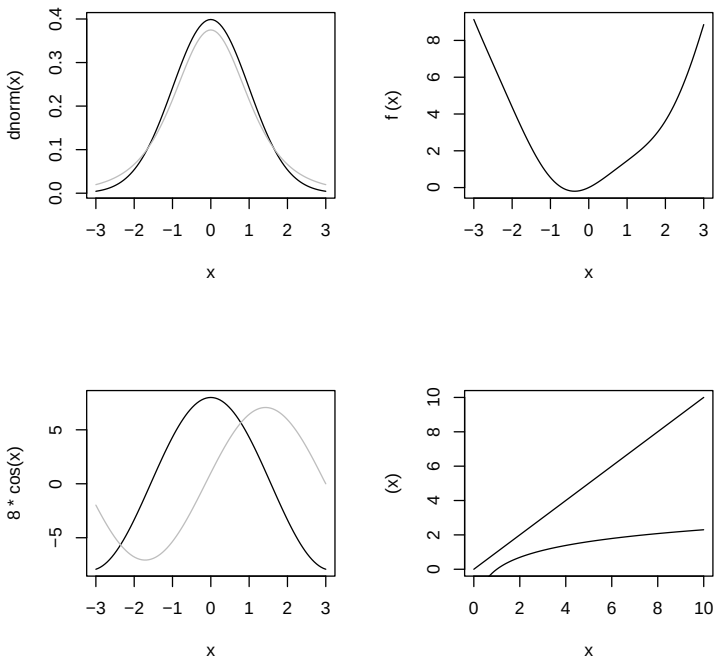
Normal	dnorm(x, mean=0, sd=1)
Exponential	dexp(x, rate=1)
Poisson	dpois(x, lambda)
Cauchy	dcauchy(x, location=0, scale=1)
Student	dt(x, df)
F	df(x, df1, df2)
χ^2	dchisq(x, df)
Binomial	dbinom(x, size, prob)
Uniform	dunif(x, min=0, max=1)

1.1.5. Otras funciones

Mencionaremos sólo `sort(x)` para ordenar un vector y `rank(x)` para obtener los rangos de los elementos del vector, es decir, las posiciones que ocupan si se ordenan de menor a mayor.

1.2. Gráficas de funciones

```
> opcion <- par(mfrow = c(2, 2))
> f = function(x) {
+     x^2 + sin(x) * cos(x)
+ }
> curve(dnorm(x), -3, 3)
> curve(dt(x, 4), -3, 3, col = "gray", add = T)
> curve(f, -3, 3)
> curve(8 * cos(x), -3, 3)
> curve(cos(x) + 7*sin(x), -3, 3, col = "gray", add = T)
> curve((x), 0, 10)
> curve(log(x), 0, 10, add = T)
> par(opcion)
```



1.2.1. Marcos de datos *(Data frame)*

Son objetos que generalizan el concepto de matriz. Se trata de arreglos rectangulares en donde las filas representan individuos y las columnas representan variables. En la misma columna todos los datos deben ser del mismo tipo (numéricos, lógicos, textos, etc), pero dos columnas pueden tener datos de tipos diferentes, por ejemplo, una contiene datos numéricos y la otra textos.

```
> a = c(12, 15, 5, 2)
> b = c("Buses", "Camiones", "Automóbiles", "Bicicletas")
> datos = data.frame(a, b)
> datos
```

```
   a           b
1 12       Buses
2 15    Camiones
3  5 Automóbiles
4  2  Bicicletas
```

1.3. Recolección de los datos

Lo ideal es contar con un diseño de base de datos y con los dispositivos que permitan ponerlos a disposición de los programas para análisis estadístico, con el debido control de calidad, la debida seguridad y la mayor facilidad de acceso posible.

Para los análisis, los programas estadísticos utilizan una estructura rectangular similar a la de una tabla de una base de datos. Las columnas se identifican con *campos* de la tabla y, en la mayoría de casos, contienen información sobre variables que se han construido o evaluado en los individuos de una determinada población, real o ficticia. Las filas se asocian con individuos o elementos de la población estudiada. Llamaremos *conjunto de datos* a la estructura así definida, junto con la información que contiene.

1.3.1. Lectura de datos con R

Los siguientes son ejemplos de instrucciones para leer los datos de archivos cuando estén disponibles:

```
datos  = read.table("f:/c20071/datos.txt", header = T)
datos1 = read.csv("f:/c20071/datos1.txt", header = T,
          sep = ",")
datos2 = read.csv("f:/c20071/datos2.txt", header = T,
           sep = "\t", na.string = ".")
datos  = read.fwf("data.txt", widths=c(1, 5, 4))
d <-read.table(
    url("http://www.stat.umn.edu/alr/data/htwt.txt"),
    header=TRUE)
```

La función `read.fwf` se utiliza para leer datos en formato fijo. Esto significa que, en el archivo donde fueron grabados, los datos de una variable aparecen siempre dentro de un mismo espacio vertical determinado por una columna inicial c_i y una columna final c_f que se reserva de manera exclusiva para ella. El ancho de esa columna, dado por $c_f - c_i + 1$, se conoce como la longitud del campo, tal como fue grabado en el archivo. Se construye un vector con las longitudes en el orden de las variables. Si hay una línea de encabezado, con los nombres de las variables, al menos por ahora, debe separarse cada nombre por un único tabulador, a menos que se especifique uno con la opción `sep=` y se coloque uno diferente entre comillas. Supongamos que el archivo `data.txt` tiene la forma siguiente, donde la primera línea contiene los nombres de las variables, separados por un solo tabulador:

```
Clase Altura Coeficiente
A 1.50 1.2
A 1.55 1.3
B 1.60 1.4
B 1.65 1.5
C 1.70 1.6
C 1.75 1.7
D      1.8
D10.8212.9
```

La siguiente instrucción lee correctamente el archivo `data_fwf.txt`. Las dos últimas líneas de datos generan problemas para la función `read.table`.

```
> datos=read.fwf("data_fwf.txt", widths=c(1, 5, 4), header=T)
> datos
```

```
  Clase Altura Coeficiente
1     A   1.50         1.2
2     A   1.55         1.3
3     B   1.60         1.4
4     B   1.65         1.5
5     C   1.70         1.6
6     C   1.75         1.7
7     D     NA         1.8
8     D  10.82        12.9
```

El archivo `Datos1.csv`, grabado con Excel en español, utiliza el punto y coma (;) como separador de campos y la coma (,) como signo decimal. Su lectura se realiza con la instrucción `read.csv2`, como se muestra más abajo.

```
> datos=read.csv2("datos1.csv", header=T, sep=";", dec=",")
> datos
```

```
  Clase Altura Coeficiente
1     A   1.50         1.2
2     A   1.55         1.3
3     B   1.60         1.4
4     B   1.65         1.5
5     C   1.70         1.6
6     C   1.75         1.7
7     D     NA         1.8
8     D  10.82        12.9
```

Los objetos *datos* son conjuntos de datos tipo *data.frame*. La función `names(datos)` proporciona los nombres de las variables incluidas. El acceso a los componentes de `datos` se logra mediante `datos$Altura`,

si se conoce el nombre de la variable o mediante `datos[, 2]`, si se sabe que la variable que se necesita está en la columna 2; `datos[1, ]` proporciona los datos de todas las variables del individuo ubicado en la fila 1; `datos[1, 3]` muestra la tercera variable (*Coeficiente*) del individuo de la fila 1 y `datos$Altura[4]` da la *Altura* del individuo de la fila 4.

Cuando se necesitan *bloques* de datos, por ejemplo, la segunda y la tercera variables de los cuatro primeros individuos, la instrucción `datos[1:4,c(2,3)]` los permitirá seleccionar. Si en lugar de las posiciones de las variables se conocen sus nombres, el mismo bloque de datos se obtiene con

```
> datos[1:4, c("Altura", "Coeficiente")]

  Altura Coeficiente
1   1.50         1.2
2   1.55         1.3
3   1.60         1.4
4   1.65         1.5
```

Funciones de textos

```
> t1 = "Grupo"
> titulo = paste(t1, "Tratamiento", 5)
> titulo

[1] "Grupo Tratamiento 5"
```

Capítulo 2

Análisis estadístico básico

Hemos definido las variables como instrumentos para observar y evaluar características en los individuos de una población. Son útiles para comparar individuos entre sí o con repecto a valores fijos utilizados como patrones. Por ejemplo, si una variable mide el tiempo empleado en recorrer cien metros planos, se pueden comparar los datos individuales entre sí para seleccionar los corredores más rápidos o determinar si un individuo ha logrado superar la meta de diez segundos para recorrer esta distancia.

Por lo general, un dato individual no tiene mayor importancia por sí mismo sino cuando se lo relaciona con otros del mismo individuo obtenidos en diferentes circunstancias, con los de otros individuos, o con un valor en el cual se tiene especial interés y que lo convierte en un patrón de comparación.

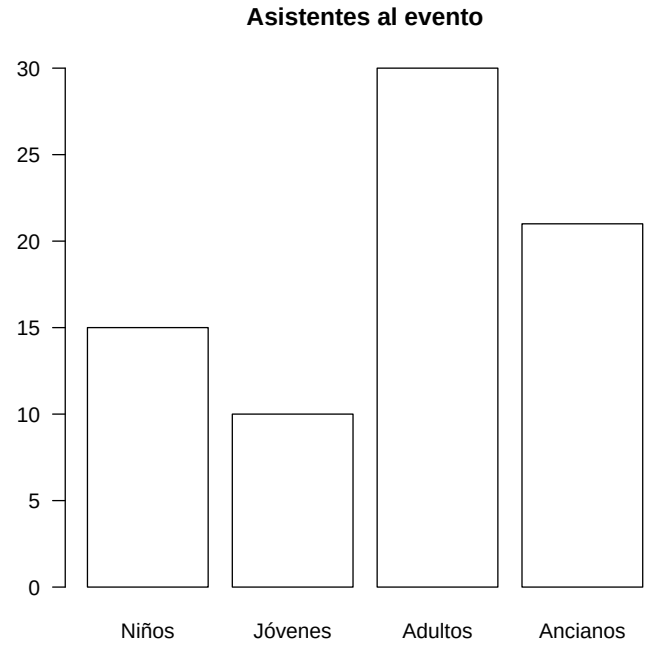
También podemos organizar los individuos en grupos según los valores obtenidos en una o más variables y conocer cómo se distribuyen en esos grupos, mirar la evolución durante un período de tiempo o relacionada con la modificación de las condiciones de observación.

2.1. Variables nominales

Estas variables no tienen incorporada la noción de cantidad, sino la de clasificación. Lo natural es evaluar la importancia de cada grupo mediante su tamaño, es decir, la frecuencia de casos identificados como pertenecientes a él, o su frecuencia relativa, calculada como el cociente entre la frecuencia del grupo y el número total de casos

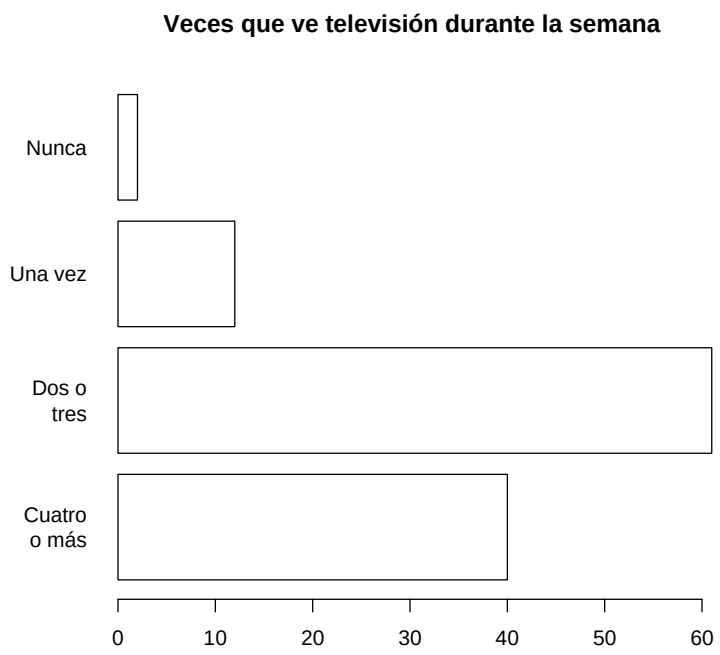
2.1.1. Diagramas de barras

```
> x = c(15, 10, 30, 21)
> n_x = c("Niños", "Jóvenes", "Adultos", "Ancianos")
> names(x) = n_x
> opcion = par(las = 1)
> barplot(x, col = "white", main = "Asistentes al evento")
```



La función `barplot` admite varios parámetros que se pueden consultar en la ayuda mediante `help(\barplot)`". En el siguiente ejemplo ilustramos la opción `horiz = T` para presentar las barras en dirección horizontal. La opción `las` puede tomar los valores 0, 1, 2 o 3 y permite controlar la orientación de la escritura de las etiquetas en los ejes.

```
x = c(40, 61, 12, 2)
n_x = c("Cuatro\n o más", "Dos o\n tres",
        "Una vez", "Nunca")
names(x) = n_x
titulo = "Veces que ve televisión durante la semana"
barplot(x, names = n_x, main = titulo, las = 1, horiz = T,
        col = "white")
```

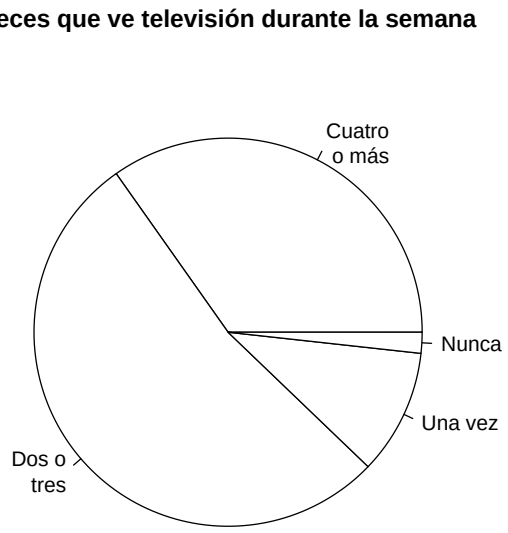


2.1.2. Tortas

Las tortas o diagramas de pastel son menos cómodas que las barras para comparar las categorías de una variable nominal, a menos que las diferencias sean grandes. Sin embargo, se utilizan de manera muy frecuente para ilustrar cómo se descompone un grupo total en las diferentes categorías.

Igual que otras funciones de R, `pie` admite varios parámetros de control del gráfico. `col =` , con un vector con los nombres de los colores que el usuario quiera asignar a cada categoría, los utiliza para colorear las tajadas de la torta. Hemos tomado la opción `col = "white"` pues consideramos que los colores agregan "espectacularidad" al gráfico, pero no información.

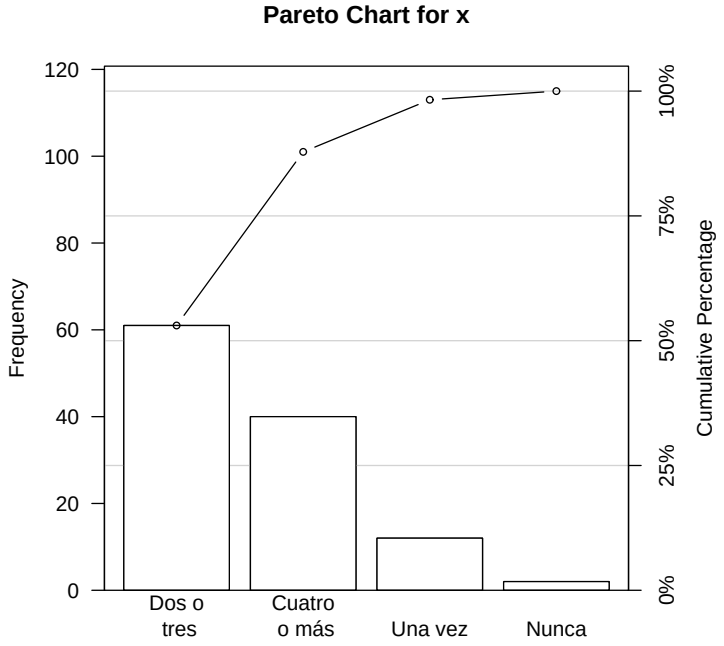
```
> titulo = "Veces que ve televisión durante la semana"
> pie(x, col = "white", main = titulo)
```



2.1.3. Diagramas de Pareto

Las categorías de una variable nominal ordenadas según su frecuencia permite poner en evidencia las más importantes. Los diagramas de Pareto las ordenan de mayor a menor y agregan una línea con la frecuencia acumulada para destacar las de mayor incidencia.

```
> library(qcc)
> pareto.chart(x, col = "white", las = 1)

Pareto chart analysis for x
                Frequency Cum.Freq. Percentage Cum.Percent.
  Dos o\n tres         61        61  53.043478     53.04348
  Cuatro\n o más       40       101  34.782609     87.82609
  Una vez              12       113  10.434783     98.26087
  Nunca                 2       115   1.739130    100.00000
```



La función `pareto.chart` forma parte del paquete `qcc` que debe cargarse antes de utilizar la función. El llamado del paquete se realiza con la instrucción `libray(qcc)`.

Para destacar las categorías de menor frecuencia se considera el extremo derecho del diagrama y el porcentaje acumulado complementario.

2.2. Una variable numérica

2.2.1. Diagramas de ramas y hojas

La función `rnorm(n, m, s)` genera un vector de 50 componentes provenientes de una distribución normal con media m y desviación estándar s. Para estos datos se construye el diagrama de ramas y hojas con la función `stem`. Es obligatorio dar como parámetro el vector de datos. El segundo parámetro en `stem(x, 2)` elabora un diagrama con el doble de ramas que cuando no se incluye. Se pueden dar valores mayores para este segundo parámetro para generar más ramas cuando la cantidad de datos es muy grande.

```
> x = rnorm(50, 100, 8)
> stem(x)

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

   8 | 24
   8 | 589
   9 | 0012222344444
   9 | 5555788999999
  10 | 11222333444
  10 | 556668
  11 | 02

> stem(x, 2)

  The decimal point is at the |
```

```
  82 | 29
  84 | 7
  86 |
  88 | 405
  90 | 26688
  92 | 315899
  94 | 35112
  96 | 97
  98 | 3789233
 100 | 68799
 102 | 5899
 104 | 00668
 106 | 018
 108 |
 110 | 5
 112 | 4
```

Ejercicio Para entender la construcción de los diagramas de ramas y hojas, compare uno a uno los valores de los diagramas y hojas anteriores.

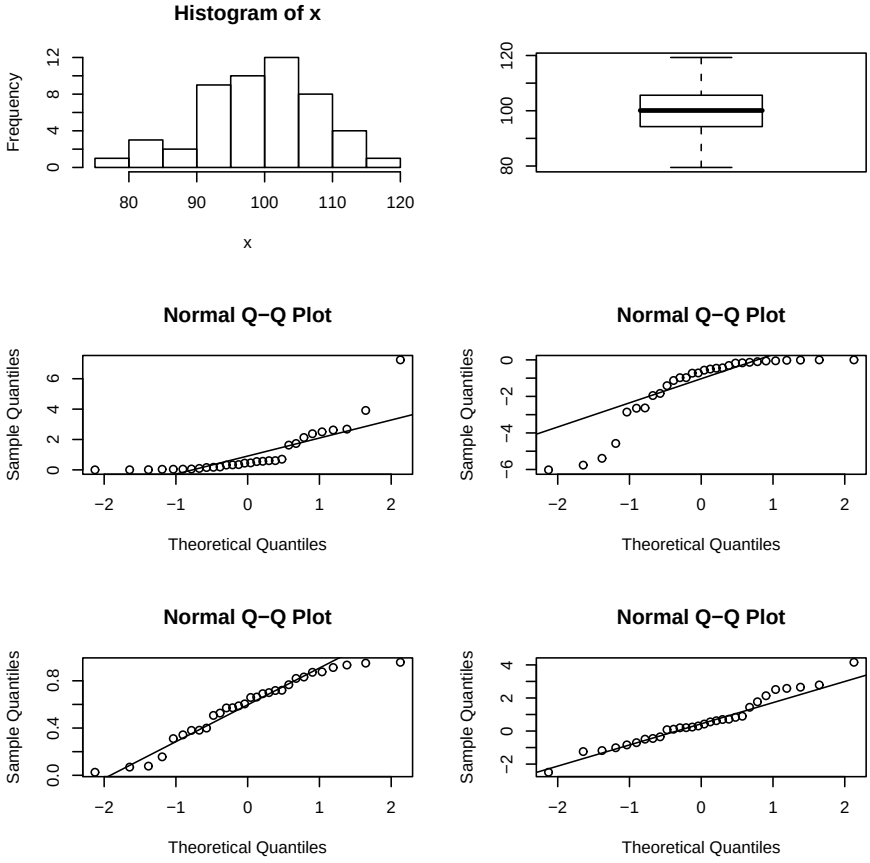
Cree un vector `y` de 40 números pseudoaleatorios de una distribución normal con media 170 y desviación estándar 7. Calcule el diagrama de ramas y hojas sin parámetros opcionales y luego otro con 2 como parámetro adicional. Compare uno a uno los valores de los dos diagramas. Imprima el vector `y` y busque una explicación de la forma como los valores se representan en los diagramas obtenidos.

2.2.2. Histogramas, boxplot y probabilidad normal

Se obtienen con las instrucciones `hist(x)`, `boxplot(x)` y `qqnorm(x)`.

```
> y = rchisq(30, 1)
> z = -rchisq(30, 1)
> u = runif(30)
> v = rt(30, 2)
```

```
> opcion <- par(mfrow = c(3, 2))
> hist(x)
> boxplot(x)
> qqnorm(y)
> qqline(y)
> qqnorm(z)
> qqline(z)
> qqnorm(u)
> qqline(u)
> qqnorm(v)
> qqline(v)
```



Capítulo 3

Inferencia

3.1. Pruebas T

3.1.1. Para una muestra

Supongamos que las hipótesis planteadas son:

$$\begin{aligned} H_0: &\quad \mu_X = 105 \\ H_1: &\quad \mu_X < 105 \end{aligned}$$

y que el vector `x` contiene los datos de la muestra observada. La función `t.test` realiza los cálculos para la prueba.

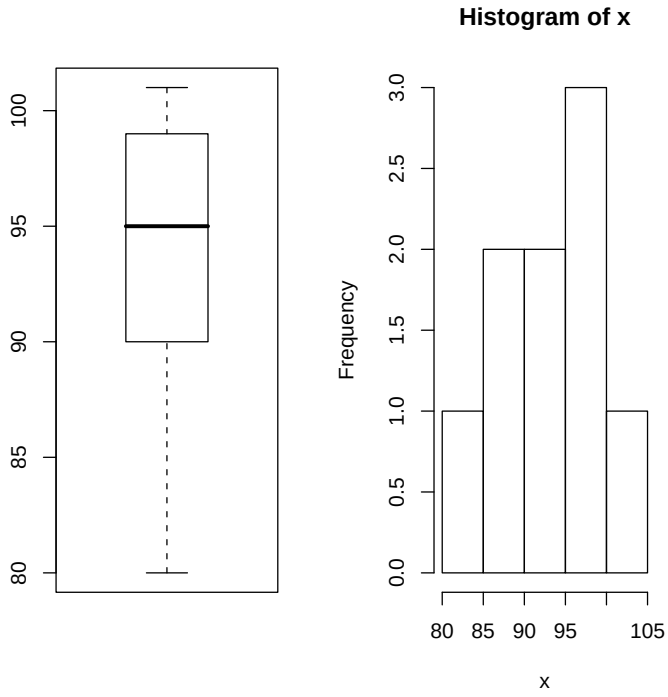
```
> x = c(90, 100, 80, 89, 95, 98, 99, 101, 94)
> t.test(x, m = 105, alt = "less")

        One Sample t-test

data:  x
t = -4.8922, df = 8, p-value = 0.0006028
alternative hypothesis: true mean is less than 105
95 percent confidence interval:
     -Inf 98.18111
sample estimates:
```

```
mean of x
       94
```

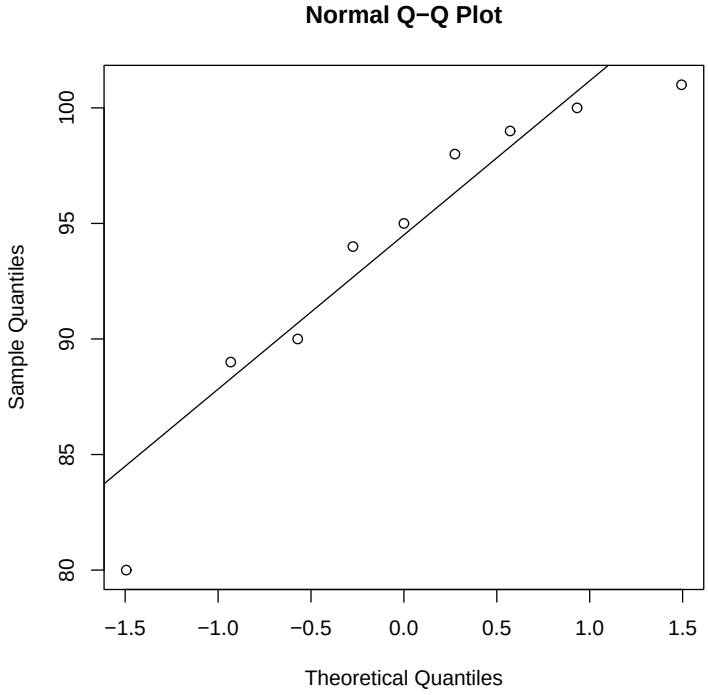
El valor-p es la herramienta clave para tomar la decisión con respecto a H_0: si el valor-p es inferior o igual al nivel de significación, se rechaza H_0 en favor de H_1. La selección adecuada de la lateralidad de la hipótesis alternativa incide directamente sobre el cálculo del valor-p.

Los resultados de la prueba incluyen la estimación puntual de la media y la estimación por intervalo de confianza utilizando un nivel de confianza implícito de 95 %. Si se desea otro valor, debe agregarse la opción `conf.level = 0.9` o el valor que se desee para el coeficiente de confianza del intervalo. La lateralidad del interalo se establece en concordancia con la de la prueba de hipótesis.

Las funciones `hist` y `boxplot` muestran algunos detalles de los datos.



Para tener información gráfica sobre la aceptabiliad de la normalidad de la distribución de la variable X, `qqnorm(x)` y `qqline(x)` presentan un diagrama de puntos asociando los cuantiles empíricos con los teóricos correspondientes a una distribución normal.



Si en un problema específico no se tiene interés o información suficientes para plantear hipótesis y la recolección de los datos se hace con fines exclusivos de estimación, conviene preguntarse si lo que se busca es una cota superior para el valor del parámetro buscado, una cota inferior o un intervalo acotado por los dos extremos. Por ejemplo, si se diseña un tratamiento para reducir el peso de personas obesas y la variable considerada es X, definida como la diferencia entre el peso anterior al tratamiento y el posterior ($D = pre - post$), lo importante para que el tratamiento sea eficaz es tener una garantía de que esa diferencia supere una cota inferior por conocer. El intervalo de confianza adecuado debe mostrar esa cota inferior, luego debe ser unilateral derecho, lo cual equivale a plantear una hipótesis alternativa

de la forma $\mu_D > 0$, así que la opción adecuada es `alt = greater` y no importa el valor que se dé para `m = .` En este caso, los resultados sobre la prueba de hipótesis no se tienen en cuenta.

3.1.2. Para dos muestras emparejadas

Supongamos que las hipótesis planteadas son:

$$H_0: \quad \mu_X - \mu_Y = -1$$
$$H_1: \quad \mu_X - \mu_Y < -1$$

y que las muestras emparejadas obtenidas son:

```
> x = c(90, 100, 80, 89, 95, 98, 99, 101, 94)
> y = c(92, 100, 84, 91, 94, 105, 102, 106, 99)
> t.test(x, y, m = -1, alt = "less", paired = T)

        Paired t-test

data:  x and y
t = -2.3534, df = 8, p-value = 0.02322
alternative hypothesis: true difference in means is less than -1
95 percent confidence interval:
      -Inf -1.419688
sample estimates:
mean of the differences
                     -3
```

La opción `paired=T` condiciona la función `t.test` a realizar los cálculos adecuados para una prueba T con datos emparejados. Como antes, se pueden utilizar las funciones `hist` y `boxplot` para mostrar algunos detalles de los datos.

3.1.3. Para dos muestras independientes

Si se consideran las varianzas poblacionales diferentes Supongamos que las hipótesis planteadas son las mismas de antes:

$$H_0: \quad \mu_X - \mu_Y = -1$$
$$H_1: \quad \mu_X - \mu_Y < -1$$

Además, supongamos que tenemos conocimiento de la igualdad de varianzas poblacionales y que hemos optado por utilizar dos muestras independientes para la prueba de las hipótesis. La función `t.test`, sin el parámetro `paired`, asume que se trata de muestras independientes. De igual manera, mientras no se indique que las varianzas poblacionales son iguales, la función `t.test` asume que son diferentes y utiliza la aproximación de Welch para los grados de libertad de la estdística de prueba.

```
> x = c(90, 100, 80, 89, 95, 98, 99, 101, 94)
> y = c(92, 100, 84, 91, 94, 105, 102, 106, 99)
> t.test(x, y, m = -1, alt = "less")

        Welch Two Sample t-test

data:  x and y
t = -0.6053, df = 15.913, p-value = 0.2768
alternative hypothesis: true difference in means is less than -1
95 percent confidence interval:
     -Inf 2.770393
sample estimates:
mean of x mean of y
       94        97
```

Si se consideran las varianzas poblacionales iguales Para las mismas hipótesis del parágrafo anterior:

$$H_0: \quad \mu_X - \mu_Y = -1$$
$$H_1: \quad \mu_X - \mu_Y < -1$$

```
> x = c(90, 100, 80, 89, 95, 98, 99, 101, 94)
> y = c(92, 100, 84, 91, 94, 105, 102, 106, 99)
> t.test(y, x, m = 1, alt = "greater", var.equal = T)

        Two Sample t-test

data:  y and x
t = 0.6053, df = 16, p-value = 0.2767
alternative hypothesis: true difference in means is greater than 1
95 percent confidence interval:
 -2.768466       Inf
sample estimates:
mean of x mean of y
       97        94
```

Pruebas bilaterales Para las hipótesis:

$$\begin{aligned} H_0 : &\quad \mu_X - \mu_Y = -1 \\ H_1 : &\quad \mu_X - \mu_Y \neq -1 \end{aligned}$$

Si las varianzas poblacionales se consideran iguales:

```
> x = c(90, 100, 80, 89, 95, 98, 99, 101, 94)
> y = c(92, 100, 84, 91, 94, 105, 102, 106, 99)
> t.test(x, y, m = -1, var.equal = T)

        Two Sample t-test

data:  x and y
t = -0.6053, df = 16, p-value = 0.5535
alternative hypothesis: true difference in means is not equal to -1
95 percent confidence interval:
 -10.004248   4.004248
sample estimates:
mean of x mean of y
       94        97
```

Si las varianzas poblacionales se consideran diferentes:

```
> x = c(90, 100, 80, 89, 95, 98, 99, 101, 94)
> y = c(92, 100, 84, 91, 94, 105, 102, 106, 99)
> t.test(x, y, m = -1)

        Welch Two Sample t-test

data:  x and y
t = -0.6053, df = 15.913, p-value = 0.5535
alternative hypothesis: true difference in means is not equal to -1
95 percent confidence interval:
 -10.007348   4.007348
sample estimates:
mean of x mean of y
       94        97
```

Cuando se tiene una variable de grupo Por lo general, en una base de datos se tiene una variable como el *puntaje* en una columna y

en otra se tiene la información sobre los grupos que se quieren comparar, por ejemplo, *sexo*. En estos casos, el llamado de la función `t.test` se hace de la siguiente manera:

```
> x = c(90, 100, 80, 89, 95, 98, 99, 101, 94)
> y = c(92, 100, 84, 91, 94, 105, 102, 106, 99)
> puntaje = c(x, y)
> sexo = rep(c(1, 2), c(9, 9))
> t.test(puntaje ~ sexo, m = -1)

        Welch Two Sample t-test

data:  puntaje by sexo
t = -0.6053, df = 15.913, p-value = 0.5535
alternative hypothesis: true difference in means is not equal to -1
95 percent confidence interval:
 -10.007348   4.007348
sample estimates:
mean in group 1 mean in group 2
             94              97
```

3.2. Comparación de varianzas

La hipótesis nula $H_0 : \sigma_X^2 = \sigma_Y^2$, frente a la alternativa $H_0 : \sigma_X^2 \neq \sigma_Y^2$, se contrasta con la prueba F, mediante la función `var.test`

```
> x = c(90, 100, 80, 89, 95, 98, 99, 101, 94)
> y = c(92, 100, 84, 91, 94, 105, 102, 106, 99)
> var.test(x, y, ratio = 1)

        F test to compare two variances

data:  x and y
F = 0.8626, num df = 8, denom df = 8, p-value = 0.8395
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.1945655 3.8239493
sample estimates:
ratio of variances
         0.8625592
```

Agregando la opción `alt = "greater"` o `alt = "less"` se obtienen las pruebas unilaterales y los intervalos de confianza, de manera similar a la prueba T.

El parámetro `ratio = 1` es opcional, pero es útil cuando en la prueba se consideran hipótesis sobre relaciones diferentes de 1 entre las varianzas, por ejemplo, si se tiene un procedimiento para reducir la variabilidad en el grupo tratado `x` a menos de la mitad de la del grupo `y`, no tratado, se utiliza:

```
> var.test(x, y, ratio = 0.5, alt = "less")

        F test to compare two variances

data:  x and y
F = 1.7251, num df = 8, denom df = 8, p-value = 0.7713
alternative hypothesis: true ratio of variances is less than 0.5
95 percent confidence interval:
 0.000000 2.965566
sample estimates:
ratio of variances
         0.8625592
```

3.3. Bondad de ajuste

```
> x = rnorm(30)
> ks.test(x, "pnorm", 0, 1)

        One-sample Kolmogorov-Smirnov test

data:  x
D = 0.1416, p-value = 0.5378
alternative hypothesis: two-sided

> shapiro.test(x)

        Shapiro-Wilk normality test

data:  x
W = 0.9699, p-value = 0.5372
```

Capítulo 4

Regresión simple

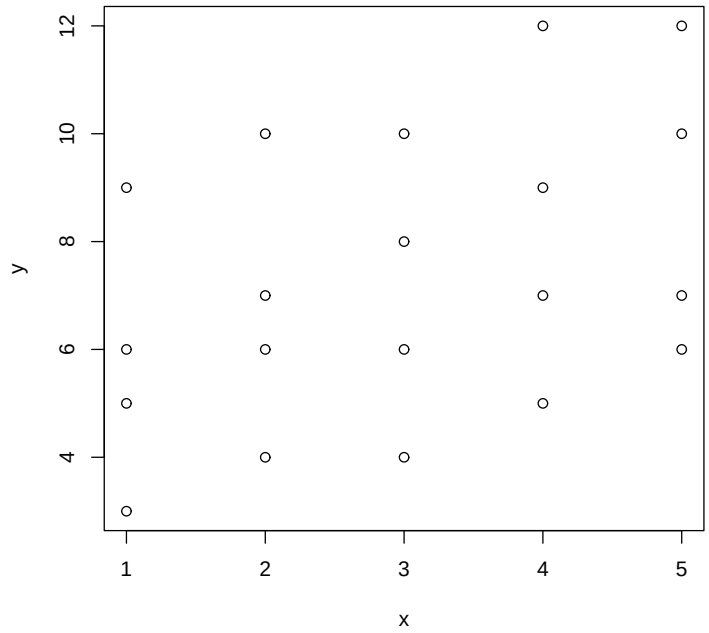
El siguiente ejemplo fue tomado de Pedhazur (1982, p. 12) [1]. Un gráfico de puntos asociando las variables muestra la tendencia en el comportamiento de la variable dependiente en función de la independiente.

```
> x = rep(c(1, 2, 3, 4, 5), c(4, 4, 4, 4, 4))
> y = c(3, 5, 6, 9, 4, 6, 7, 10, 4, 6, 8, 10, 5, 7, 9, 12, 6, 7,
+     10, 12)
> cor(x, y)
```

```
[1] 0.4157055
```

```
> plot(x, y)
```

[1] Pedhazur, E. J., Multiple Regression in Behavioral Research, CBS College Publishing, New York, 1982




```
> lmyx = lm(y ~ x)
> summary(lmyx)

Call:
lm(formula = y ~ x)

Residuals:
   Min     1Q Median     3Q    Max
-3.300 -1.988 -0.175  1.575  3.950

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.0500     1.2827   3.937 0.000967 ***
x             0.7500     0.3868   1.939 0.068318 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.446 on 18 degrees of freedom
Multiple R-Squared: 0.1728,        Adjusted R-squared: 0.1269
```

```
F-statistic:  3.76 on 1 and 18 DF,  p-value: 0.06832
```

```
> anova(lmyx)

Analysis of Variance Table

Response: y
          Df  Sum Sq Mean Sq F value  Pr(>F)
x          1  22.500  22.500  3.7604 0.06832 .
Residuals 18 107.700   5.983
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> opcion = par(mfrow = c(2, 2))
> plot(fitted.values(lmyx), residuals(lmyx))
> hist(residuals(lmyx))
> boxplot(residuals(lmyx))
> qqnorm(residuals(lmyx))
> qqline(residuals(lmyx))
> par(opcion)
```