Prácticas de Estadística con R

Departamento de Matemáticas
Universidade da Coruña

Curso 2018-2019

Práctica 1

Introducción a R

1.1. Qué es R

- En 1976 los Laboratorios Bell desarrollan el lenguaje S, llamándose su implementación comercial S-PLUS.
- En 1993, en la Universidad de Auckland, se implementó otro dialecto de S, al que llamaron R; desde 1995 es código abierto.
- ¿Qué es R?
  R es un lenguaje de programación orientado a objetos y un sistema de análisis de datos.
- ¿Qué permite hacer R?
  Análisis estadísticos y gráficos.
- ¿Cómo funciona R?
  Utilizando un intérprete de comandos, no a través de un sistema GUI (interfaz gráfica de usuario). Es más difícil de aprender, pero a cambio es mucho más flexible y potente. No obstante, el paquete R-Commander `(Rcmdr)` permite utilizar una interfaz de menús.
- ¿Cuánto cuesta R?
  R es gratuito (software libre) y accesible bajo Licencia Pública General de GNU.
  Lo podemos encontrar en *www.r-project.org*.

1.2. Primeros pasos con R

- Símbolo del sistema: >
- Comentarios: #
- Separador de comandos: habitualmente un cambio de línea. También punto y coma (;).
- Calculadora: $+$, $-$, $*$, $/$, $\%\%$ (módulo), $\%/\%$ (división entera), $\hat{}$
- Operadores de asignación: $<-$ (siempre válido), $=$ (válido en el nivel superior; no se recomienda su uso).
- Constantes lógicas: `TRUE` o `T`, y `FALSE` o `F`.
- Operadores lógicos y relacionales: `!A` (not), `A&B` (and), `A | B` (or), `xor(A,B)`, $<$, $>$, $==$, $>=$, $<=$, $!=$
- Valores especiales: `NA` (no disponible), `NaN` (no es un número), `Inf`.

- Distingue entre mayúsculas y minúsculas.
- Por defecto utiliza doble precisión: 53 dígitos binarios (15 ó 16 decimales). La visulización en pantalla de los números se redondea a un número razonablemente pequeño de decimales.
- *Workspace*: al tratarse de un lenguaje interpretado, es importante saber en todo momento los objetos disponibles en memoria. Debemos distinguir entre los objetos predefinidos por el sistema y los definidos por nosotros. Para reutilizar en sesiones próximas nuestos objetos tenemos dos opciones: dejar que el propio R los almacene y recupere, o guardarlos nosotros en un fichero de texto plano.
- *Consola*: las instrucciones pueden escribirse directamente en la consola, y se ejecutan según se van introduciendo ya que no hay un proceso de compilación.
- *Script*: en general, es más comodo escribir las instrucciones en un fichero de texto. Se pueden utilizar los ficheros *script* que incorpora el entorno, muy básicos, o instalar un editor más potente. Nosotros utilizaremos el IDE gratuito RStudio. Otra alternativa es Tinn-R.

1.3. Funciones internas

R tiene predefinidas cientos de funciones en su paquete básico (paquete es el nombre que se le da en R a las bibliotecas de funciones). Adicionalmente se pueden utilizar otros paquetes, sin más que cargarlos en la memoria. Hay una cantidad sorprendente de recursos disponibles en estos paquetes. Para utilizar una función se escribe:

`nombre_función(argumento1,argumento2,...)`, si no se especifican los valores de los argumentos se tomarán valores predefinidos por defecto. Un nombre seguido de paréntesis hace siempre referencia a una función. Observemos lo que ocurre si escribimos el nombre de la función, con y sin los paréntesis, en los siguientes ejemplos:

- `q()` (acrónimo de `quit`).
- `ls()`
- `rm()` (acrónimo de `remove`). Si escribimos `rm(list=ls())`, se borrarán todos los objetos definidos por el usuario.
- `dir()`
- `library()`
- `getwd()`

1.4. Ayuda

- `help()` (online), por ejemplo `help(mean)`, `?mean`
- `help.start()` (html)
- `demo()`, por ejemplo `demo(graphics)`
- `help.search("optimization")`
- `apropos("texto")` (lista objetos que contienen *"texto"*)
- `example("texto")`
- `RSiteSearch("missing")` (función que busca en internet)

1.5. Una interfaz gráfica: el paquete R-Commander

El paquete R-Commander (`Rcmdr`) permite utilizar una interfaz de menús. Una vez iniciado R, este paquete se carga mediante la instrucción `library(Rcmdr)`. Si se cierra la ventana de R-Commander se puede volver a iniciar con la instrucción `Commander()`.

1.6. Unos primeros cálculos

- Realiza las siguientes operaciones:
  `n <- 1:10` (calcula la secuencia de los 10 primeros números naturales)
  `1.25*(n*0.8)-n`
  ¿Cómo se explican los resultados?
- Realiza la comprobación $(49 * (4/49) == 4)$.
- Procedemos ahora a calcular la cuota mensual R de devolución de un préstamo de un capital $P = 1500$ a un interés mensual $i = 0.01$ en $n = 10$ meses:

$$R = P\frac{i}{1-(1+i)^{-n}}$$

En la siguiente sección se explican las estructuras básicas de programación. Utilízalas para escribir una función que calcule la fórmula anterior.

1.7. Programación

La programación en R no es declarativa (HTML, LaTeX, programación funcional (Lisp, Haskell), programación lógica (Prolog), por ejemplo), sino imperativa, y los programas pueden escribirse en estilo procedimental (o procedural), que es el estilo clásico de programación (C, Pascal), en estilo modular, funcional (Mathematica), orientado a objetos (Java)...

1.7.1. Funciones

```
nombre <- function(parámetros) {
      comandos
      return(nombre_objeto_salida)
}
```

Las funciones en R producen una única salida y no salidas múltiples. Si nos interesa retornar varios valores, los agrupamos dentro de un único objeto. Mediante `return()` podemos especificar la salida. Caso de faltar esta instrucción, sería la última orden ejecutada.

Al llamar a una función podemos identificar los parámetros de dos formas distintas: poniéndolos en el mismo orden en el que están definidos en la función; y especificando el nombre de los parámetros, en cuyo caso los podemos poner en cualquier orden.

En la lista de parámetros podemos poner valores por defecto, en la forma `nombre = valor`. Lo que no se puede hacer es: `nombre <- valor`. Las dos formas pueden usarse al *llamar* a una función, pero con significados distintos: en el primer caso le pasamos los valores de los parámetros (argumentos) identificándolos por los nombres, mientras en el segundo caso asignamos valores a variables y simultáneamente llamamos a la función pasándole unos argumentos que *no* están siendo identificados por el nombre, sino por el orden de aparición.

El *entorno* de una función incluye los objetos definidos en la consola; y las variables definidas en una función serán también reconocidas en cualquier otra nueva función definida dentro de la anterior. Pero no al revés.

1.7.2. Bucles

`for (índice in vector) {comandos}`

`while (condición) {comandos}`

`repeat {comandos}` repite los comandos sin fin.

Estos bucles se pueden combinar con las órdenes `break` (salir del bucle) y `next` (volver al principio del bucle).

1.7.3. If

```
if (condición) {
      comandos cuando TRUE
} else {
      comando cuando FALSE
}
```

1.8. Vectores, indexado

Comprueba lo que realizan los siguientes comandos:

- `a <- c(10,20)` (la función `c` (de combinar) es la forma más sencilla de crear un vector).
- `1:21; seq(1,21)` (estas dos instrucciones son equivalentes).
- `seq(1,21,by=2)`
- `seq(3.5,21.4,by=2)`
- `seq(1,21,length=6)`
- `a <- c(a,4:1)`
- `a[4]` (los índices se especifican usando corchetes).
- `a[c(2,4)]` (podemos hacer referencia a más de un índice a la vez).
- `a[1:5]`
- `a[c(T,F)]`
- `a[-2]`
- `a[a>3]`
- `rep(3,12)` (observa lo que hace la función `rep`).
- `rep(c(1,4),3)`
- `rep(c(1,4),c(3,2))`
- `rep(c(1,4), each=3)`
- `b <- c(2,-2)`
- `2+a; 2*a; a*b` (operaciones con funciones de distinta longitud (reciclado)).

1.9. Unas primeras fórmulas de estadística

Al ser R un lenguaje con orientación estadística, no solo tiene predefinidas las funciones matemáticas estándar, sino también una cantidad importante de funciones estadísticas. Por ejemplo:

`min(), max(), length(), sum(), prod(), sort()`

`mean(), sd(), var(), summary()`

Media

La media de los datos $(x_1, \ldots, x_n)$ se define

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

y en R la podemos calcular de las siguientes formas:

- `x <- 1:11; n <- length(x); sum(x)/n`
- `x <- 1:11; mean(x)`
- Escrito en forma de función:

```
media <- function(x) {
       n <- length(x)
       m <- sum(x)/n
       return(m)
}
```

Varianza

La varianza se define

$$s^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

y en R la podemos calcular de las siguientes formas:

- `x <- 1:11; n <- length(x); sum((x-mean(x))^2)/n`
- `x <- 1:11; mean((x-mean(x))^2)`

Cuasi-varianza

La cuasi-varianza se define

$$\widehat{s}^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

y en R se calcula utilizando la función interna `var()`. A continuación vamos a comprobar los peligros de utilizar una expresión equivalente, conocida como fórmula rápida porque su cálculo requiere menos iteraciones.

- Crea una función `cuasi_var(x)` que calcule la fórmula

$$\frac{1}{n-1} \left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2 \right)$$

- Define un vector `x` que contenga los 11 primeros números naturales y calcula su cuasi-varianza utilizando `var(x)` y `cuasi_var(x)`.
- Define `y <- x + 1.e10` y repite el apartado anterior con los nuevos datos. ¿Qué ha sucedido?

Operaciones

- Dados los datos `x <- 1:11`, calcula su media y su varianza.
- Dados los nuevos datos `y <- 2+3*x`, calcula su media y su varianza.
- Intenta sugerir una relación entre los valores de los dos apartados anteriores. Prueba con nuevos valores para `x` y diferentes transformaciones `y`.
- Dados los nuevos datos `z <- x^2`, calcula su media. Intenta relacionarla con $\overline{x}^2$. Ahora intenta relacionarla con $\overline{x}^2$ y s_x^2.

Media truncada

Escribe una función que implemente el cálculo de la media truncada al 20 % (una media alternativa) conforme al siguiente algoritmo:

Input: un vector $x = \{x_1, \ldots, x_n\}$.

Output: la media truncada de x al 20 %.

1. Calcular el número de elementos de x; llamarle n.
2. Calcular el entero más próximo al 20 % de n; llamarle k.
3. Ordenar los elementos de x; seguir llamándole x.
4. Eliminar los k primeros y los k últimos elementos de x; llamarle y al vector formado por los elementos no eliminados.
5. Devolver la media de y.

1.10. Data frame

- *Data frame*: es una estructura de datos en forma de tabla en la que cada columna representa una variable. Las columnas pueden ser de tipos diferentes y tienen nombre (su generalización se llama `list`). Por ejemplo `iris` (se trata de un objeto predefinido que contiene datos muy famosos en estadística), del que podemos obtener información utilizando `?iris` y `str(iris)`. Otro ejemplo:
```
x <- 2:4; y <- seq(2,6,by=2)
a <- data.frame(x, y, z=c(2,4,5))
```
- `datos <- read.table("C:/.../practica2.txt", header=T)`
`read.table` asume que los datos están separados por espacios, y guarda los datos en un *data frame*.
- `datos$x` accede a la columna `x` del objeto `datos`.
- La función `names(datos)` lista los nombres de las columnas (objetos).
- Existe un *Editor de datos* (equiv. `fix(datos)`), pero el *data frame* `datos` ha de existir previamente; se puede crear con `datos <- data.frame()`. `Rcmdr` es una alternativa cómoda.

1.11. Un poco sobre gráficas

Histograma

- `x <- rnorm(100)`
- `hist(x)`
- `hist(x, freq=F)`
- `h <- hist(x, plot=F)`
- `h`
- `h$density`

Scaterplot (nube de puntos)

- `x <- seq(-2,2,0.1)`
- `y1 <- x^3`
- `y2 <- x^2`
- `plot(x,y1)`
- `plot(x,y1,type="l")`
- `lines(x,y2)`
- `points(x,x)`
- `plot.new()`
- `lines(x,y2)`

Curvas

- `curve(expr=sin, from=0, to=6*pi)`
- `curve(cos, add=T)`
- `curve(expr=x^2-2*x, from=1, to=10)`
- `curve(x^3, add=T)`

1.12. Ejemplo de función recursiva

La conocida sucesión de Fibonacci se define, matemáticamente, de forma recursiva:

$$a_1 = 1,\ a_2 = 1,\ a_n = a_{n-2} + a_{n-1}\ \forall n \geq 3$$

Si nos interesa el cálculo de su término n-ésimo, con $n = 20$, podemos definir una función recursiva en R, aunque resulta más eficiente resolver este problema con un simple bucle.

- Definiendo una función recursiva:

```
f <- function(n) {
       if (n<=2) {
            a <- 1
       } else {
            a <- f(n-1) + f(n-2)
       }
       return(a)
}
for (i in 1:20) {print(f(i))}
system.time( f(20) )
system.time( f(30) )
```

- Utilizando un bucle:

```
tiempo.inicial <- proc.time()
Fibonacci <- rep(NA,20)
Fibonacci[1] <- Fibonacci[2] <- 1
for ( i in 3:20) {
       Fibonacci[i] <- Fibonacci[i-2] + Fibonacci[i-1]
}
print(Fibonacci[20])
tiempo.final <- proc.time()
print(tiempo.final - tiempo.inicial)
```

1.13. Ejercicio: quicksort

El algoritmo *quicksort con pivote aleatorio* es un método de ordenación simple, práctico y eficiente. Debemos escribir una función recursiva de nombre `Quicksort` que lo implemente, conforme a las siguientes instrucciones:

Input: un vector $x = \{x_1, \ldots, x_n\}$.

Output: los elementos de x ordenados.

1. Si x tiene cero o un elementos, devolver x. En caso contrario continuar.
2. Elegir al azar un elemento de x como pivote; llamarle a.
3. Comparar cada uno de los restantes elementos de x con a y construir dos nuevos vectores:
   a) $x1$ contiene todos los elementos de x, excepto a, que son menores o iguales que a.
   b) $x2$ contiene todos los elementos de x que son mayores que a.
4. Utilizar `Quicksort` para ordenar $x1$ y $x2$.
5. Devolver la lista $x1, a, x2$.

1.14. Ejercicio: media recortada

Escribe una función que implemente el cálculo de la media recortada al 20 %:

La media recortada es una variante de la media truncada. Se diferencia de esta en que en lugar de simplemente eliminar los primeros elementos del vector (ordenado), se reemplazan por el menor de los datos restantes; y de forma análoga los últimos elementos del vector se reemplazan por el mayor de los restantes.

Práctica 2

Probabilidad

2.1. Problema del cumpleaños

El problema del cumpleaños se puede enunciar: dada una habitación con n personas, calcula la probabilidad de que al menos dos de ellas tengan el mismo cumpleaños.

Nos interesa escribir una función que calcule las probabilidades en función de n ($1 \leq n \leq 100$) y las represente gráficamente. Además, queremos calcular el número mínimo de personas necesario para que la probabilidad pedida sea mayor o igual a 0.5.

```
# Paradoja del cumpleaños
# Probabilidad de que al menos 2 personas de n compartan cumpleaños
cumple <- function(nmax=100){
      dif <- numeric(nmax)
      for (n in 1:nmax){
            dif[n] <- prod(365:(365-n+1))/365^n
      }
      p <- 1-dif
      plot(1:nmax, p, main="Paradoja del cumpleaños", xlab="Número de personas",
      ylab="Probabilidad de que al menos dos personas compartan cumpleaños")
      abline(h=0.5,v=23)
      return(p)
}
p <- cumple()
which(p==min(p[p>=0.5])) # Alternativa: método dicotómico
```

2.2. Ejercicio: filtro de SPAM

El teorema de Bayes es una de las herramientas que se utilizan para decidir de forma automática si un mensaje de correo electrónico se clasifica como SPAM. En este ejercicio vamos a implementar una versión muy simple de este método.

Disponemos de una muestra de aprendizaje (se llama así porque de ella aprenderemos lo que tenemos que hacer en el futuro) formada por 10 mensajes de los que sabemos cuales son SPAM y cuales no. Seleccionamos 3 palabras que consideramos críticas (A, B, C) y comprobamos cuales de los mensajes las contienen. La información obtenida la mostramos en la tabla siguiente, en la que cada fila representa un mensaje, las tres primeras columnas indican si la palabra en cuestión está en el mensaje o no, y la última columna si el mensaje es SPAM o no:

A	B	C	S
0	1	0	1
1	1	1	1
0	0	1	0
0	0	0	0
0	1	1	1
1	0	1	1
0	0	0	0
1	0	0	0
1	1	0	1
1	1	1	1

1. La tabla anterior se encuentra en el fichero `practica2.txt`. Lee la tabla en R utilizando la instrucción `read.table` y ponle el nombre `tabla`.

2. Basándote en la tabla, calcula las probabilidades a priori de que un mensaje sea SPAM ($S = 1$) y de que no lo sea ($S = 0$)

$$P(S = 0), P(S = 1)$$

   Puede resultarte de utilidad:

```
n<-10; m<-4; sum(tabla[,m])/n
```

3. Calcula las probabilidades:

$$P(A = 1 \mid S = 0), P(B = 1 \mid S = 0), \ldots, P(C = 1 \mid S = 1)$$

   Puede resultarte de utilidad:

```
rowsum(tabla[,-m],group=tabla[,m])
```

4. Recibimos un nuevo mensaje, comprobamos si contiene las palabras críticas y obtenemos que ($A = 0$, $B = 0$, $C = 1$), que por comodidad denotamos 001. Suponiendo que las tres palabras son independientes, calcula las probabilidades

$$P(001 \mid S = 0), P(001 \mid S = 1)$$

5. Utilizando el teorema de la probabilidad total, calcula

$$P(001)$$

6. Utilizando el teorema de Bayes, calcula la probabilidad de que el nuevo mensaje sea SPAM condicionada a que ($A = 0$, $B = 0$, $C = 1$)

$$P(S = 1 \mid 001)$$

7. Si la probabilidad anterior es mayor que 0.8 clasifica el nuevo mensaje como SPAM.

El procedimiento anterior se puede generalizar para clasificar un nuevo mensaje arbitrario $x_1x_2x_3$. Para ello, podría ser útil el "truco":

$$P(A = x_1 \mid S = 0) = P(A = 0 \mid S = 0)^{1-x_1} P(A = 1 \mid S = 0)^{x_1}$$

2.3. Ejercicio: bolas y urnas (generalización del problema del cumpleaños)

1. Escribe una función que calcule las probabilidades de que al repartir aleatoriamente $n = 10$ tareas entre r procesadores ($10 \leq r \leq 500$), ningún procesador tenga más de una tarea asignada; y que represente gráficamente las probabilidades en función de r.

2. Calcula el valor mínimo de r que garantiza una probabilidad mayor o igual a 0.9.

Práctica 3

Introducción a la simulación

3.1. Introducción

La realidad puede ser muy compleja por lo que se suele emplear un modelo para tratar de explicarla. En muchos casos, como no se dispone de la suficiente información sobre las variables que influyen en el fenómeno en estudio, se recurre a modelos estocásticos (con componente aleatoria) que tienen en cuenta esta incertidumbre (la inferencia estadística proporciona herramientas para estimar los parámetros y contrastar la validez del modelo a partir de los datos observados).

La idea es emplear el modelo para resolver el problema de interés. Cuando la solución no se puede obtener de modo analítico (teórico) se puede recurrir a la simulación (suele ser más lento y computacionalmente costoso). Nos centraremos en el caso de la simulación estocástica, donde las conclusiones se obtienen habitualmente generando repetidamente simulaciones del modelo aleatorio. Se basan por tanto en la generación de números aleatorios.

Entre las aplicaciones prácticas de la generación de números aleatorios podríamos destacar:

- Estadística: muestreo, comparación de métodos (de estimación o contraste de hipótesis), distribución estadísticos...
- Optimización: Algoritmos genéticos, ...
- Computación: Diseño, verificación y validación de algoritmos, ...
- Criptografía: Protocolos de comunicación segura (SSH, SSL), hacking ...
- Física: Simulación de fenómenos naturales, ...
- Análisis numérico, etc.

3.2. Generación de números (pseudo)aleatorios

Una sucesión de números aleatorios puros (true random), se caracteriza por que no existe ninguna regla o plan que nos permita conocer sus valores.

3.2.1. Tablas de números aleatorios

Originalmente (antes del desarrollo de los ordenadores) los números aleatorios se obtenían por procesos físicos (loterías, ruletas, ruidos,...) y se almacenaban en tablas (true random), de la forma:

```
73735 45963  78134 63873
02965 58303  90708 20025
98859 23851  27965 62394
33666 62570  64775 78428
81666 26440  20422 05720

15838 47174  76866 14330
89793 34378  08730 56522
78155 22466  81978 57323
16381 66207  11698 99314
75002 80827  53867 37797

99982 27601  62686 44711
84543 87442  50033 14021
77757 54043  46176 42391
80871 32792  87989 72248
30500 28220  12444 71840
```

Para seleccionar números aleatorios en un rango de 1 a m:

- Se selecciona al azar un pto de inicio en la tabla y una dirección
- Se agrupan los dígitos de forma que "cubran" el valor de m
- Se seleccionan los valores menores o iguales que m (se descarta el resto)

Algunos enlaces:

- A Million Random Digits with 100,000 Normal Deviates. RAND Corporation. 1955: *http://en.wikipedia.org/wiki/AMillionRandomDigitswith100,000NormalDeviates*
- Generador de números aleatorios online (ver paquete **random** en R): *http://www.random.org/integers*

3.2.2. Generación de números pseudoaleatorios mediante software

En la actualidad estos números son generados por ordenador empleando algoritmos recursivos y se denominan pseudoaleatorios. Los números de la sucesión serán predecibles, conociendo el algoritmo y el primer valor, llamado semilla. Sin embargo, si no se conoce previamente el algoritmo, no se debería poder distinguir una serie de números pseudo-aleatorios de una sucesión de números verdaderamente aleatoria (utilizando recursos computacionales razonables). En caso contrario esta predicibilidad puede dar lugar a serios problemas (e.g. http://eprint.iacr.org/2007/419)

La mayoría de los métodos de simulación se basan en la posibilidad de generar números pseudo-aleatorios con distribución $U(0,1)$. Hay una gran cantidad de algoritmos de este tipo:

- Cuadrados medios, Lehmer,...
- Congruenciales
- Registros desfasados
- Combinaciones,...

(Código fuente disponible en múltiples librerias, e.g. GNU Scientific Library).

En cualquier caso, todo generador de números pseudoaleatorios mínimamente aceptable debe comportarse como si se tratase de una muestra genuina de datos independientes de una $U(0,1)$. Otras propiedades de interés serían:

- Reproducibilidad a partir de la semilla.

- Periodo suficientemente largo.
- Rapidez.
- Consumir poca memoria.

3.2.3. Generación de números pseudoaleatorios en R

La generación de números pseudoaleatorios en R es una de las mejores disponibles en paquetes estadísticos.

Algunas de las funciones básicas:

- `set.seed`: permite establecer la semilla (y el generador)
  - `set.seed(entero)`
- `rdistribución(n,...)`: genera valores aleatorios de la correspondiente distribución.
  - `runif(n, min=0, max=1)`
- `sample`: muestras aleatorias y permutaciones.

La semilla se almacena en `.Random.seed`, un vector de enteros cuya dimensión depende del tipo de generador. No debe ser modificado manualmente; se guarda con el entorno de trabajo. Si no se especifica con `set.seed` (o no existe) se genera a partir del reloj del sistema.

Como regla general, por lo menos mientras se está desarrollando un programa, interesa fijar la semilla de aleatorización. Esto permite la reproductibilidad de los resultados y facilita la depuración del código.

3.3. Método congruencial

El generador de números pseudoaleatorios *congruencial lineal* consiste en, partiendo de una semilla inicial x_0, construir para cada $n = 0, 1, 2, \ldots$

$$\begin{aligned} x_{n+1} &= (a + bx_n) \mod m \\ u_{n+1} &= \frac{x_{n+1}}{m} \end{aligned}$$

donde a, b y m son parámetros fijados de antemano.

Algunas de las combinaciones de parámetros serían las siguientes:

- $a = 0$, $b = 2^{16}+3 = 65539$ y $m = 2^{31}$, esta es la rutina `RANDU` que implementaban los ordenadores de IBM (no recomendable, ver ejercicio siguiente),
- $a = 0$, $b = 7^5 = 16807$ y $m = 2^{31}-1$, empleados por la librería IMSL (International Mathematics and Statistics Library).

A continuación vamos a comprobar la sensibilidad de este método a la elección de los parámetros:

1. Desarrolla el código necesario para la generación de números pseudoaleatorios con el método congruencial lineal, utilizando como parámetros por defecto $a = 0$, $b = 7^5$ y $m = 2^{31} - 1$.
2. Obtén 500 números entre 0 y 1 utilizando el método anterior, con parámetros $a = 1$, $b = 5$ y $m = 512$ (de ciclo máximo).

3. Representa el histograma de la variable resultante y compárala con la densidad teórica.

4. Calcula la media de las simulaciones (`mean`) y compárala con la teórica.

5. Aproxima (mediante simulación) la probabilidad del intervalo $(0.4; 0.8)$ y compárala con la teórica.

6. Representa los pares de datos (u_i, u_{i+1}) , ¿se observa algún problema?

7. Repite los apartados anteriores utilizando los parámetros por defecto.

3.4. Ejemplo de simulación

En este ejemplo vamos a simular una probabilidad geométrica. El problema es el siguiente: si elegimos un punto (X, Y) al azar dentro de un cuadrado de área 4 centrado en el origen, $[-1, 1]\times[-1, 1]$, ¿cuál es la probabilidad de que $X + Y < 0$? La forma de proceder es la siguiente:

1. Elige dos números al azar entre -1 y 1, es decir, según una distribución $U(-1, 1)$, y comprueba si su suma es menor que 0.

2. Repite el apartado anterior un número grande de veces y comprueba la proporción de aciertos.

3. Adicionalmente, compara la probabilidad aproximada mediante simulación con la teórica (obtenida aplicando la regla de Laplace $\frac{\text{área favorable}}{\text{área posible}}$).

- Código propuesto:

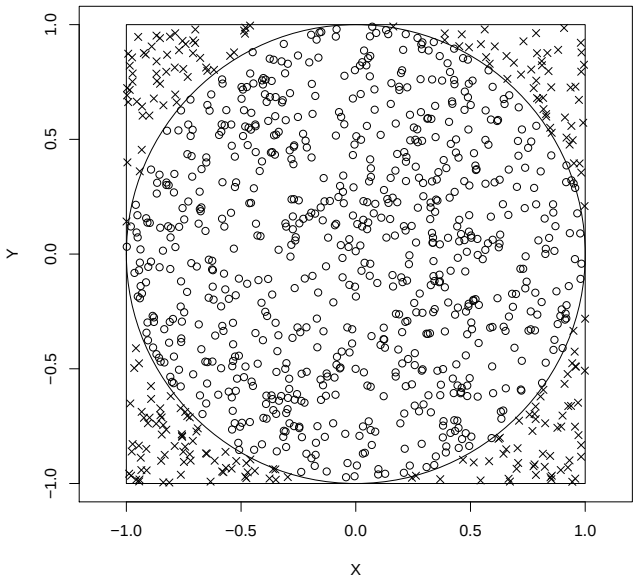
```
set.seed(1)
n <- 10000
x <- runif(n, -1, 1)
y <- runif(n, -1, 1)
indice <- (x+y < 0)
solucion <- mean(indice)
print(solucion)
```

3.5. Aproximación del valor de π mediante simulación

Continuando con el ejemplo anterior, aproximar el valor de π mediante simulación a partir de

$$P\left(X^2 + Y^2 \leq 1\right)$$

donde X e Y son variables aleatorias independientes de distribución $U(-1, 1)$.



3.6. Simulación de variables discretas

El procedimiento habitual para simular una variable aleatoria discreta X consiste en generar valores aleatorios con distribución uniforme en el intervalo (0,1) (por ejemplo utilizando la función `runif`) y asociar a cada posible valor x_i de X un subintervalo con la correspondiente probabilidad. Otra opción es utilizar `sample(valores, numsim, replace = TRUE, prob = probabilidades)`.

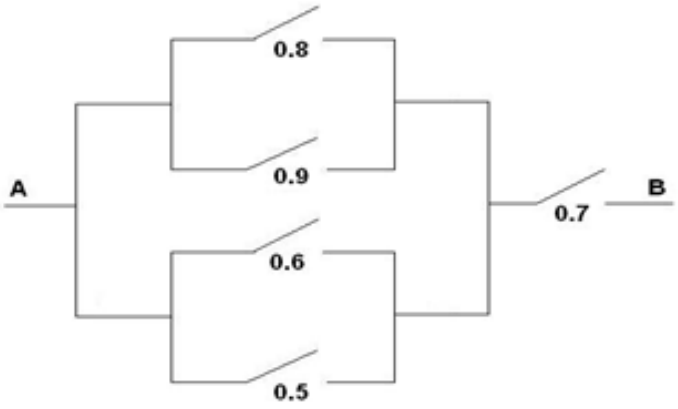
A continuación vamos a simular la probabilidad de cara en el lanzamiento de una moneda:

- Código propuesto:

```
set.seed(1)
n <- 10000
resultado <- c(cara=1,cruz=0)
x <- sample(resultado, n, replace=TRUE, prob=c(0.5,0.5) )
mean(x)
# Alternativa
y <- runif(n)<0.5
mean(y)
```

3.7. Ejercicio

Simula el paso de corriente a través del siguiente circuito, donde figuran las probabilidades de que la corriente pase por cada uno de los interruptores:



Considera que cada interruptor es una v.a. de Bernoulli independiente y emplea la rutina `rbinom` para simular 10000 valores de cada una de ellas.

3.8. Ejercicio

En 1651, el Caballero de Méré le planteó a Pascal una pregunta relacionada con las apuestas y los juegos de azar: ¿es ventajoso apostar a que en cuatro lanzamientos de un dado se obtiene al menos un seis? Este problema generó una fructífera correspondencia entre Pascal y Fermat que se considera, simbólicamente, como el nacimiento del Cálculo de Probabilidades.

1. Escribe una función que simule el lanzamiento de n dados. El parámetro de entrada es el número de lanzamientos n, que toma el valor 4 por defecto, y la salida debe ser `TRUE` si se obtiene al menos un 6 y `FALSE` en caso contrario.

2. Escribe un programa que utilice la función anterior para simular $N = 10000$ jugadas de este juego y calcule la proporción de veces que se gana la apuesta (obtener al menos un 6 en n lanzamientos), usando $n = 4$.

3. Compara la proporción anterior con la probabilidad teórica de ganar, $1 - (5/6)^n$.

Práctica 4

Variables aleatorias

4.1. Introducción

En esta práctica vamos a estudiar las principales distribuciones de probabilidad univariantes discretas y continuas utilizando el lenguaje R. Gran parte de lo que practicaremos se podría hacer también con R-Commander; para ello cargamos el paquete

```
library(Rcmdr)
```

y vamos al menú:

Distribuciones→Distribuciones Discretas

Distribuciones→Distribuciones Continuas

Desde este menú se pueden calcular probabilidades, cuantiles, representaciones gráficas y muestras de la distribución de probabilidad elegida, tanto discretas como continuas.

Para trabajar en lenguaje R debemos tener en cuenta las siguientes asignaciones de nombre:

Distribución	**Terminación asociada**
Binomial	`binom`
Binomial Negativa	`nbinom`
Poisson	`pois`
Uniforme	`unif`
Exponencial	`exp`
Normal	`norm`

Y las siguientes aplicaciones

Aplicación	**Prefijo asociado**
Densidad	`d`
Función de distribución	`p`
Cuantil	`q`
Muestras aleatorias	`r`

Para conocer cómo llamar a las funciones **densidad**, **distribución** y **cuantil** de las distribuciones de interés utiliza la ayuda de R:

```
?dbinom  ?pbinom  ?qbinom
```

4.2. Principales distribuciones univariantes discretas

Las distribuciones discretas que estudiamos son: binomial, binomial negativa y Poisson. En estos casos, la aplicación "densidad" (prefijo `d`) calcula las probabilidades en los puntos que le indiquemos. Por ejemplo:

```
dbinom(5, size=10, prob=0.5)
```

calcula $P(X = 5) = 0.246$, siendo $X \in B(10, 0.5)$, es decir, calcula la probabilidad de que el número de éxitos obtenidos sea 5 en una distribución binomial con 10 intentos y probabilidad de éxito 0.5.

```
dbinom(0:10, 10, 0.5)
```

nos devuelve

```
 [1] 0.0009765625 0.0097656250 0.0439453125 0.1171875000 0.2050781250 0.2460937500
 [7] 0.2050781250 0.1171875000 0.0439453125 0.0097656250 0.0009765625
```

que son las probabilidades de obtener 0 éxitos, 1 éxito, ..., 10 éxitos, respectivamente. Si nos interesa el cálculo de la probabilidad acumulada hasta 3 éxitos $P(X \leq 3) = P(X = 0) + \ldots + P(X = 3)$ (llamada **función de distribución** en el valor 3) podemos utilizar

```
sum(dbinom(0:3, 10, 0.5))
```

y también podemos usar la aplicación "distribución" (prefijo `p`)

```
pbinom(3, size=10, prob=0.5)
```

Más ejemplos:

- $P(X \geq 4)$
  `sum(dbinom(4:10, 10, 0.5))` y también `1 - pbinom(3, 10, 0.5)`
- $P(3 < X \leq 8)$
  `sum(dbinom(4:8, 10, 0.5))` y también `pbinom(8, 10, 0.5) - pbinom(3, 10, 0.5)`

Para calcular los cuantiles utilizamos el prefijo `q` de la siguiente forma:

```
qbinom(c(0.25,0.75), size=10, prob=0.5),
```

obteniendo el **cuantil 0.25** y el **cuantil 0.75**, es decir el número de éxitos necesarios en una $B(10, 0.5)$ que garantiza una probabilidad acumulada del 25 % y 75 %, respectivamente.

4.2.1. Unos primeros cálculos

Comenzamos realizando unos primeros cálculos rápidos relativos a distribuciones discretas. Calcula:

- $X \in B(10, 0.7)$, $P(X = 0)$
- $X \in B(10, 0.7)$, $P(X \leq 3)$ y $P(X < 3)$
- $X \in B(10, 0.7)$, $P(X > 2)$ y $P(X \geq 2)$
- $X \in B(100, 0.05)$, los cuartiles (cuantiles 0.25, 0.5 y 0.75)
- $X \in BN(10, 0.8)$, $P(1 < X < 4)$ y $P(1 \leq X \leq 4)$
- $X \in BN(10, 0.8)$, la mediana
- $X \in P(\lambda = 3)$, $P(X = 0)$ y $P(X \geq 3)$

Nota: Observa la diferencia entre: $P(X \leq 3)$ y $P(X < 3)$, $P(X > 2)$ y $P(X \geq 2)$, $P(1 < X < 4)$ y $P(1 \leq X \leq 4)$.

4.2.2. Representaciones gráficas

- Representa gráficamente la función de masa de probabilidad de $X \in B(10, 0.5)$. ¿Qué ocurre gráficamente si el número de pruebas sube a 100?, ¿y si la probabilidad es menor de 0.5?
  Puedes utilizar el siguiente código en R:
  ```
  x<-0:10
  p<-dbinom(x,10,0.5)
  plot(x,p,type='h')
  ```
- Representa gráficamente la función de masa de probabilidad de $X \in P(2)$, indica cuáles son los valores de la variable con probabilidades crecientes, en qué valor alcanza su máximo y observa que desde ese punto las probabilidades son monótonas decrecientes. ¿Qué observas si subimos λ a 10?

4.2.3. Ejercicio

Para unas oposiciones se realiza un test con 20 preguntas. Una determinada persona tiene una probabilidad de 0.8 de contestar bien cada pregunta.

- Calcula la probabilidad de que la primera pregunta que contesta bien sea la tercera que hace.
- Para aprobar el test es necesario contestar 10 preguntas bien. ¿Cuál es la probabilidad de que apruebe al contestar la pregunta número 12?

4.3. Principales distribuciones univariantes continuas

Las distribuciones continuas que estudiamos son: uniforme, exponencial y normal. Cuando trabajamos con variables continuas la aplicación "densidad" (prefijo `d`) nos sirve para las representaciones gráficas, pero no para calcular (directamente) las probabilidades: para ello utilizamos la función de distribución (prefijo `p`). Recuerda que en el caso continuo que nos ocupa en esta sección solo estamos interesados en el cálculo de probabilidades en intervalos y que los extremos (intervalos abiertos, semiabiertos o cerrados) no afectan al resultado. Por ejemplo, si $X \in N(0, 1)$:

- $P(X < 0.3) = P(X \leq 0.3)$
  ```
  pnorm(0.3, 0, 1)
  ```
- $P(X > 1.4) = P(X \geq 1.4)$
  ```
  1 - pnorm(1.4, 0, 1)
  ```
- $P(0.3 < X < 1.4)$
  ```
  pnorm(1.4, 0, 1) - pnorm(0.3, 0, 1)
  ```

4.3.1. Unos primeros cálculos

Comenzamos realizando unos primeros cálculos rápidos relativos a distribuciones continuas. Calcula:

- $X \in U(2, 5)$, $P(X < 3.7)$
- $X \in \text{Exp}(\lambda = 3)$, $P(X < 0.5)$, $P(X > 1.5)$ y $P(0.5 < X < 1.5)$
- $X \in \text{Exp}(\lambda = 3)$, los cuantiles 0.05 y 0.95
- $X \in N(1.5, 3.2)$, $P(X < -1.8)$, $P(X > 2.7)$ y $P(-1.8 < X < 2.7)$
- $X \in N(1.5, 3.2)$, los cuartiles
- $X \in N(1.5, 3.2)$, el valor a tal que $P(X > a) = 0.3$

4.3.2. Representaciones gráficas

- Representa la función de densidad de una distribución uniforme en el intervalo (-1,2). Observa que es una función constante en ese intervalo, ¿podrías decir qué valor toma esa constante? Para la representación gráfica puedes hacer uso del siguiente código en R:
  `curve(dunif(x,-1,2),xlim=c(-1,2),ylim=c(0,1))`

- Representa gráficamente la función de densidad de una distribución exponencial de $\lambda = 0.5$. ¿Qué observas si subimos λ a 1 y a 2? Puedes hacer uso del siguiente código en R:
  ```
  curve(dexp(x,rate=0.5),xlim=c(0,4),ylim=c(0,2))
  curve(dexp(x,rate=1),col='red', add=T)
  curve(dexp(x,rate=2),col='blue', add=T)
  ```

- Representa gráficamente la función de densidad de una $N(0,1)$. ¿Qué ocurre si aumentamos la varianza a 4? ¿y si disminuye a 0.5? Representa las tres distribuciones en el mismo gráfico (utiliza el código dado en los apartados anteriores).

- Representa gráficamente la función de densidad de una $N(0,1)$. ¿Qué ocurre si aumentamos la media a 2? ¿y si disminuye a -2? Representa las tres distribuciones en el mismo gráfico (utiliza el código dado en los apartados anteriores).

4.3.3. Ejercicio

La envoltura de plástico de un pen-drive está formada por dos láminas, una interior y otra exterior. El grosor de cada una sigue una distribución normal, con media 1 milímetro y desviación típica 0.05 milímetros para la interior, y con media 1.5 milímetros y desviación típica 0.1 milímetros para la exterior. Un pen-drive es considerado como defectuoso si el grosor total de su envoltura es menor que 2.4 milímetros o mayor o igual que 2.55 milímetros. Además, los grosores de cada una de las láminas son independientes.

- Vamos a seleccionar al azar un pen-drive.
  - Representa la función de densidad del grosor total de su envoltura.
  - ¿Cuál es la probabilidad de que resulte defectuoso?
  - ¿Cuál debería ser el grosor total de la envoltura por encima del cual solo quedan el 1 % de los pen-drives fabricados?
- Vamos a seleccionar al azar 10 pen-drives de un gran lote. ¿Cuál es la probabilidad de que resulten exactamente 5 defectuosos?

4.4. Ejercicio

Un canal de transmisión está formado por una fuente que emite dígitos binarios y un receptor. Debido al ruido del canal, hay una probabilidad $p = 0.1$ de que ocurra un error en la transmisión de cada dígito (si se emite 0, recibe 1 y, si se emite 1, se recibe 0). El error se produce con independencia de lo que haya ocurrido en los dígitos emitidos antes.

Responde a las siguientes preguntas utilizando el lenguaje R:

- Representa la función de masa de probabilidad del número de dígitos recibidos con error si emitimos un mensaje de 200 dígitos. ¿De qué distribución se trata?
- Calcula la probabilidad de recibir con error más de 10 dígitos en un mensaje de 200 dígitos.
- Si emitimos dígitos consecutivamente, ¿qué distribución tiene la variable número de dígitos emitidos hasta el primer error? Representa su función de masa de probabilidad.

- Representa la función de densidad $N(20, 4.24)$ y compárala con la gráfica del primer apartado. ¿Podría decirse que la aproximación por esta normal es buena?
- Compara la probabilidad de que la variable $N(20, 4.24)$ tome un valor superior a 10 con el resultado obtenido en el segundo apartado.

4.5. Ejercicio

En una página web se registra una media de 10 visitas por minuto. Calcula:

- La probabilidad de que en un minuto haya más de tres visitas.
- Representa la función de masa de probabilidad de la variable número de visitas en 30 segundos. ¿Qué observas en la gráfica si aumenta el número medio de visitas al doble? ¿Y si disminuye a la mitad?
- La probabilidad de que durante medio minuto no haya ningún acceso a la web.
- Representa la función de densidad de la distribución que mide el tiempo transcurrido entre dos visitas consecutivas a la web. ¿Qué observas en la gráfica si aumenta el número medio de visitas al doble? ¿Y si disminuye a la mitad?
- Calcula el tiempo mediano y el intervalo de tiempo para que la probabilidad de que no haya ningún acceso sea de 0.90.

Práctica 5

Estadística descriptiva básica con R-Commander

5.1. Qué es R-Commander

- Es una Interfaz Gráfica de Usuario (GUI en inglés), creada por John Fox, que permite acceder a muchas capacidades del entorno estadístico R sin que el usuario tenga que conocer el lenguaje de comandos.

- R-Commander se arranca ejecutando en la consola de R `library(Rcmdr)`. En ese momento se abre una ventana propia del entorno R-Commander que nos permite ejecutar operaciones estadísticas que se encuentren implementadas dentro del entorno. La ventana del R-Commander se divide en tres ventanas: *ventana de instrucciones (script)*, en esta ventana se muestra el comando R que ejecuta la tarea que se haya solicitado a través del menú, el script puede guardarse y ejecutarse en cualquier otro momento; *ventana de resultados* y *mensajes*.

- ¿Qué permite hacer R-Commander?
  Análisis estadísticos básicos y gráficos.

- ¿Cómo funciona R-Commander?
  Permite utilizar una interfaz de menús. Las opciones son:

  - *Ficheros*: Abrir ficheros de instrucciones, o bien guardar datos, resultados, sintaxis, etc.

  - *Editar*: Típicas opciones de cortar, pegar, borrar, seleccionar todo, etc.

  - *Datos*: Utilidades para la gestión de datos (crear datos, cargar, importar, recodificar, etc.)

  - *Estadísticos*: Ejecución de procedimientos propiamente estadísticos.

  - *Gráficas*: Representaciones gráficas.

  - *Distribuciones*: Cálculo de probabilidades, cuantiles, generación de datos y gráficos de las distribuciones de probabilidad más habituales.

  - *Herramientas*: Carga de librerías y definición del entorno.

- *Ayuda*: Ayuda sobre el R-Commander.

- Si se cierra R-Commander (sin cerrar R) para volver a cargarlo debe ejecutarse la instrucción `Commander()`

5.2. Etapas en el análisis de datos con un paquete estadístico

Hay tres etapas básicas en un análisis estadístico de un conjunto de datos:

1. Obtención y preparación de los datos (transformando otros si es necesario).
2. Selección del procedimiento necesario y de las opciones deseadas (generalmente a través de los menús del Rcmdr, o también empleando código).
3. Análisis de los resultados obtenidos.

5.3. Lectura de datos desde un fichero externo

Para que R pueda utilizar los datos deben introducirse de modo que cada variable figure en una columna y cada fila represente un caso. Asimismo es conveniente que cada columna esté encabezada con el nombre de la variable. Cuando falte algún dato conviene introducir el valor `NA`, que R interpretará como *Not Assigned* (valor no asignado).

Para leer datos con R-Commander:

- Datos en formato de R, es decir extensión *.RData*
  `Datos > Cargar conjunto de datos`
- Datos en formato texto, excel, SPSS, dBase, etc
  `Datos > Importar datos`

Ejercicio

Consideremos los datos del fichero *top500.RData*, estos datos recogen la información de los 500 ordenadores más rápidos del mundo a diciembre de 2010.

Lee los datos del fichero *top500.RData*, y clasifica las variables en cualitativas o cuantitativas. Puedes acceder a la información de la base de datos y su estructura utilizando las siguientes instrucciones: `as.data.frame(attr(top500, "variable.labels"))` y `str(top500)`.
Nota: Se trata de una base de datos en formato R (extensión .RData).

5.4. Preparación del fichero de datos

El menú `Datos > Modificar variables del conjunto de datos activo > Calcular nueva variable` permite generar nuevas variables a partir de las existentes.

- Especificar el nombre de la varible destino en la ventana `Nombre de la nueva variable`.
- En la ventana `Expresión a calcular` debemos introducir la fórmula de la transformación, se pueden incluir variables a partir de la lista que aparece en `Variables actuales`.

Siguiendo los pasos anteriores vamos a crear una nueva variable LogRMax, que será la transformación a logaritmo neperiano del rendimiento alcanzado por el equipo.

- Introduce el nombre de la nueva variable en la ventana `Nombre de la nueva variable`: `LogRMax`
- En la ventana `Expresión a calcular` introduce la expresión `log(RMax)`

Se puede crear directamente en la ventana de comandos con la instrucción:

```
top500$LogRMax <- log(top500$RMax)
```

Ahora guardaremos la nueva base de datos en un nuevo fichero con el nombre *top500b.RData*. Para ello vamos al menú `Datos > Conjunto de datos activo > Guardar el conjunto de datos activo...` elegimos el directorio y escribimos el nombre del nuevo fichero.

Ejercicio

Crea una nueva variable LogRPeak que será la transformación a logaritmo neperiano del rendimiento teórico del equipo. Guarda la nueva base de datos en el fichero *top500b.RData* que ya contiene el LogRMax.

5.5. Análisis descriptivo de variables cualitativas

El menú `Estadísticos > Resúmenes > Conjunto de datos activo` (comando `summary()`) permite obtener las frecuencias de las variables cualitativas y valores máximos y mínimos, los cuartiles, la media y el número de datos faltantes de las variables cuantitativas del conjunto de datos. A continuación se da algún ejemplo de los resultados que se pueden obtener con este menú

Familia		Potencia	
AMD:	57	Min.:	0.02937
Intel:	398	1st Qu.:	0.17260
Power:	40	Mean :	0.47650
Otros:	5	Median :	0.23310
		3rd Qu.:	0.42449
		Max. :	6.95060
		NA's :	237.00000

Por defecto, Rcmdr solo hace tablas de frecuencias para variables que hayan sido definidas como "factores". Con la instrucción `str(top500)`) puedes comprobar cuáles de las variables de la base de datos están definidas como factores. Además, el Rcmdr permite convertir variables numéricas en factor, para ello se utiliza el menú `Datos > Modificar variables del conjunto de datos activo > Convertir variable numérica en factor...`.

A través del menú `Estadísticos > Resúmenes > Distribución de frecuencias...` se pueden obtener las tablas de frecuencias absolutas y porcentajes de las variables definidas como factores. Los resultados que se generan son de la forma:

(Tabla de frecuencias absolutas)

```
.Table <- table(top500$Familia)
.Table # counts for Familia
```

AMD	Intel	Power	Otros
57	398	40	5

(Tabla de porcentajes)

```
100*.Table/sum(.Table) # percentages for Familia
```

AMD	Intel	Power	Otros
11.4	79.6	8.0	1.0

Es decir, de la familia Intel hay 398 registros, lo que supone un 79.6 % del total de los datos, AMD representa un 11.4 % y Power un 8 %.

Representaciones gráficas

Si se quiere representar gráficamente, el *diagrama de sectores* es una de las representaciones más habituales, y se obtiene a través del menú `Gráficas > Gráfica de sectores...`. Como puede verse la familia Intel es la más frecuente seguida de AMD y Power.

```
pie(table(top500$Familia), labels=levels(top500$Familia), main="Familia",
col=rainbow(length(levels(top500$Familia))))
```

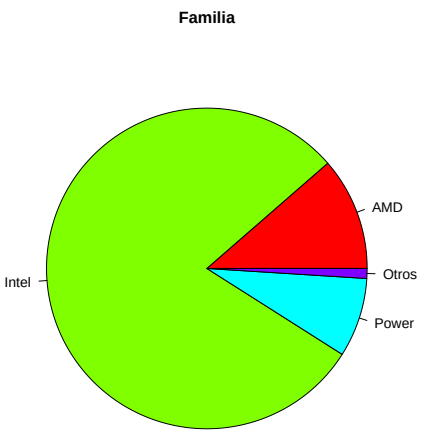


Figura 5.1: Diagrama de sectores de la variable Familia

Otra forma de respresentar estas variables es con el *diagramas de barras*. En Rcmdr se selecciona `Gráficas > Gráfica de barras...`, el código R correspondiente es

```
barplot(table(top500$Familia), xlab="Familia", ylab="Frequency")
```

Las conclusiones obtenidas a partir del diagrama de barras Figura 5.2 son las mismas que a partir del gráfico de sectores Figura 5.1.

Ejercicio

1. Siguiendo con el estudio de la base de datos contenida en el fichero *top500b.RData*, haz un estudio descriptivo de las variables cualitativas `Sistema`, `Arquitectura` y `Procesador`. Interpreta los resultados obtenidos.

2. ¿Podrías repetir el mismo estudio con la variable `Año`? Justifica la respuesta.

3. Convierte la variable numérica `Año` en factor y haz el estudio estadístico de la misma. Para convertir una variable numérica en factor ejecuta los siguientes pasos:

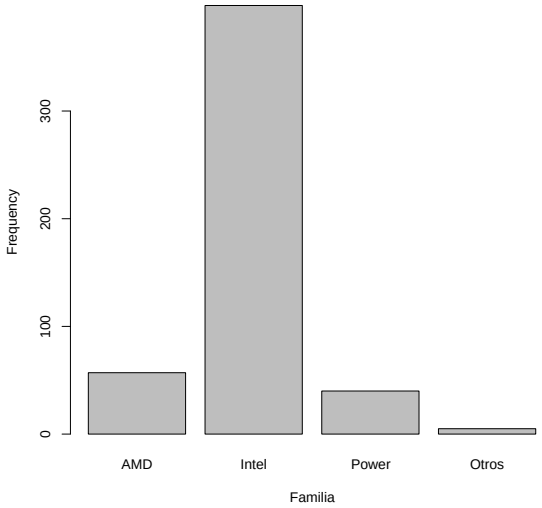


Figura 5.2: Diagrama de barras de la variable Familia

- En el menú de Rcmdr `Datos > Modificar variables del conjunto de datos activo > Convertir variable numérica en factor...` selecciona la variable `Año`
- Marca la opción `Utilizar números` (con esta opción Rcmdr mantiene los valores numéricos como nombres de las categorías de la variable factor)
- Asigna el nombre `AñoR` a la nueva variable obtenida
- Guarda la base de datos en el fichero *top500b.RData*

5.6. Análisis descriptivo de variables continuas

En esta sección se estudia el tratamiento de variables continuas o discretas que toman muchos valores. En particular estudiaremos la variable Frecuencia del procesador de *top500.RData* (varible continua cuyas unidades son GHz). Para el análisis numérico las medidas básicas recomendadas se calculan desde `Estadísticos > Resúmenes > Resúmenes numéricos...` seleccionando las opciones deseadas:

```
numSummary(top500[,"Frecuencia"], statistics=c("mean", "sd", "quantiles"),
quantiles=c(.25,.5,.75))
```

mean	sd	25 %	50 %	75 %	n
2.612666	0.5868782	2.333	2.63	2.93	500

Esta variable no toma valores perdidos (NA), en caso de que los tuviese aparecería una columna al final de la tabla indicando la frecuencia de NA. En particular la frecuencia media del procesador es de 2.613 GHz, con una dispersión de 0.587 GHz, por lo tanto el coeficiente de variación ($\frac{sd}{mean} = 0.225$) es del 22.5 %. Obsérvese que el valor de la desviación típica que aquí se da (`sd`) es en realidad la cuasi-desviación típica. Si nos fijamos en las medidas de posición central, el rango intercuartílico (diferencia entre el tercer y primer cuartil) es de 0.597 GHz, es el rango de valores en el que se encuentra el 50 % central de la distribución de la frecuencia del procesador. La mediana (o segundo cuartil) es 2.63 GHz, es decir si ordenamos los equipos por su frecuencia de menor a mayor hasta 2.63 GHz se encuentra el 50 % de los equipos y por encima de este valor el otro 50 %. También podemos ver que el valor de la mediana está bastante próximo a la media.

Nota: El Rcmdr obtiene los cuantiles a partir de la instrucción `quantile(x, c(0.25,0.50,0.75)))` con el valor `type` que toma por defecto.

Si lo que queremos es obtener la tabla de frecuencias de una variable continua, tenemos que hacer uso del comando `hist` como vimos en la primera práctica. Tecleando en la consola de R:

```
h<-hist(top500$Frecuencia, breaks="Sturges",plot=F)
```

obtenemos información de la frecuencia del procesador agrupando sus valores en intervalos equidistantes. En `h` se guardan los intervalos en los que se agrupan los valores de la variable, las frecuencias, frecuencias relativas, marcas de clase de los intervalos, si los intervalos son equidistantes, etc. Tecleando `h$breaks` se obtienen los extremos de los intervalos y con `h$counts` se obtienen las frecuencias absolutas de los intervalos. Para ver toda la información teclear `h`.

Representación gráfica

La representación gráfica más usual para este tipo de variables es el histograma. Se puede ejecutar con el menú `Gráficas > Histograma...`, código asociado

```
Hist(top500$Frecuencia, scale="frequency", breaks="Sturges", col="darkgray")).
```

O bien directamente en la consola de R

```
hist(top500$Frecuencia)
```

con las opciones que interese (consultar la ayuda). Notar la diferencia en el nombre de la función que calcula el histograma en Rcmdr (**Hist(...)**) y el nombre de la función de R (**hist(...)**), ambas representan histogramas pero son funciones distintas.

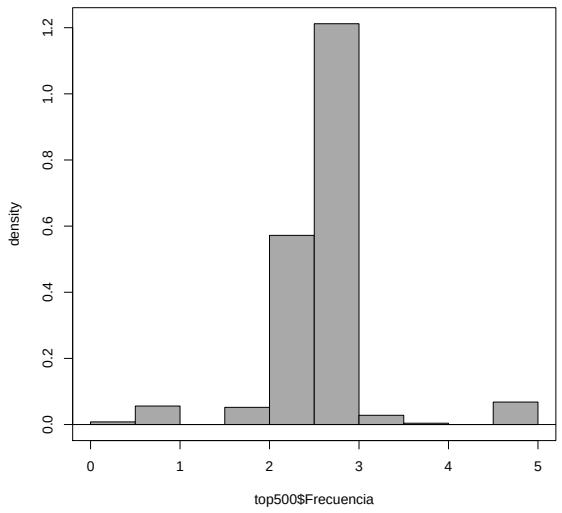


Figura 5.3: Histograma de la frecuencia del procesador

En el gráfico se observa que la mayor frecuencia de datos se concentra en ordenadores con frecuencia de procesador entre 2 y 3 GHz, estando la moda (valor más frecuente) en el intervalo de 2.5 a 3 GHz.

Otra representación gráfica muy utilizada es el diagrama de cajas `Gráficas > Diagrama de caja...`

```
boxplot(top500$Frecuencia, ylab="Frecuencia")
```

Se puede observar en el gráfico la simetría de los datos, la mediana (marcada en el gráfico por la línea negra) divide a la mitad la caja (que va desde el primer al tercer cuartil). Se observa que el 50 % central

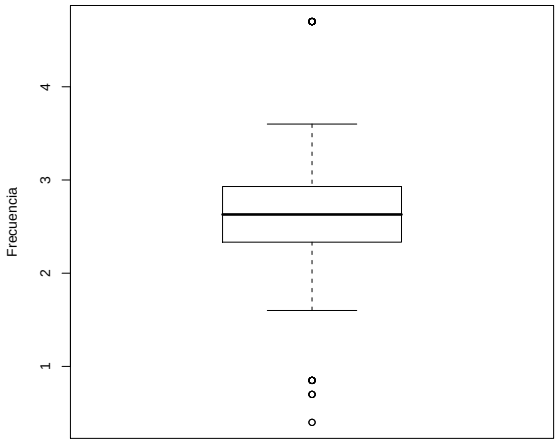


Figura 5.4: Diagrama de cajas de la frecuencia del procesador

de los datos están entre $Q_1 = 2.33$ y $Q_3 = 2.93$. Este gráfico es útil para la identificar valores atípicos, que se repesentan como los valores que caen fuera del intervalo $L1 = \text{mín}\{x_i / x_i \geq Q_1 - 1.5(Q3 - Q1)\}$ y $L2 = \text{máx}\{x_i / x_i \leq Q_3 + 1.5(Q3 - Q1)\}$, los bastones de la gráfica. En este caso tenemos 3 valores atípicos pequeños y uno grande.

Análisis por grupos

En *top500.RData* tenemos alguna variables respecto a las que puede ser interesante estudiar la frecuencia del procesador, por ejemplo la variable factor `Familia`, `Modelo`, `Arquitectura`, etc. Veamos como se comporta la frecuencia del procesador teniendo en cuenta la familia de procesadores. Entonces haremos un análisis por grupo, tanto numérico como gráfico que permita evaluar las diferencias de la frecuencia de procesador en función de la familia de procesadores. Los valores que toma la varible `Familia` son: AMD, Intel, Power, Otros.

Para el análisis numérico, como antes, iremos al menú `Estadísticos > Resúmenes > Resúmenes numéricos...` seleccionando las opciones deseadas y en `Resumir por grupos...` indicar la variable factor por la que se hará el estudio

```
numSummary(top500[,"Frecuencia"], groups=top500$Familia,
statistics=c("mean", "sd", "quantiles"), quantiles=c(.25,.5,.75))
```

Para el análisis gráfico recurriremos al diagrama de cajas, `Gráficas > Diagrama de caja...` con la opción `Gráficas por grupos`.

```
boxplot(Frecuencia~Familia, ylab="Frecuencia", xlab="Familia", data=top500)
boxplot(top500$Frecuencia, ylab="Frecuencia")
```

Se puede observar la diferencia entre las cajas atendiendo a la familia de procesadores, por ejemplo la frecuencia del procesador presenta mucha variabilidad en Power. En AMD e Intel la distribución de los datos está concentrada en valores entre 2 y 3 GHz, presentando en el caso de Intel 2 valores atípicos.

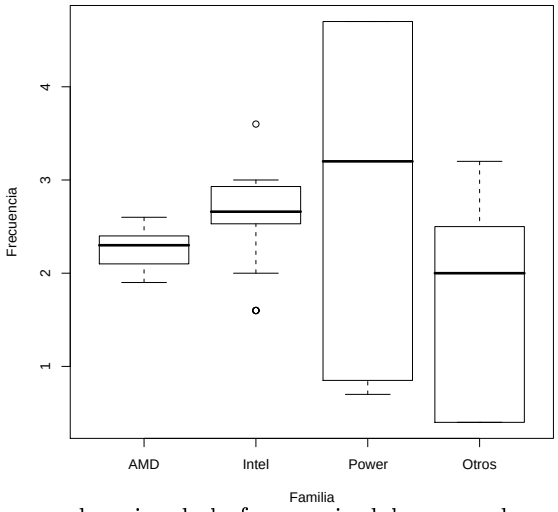


Figura 5.5: Diagrama de cajas de la frecuencia del procesador respecto a la familia.

5.7. Ejercicio

1. Haz un estudio completo (numérica y gráficamente) de las variables rendimiento máximo (`RMax`) y rendimiento teórico (`RPeak`). Compara ambos rendimientos.
2. Compara los rendimientos máximos y teóricos teniendo en cuenta la familia de procesadores.
3. ¿Cómo se comporta la frecuencia del procesador a lo largo de los años 2005 a 2010? Estudia la frecuencia del procesador respecto a la variable Año.

5.8. Ejercicio

1. Cargar el fichero de datos objeto de estudio, *Internet.RData*.
2. Estudia el tipo de variables que hay en el fichero: factor (variables cualitativas), numéricas (variables cuantitativas).
3. De acuerdo a lo visto en esta práctica, haz un estudio estadístico (numérico y gráfico) de las siguientes variables:
   - Sexo (Sexo)
   - Estatura (Estatura)
   - Principal equipo desde el que accedes a internet (Equipo)
   - Años con conexión a internet (AñosInternet)
   - Principal red social en la que estás registrado (RedSocial)
4. Haz un estudio estadístico completo de las siguiente variables, teniendo en cuenta los grupos de otra variable por la que puede ser interesante hacer el estudio:
   - La estatura teniendo en cuenta el sexo (Estatura–Sexo)
   - La edad respecto a la red social en la que estás registrado (Edad–RedSocial)
   - Los años conectado a internet considerando el equipo desde el que accedes (AñosInternet–Equipo).

Práctica 6

Inferencia estadística

6.1. Introducción

La inferencia estadística se utiliza cuando no se puede observar toda la población. Nuestro objetivo es obtener información sobre un parámetro de la población (media, varianza, proporción, etc.) a partir de una muestra. Para ello utilizamos distintas técnicas estadísticas: estimación puntual, intervalo de confianza y contraste de hipótesis. Todas ellas se pueden ejecutar en R desde el menú: `Estadísticos > Medias`, `Estadísticos > Proporciones` y `Estadísticos > Varianzas`.

La inferencia paramétrica clásica toma como punto de partida la **hipótesis de normalidad**. Es por ello por lo que lo primero que necesitamos es un procedimiento que nos permita concluir sobre el grado de cumplimiento de esta hipótesis.

Así un proceso de inferencia sigue inicialmente dos pasos, a continuación se muestran para el caso particular del parámetro media:

Paso 1. **Contraste de normalidad**. ¿Proceden nuestros datos de una distribución normal? Si la respuesta es afirmativa se continúa en el Paso 2, en otro caso en el Paso 2*.
`Estadísticos > Resúmenes > Test de normalidad de Shapiro-Wilk`

Paso 2. **Contrastes paramétricos clásicos**. Se elige el contraste adecuado a la información que tenemos.

- Una muestra:
  - Contraste sobre la media.
    `Estadísticos > Medias > Test t para una muestra`
- Dos muestras:
  - Muestras independientes:
    - Igualdad de varianzas.
      `Estadísticos > Varianzas > Test F para dos varianzas`
    - Igualdad de medias: varianzas desconocidas e iguales, varianzas desconocidas y distintas.
      `Estadísticos > Medias > Test t para muestras independientes`
  - Datos pareados: Diferencia de medias.
    `Estadísticos > Medias > Test t para muestras relacionadas`

Paso 2*. **Contrastes no paramétricos**. Alternativa a los métodos clásicos paramétricos.
`Estadísticos > Test no paramétricos`

En esta práctica se trabajará con el fichero de datos "topinferencia.RData" que contiene una muestra de la base de datos "top500.RData" (analizada en la práctica anterior). Se trataría por tanto de llegar a conclusiones (realizar inferencias) sobre la población a partir de la información disponible en la muestra.

6.2. Contrastes de normalidad

Queremos estudiar si la variable `Potencia` sigue una distribución normal. Empezaremos haciendo un estudio gráfico para concluir realizando el contraste de hipótesis

$$\begin{cases} H_0: & \text{La variable } \texttt{Potencia} \text{ sigue una distribución normal} \\ H_1: & \text{La variable } \texttt{Potencia} \text{ NO sigue una distribución normal} \end{cases}$$

Utilizando el conjunto de datos "topinferencia.RData":

1. Generar el histograma de la variable `Potencia` (consumo eléctrico en MW). ¿Parece razonable suponer que la distribución de esta variable es normal? Dibuja sobre el histograma la curva de una distribución normal de media y varianza las que presentan los datos de `Potencia`.

   ```
   Hist(topinferencia$Potencia, scale="density", breaks="Sturges", col="darkgray")
   curve(dnorm(x,mean(topinferencia$Potencia,na.rm=TRUE),sd(topinferencia$Potencia,
   na.rm=TRUE)), add= TRUE)
   ```

2. Utilizando el menú `Gráficas > Gráfica de comparación de cuantiles...`, generar el gráfico que permita comparar los valores observados de `Potencia` con los esperados bajo normalidad.

3. Repetir los dos apartados previos empleando la variable `LogPotencia` (logaritmo del consumo eléctrico).

4. Contrastar la hipótesis de normalidad de estas variables empleando el test de Shapiro-Wilk. Para ello ir al menú `Estadísticos > Resúmenes > Test de normalidad de Shapiro-Wilk...`.

5. Repetir el estudio para las variables `LogRPeak` y `LogRMax`.

6.3. Contrastes paramétricos

En esta sección trabajaremos con los contrastes estadísticos clásicos para estudiar la media de una variable si disponemos de una muestra y para comparar dos medias (de dos variables) si disponemos de dos muestras. Estos métodos exigen que las variables con las que estemos trabajando se comporten conforme a una distribución normal, aunque este requisito se puede relajar mucho si los tamaños muestrales son grandes.

6.3.1. Test t para una muestra

Queremos contrastar como es la media μ (desconocida) de una variable (e.g. `LogPotencia`) en relación a un valor (conocido) μ_0. Los tres posibles contrastes que podemos hacer son:

$$\begin{cases} H_0: \mu = \mu_0 \\ H_1: \mu \neq \mu_0 \end{cases} \qquad \begin{cases} H_0: \mu = \mu_0 \\ H_1: \mu < \mu_0 \end{cases} \qquad \begin{cases} H_0: \mu = \mu_0 \\ H_1: \mu > \mu_0 \end{cases}$$

Empleando el conjunto de datos "topinferencia.RData":

1. Obtener una estimación puntual y por intervalo de confianza de la media de la variable `LogPotencia`. Para ello emplear el menú `Estadísticos > Medias > Test t para una muestra` (dejar el resto de opciones con los valores por defecto).

2. Para contrastar la hipótesis de que la potencia media es 0.2 MW lo que debemos es contrastar que la media de `LogPotencia` es -1.61[1].

3. ¿Podemos afirmar que la media de `LogPotencia` es (significativamente) superior a -1.61?

4. Repetir el estudio sustituyendo `Potencia` por `RPeak` y considerando 0.2 PetaFLOPS como valor de referencia.

6.3.2. Test t para dos muestras independientes

Queremos contrastar la hipótesis de que son iguales los rendimientos (teóricos) medios de los sistemas con arquitectura Cluster y con arquitectura MPP. Debido a que no es razonable suponer que la distribución de la variable `RPeak` es normal, en su lugar vamos a utilizar la variable `LogRPeak`. Observemos que disponemos de dos muestras, cada una correspondiente a una arquitectura, y que ambas muestras son independientes entre si. Así, podemos hablar de μ_X para referirnos a la media de `LogRPeak` de los sistemas con arquitectura Cluster y de μ_Y para la media de `LogRPeak` de los sistemas con arquitectura MPP. Los contrastes que podemos hacer son:

$$\begin{cases} H_0 : \mu_X = \mu_Y \\ H_1 : \mu_X \neq \mu_Y \end{cases} \qquad \begin{cases} H_0 : \mu_X = \mu_Y \\ H_1 : \mu_X < \mu_Y \end{cases} \qquad \begin{cases} H_0 : \mu_X = \mu_Y \\ H_1 : \mu_X > \mu_Y \end{cases}$$

Cuando utilicemos el menú `Estadísticos > Medias > Test t para muestras independientes` comprobaremos que el método nos va a preguntar si las varianzas de las dos variables son iguales. Para saber la respuesta a esta pregunta debemos realizar, previamente, un contraste de igualdad de varianzas:

$$\begin{cases} H_0 : \sigma_X^2 = \sigma_Y^2 \\ H_1 : \sigma_X^2 \neq \sigma_Y^2 \end{cases}$$

Antes se comentó que la normalidad de las variables no era tan importante para hacer contrastes clásicos sobre las medias si los tamaños muestrales eran grandes. No pasa lo mismo con la varianza: si hay sospechas de que pueda fallar la normalidad hay que renunciar al contraste clásico de igualdad de varianzas `test F para dos varianzas` y utilizar en su lugar un contraste menos dependiente de la hipótesis de normalidad, como por ejemplo el `test de Levène`.

1. Generar un gráfico de cajas comparando `LogRPeak` entre los dos tipos de arquitecturas.

2. Obtener una estimación puntual y por intervalo de confianza del cociente de las varianzas σ_X^2/σ_Y^2 al 99 % utilizando `Estadísticos > Varianzas > Test F para dos varianzas`.

3. ¿Se puede asumir que hay igualdad de varianzas? Utilizar el `test F para dos varianzas` y el `test de Levène`.

4. Utilizando el menú `Estadísticos > Medias > Test t para muestras independientes`, obtener una estimación por intervalo de confianza de la diferencia de medias de `LogRPeak` entre los dos tipos de Arquitecturas.

5. ¿Hay diferencias significativas entre los rendimientos medios? Responder utilizando el p-valor.

6. Repetir el estudio sustituyendo `RPeak` por `Potencia`.

6.3.3. Diferencia de medias con datos pareados

En muchos casos surgen problemas en los que se dispone de dos muestras aleatorias dependientes, donde cada observación de una muestra está emparejada con una observación de la otra. Por ejemplo, dos mediciones sobre el mismo individuo de la población. En estos casos, el procedimiento equivalente

[1] Realmente, el valor que se debe contrastar es $E(\log(X)) = \log(E(X)) - \frac{1}{2}\log\left(1 + \frac{Var(X)}{E(X)^2}\right)$ pero hemos utilizado $E(\log(X)) = \log(E(X))$ por simplicidad.

al anterior es la `prueba t para muestras relacionadas` (las muestras deben estar en el fichero de datos en dos variables de forma que los valores relacionados se correspondan al mismo individuo).

A continuación vamos a comparar los rendimientos medios alcanzados y teóricos utilizando `LogRMax` y `LogRPeak`.

1. Obtener un intervalo de confianza para la diferencia entre las medias de `LogRMax` y `LogRPeak`.
2. ¿Podemos afirmar que el rendimiento medio alcanzado (RMax) es significativamente inferior al rendimiento medio teórico (RPeak)?

6.4. Contrastes no paramétricos

Los métodos que vemos a continuación son una alternativa a los métodos clásicos paramétricos. No exigen que la variable que estamos estudiando se comporte como una distribución normal y son utilizables bajo condiciones muy generales. Son métodos genéricos, no específicos, con las ventajas e inconvenientes que eso conlleva.

1. Contrastar si el rendimiento alcanzado (RMax) es significativamente distinto del rendimiento teórico (RPeak) empleando el contraste no paramétrico: `Estadísticos > Test no paramétricos > Test de Wilcoxon para muestras pareadas` (es una alternativa al test t paramétrico de comparación de medias de muestras pareadas).
2. Contrastar si hay diferencias significativas en el rendimiento medio alcanzado (RMax) entre arquitecturas (Cluster y MPP) empleando el contraste no paramétrico `Estadísticos > Test no paramétricos > Test de Wilcoxon para dos muestras` (alternativa al test t paramétrico de comparación de medias de dos muestras independientes).

6.5. Ejercicio

Con el conjunto de datos `Internet.RData`, asumiendo que las variables Estatura, Peso y RPeso siguen una distribución normal:

1. Obtener una estimación puntual y por intervalo de confianza de la media de la variable Estatura.
2. ¿La estatura media es significativamente superior a 175cm? Responder utilizando el p-valor.
3. Generar un gráfico de cajas comparando el Peso dependiendo de si se dispone de página web (Web). ¿Se puede asumir que hay igualdad de varianzas? Obtener una estimación puntual y por intervalo de confianza de la diferencia de medias de pesos. ¿Hay diferencias significativas entre las medias? (responder con los p-valores).
4. Generar la variable `RPeso` con el peso corporal de referencia de los encuestados según su estatura `Internet$RPeso <- Internet$Estatura-100` (usando el índice Broca de 1871). Obtener un intervalo de confianza para la diferencia de medias entre peso y peso de referencia. ¿Podemos afirmar que el peso medio es significativamente superior al de referencia?
5. Generar el histograma de la variable `Edad`. ¿Parece razonable suponer que la distribución de esta variable es normal? Obtener el gráfico de comparación de cuantiles bajo normalidad y el p-valor del contraste de Shapiro-Wilk.
6. Contrastar si hay diferencias significativas en la edad media dependiendo de si se dispone de página web (Web), empleando el test no paramétrico de Wilcoxon.

Práctica 7

Regresión simple

Un modelo de regresión se utiliza para estudiar el comportamiento de una variable de interés (variable respuesta), Y, a partir de su relación con una o varias variables explicativas (X).

En regresión **simple** se considera **una única variable explicativa**.

En el modelo de regresión **lineal** simple la relación es **lineal** (geometricamente, una recta).

El modelo matemático es:

$$Y = \alpha + \beta X + \varepsilon,$$

α y β son parámetros desconocidos que hay que estimar a partir de una muestra: $\{(X_i, Y_i : i = 1, \ldots, n\}$.

ε es el error aleatorio, se supone que es una variable aleatoria con distribución normal de media cero. Este error aglutina el efecto de (muchas) variables no controladas y que influyen en la respuesta.

Una vez estimado el modelo de regresión (α y β), este modelo ajustado se utiliza para **predecir** el valor de la variable respuesta, Y, asociado a un valor de la variable explicativa, X.

$$\hat{Y} = \hat{\alpha} + \hat{\beta} X$$

7.1. Ajuste, diagnosis e inferencia

En esta práctica se utiliza el análisis de **regresión lineal simple (RLS)** para estudiar y predecir el comportamiento del rendimiento observado (Y = RMax) de los ordenadores incluidos en la lista top500 a partir del conocimiento del rendimiento teórico (X = RPeak).

El conjunto de los pares (RPeak , RMax) es la población (desconocida) de la que se quiere obtener información, en particular, estamos interesados en conocer el comportamiento del rendimiento observado (Y = RMax).

Para hacer este estudio se necesita una muestra de partida. Nosotros dispondremos de una muestra de tamaño $n = 20$ de los pares (RPeak , RMax). El fichero *topregresion.RData* es igual al fichero *top500.RData* excepto en que tiene una columna adicional que se corresponde con la variable *Muestra*. Los ordenadores en los que la variable *Muestra* toma el valor 1 son los datos muestrales, los que se conocen en la práctica. Los restantes serían desconocidos.

1. Representa la población a través de un diagrama de dispersión.

   A través del menú `Gráficas > Diagrama de dispersión` se pueden obtener diagramas de dispersión.

   En primer lugar hacerlo desactivando las opciones. Mejoramos nuestra representación si la variable *Muestra* se convierte en un factor (categórica) y en el campo `Gráfica por grupos` intro-

ducimos este factor. De esta forma distinguimos los datos muestrales (conocidos) del resto de la población.

Alternativa, se obtiene una gráfica similar en el menú `Gráficas > Gráfica XY`.

2. Representamos graficamente la muestra de 20 puntos, en los que la variable *Muestra* toma el valor 1.

   A través del campo `Expresión de selección` podemos seleccionar un subconjunto de datos.

   a) Representa la muestra a través de un diagrama de dispersión.
   En nuestro caso la expresión a introducir es: **Muestra==1**.

   b) ¿Se intuye la presencia de relación lineal entre ambas variables?
   Activar las opción: `Líneas suavizadas`.

3. Ajusta una **recta de regresión** a la muestra.

   a) ¿Cuáles son las estimaciones de la ordenada en el origen (α) y la pendiente de la recta (β)?
   A través del menú
   `Estadísticos > Ajuste de modelos > Regresión lineal`
   se pueden obtener estimaciones asociadas a un modelo de regresión lineal. (No olvidarse de la `Expresión de selección`)

   b) ¿Cuál es la estimación del coeficiente de correlación lineal?
   Puede obtenerse a través del menú del apartado anterior, sin más que tener en cuenta que tiene el mismo signo que la estimación de β, y es la raíz cuadrada de *Multiple R-squared*.

   c) Representa la recta ajustada junto con el diagrama de dispersión de la muestra.
   Puede obtenerse a través del menú `Gráficas > Diagrama de dispersión`. Opción *Línea de mínimos cuadrados*.

4. ¿Se puede asumir que los errores (ε) del modelo de regresión siguen una distribución normal?

   Solicitando la ayuda presente en el cuadro de diálogo del menú

   `Estadísticos > Ajuste de modelos > Regresión lineal`

   se puede observar el listado de los resultados asociados a la ejecución de la regresión. Escribe en R `>names(RegModel.1)`

   donde `RegModel.1` es el objeto que contiene a la salida de la regresión.

   Entre los resultados están los residuos (*residuals*) del ajuste, que son valores próximos a los errores del modelo. El **test de Shapiro-Wilk** está diseñado para contrastar la normalidad. Por lo tanto, la orden a ejecutar en R es:

   ```
   > shapiro.test(RegModel.1$residuals)
   ```

   También te pueden servir de ayuda los siguientes gráficos de R:

   - > hist(RegModel.1$residuals,freq=F)
   - > qqnorm(RegModel.1$residuals)
     > qqline(RegModel.1$residuals)

5. Construye intervalos de confianza para los parámetros de la recta de regresión: α y β.

   En el menú `Modelos > Intervalos de confianza` se obtienen los IC para α y β.

   La siguiente orden de R calcula la estimación y el Intervalo de Confianza para el coeficiente de correlación lineal (ρ):

   ```
   > cor.test(~RMax+RPeak, alternative="two.sided", method="pearson",
   data=topregresion, subset=Muestra==1)
   ```

6. Contrasta la hipótesis de que el coeficiente de regresión es cero: $H_0 : \beta = 0$

   En el menú `Modelos > Resumir el modelo` se realiza este contraste. **Contraste individual de la t**.

   También proporciona el contraste de la hipótesis $H_0 : \alpha = 0$

   La orden de R del apartado anterior: `cor.test(...)` proporciona el resultado del contraste $H_0 : \rho = 0$ relativo al coeficiente de correlación lineal.

7. **Estimación de la media condicionada,** $E(RMax/(RPeak = 0.14)) = E(Y/(X = 0.14))$.

   Estima la media del rendimiento observado de los ordenadores que tienen un rendimiento teórico de 0.14. Calcula el *intervalo de confianza* para esta estimación.

   La siguiente orden de R proporciona el resultado:

   ```
   > predict.lm(RegModel.1, newdata=data.frame(RPeak=0.14), interval="confidence")
   ```

   por defecto trabaja con un nivel (level) de 0.95.

8. **Predicción para una observación,** $RMax/(RPeak = 0.14)$, esto es, $Y/(X = 0.14)$.

   El ordenador de mi despacho tiene un rendimiento teórico de 0.14. **Predice** el rendimiento observado para este ordenador y calcula un *intervalo de predicción*.

   La siguiente orden de R proporciona el resultado:

   ```
   > predict.lm(RegModel.1, newdata=data.frame(RPeak=0.14), interval="prediction")
   ```

   Igual que la del apartado anterior cambiando en el parámetro `interval` la término *confidence* por *prediction*

9. Ajustar un modelo de regresión lineal que pase por el origen, $\alpha = 0$.

   Al hacer el apartado 6 se obtiene que el parámetro α no es significativamente distinta de 0. Si queremos forzar a que la recta de regresión pase por el origen utilizamos el menú de Rcommander

   `Estadísticos > Ajuste de modelos > Modelo lineal`

   y se incluye un -1 en la fórmula del modelo: `RMax ~ RPeak-1`

10. ¿Es conveniente utilizar un modelo de regresión cuadrática en lugar de un modelo de regresión lineal?

    En el menú

    `Estadísticos > Ajuste de modelos > Modelo lineal`

    se pueden incluir varias variables en la fórmula del modelo; por ejemplo,

    - *RPeak* y su cuadrado: *RPeak + I(RPeak ^2)*
    - ambas variables pero no la ordenada en el origen: *RPeak + I(RPeak ^2) -1*.

    Si el coeficiente estimado correspondiente a la variable *I(RPeak ^2)* es significativamente distinto de cero, puede ser conveniente utilizar un modelo de regresión cuadrática.

7.2. Ejercicio

Construye y analiza un modelo de regresión lineal simple que explique el comportamiento de la variable *Peso* a partir de la variable explicativa *Estatura*.

La muestra a utilizar para este estudio se corresponde con los pesos y estaturas contenidos en el conjunto de datos `Internet.RData`.

El análisis debe resolver cuestiones similares a las realizadas en la práctica actual. Para resolver los apartados 7 y 8, se pide calcular estimaciones y predicciones (tanto puntuales como a través de intervalos) del peso medio y del peso, respectivamente, para un individuo cuya estatura es de 180. Los apartados 9 y 10 no es necesario hacerlos.