UNIVERSIDAD POLITÉCNICA DE MADRID
E.T.S Ingenieros Industriales
Unidad Docente de Estadística

INDUSTRIALES
ETSII | UPM

Estadística con R

Eduardo Caro Huertas
Jaime Carpio Huertas
Jesús Juan Ruiz
Alberto Rodríguez Gallego
Francisco Santos Penido

Versión: Febrero 2012

Contenido

0. INTRODUCCIÓN

0.1. ¿QUÉ ES R?

El programa estadístico R es un entorno de trabajo orientado a resolver problemas de Estadística. Inicialmente desarrollado por Robert Gentleman y Ross Ihaka del Departamento de Estadística de la Universidad de Auckland en 1993, es un programa en continuo proceso de actualización gracias al esfuerzo cooperativo de personas e instituciones académicas relevantes del área de Estadística y de Computación en todo el mundo. Su desarrollo actual es responsabilidad del *R Development Core Team*.

El programa R permite trabajar con una ventana de interacción con usuario, *R-Console*, que ofrece posibilidades para gestionar archivos en disco, guardar resultados, etc. R está desarrollado en un lenguaje de programación que utiliza constantes, variables de diversos tipos (numéricas, lógicas, cualitativas-factor y caracteres), estructuras numéricas complejas (vector, matriz, data.frame, lista), y estructuras lógicas como bucles controlados por los comandos *for*, *if* o *while*. Otra interesante característica de R es su gran capacidad gráfica, que permite generar gráficos muy variados y de extraordinaria calidad y flexibilidad. Finalmente, tiene la capacidad de llamar a otras funciones y de desarrollar nuevas funciones en un lenguaje de programación muy sencillo de manejar.

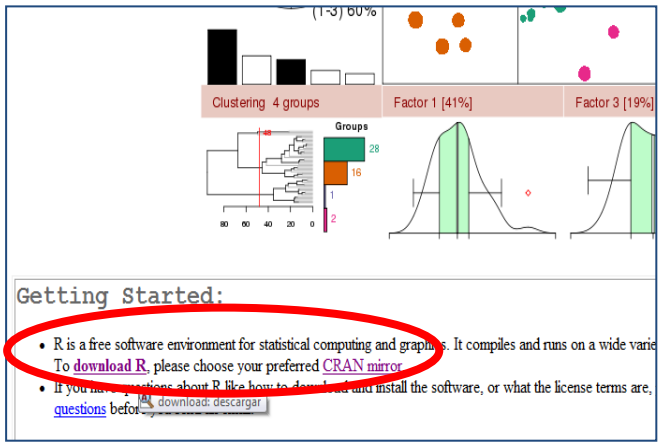
Además del entorno básico, R dispone de un módulo adicional llamado *R-Commander*, que proporciona una serie de menús que facilita el uso inicial del programa.

Como ya se ha dicho, R forma parte de un proyecto colaborativo y abierto. Sus usuarios pueden publicar paquetes que extienden su configuración básica. Existe un repositorio oficial de paquetes cuyo número superó en otoño de 2009 la cifra de los 2000. Dado el enorme número de nuevos paquetes, éstos se han agrupado en temas según su naturaleza y función (http://www.cran.r-project.org/web/views/). Por ejemplo, hay grupos de paquetes relacionados con estadística bayesiana, econometría, series temporales, regresión, diseño de experimentos, fiabilidad, control de calidad, probabilidad, etc.

0.2. INSTALACIÓN DE R

R se descarga desde la página oficial del equipo que desarrolla el programa: www.r-project.org. En esta página pulse en ***download R***.

Este vínculo conduce a una página en la que debe seleccionar su país. Seleccione el único



vínculo asociado a España.

Ahora debe elegir su sistema operativo (**Windows**, Linux o Macintosh). Selecciona el programa "**base**" y le lleva a otra página en la que debe hacer click sobre **Download R** y de nuevo click sobre la versión más reciente que aparezca en este momento.

Así podrá dar comienzo a la descarga del programa que se recomienda que guarde en una carpeta accesible.

Ejecute el archivo que acaba de descargar y:

a) Cuando pregunta sobre los componentes que desea instalar, selecciona instalación completa.

b) Cuando pregunta si desea establecer opciones de configuración, escoja Sí

c) Para el modo de presentación (MDI o SDI), escoja SDI (es conveniente por la implementación actual del módulo *R-Commander*).

d) Seleccione ayuda html.

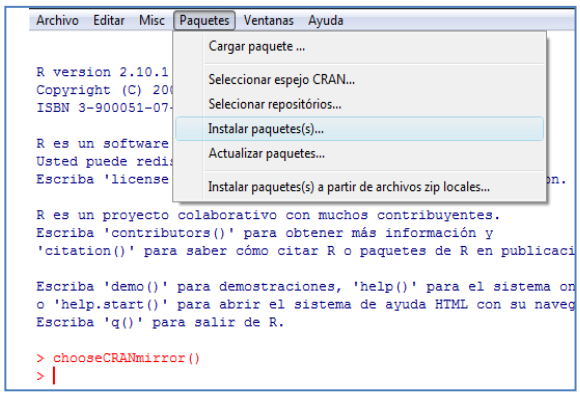
e) Acceso a Internet, elija la opción Standard.

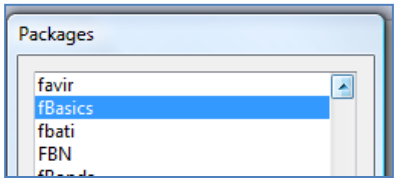
f) Cree un acceso directo en el escritorio.

Una vez instalado, ejecute el programa R pulsando el icono de acceso directo creado en el escritorio. Al ejecutar el programa, aparece la ventana básica *R-Console* donde ya puede empezar a trabajar con R, pero antes de ello vamos a instalar varios paquetes que nos serán de gran ayuda a lo largo de las prácticas de Estadística.

En el menú **Paquetes**, pinche en **Seleccionar espejo CRAN**. Esto le permitirá seleccionar la zona geográfica donde se encuentra para descargar los paquetes informáticos de la zona más cercana. En el cuadro de diálogo que aparece, escoja **Spain (Madrid)**, France (Toulouse), Portugal o algún otro cercano, y pulsa OK.

A continuación descargaremos los paquetes de Internet que nos son útiles. Para ello en el menú **Paquetes**, pincha en **Instalar paquete(s)**. De esta manera, se va a completar la instalación instalando paquetes con programas adicionales al paquete base. Este proceso de Instalar paquetes se hace sólo la primera vez que se instala un paquete.



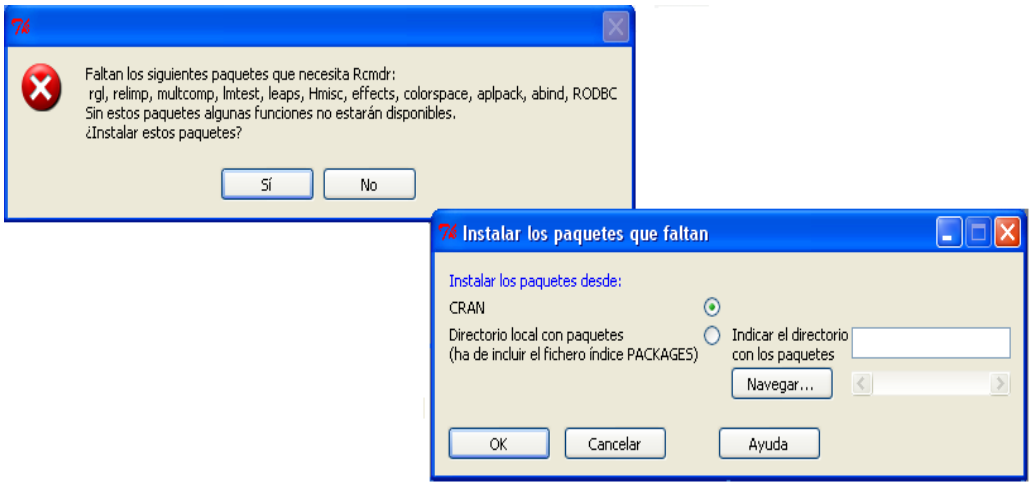


Debe instalar dos paquetes distintos: **fBasics** y **Rcmdr** (puede seleccionarlos juntos con la manteniendo pulsada la tecla Alt Gr). Esta opción descargará de Internet varios archivos comprimidos que se guardarán en el directorio que te indique la *R-Console*.

El paquete **fBasics** contiene funciones para la realización de resúmenes estadísticos básicos, mientras que el paquete **Rcmdr** contiene la interfaz *R-Commander* que permite al usuario una interacción de menús más rica que la de *R-Console*, y también permite trabajar en modo comando.

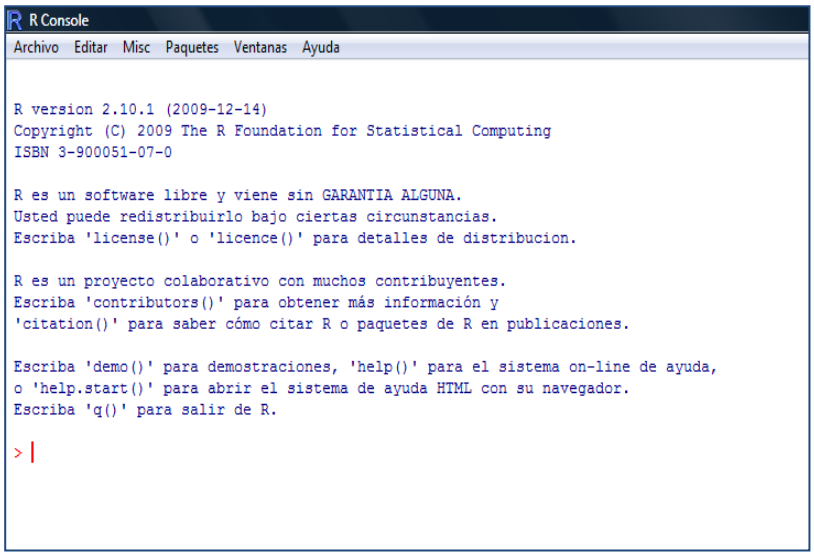
Siempre que se quieran utilizar los paquetes que se han descargado al inicial una sesión en el entorno R se deberán cargar. Para ello, una vez instalados, se escoge en el menú **Paquetes** la opción **Cargar paquete…** y de una lista desplegable se escoge el paquete **fBasics** o **Rcmdr** que se vaya a utilizar. La opción **Cargar paquete…** sirve para permitir que los paquetes, que tengamos instalados, estén disponibles para la ejecución.

La primera vez que cargas el paquete Rcmdr (*R-Commander*), te recomienda instalar paquetes adicionales necesarios para ejecutar de forma satisfactoria *R-Commander*. Pulsar **Sí** y descargar los paquetes que faltan de Intenet utilizando la opción **CRAN**. Esta operación puede tardar varios minutos. En lo sucesivo cuando pulses cargar paquete Rcmdr se cargan todos los paquetes necesarios de forma rápida y automática.



0.3. UNA VISTA RÁPIDA DEL ENTORNO R-CONSOLE Y R-COMMANDER

Al ejecutar R, pulsando el icono de acceso directo creado en el escritorio, se abre por defecto la ventana de *R-Console*. Es un entorno que recuerda mucho a un editor de texto.



La barra de tareas, en la parte superior, contiene los menús clásicos de cualquier programa de Windows a excepción de Misc (no lo utilizaremos) y Paquetes (descrito anteriormente).

Además dispone de una línea de comandos mancada por el símbolo > donde se escriben las instrucciones siguiendo el lenguaje de programación y la sintaxis propia de R que veremos a lo largo de este manual.

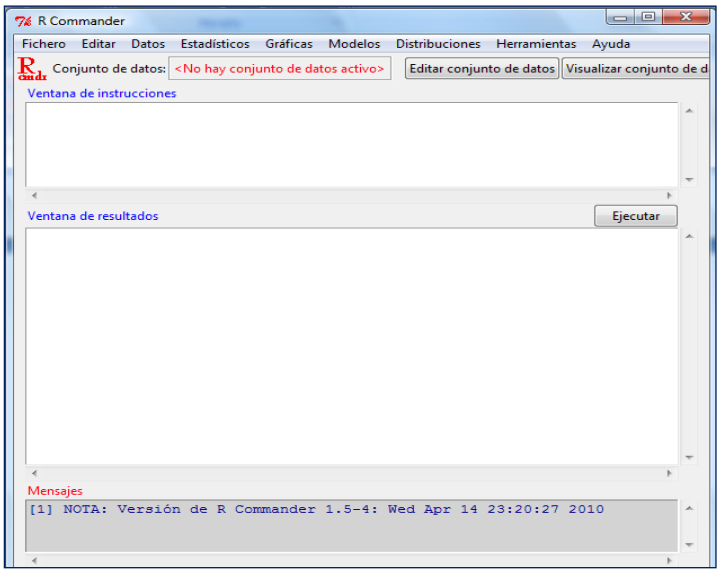
Sin embargo existen diversos interfaces gráficos que facilitan el trabajo en R al usuario novel, ya que así no hace falta memorizar los comandos que estamos utilizando. De todas formas, no se debe confundir interfaz gráfica con la posibilidad de generar gráficos, lo cual tanto *R-Console*, como cualquier interfaz gráfica, hacen de manera más que satisfactoria.

Algunas de estas interfaces son:

- JGR o *Java GUI for R*, una terminal de R multiplataforma basada en Java
- R Commander, una interfaz gráfica multiplataforma basada en tcltk
- RExcel, que permite usar R y Rcmdr desde Microsoft Excel
- rggobi, un interfaz a GGobi para visualización
- Statistical Lab

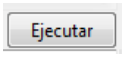
Durante estas prácticas se utilizará como interfaz gráfica *R-Commander*.

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 propio de este entorno. Para poder ejecutar *R-Commander*, hay que cargar el paquete Rcmdr que previamente se ha instalado como se ha indicado antes. Para cargar el paquete Rcmdr se pulsa la opción **Paquete/Cargar Paquete** y se selecciona la opción **Rcmdr**.



Al cargar el paquete Rcmdr aparecerá una ventana correspondiente al entorno de *R-commander*. No obstante, *R-Commander* no pretende ocultar el lenguaje R. Si observamos de cerca la ventana de *R-Commander*, vemos que se divide en tres subventanas: instrucciones, resultados y mensajes.

Cada vez que, a través de los menús de *R-Commander* acceda a las capacidades de R (gráficos, procedimientos estadísticos, modelos, etc.), en la ventana de instrucciones, se mostrará el comando que ejecuta la tarea que hayamos solicitado, y en la ventana de resultados se mostrará el resultado de dicho comando. De este modo, aunque el usuario no conozca el lenguaje de comandos de R, simplemente observando lo que va apareciendo en la ventana de instrucciones se irá familiarizando con dicho lenguaje.

Es más, el usuario puede introducir comandos directamente en dicha ventana, y tras pulsar el botón 

dichos comandos serán ejecutados y su resultado mostrado en la ventana de

resultados. Las instrucciones pueden guardarse y volver a ser ejecutadas directamente con otros conjuntos de datos diferentes.

La ventana de mensajes es de gran utilidad, ya que advierte al usuario de los posibles errores de sintaxis que haya cometido durante la ejecución.

El acceso a las funciones implementadas en *R-Commander* es muy simple y se realiza utilizando el ratón para seleccionar, dentro del menú situado en la primera línea de la ventana, la opción a la que queramos acceder.

Las opciones son:

- **Ficheros**:	Para abrir ficheros con instrucciones a ejecutar, o para guardar datos, resultados, sintaxis, etc
- **Editar**:	Las típicas opciones para cortar, pegar, borrar, etc.
- **Datos**:	Utilidades para la gestión de datos (creación de datos, importación desde otros programas, recodificación de variables, etc.).
- **Estadísticos**:	Ejecución de procedimientos propiamente estadísticos.
- **Graficas**:	Contiene todos los gráficos disponibles.
- **Modelos**:	Definición y uso de modelos específicos para el análisis de datos..
- **Distribuciones**:	Probabilidades, cuantiles y gráficos de las distribuciones de probabilidad más habituales (Normal, t de Student, F de Fisher, binomial, etc.).
- **Herramientas**:	Carga de librerías y definición del entorno.
- **Ayuda**:	Ayuda sobre *R-Commander* (en inglés).

Como conclusión, es importante destacar que aunque se pueda utilizar R a través de la interfaz gráfica *R-Commander*, no es posible aprovechar todo el potencial que ofrece esta herramienta sin tener unos conocimientos previos del lenguaje de programación que utiliza R. Por ello, aprenderemos a utilizar R a partir de las instrucciones en *R-Console*, tratando de entender las posibilidades que nos ofrecen estas instrucciones, para más adelante poder manipularlas en *R-Commander*.

Para más información sobre el entorno de R Console y de R Commander se recomienda utilizar **la ayuda que ofrece el programa** o el uso de páginas web oficiales como:

http://cran.r-project.org/doc/contrib/rdebuts_es.pdf

http://cran.r-project.org/other-docs.html

Además en internet hay un gran número de manuales en español como:

Universidad de las Palmas de Gran Canaria.

http://www.dma.ulpgc.es/profesores/personal/asp/Documentacion/Manual%20R%20commander.pdf

Universidad de Cádiz

http://knuth.uca.es/R/doku.php?id=documentacion

Universidad Carlos III de Madrid

http://ocw.uc3m.es/estadistica/aprendizaje-del-software-estadistico-r-un-entorno-para-simulacion-y-computacion-estadistica/material-de-clase/

O incluso vídeos en youtube donde puedes ver en tiempo real los resultados de las instrucciones:

http://www.youtube.com/user/bebilda

1. PRIMEROS PASOS EN R

1.1. OPERACIONES NUMÉRICAS

R puede usarse como herramienta de cálculo numérico, campo en el que puede ser tan eficaz como otras herramientas específicas en la materia, tales como GNU Octave o MATLAB.

En la ventana *R-Console* cada línea en la que el usuario puede introducir información se inicia con el carácter '>' que pone automáticamente el sistema R. Las instrucciones en una misma línea se separan por ';'. Para ejecutar las instrucciones que están en una línea, se pulsa la tecla *Enter*. Para visualizar en la ventana el valor de una variable se escribe su nombre como instrucción. El operador de asignación puede ser '='o '->' o '<-', aunque es preferible el uso de este último ya que el signo igual tendrá en algunas ocasiones connotaciones lógicas y '->' es un poco más complicado de introducir por teclado.

```
> a=3;
> a<-3;
> 3->a;
> a
[1] 3
>
```

Las instrucciones son escritas en rojo, mientras que los resultados son mostrados en azul. Además es bueno saber que se pueden recuperar líneas de instrucciones introducidas anteriormente pulsando la tecla con la flecha ascendente del teclado, a fin de reejecutarlas o modificarlas. Para abortar la ejecución de una instrucción y devolver el control al usuario, basta pulsar la tecla '*Esc*' del teclado. Así recuperaremos el símbolo '>' para volver a escribir instrucciones.

Las operaciones matemáticas simples son inmediatas y no requieren más que su escritura y ejecución. Para su escritura, se debe tener en cuenta que el lenguaje de R respeta los paréntesis, es decir, hace en primer lugar las operaciones entre paréntesis y luego el resto:

```
> 1+1
[1] 2
> (1+1)*(7+8)
[1] 30
> 1269/31
[1] 40.93548
> 2^4
[1] 16
```

Además también es capaz de hacer operaciones de tipo lógico:

```
> 235<2
[1] FALSE
```

Si se va a utilizar de forma repetida un cierto número, se puede definir como una constante. Para ello se tiene que asociar un nombre a la constante (por ejemplo *k*), utilizar el operador de asignación ('<–') y darle su valor correspondiente (por ejemplo 2). De esta manera el nombre de la constante sustituye a su valor en las futuras operaciones:

```
> k<-2
> 20+k
[1] 22
> k^4
[1] 16
```

Algunos de los operadores y funciones más básicas para el cálculo numérico son los siguientes:

Operadores:

Aritméticos		Comparativos		Lógicos	
Suma	+	igualdad	==	Y lógico	&
Resta	–	Diferente de	!=	No lógico	!
Multiplicación	*	Menor que	<	O lógico	|
División	/	Mayor que	>		
Potencia	^	Menor o igual	<=		
División entera	%/%	Mayor o igual	>=		

Funciones:

Raíz cuadrada	`sqrt(x)`	Seno	`sin(x)`
Exponencial	`exp(x)`	Coseno	`cos(x)`
Logaritmo neperiano	`log(x)`	Tangente	`tan(x)`

1.1.1 *VECTORES Y MATRICES (ARRAY)*

El uso de vectores y matrices es fundamental para poder organizar los datos de una forma adecuada. Por eso, debemos de saber utilizarlos en R y aprender las instrucciones básicas para manipularlos.

1.1.1.1. Vectores

Construcción de vectores: Para construir un vector no hay más que seguir la misma sintaxis que se utiliza para las constantes. En primer lugar, se asigna un nombre (por ejemplo v), seguidamente utilice el operador de asignación y después se pone la letra 'c' (de concatenar) seguida de las componentes del vector entre paréntesis y separadas por comas. Para poder visualizar el vector por pantalla no tiene más que escribir su nombre:

```
> v <- c(1,2,3,4,1,2,3,4)
> v
[1] 1 2 3 4 1 2 3 4
```

Por ejemplo podemos crear un vector x cuyos valores sigan una distribución de probabilidad Normal estándar de 5 componentes con la siguiente instrucción (veremos más detenidamente como generar número aleatorios en el próximo capítulo):

```
> x <- rnorm(5)
> x
[1]  0.7155483 -0.2217395 -1.0279464 -0.8348127  0.2154786
```

También se pueden introducir los datos por teclado con la instrucción `scan()`, por ejemplo

```
> x <- scan()
1: 5.7 3.2 7.5 5.9 4.4 7.9 8.8
8: 3.6 8.7 9.2 5.6 7.9 4.7 6.9
15: 5.5 6.5 6.0 5.0 9.0 3.5 2.2
22:
Read 21 items
```

Los valores se teclean dejando espacios en blanco, cada vez que se pulsa la tecla ENTER cambia de línea y se puede continuar introduciendo valores. Para terminar se pulsa ENTER en una línea vacía. El resultado grabado en x es

```
> x
 [1] 5.7 3.2 7.5 5.9 4.4 7.9 8.8 3.6 8.7 9.2 5.6 7.9 4.7 6.9 5.5 6.5 6.0
5.0 9.0
[20] 3.5 2.2
>
```

Sin embargo, lo habitual será extraer los datos de archivos previamente creados. Esto se verá en el epígrafe referido a cargar archivos.

Si se quiere trabajar con una componente de un vector no hay más que poner el nombre del vector y el número de la componente entre corchetes.

```
> v[5]
[1] 1
> v[6]^4
[1] 16
```

Las operaciones y funciones vistas anteriormente para variables escalares pueden aplicarse a vectores, con la salvedad que las operaciones se harán a cada componente del vector.

```
> v<-1:5
[1] 1 2 3 4 5
> v+2
[1] 3 4 5 6 7
> v^2
[1] 1  4  9 16 25
> sqrt(v)
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
> v==3
[1] FALSE FALSE  TRUE FALSE FALSE
```

Para crear un vector como secuencia de valores equidistantes se puede utilizar la función `seq()`.

```
> x<-seq(1,5); x
[1] 1 2 3 4 5
> x<-seq(1,2,0.1); x
[1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
```

Para seleccionar ciertos elementos del vector anterior se puede utilizar el siguiente comando

```
> xsel<-x[x>=1.5];
> xsel
[1] 1.5 1.6 1.7 1.8 1.9 2.0
```

Algunas de las **funciones básicas** para usar con vectores son las siguientes:

Nº de elementos de un vector	`length(x)`
Suma los elementos del vector	`sum(x)`
Máximo y Mínimo	`range(x)`
Ordenación	`sort(x)`
Resumen estadístico del vector summary	`summary (x)`

Por ejemplo para ver la longitud del vector v

```
> length(v)
[1] 8
```

O para obtener un resumen estadístico del vector v:

```
> summary(v)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
   1.00    1.75    2.50    2.50    3.25    4.00
```

Es importante conocer como maneja R los datos faltantes. Es una situación muy frecuente en Estadística, que tengamos un conjunto de variables que describen a una serie de individuos, por ejemplo, y que para un individuo concreto no se disponga del valor de una de esas variables. Puede ocurrir esto con varios individuos y en varias variables. R tiene en cuenta esta posibilidad, en estos casos aparece el valor faltante como NA (Non Available, No Accesible).

```
> v1<-c(1,NA,3,5,6,NA);v1
[1]  1 NA  3  5  6 NA
```

Si lo que queremos es que R diga donde hay un valor no accesible, utilizaremos la función "`is.na(vector)`":

```
> is.na(v1)
[1] FALSE  TRUE FALSE FALSE FALSE  TRUE
```

Si lo que queremos es que R efectúe una operación sobre el vector sin tener en cuenta el valor no accesible, utilizaremos dentro del paréntesis de la instrucción que debemos escribir el atributo `na.rm=TRUE` (`rm` significa remove (quitar) y `TRUE` debe ir siempre en mayúsculas porque de lo contrario no lo reconoce)

```
> sum(v1)
[1] NA
> sum(v1,na.rm=TRUE)
[1] 15
```

Nota: Con estos dos ejemplos, el lector podrá distinguir fácilmente lo que es una función de lo que son los atributos. Mientras la función realiza una operación o una instrucción, los atributos se encargan de modificar dichas instrucciones, de tal manera que se puedan tener en cuenta otros factores que la función no tendría por defecto. El aprendizaje de los atributos va a ser clave para poder aprovechar el lenguaje de programación; además serán de gran ayuda para el entorno *R-Commander*, ya que este trabaja únicamente con funciones por defecto.

También los vectores pueden almacenar cadenas de caracteres, la sintaxis es similar, lo único hay que poner la cadena de caracteres entre comillas dobles

```
> v<-c("Jesus","Jaime","Javier")
[1] "Jesus"  "Jaime"  "Javier"
```

1.1.1.2. Matrices (`array`)

Empezamos construyendo una matriz de 2 filas y 4 columnas. La instrucción a utilizar es `array`, poniendo entre paréntesis todas las componentes de la matriz por columnas igual que se escribía para definir un vector, es decir `c( , , , )`. Finalmente, es muy importante definir la dimensión; para ello se pone una coma y a continuación el atributo dim=c(filas, columnas).

```
> M <- array(c(2,7,4,9,3,1,2,8),dim=c(2,4))
> M
```

```
     [,1] [,2] [,3] [,4]
[1,]    2    4    3    2
[2,]    7    9    1    8
```

Para acceder a elementos de la matriz se utilizan los corchetes de la siguiente forma:

```
> M[2,3]
[1] 1
> M[1,c(1,2,4)]
[1] 2 4 2
```

Obtener toda la fila 1 y toda la columna 4:

```
> M[1,]
[1] 2 4 3 2
> M[,4]
[1] 2 8
```

Para sustituir valores dentro de la matriz:

```
> M[2,c(2,4)]=c(17,18)
> M
     [,1] [,2] [,3] [,4]
[1,]    2    4    3    2
[2,]    7   17    1   18
```

Algunas funciones básicas para trabajar con matrices:

Nº de filas	`nrow(x)`
Nº de columnas	`ncol(x)`
Dimensión de la matriz	`dim(x)`

Operaciones básicas con matrices (`array`):

R al igual que con vectores permite sumar a todos los elementos de la matriz un mismo número. De la misma manera permite multiplicar una matriz por un escalar. Las instrucciones son muy sencillas:

```
> M<-array(c(2,4,3,2),dim=c(2,2));M
     [,1] [,2]
[1,]    2    3
[2,]    4    2
> M+2
     [,1] [,2]
[1,]    4    5
[2,]    6    4
> M*5
```

```
     [,1] [,2]
[1,]   10   15
[2,]   20   10
```

Nota: obsérvese que R muestra directamente por pantalla el resultado de las operaciones. Sin embargo, cuando se define una variable o cuando se utilizan algunas funciones no muestra el resultado de aplicar esas instrucciones. Por ello se hace necesario escribir el nombre de la variable modificada o creada para observar los cambios.

Suma de Matrices:

Una vez definidas dos matrices que puedan ser sumadas dimensionalmente, la instrucción es básica:

```
> M<-array(c(2,4,3,2),dim=c(2,2))
> N<-array(c(8,6,7,8),dim=c(2,2))
> M+N
     [,1] [,2]
[1,]   10   10
[2,]   10   10
```

Traspuesta de una matriz

Se utiliza la función `t(`*`nombre de la matriz`*`)`. Si se quiere definir una nueva matriz transpuesta no hay más que asignar la función al nuevo nombre (como en otros lenguajes de programación, no hay problema en definirlo con el mismo nombre, pero se recomienda extremar la precaución en dichos casos):

```
> t(M)
     [,1] [,2]
[1,]    2    4
[2,]    3    2
> Nt<-t(N);Nt
     [,1] [,2]
[1,]    8    6
[2,]    7    8
```

Producto de matrices (operador `%*%` para producto):

Para multiplicar dos matrices hay que utilizar el operador `%*%`. Se escribe el nombre de las matrices a multiplicar (dimensionalmente correctas) y entre ellas se escribe dicho operador.

```
> P<-array(c(1,2,2,1),dim=c(2,2))
> Q<-array(c(1,2,3,4),dim=c(2,2))
> P%*%Q
     [,1] [,2]
[1,]    5   11
[2,]    4   10
```

El operador `%*%` también se utiliza para realizar el producto escalar entre vectores, y productos entre matriz y vector.

```
> v1<-c(1,1);
> v2<-c(1,-2);
> v1%*%v2
     [,1]
[1,]   -1
> t(v1)%*%v2
     [,1]
[1,]   -1
> P%*%v1
     [,1]
[1,]    3
[2,]    3
```

Nota: el operador `*` por sí solo, realiza una operación muy curiosa entre matrices de la misma dimensión. Multiplica cada elemento de la matriz con el que se encuentra en la misma posición en la otra matriz. Por ejemplo:

```
> P*Q
     [,1] [,2]
[1,]    1    6
[2,]    4    4
```

Inversión matricial:

Se utiliza la instrucción `solve`*(nombre de la matriz).* Se utiliza `solve` porque esta función también permite resolver sistemas lineales de ecuaciones .

```
> solve(Q)
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
```

1.2. OPERACIONES CON ESTRUCTURAS DE DATOS (`DATA.FRAME`)

La forma más habitual de almacenar datos es hacer uso de tablas (`data.frames` en R). Esto es como una matriz, formada por filas y columnas, con la diferencia que cada columna puede ser una variable de tipo diferente. En una tabla pueden coexistir columnas con información numérica, entera, decimal, otras con información cualitativa de caracteres, otras lógicas, etc. Lo más frecuente es que estas tablas tengan dos dimensiones (filas y columnas), pero en algún caso puede tener más de dos dimensiones.

Para construir una estructura de tipo `data.frame` se utilizará la función `data.frame()`. Un ejemplo ayudará a comprender mejor esta estructura de datos, para ello vamos a construir una tabla que nos ayude a almacenar el nombre y la edad de los alumnos matriculados en la

asignatura de Estadística. Para ello primero creamos dos vectores con la información del nombre y de la edad y luego los unimos con la función `data.frame()`

```
> nombre<-c("Laura", "Pedro", "Juan", "Maria");
> edad<-c(19,20,18,20);
> tabla<-data.frame(nombre,edad);
> tabla
  nombre          edad
1  Laura          19
2  Pedro          20
3  Juan           18
4  Maria          20
```

El nombre que se le asocia a cada columna dentro de la estructura es el nombre que tenga los vectores. Si quisiéramos cambiar el nombre de alguna columna lo deberíamos notificar en los argumentos de la función `data.frame()`

```
> tabla<-data.frame(Alumno=nombre,edad);
> tabla
   Alumno  edad
1  Laura   19
2  Pedro   20
3   Juan   18
4  Maria   20
```

Para referirnos a cada columna de la estructura de datos por separado utilizamos el signo $ entre el nombre del `data.frame` y el de la columna, o ponemos el índice de la columna que queremos mostrar como en matrices `tabla[,i]`:

```
> tabla$Alumno
[1] Laura Pedro Juan  Maria
Levels: Juan Laura Maria Pedro
> tabla$edad
[1] 19 20 18 20
> tabla[,1]
[1] Laura Pedro Juan  Maria
Levels: Juan Laura Maria Pedro
```

Trabajar con `data.frames` es similar a trabajar con matrices, así pues, muchas de las funciones usadas con matrices pueden usarse también aquí.

Búsqueda de componentes y filas o columnas:

```
> tabla[3,2]
[1] 18


> tabla[1,]
```

```
  Alumnos edad
1   Laura   19
```

Se puede emplear la función `dim()`:

```
> dim(tabla)
[1] 4 2
```

`summary()` nos da el resumen estadístico de cada una de sus columnas:

```
> summary(tabla)
  Alumnos       edad
 Juan :1   Min.   :18.00
 Laura:1   1st Qu.:18.75
 Maria:1   Median :19.50
 Pedro:1   Mean   :19.25
           3rd Qu.:20.00
           Max.   :20.00
```

1.2.1 MANIPULAR FICHEROS DE DATOS CON LA R CONSOLE

En primer lugar para leer y escribir archivos de datos con R hay que saber el directorio de trabajo en el que nos encontramos. El comando `getwd()` (get working directory) indica el directorio en el que nos encontramos. Para cambiar el directorio de trabajo, lo más sencillo es utilizar la barra de herramientas que aparece en el borde superior de la ventana de R, elegir **Archivo y** después **Cambiar dir ...** Otra opción es utilizar la función `setwd()`; por ejemplo, `setwd("C:/data")` o `setwd ("/home/paradis/R")`. Para ver qué elementos (archivos y carpetas) hay en el interior del directorio seleccionado se utiliza la función `dir()`.

R puede leer y escribir datos en archivos de texto (ASCII) con las siguientes funciones: `read.table` y `write.table`. También se pueden leer archivos en otros formatos (Excel, SAS, SPSS, . . .), y acceder a bases de datos tipo SQL, pero las funciones necesarias no están incluidas en el paquete base. Aunque esta funcionalidad es muy útil para el usuario avanzado, nos restringiremos a describir las funciones para leer archivos en formato ASCII únicamente.

1.1.1.3. Leer datos: read.table

La función `read.table()` crea una estructura de datos tipo `data.frame`, vista anteriormente, y constituye la manera más usual de leer datos en forma de tabla. En la carpeta que acompaña a este manual tenemos dos ficheros de datos llamados "`Alumnos.txt`" y "`Alumnos_cabecera.txt`" cuyo contenido se muestra a continuación:

Alumnos.txt

Juan	21
Pedro	22
Laura	19
Miguel	20
Maria	18

Alumnos_cabecera.txt

Nombre	Edad
Juan	21
Pedro	22
Laura	19
Miguel	20
Maria	18

Nota: Muy importante, la última línea del fichero debe estar en blanco para que éste pueda ser leído.

```
> misdatos <- read.table("Alumnos.txt")
> misdatos
      V1 V2
1   Juan 21
2  Pedro 22
3  Laura 19
4 Miguel 20
5  Maria 18
```

Creará una tabla de datos denominado `misdatos`, y cada variable recibirá por defecto el nombre V1, V2. Se pueden cambiar los nombres de la cabecera de las columnas utilizando argumentos de entrada en la función read.table.

```
> misdatos <- read.table("Alumnos.txt",col.names=c("Nombre","Edad"))
```

Es usual que el fichero de datos tenga escritas las cabeceras para saber a que datos hace referencia cada columna. El típico ejemplo es el fichero "Alumnos_cabecera.txt". Para indicarle a R que el fichero tiene cabeceras hay que utilizar otro argumento

```
> misdatos <- read.table("Alumnos_cabecera.txt",header=TRUE)
```

Existen más argumentos de entrada que se pueden utilizar con la función `read.table`, para ver esta información basta con escribir `help(read.table)` y te despliega un panel con la ayuda. Brevemente resumimos aquí esa información en castellano:

```
read.table(file, header = FALSE, sep = "", quote = "\"'", dec=".",
row.names, col.names, as.is = FALSE, na.strings = "NA",
colClasses = NA, nrows = -1,skip = 0, check.names=TRUE, fill =
!blank.lines.skip,strip.white = FALSE, blank.lines.skip =
TRUE,comment.char = "#")
```

file	el nombre del archivo (entre "" o como una variable de tipo caracter), posiblemente con su dirección si se encuentra en un directorio diferente al de trabajo (el símbolo \no es permitido y debe reemplazarse con /, inclusive en Windows), o una dirección remota al archivo tipo URL (http://...)
header	una variable lógica (FALSE (falso) o TRUE (verdadero)) indicando si el archivo contiene el nombre de las variables en la primera fila o línea
sep	el separador de campo usado en el archivo; por ejemplo sep="\t" si es una tabulación
quote	los caracteres usados para citar las variables en modo caracter
dec	el caracter usado para representar el punto decimal
row.names	un vector con los nombres de las líneas de tipo caracter o numérico (por defecto: 1, 2, 3, ...)
col.names	un vector con los nombres de las variables (por defecto: V1, V2, V3, ...)
as.is	controla la conversión de variables tipo caracter a factores (si es FALSE) o las mantiene como caracteres (TRUE); as.is puede ser un vector lógico o numérico que especifique las variables que se deben mantener como caracteres
na.strings	el valor con el que se codifican datos ausentes (convertido a NA)
colClasses	un vector de caracteres que proporciona clases para las columnas
nrows	el número máximo de líneas a leer (se ignoran valores negativos)
skip	el número de líneas ignoradas antes de leer los datos
check.names	si es TRUE, chequea que el nombre de las variables sea válido para R
fill	si es TRUE y todas las filas no tienen el mismo número de variables, agrega "blancos"
strip.white	(condicional a sep) si es TRUE, borra espacios extra antes y despues de variables tipo caracter
blank.lines.skip	si es TRUE, ignora líneas en "blanco"
comment.char	un caracter que define comentarios en el archivo de datos; líneas que comienzen con este caracter son ignoradas en la lectura (para desactivar este argumento utilize comment.char ="")

1.1.1.4. Exportar datos: función write.table

La función `write.table()` exporta a un fichero de texto cualquier variable `data.frame` que haya en `R-Console`. Consideremos que existe la tabla `misdatos` del punto anterior y queremos grabarla en un fichero que se llame "`Prueba.txt`"

```
> write.table(misdatos,file="Prueba.txt");
```

El resultado seria que en el directorio de trabajo aparecería el fichero "Prueba.txt" así:

```
"Nombre" "Edad"
"1" "Juan" 21
"2" "Pedro" 22
"3" "Laura" 19
"4" "Miguel" 20
"5" "Maria" 18
```

Incorporando más argumentos de entrada a la función `write.table()` podemos quitarle las comillas, respetar la cabecera y dejarlo en un formato similar a "`Alumnos_cabecera.txt`".

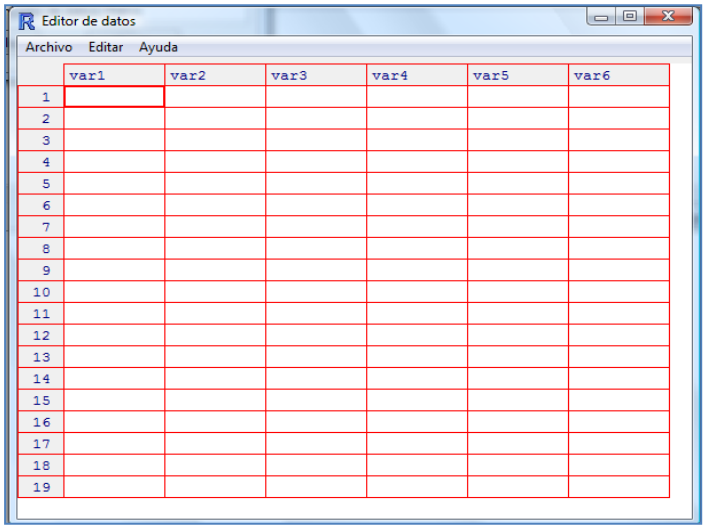
```
> write.table(misdatos,file="Prueba.txt",quote=FALSE,row.names=FALSE);
```

El argumento '`quote`' se refiere a las comillas dobles que se escriben en el fichero y el argumento '`row.name`' hace referencia a la numeración de las filas de la tabla. De todas formas para una descripción detallada de los posibles argumentos de entrada para la función `write.table` utilícese la ayuda `help(write.table)`.

1.2.2 MANIPULAR FICHEROS DE DATOS CON R COMMANDER

Para manipular tablas de datos (importarlas de un fichero, crearlas, editarlas, o exportarlas a ficheros) nos puede ser muy útil la interfaz gráfica *R-Commander*. Para ello cargamos el paquete `Rcmdr` y acto seguido utilizaremos la barra de herramientas de la parte superior.

Para **Crear** una estructura de datos procederemos de la siguiente forma: **Datos/Nuevo conjunto de datos….**



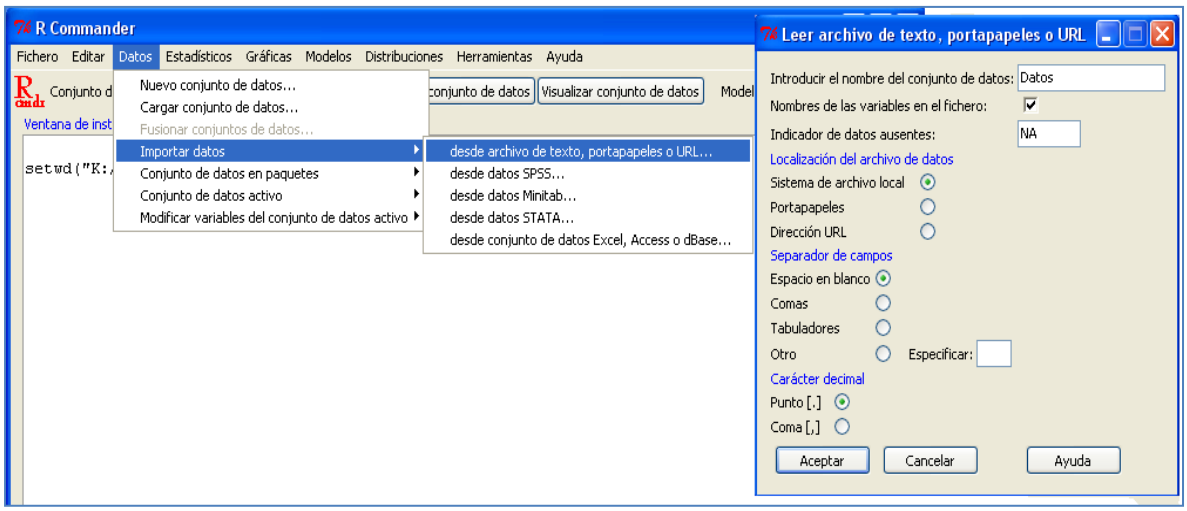
Se da un nombre al nuevo conjunto de datos y se puede empezar a crear la tabla, dando nombre a las cabeceras e introduciendo los valores (numérico o caracteres) en cada una de las casillas de la tabla que ha aparecido:

Pinchando en `var1, var2…,` se puede cambiar el nombre de las cabeceras de cada columna de la tabla. Además podemos indicar si los datos que almacenaremos son datos numéricos o cadena de caracteres.

Para terminar de crear los datos simplemente se cierra la ventana.

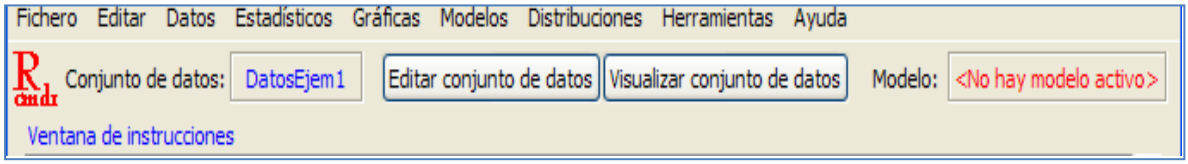
También se puede utilizar el `R-Commander` para **Importar** datos de ficheros de texto creados con diversas aplicaciones: editor de texto puro, hoja electrónica Excel, u otros programas (SPSS, Minitab, Stata. Access) que provienen de varios formatos: desde Excel, Acces,….

Para importar datos de ficheros procedemos **Datos/Importar datos/desde archivo de texto o portapapeles….**

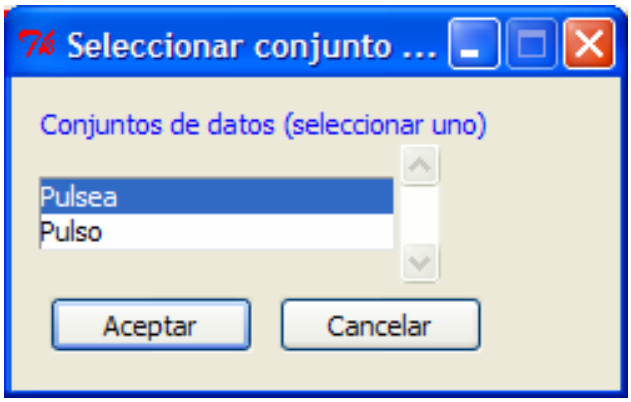


Acto seguido se abre una ventana donde tenemos que introducir el nombre que queremos dar al conjunto de datos que vamos a leer, si el fichero posee cabecera, … Al aceptar, buscaremos con un explorador el fichero de datos que queremos importar.

Para **Editar/Visualizar** un conjunto de datos que hemos importado o creado debajo de del menú principal tenemos botones que nos pueden servir para ello.



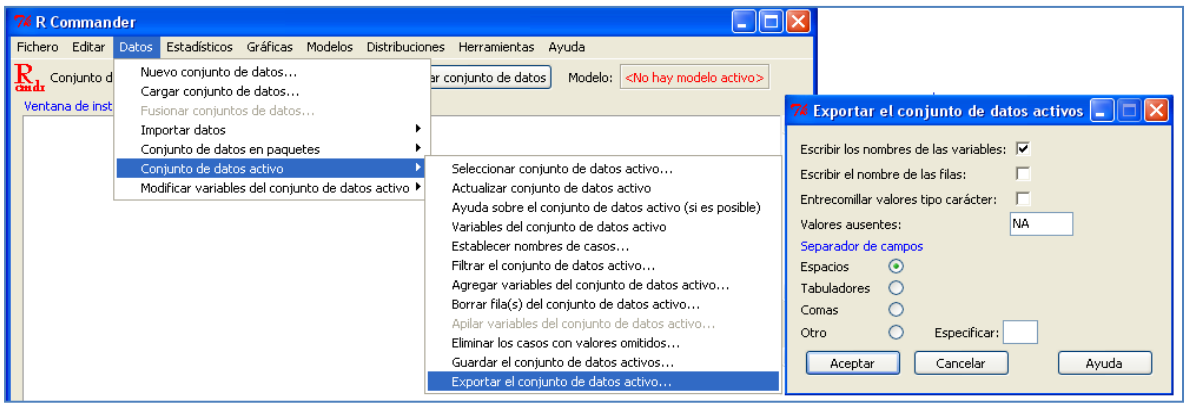
En primer lugar tenemos que activar el **Conjunto de datos** con el que queremos trabajar, por si hubiésemos cargado más de un fichero de datos. Si se pincha sobre el botón próximo, se despliega un menú con el conjunto de datos entre los que se puede seleccionar el activo.



Cada uno de estos conjuntos de datos debe ser del tipo `data.frame`, como una tabla rectangular.

Una vez seleccionados los datos con los que queremos trabajar se pueden Editar o solamente Visualizar pulsando los botones **Editar conjunto de datos** ó **Visualizar conjunto de datos** respectivamente.

Finalmente, si se quiere **Exportar** a fichero una de las variables seleccionadas en Conjunto de datos, se procede de la siguiente forma **Datos/Conjunto de datos archivo/Exportar el conjunto de datos activo…**.



Acto seguido se abre una ventana donde tenemos que introducir las opciones con las que queremos guardar los datos (escribiendo el número de las filas, entrecomillando los caracteres,...). Por ultimo, se debe introducir el nombre dentro de la carpeta donde se quiera guardar el fichero con los datos exportados.

1.3. OPERACIONES DE GESTIÓN DE VARIABLES: `LS( ), RM( ) Y SAVE( ).`

Ya hemos visto como generar variables (escalares, vectores, matrices, `data.frame`) y operaciones sobre ellos. A continuación veremos una serie de instrucciones para gestionar todas las variables en el entorno de trabajo *R-Console*.

Muchas veces cuando la ventana de *R-Console* está llena de instrucciones conviene limpiarla, para ello se puede seleccionar la opción **Limpiar Consola** en el menú **Editar** de `R-Console`, o simplemente en la `R-Console` se pulsa el botón derecho del ratón y elegir **Limpiar pantalla**. O pulsar la siguiente combinación de teclas `Crl+ L`.

Para ver las variables que se están gestionando simultáneamente con R se utiliza el comando `ls()` (lista de objetos).

```
> ls()
[1] "a" "s" "v"
```

Para eliminar variables de trabajo se utiliza la función `rm()` ('remove' en ingles). Se pueden borrar un conjunto de variables o borrar todas las variables de trabajo

```
> rm(a)            //Borra solamente la variable "a".
> rm(a,s)          //Borra las variables "a" y "s".
> vect<-c("a","s"); rm(list=vect);     //Borra las variables "a" y "s".
> rm(list=ls())    //Borra todas las variables.
```

Muchas veces se quiere guardar el entorno de trabajo, es decir, todas ó parte de las variables que se han creado. Se pueden guardar las variables de trabajo con la función `save()`

```
> save(list=c("a","c"),file="Variables.RData")
```

Es conveniente poner la extensión a los ficheros .RData para saber que son datos que únicamente se pueden leer con el programa R y no se pueden abrir como si fueran ficheros de texto normal. Si se quieren guardar todas las variables definidas en el entorno de trabajo se

puede utilizar directamente la instrucción `save.image("Variables.RData")` ó en **Archivo/Guardar área de trabajo**.

Para leer las variables que han sido guardadas en el fichero se puede utilizar el comando `load("Variables.RData")` ó en **Archivo/Cargar área de trabajo**.

1.4. Procedimientos y Funciones en R

Una de las grandes ventajas de R es su facilidad para programar nuevas funciones. Es muy útil cuando se quiere repetir un conjunto de instrucciones para diferentes conjuntos de datos. Una función es un conjunto de instrucciones del lenguaje R escritas en un archivo de texto que se puede abrir, editar y ejecutar utilizando el menú Archivo en *R-Console* y también en *R-Commander*. R reconoce como ejecutables los archivos texto con la extensión '.R'. Las funciones pueden ser guardadas y posteriormente editadas, para actualizar y modificar aquello que se desee.

Podemos distinguir 2 tipos de programas ejecutables, Procedimientos y Funciones:

a) Procedimientos: Son líneas ó sentencias de R que se pueden guardar en un fichero. El resultado de leer estos ficheros de texto es como si se copiaran y pegaran las instrucciones en el entorno de R.

   Ejemplo de procedimiento: "Procedimiento.R"

```
v<-c(1,2);
M<-array(c(1,1,2-1),dim=c(2,2));
a=M%*%v;
```

   Para ejecutar este procedimiento basta con leerlo con la instrucción `source()` y ejecuta de inmediato todas las instrucciones creando en el entorno de trabajo todas las variables que están definidas en el fichero.

```
> source("Ejecutable.R")
> ls()
[1] "a" "M" "v"
```

b) Funciones: R permite definir funciones que luego se pueden utilizar cada vez que se quiera. La estructura general de una función en R es la siguiente:

```
Nombre = function(argumento_1, argumento_2, …)
{
sentencia_1,
sentencia_2, …
}
```

   La función se puede escribir directamente en la consola o guardar en un archivo de texto con extensión .R que luego podemos leer con la instrucción `source()`.

Un ejemplo muy sencillo puede ser la función que calcula el factorial de un número entero, mostrando la función el resultado por pantalla y en una variable. Escribimos directamente en la consola

```
fact = function(n)
{
i=n;
fact=i;
while(i>1){fact=(i-1)*fact; i=i-1
}

cat("------------------------\n")  #cat sirve para visualizar cosas
cat("El factorial del número",n,"es igual a:",fact,"\n")
cat("------------------------\n")

# Otra opción es poner simplemente la instrucción print(fact)
}

> source("fact.R")    #Comprobación de que la función funciona bien
> a=fact(4)
------------------------
El factorial del número 4 es igual a: 24
------------------------
> a
[1] 24
```

Las variables definidas dentro del cuerpo de una función son locales, esto es, desaparecen al terminar la ejecución de la función. Por ejemplo en el ejemplo anterior la variable i desaparece

```
> i
Error: objeto 'i' no encontrado
```

Las funciones son consideradas en el entorno de trabajo como variables

```
> ls()
[1] "fact"
```

También estas funciones se pueden guardar en archivos de texto con la extensión .R para que sean reconocidas como programas. Por ejemplo la función anterior la hemos guardado en un archivo en la carpeta llamada `fact.R`. Para cargarla en el entorno de trabajo basta con ejecutar las siguientes order:

```
> source("fact.R")
> ls()
[1] "fact"
```

```
> fact(3)
[1] 6
```

R permite crear estructuras repetitivas o bucles (loops) para la ejecución condicionada de instrucciones. Entre estas estructuras repetitivas destacamos los bucles `for`, `while` e `if`, la ejecución condicional de sentencias.

1.4.1 *EL BUCLE FOR*

El bucle for se caracteriza por repetir una instrucción o varias de ellas un número determinado de veces con un índice que va variando desde un número inicial al uno final, ambos conocidos. La sintaxis de la instrucción es la siguiente:

```
for (i in vector_de_valores)

{sentencia_1; sentencia_2; ….;sentencia_n;}
```

Un ejemplo de aplicación puede ser el siguiente: Dato un vector v de datos vamos a calcular la varianza de los datos.

```
> v<-c(1,1.3,0.9,1.5,0.5);
> media=mean(v);
> n=length(v);
> varianza=0;
> for(i in 1:n) {varianza=varianza+1/n*(v[i]-media)^2}
> varianza
[1] 0.1184
```

No obstante, los bucles `for` son muy lentos en R y deben ser evitados siempre que se pueda.

1.4.2 *EL BUCLE WHILE*

El bucle `while` se caracteriza por repetir una instrucción ó un conjunto de ellas mientras se cumpla una condición dada por una sentencia lógica. El número de veces que se ejecutará el bucle no se conoce a priori. La sintaxis de la instrucción es la siguiente:

```
while (condicion lógica)

{sentencia_1; sentencia_2; ….;sentencia_n;}
```

Un ejemplo de aplicación puede ser el siguiente: Calcular el factorial n! de un número entero n.

```
> n=6;
> factorial_n=n;
> while(n>1){factorial_n=(n-1)*factorial_n; n=n-1}
> factorial_n
[1] 720
```

1.4.3 *EJECUCIÓN CONDICIONAL: IF*

El comando `if` permite ejecutar una serie de instrucciones si se cumple una condición, por el contrario, si no se cumple esa condición se pueden ejecutar otro conjunto de ellas. En

principio esta sentencia solo se ejecuta una vez a no ser que este metida en el interior de un bucle de los anteriores. La sintaxis de la instrucción es la siguiente:

```
if (condicion lógica)
{sentencia_1; . . ; sentencia_n;}
else
{sentencia_1; . . ; sentencia_n}
```

Un ejemplo de aplicación puede ser el siguiente: Dado un vector v lo dividiremos en 2 uno para almacenar los números pares y otros para los impares. Para ello utilizaremos el operador a**%%**b que da el resto de la división a/b.

```
> v<-c(1,5,2,6,8,7);
> pares<-c();
> impares<-c();
> for (i in 1:length(v))
+ {
+ if (v[i]%%2==0) {pares<-c(pares,v[i])} else { impares<-c(impares,v[i])}
+ }
> pares
[1] 2 6 8
> impares
[1] 1 5 7
```

1.5. GRÁFICOS EN R

Otra gran ventaja de R son sus capacidades gráficas. Los gráficos se pueden exportar en diferentes formatos (pdf, eps, jpg,...). Para ver una selección de gráficos realizados con R puede ejecutarse un programa demostración mediante la siguiente instrucción:

```
> demo("graphics")
```

Ejecutando cualquier instrucción de dibujo se abrirá una ventana donde aparecerán todos los gráficos. Pero con el comando `windows()` se abrirán ventanas para gráficos.

```
> windows()
> windows()  # Crea 2 ventanas para gráficos: Device 2 y Device 3
```

Pueden existir distintas ventanas gráficas abiertas, pero solamente una estará activa. En la parte superior de la ventana te indica el número de la pantalla y si esta activa (ACTIVE) ó inactiva (inactive). Para activar una ventana habría que utilizar el comando `dev.set()`

```
> dev.set(2)  # Activa la ventana 2
```

También se pueden eliminar ventanas de gráficos con la instrucción `dev.off()`

```
> dev.off(3)  # Elimina la ventana 3
```

A continuación vamos a comentar las instrucciones más usuales para crear gráficas, podemos dividir estas en instrucciones de alto nivel y de bajo nivel.

1.5.1 *INSTRUCCIONES DE ALTO NIVEL*

Si en la ventana activa hubiera un gráfico, estas instrucciones lo eliminarían y dibujarían el gráfico nuevo acorde con la instrucción. Entre las más usadas citaremos:

Realizar un histograma	`hist()`
diagrama de cajas	`boxplot()`
hacer un gráfico de puntos	`plot()`
Gráficos 3D	`persp()`
Gráficos de contorno	`contour()`
Varios gráficos de dispersión	`pairs()`

Las instrucciones anteriores las veremos más detenidamente en el capítulo de Estadística Descriptiva. Para ver un ejemplo de las posibilidades gráficas vamos a describir brevemente la función `persp()`: Crea gráficos en 3 dimensiones, para ello necesita los siguientes parámetros, los vectores x e y que indiquen las coordenadas del recinto cuadrado donde quiera representarse la solución y los valores z en una matriz de todos los puntos (x_i, y_j). Para definir z se puede utilizar la función `outer(x,y,f)` que evalua la función f en todos los puntos (x_i, y_j). Para más claridad vease el siguiente ejemplo:

```
> x <- seq(-10,10,length=50)    #Generamos 50 puntos en la dirección x entre -10 y 10
> y = x;
> f = function(x,y){x^2-y^2}
> z = outer(x,y,f);
> persp(x,y,z);
> persp(x, y, z, theta = 30, phi = 30);
```

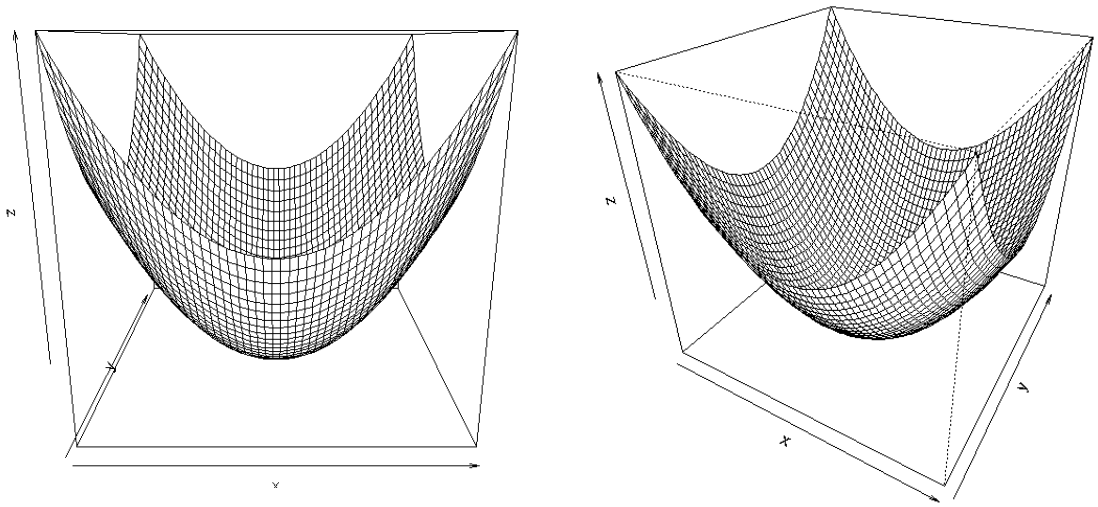


Figura 2.1. Distintas perspectivas de una representación en 3D de la función $f=x^2+y^2$.

Para ver más argumentos de esta función puede usarse la ayuda `help(persp)`.

Función `plot()`: Rutina de gráficos generales en 2D. Los argumentos de entrada son un vector de puntos `x` y otro vector `y`. Veamos un ejemplo:

```
> x<-seq(-10,10)    #Generamos los números -10,-9,…,9,10
> y=x^2;
> plot(x,y)
```

Esta función admite bastantes argumentos, estos pueden ver utilizando la ayuda `help(plot)`. Vamos a destacar los argumentos más importantes

---

`axes=F` Suprime la generación de los ejes

`log="x"` Hace que alguno de los ejes se tome en escala logarítmica
`log="y"`
`log="xy"`

`type="p"` Dibuja puntos individuales (opción por defecto)
`type="l"` Dibuja líneas
`type="b"` Dibuja puntos y líneas
`type="o"` Dibuja puntos atravesados por líneas
`type="h"` Dibuja con líneas verticales
`type="s"` Dibuja a base de funciones escalera
`type="S"` Casi lo mismo
`type="n"` No dibuja nada. Pero deja marcados los puntos para manejos posteriores

---

`xlab="cadena"` Etiqueta para el eje de las x
`ylab="cadena"` Etiqueta para el eje de las y
`main="cadena"` Título del gráfico
`sub="cadena"` Subtítulo del gráfico

`pch="simbolo"` Se dibuja con el simbolo especificado. Por ejemplo:
`pch=18`
`pch="x"`
`pch="P"`

`col="red", "green", "black",…` Color para pintar la curva

A continuación mostramos algunos ejemplos:

```
> x<-seq(-10,10)    #Generamos los números -10,-9,…,9,10
> y=x^2;
> plot(x,y,type="l",xlab="eje de las x", ylab="eje de las y",  main="Parabola")
> plot(x,y,type="h",xlab="eje de las x",ylab="eje de las y", main="Parabola",axes=F)
> plot(x,y,pch=18,col=2,type="b")
```

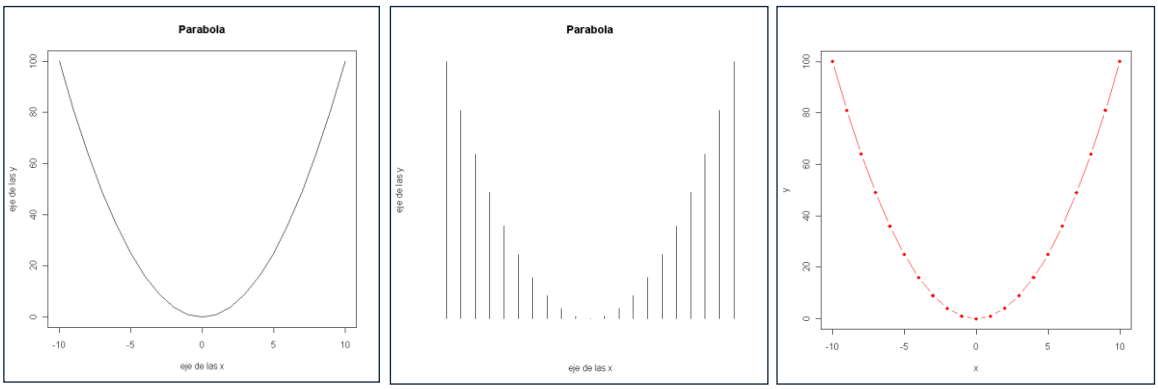


FIGURA 2.2. DISTINTOS DIBUJOS OBTENIDOS PARA LA FUNCIÓN $Y=X^2$.

1.5.2 *INSTRUCCIONES DE BAJO NIVEL*

Los comandos de bajo nivel sirven para añadir información extra a los gráficos que producen los comandos de alto nivel. Por ejemplo, podemos añadir texto a un gráfico, puntos extra, líneas, o cosas parecidas. Las funciones más importantes son las siguientes:

`points(x,y)` Añade puntos o líneas conectadas al gráfico actual
`lines(x,y)`

`text(x,y,etiquetas)` Añade texto al gráfico actual en la posición x,y

`abline(a,b)` Añade una línea de pendiente a y que corta al origen en b
`abline(h=y)` Añade línea horizontal que corta al eje y en h=y
`abline(v=x)` Lo análogo para línea vertical

`polygon(x,y)` Dibuja un polígono

`title(main,sub)` Añade título y subtítulo al gráfico actual

`axis(side)` Añade ejes al gráfico actual (de 1 a 4)

A continuación mostramos algunos ejemplos:

```
> x<-seq(-10,10)    #Generamos los números -10,-9,…,9,10
> y=x^2;
> plot(x,y,axes=F)
> axis(1)

> plot(x,y)
> abline(h=40,col="red");
> abline(v=0,col="blue");
> text(-5,60,"grafico curioso");
```

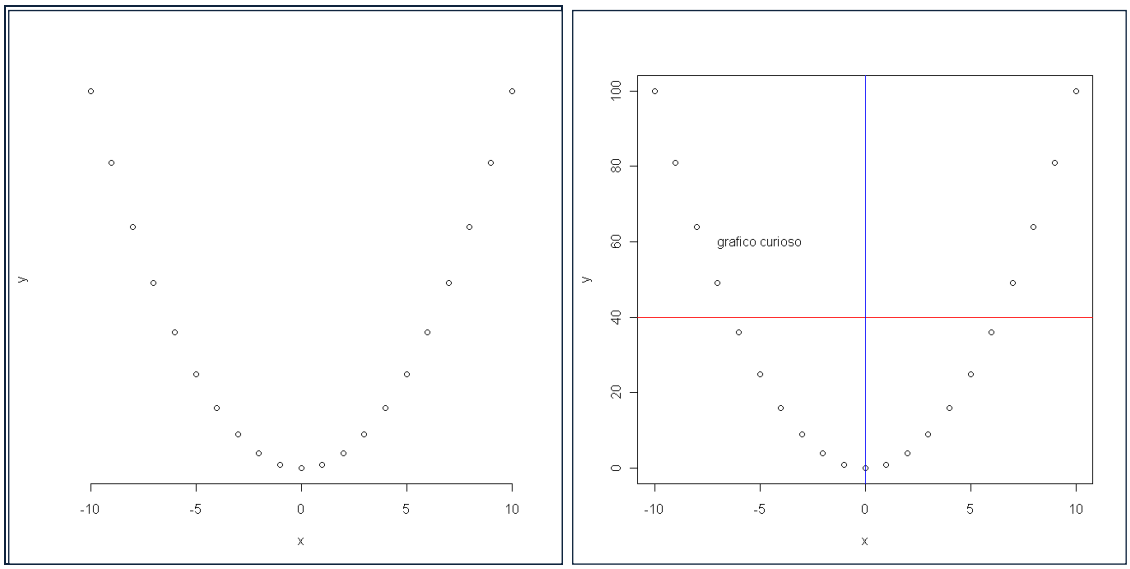


FIGURA 2.3. GRÁFICOS EN 2D OBTENIDOS CON FUNCIONES DE BAJO NIVEL ACOMPAÑANDO A $Y=X^2$.

2. ESTADÍSTICA DESCRIPTIVA

2.1. INTRODUCCIÓN

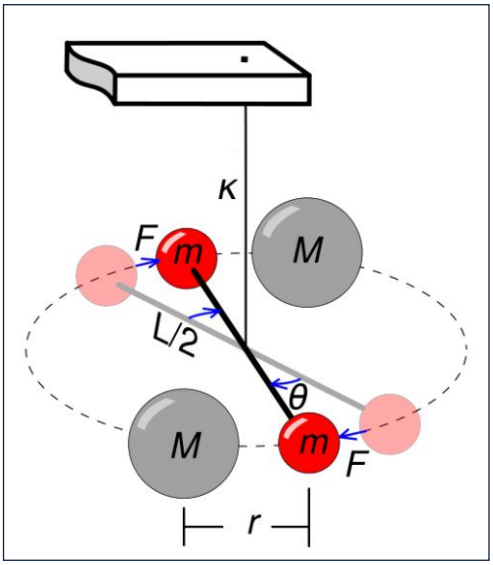
En este capítulo vamos a ver las funciones de R que nos permitirán realizar la estadística descriptiva de un conjunto de datos. Para ir viendo las funciones de una forma más amena y didáctica iremos resolviendo varios ejercicios. Empezaremos con la resolución de un ejercicio de estadística descriptiva de una sola variable, para acto seguido pasar a otro de varias variables. Más adelante en el capítulo se proponen una serie de ejercicios para que el alumno pueda practicar y profundizar en este tema. Por último, se hace un resumen de las instrucciones más interesantes que han aparecido en este capítulo.

2.2. ESTADÍSTICA DESCRIPTIVA DE UNA VARIABLE

Vamos a realizar la estadística descriptiva de una variable. Para ello se van a utilizar los datos de las medidas de la densidad de la Tierra tomados por J. Michell y H. Cavendish almacenados en el fichero **Cavendish.txt**. El enunciado de ejercicio es el siguiente:

En el siglo XVIII John Michell y Henry Cavendish se propusieron pesar el mundo, o lo que es lo mismo, determinar cuál es la densidad de la tierra. Con este fin Michell construyó una balanza de torsión, (similar a la ideada por Coulomb, pocos años antes, para estudiar la atracción entre cargas eléctricas) y Cavendish realizó el experimento.

La balanza de torsión consistía en una barra ligera, que tenía en sus extremos dos esferas pequeñas de plomo, y que fue suspendida de un alambre delgado. La barra podía girar libremente alrededor del alambre cuando una pequeña fuerza era aplicada a las esferas (torciendo el alambre).



Cavendish, después de calibrar el aparato, puso dos grandes bolas de plomo cerca de las pequeñas, una en cada lado. La fuerza de atracción gravitatoria entre las bolas grandes y las pequeñas torció el alambre y, a partir de ese giro, calculó la constante de gravitación. Él conocía la masa de las esferas, la distancia entre sus centros y la fuerza con que se atraían (la que produjo la torsión).

Basándose en la ley de gravitación de Newton, determino cuál era el valor de la constante G. Con el valor de la constante de la gravitación universal G puede determinarse fácilmente la masa de la Tierra, ya que se conoce la masa y el peso (fuerza con que la Tierra lo atrae) de los cuerpos que en ella descansan y se conocía el radio de la Tierra.

En la siguiente tabla se muestran 29 medidas de la densidad de la tierra, expresadas como un múltiplo de la densidad del agua, obtenidas por Henry Cavendish en 1798 empleando la balanza de torsión. Las discrepancias observadas entre los datos son debidas a la falta de precisión en las mediciones y a factores no controlables en la época como son cambios de temperatura, de presión, efectos magnéticos e incluso fenómenos de electrización.

5,50	*5,57*	*5,42*	*5,61*	*5,53*	*5,47*	*4,88*	*5,62*	*5,63*	*4,07*
5,55	*5,34*	*5,30*	*5,36*	*5,79*	*5,75*	*5,29*	*5,10*	*5,86*	*5,58*
5,29	*5,34*	*5,26*	*5,44*	*5,46*	*5,27*	*5,85*	*5,65*	*5,39*	

2.2.1 PARÁMETROS ESTADÍSTICOS

En primer lugar se deben cargar los datos correspondientes a este ejercicio. El nombre del archivo que contiene los datos es **Cavendish.txt**. Recuerda que para cargar los datos es necesario que conozcas la carpeta en la que se encuentran.

Como ya se vio en el capítulo anterior, hay varias maneras para cargar datos en R. Sin embargo, la más cómoda resulta ser a través de la instrucción read.table("…").

El único argumento que se utiliza en esta ocasión es header=T, ya que los datos tienen un título: densidad. Por último el nombre que se le va dar a los datos es Cavendish:

```
>Cavendish<-read.table("Cavendish.txt", header=T)
>Cavendish
   Densidad
1      4,07
2      4,88
3       5,1
4      5,26
      […]
```

Así se obtiene una tabla en la que aparecen los datos asociados a la variable Cavendish. Esta tabla solo contiene una columna que se llama Densidad, así pues para que todo resulte más cómodo se puede definir un vector de trabajo llamado v1 como:

```
> v1<-Cavendish$Densidad
```

Existen una serie de instrucciones que devuelven los parámetros estadísticos de mayor aplicación. Estos se muestran en la siguiente tabla:

Media	`mean(x)`

Mediana	`median(x)`
Quantiles	`quantile(x,p)`
Varianza	`var(x)`
Desviación Típica	`sd(x)`
Resumen estadístico	`summary`

Para hallar la media:

```
>mean(v1)
[1] 5.419655
```

Para hallar la varianza:

```
>var(v1)
[1] 0.1148392
```

Para hallar la desviación típica:

```
>sd(v1)
[1] 0.3388793
```

Para hallar el tercer quantil:

```
>quantile(v1,0.75)
75%
5.61
```

Para hacer un resumen estadístico de los datos:

```
>summary(v1)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   4.07    5.30    5.46    5.42    5.61    5.86
```

Si se quieren hallar el coeficiente de apuntamiento y de asimetría de la distribución de datos es necesario instalar el paquete "moments", de la misma forma que se ha indicado en el Capítulo 1. Una vez instalado se carga el paquete y se pueden utilizar las siguientes funciones:

Momentos de cualquier orden se calculan con la función `moment()`:

```
> moment(v1,order=2,central=TRUE)
Densidad
0.1108792
```

Coeficiente de asimetría se obtiene con la función `skewness()`:

```
> skewness(v1)
Densidad
-2.206814
```

El valor de asimetría (skewness) negativo y grande indica que los datos tienen una distribución asimétrica, con mayor concentración a la izquierda de la media y con posibles valores extremos a la izquierda.

Kurtosis o coeficiente de apuntamiento respecto a la normal (esceso de kurtosis) se obtiene con la funcion `kurtosis()`:

```
> kurtosis(v1)
Densidad
9.89353
```

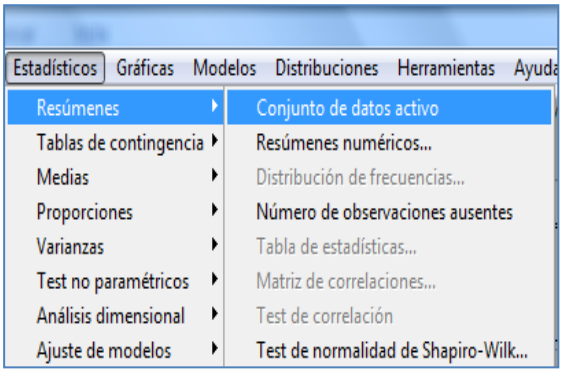
La medida de kurtosis nos da una idea de la forma de la distribución en cuanto a su apuntamiento. La distribución normal tiene un grado de apuntamiento igual a 3. Es una medida difícil de interpretar, pues mide la proporción de datos que hay en el centro de la distribución en relación con la que hay en los extremos. Un apuntamiento mayor que 3 indica que la distribución (estandarizada) tiene más punta que la normal y una distribución con apuntamiento menor que 3, que es menos puntiaguda que la normal. Por ejemplo, la distribución uniforme tiene un apuntamiento de 9/5=1.8, emnor que la normal. El coeficiente de apuntamiento siempre es mayor o igual que 1. En muchos casos, como es el caso de R, el valor de kurtosis proporcionado es el coeficiente de apuntamiento menos 3, es una medida relativa a la normal.

En la siguiente página web viene toda la descripción sobre el paquete moments:

http://cran.r-project.org/web/packages/moments/moments.pdf

Todo lo que se ha hecho hasta ahora, también se puede hacer con *R-Commander*. En primer lugar para cargar el archivo se busca en el menú **Datos/Importar datos** como se vio en el Capitulo 2. Una vez cargados los datos debe aparecer el nombre de los mismos en el cuadro Conjunto de datos en azul.

Para mostrar los parámetros estadísticos más importantes se elige la opción **Estadísticos/Resúmenes**



- Para mostrar un resumen estadístico equivalente a la instrucción **summary** se selecciona **Conjunto de datos activo**

- Si se desea una información más detallada se puede seleccionar **Resúmenes numéricos**

2.2.2 *GRÁFICOS ESTADÍSTICOS DE INTERÉS*

Los gráficos se representan en ventanas nuevas; diferentes de la ventana de instrucciones y resultados.

En primer lugar estudiaremos los histogramas. Si se desea representar el histograma se empleará la función `hist()`, argumentos de la función deben ser la variable que se quiera representar (v1 en este caso), el número de clases (8 en este caso), incluso el color del histograma (en este caso rojo)

```
>hist(v1,breaks=8,freq=T,col="red")   >hist(v1,breaks=8,freq=F,col="red")
```

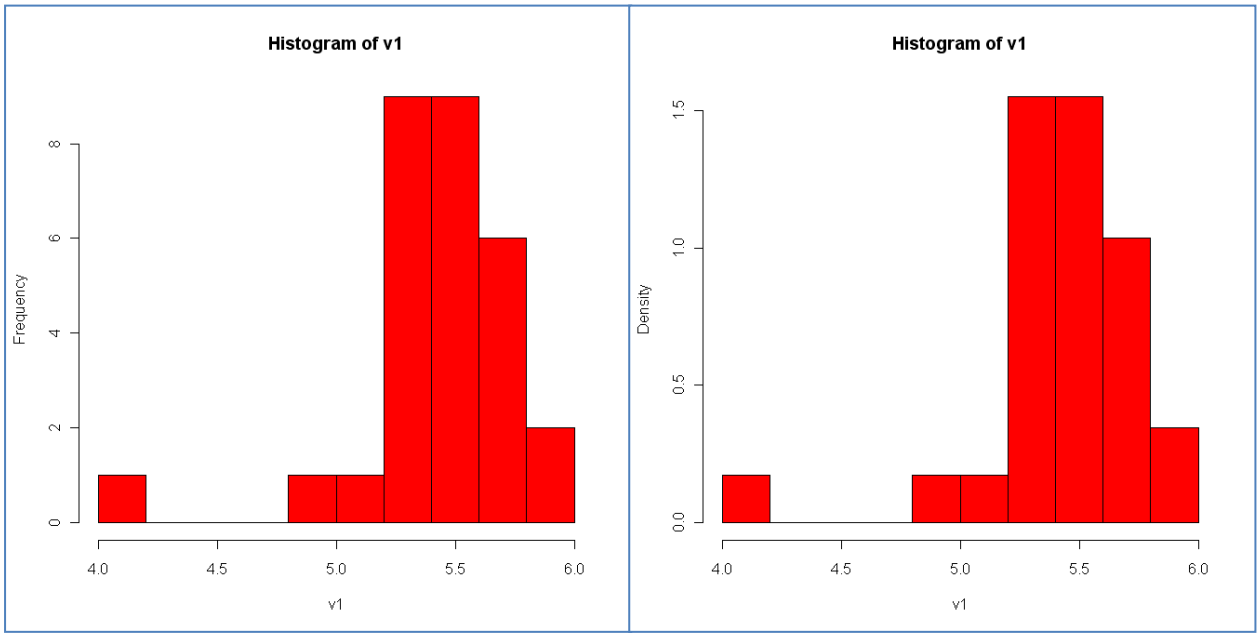


FIGURA 2.1. HISTOGRAMA DE LA VARIABLE DENSIDAD

El argumento `freq` puede tomar 2 valores T (TRUE) ó F (FALSE). En el caso de que tome el valor TRUE (figura 2.1 Izquierda), en el eje y se representará la Frecuencia absoluta de cada clase. Sin embargo, al elegir FALSE (figura 2.1 Derecha) el eje y representará la densidad de frecuencias, es decir, el dibujo se escalará para que la integral o area bajo el histograma sea la unidad. La representación del histograma con esta última opción permitirá relacionar el histograma con la función de densidad de probabilidad de una variable aleatoria.

Los diagramas de cajas o diagramas Box-plot se obtienen con la orden `boxplot(x)`. En este caso vamos a colocar el gráfico en horizontal y de color rojo, ver figura 2.2

```
> boxplot(v1,col="red",horizontal=T)
```

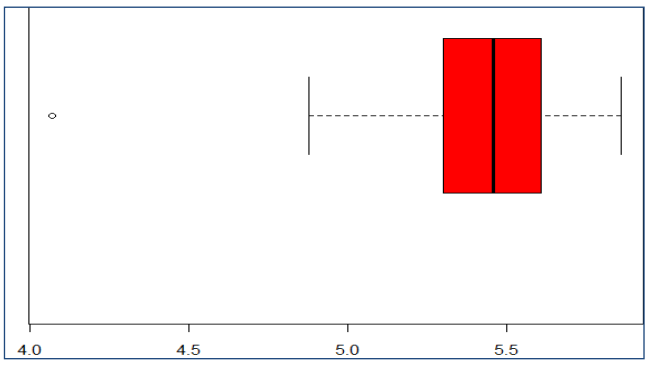


FIGURA 2.2. DIAGRAMA DE CAJAS DE LA VARIABLE DENSIDAD

Se observa que existe un valor atípico, en este caso un valor que es menor que 1,5 veces el rango intercuartílico. Si se considera que este dato es anómalo y es producto de una mala medición se puede eliminar para realizar un mejor análisis de los datos. El dato atípico se puede eliminar utilizando el argumento `outline=FALSE`,

```
> boxplot(v1,col="blue",horizontal=T,outline=FALSE)
```

O como ya conocemos donde está el dato atípico podemos hacer una selección de los elementos de v1 mayores a 4.5.

```
> boxplot(v1[v1>4.5],col="blue",horizontal=T)
```

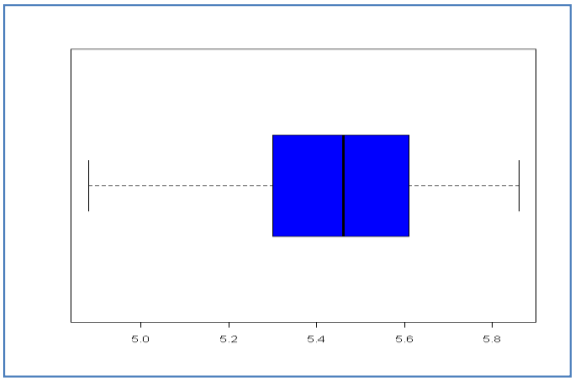
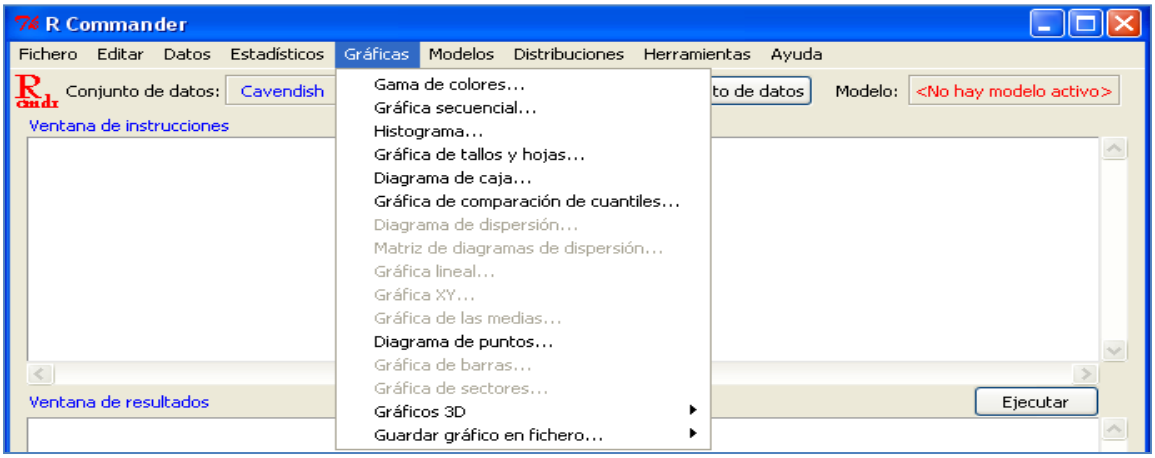


FIGURA 3.3. DIAGRAMA DE CAJAS DE LA VARIABLE DENSIDAD CON EL VALOR ATÍPICO ELIMINADO

Por último, para representar el diagrama de tallos y hojas se usa la función `stem()` del pauqte básico. También se puede utilizar la instrucción más elaborada `stem.leaf()`, pero antes hay que cargar el paquete "aplpack" bien desde la pestaña Paquetes/Cargar paquete o mediante línea de comandos. Utilizaremos este para a la vez enseñaros como cargar paquetes nuevos.

```
> library(aplpack)
> stem.leaf(v1)
1 | 2: represents 0.12
 leaf unit: 0.01
            n: 29
LO: 4.07
   2     48  | 8
```

```
      49  |
      50  |
   3  51  | 10
   7  52  | 6799
  12  53  | 446910
 (4)  54  | 2467
  13  55  | 03578
   8  56  | 1235
   4  57  | 59
   2  58  |

56
```



En el diagrama se indica que hay n=29 datos, que las hojas representan unidades físicas de 0.01, es decir el numero 48|8 representa realmente el dato 4.88. El valor atípico lo representa aparte LO:4.07, para que sea más cómoda la lectura del diagrama. La primera columna del diagrama representa la frecuencia acumulada por arriba y por abajo hasta llegar al dato de la mediana que se encuentra en la fila marcada por un paréntesis (4), 4 en este caso indica el número de datos ó hojas que comparten un mismo tallo.

Con el entorno gráfico *R-Commander* se puede crear los gráficos anteriores de forma muy sencilla. Una vez que los datos están seleccionados se despliega el menú **Gráficas** y se pueden elegir los gráficos que hemos visto y otros adicionales. Se pincha en gráficas, histograma y se abrirá una ventana en la que se puede elegir directamente el número de clases en las que se desea dividir el histograma. Además, se puede elegir la escala de representación del eje.

Una vez que se selecciona el gráfico que se quiere representar se abre otra ventana en la que puedes marcar algunas opciones para mejor representación. Además de todo esto, si se quieren introducir más modificaciones en el gráfico en la línea de comandos del *R-Commander* te aparece la instrucción y se puede modificar introduciendo más argumentos. Se recuerda para ver todas las posibilidades que tiene una función se puede llamar a la función `help(), help(hist), help(box.plot),…`

2.3. DESCRIPTIVA DE VARIAS VARIABLES: CALLES DE MADRID

El siguiente ejercicio esta basado en los datos que tomó un alumno de la titulación de Ingeniero Industrial allá por los años 90. Él quería estudiar la longitud y la anchura de las calles más populares que componen el casco antiguo de Madrid, también conocido como Madrid de los Austrias.

En el fichero `Calles.txt` *encontramos un conjunto de datos que contiene los nombres, longitudes y anchuras (en metros) de 112 calles del casco viejo de Madrid. El objetivo de este ejercicio es hacer un análisis descriptivo multivariante de las dos variables y estudiar la si puede existir relación entre las variables.*

2.3.1 ESTUDIO DE LAS VARIABLES ESTADÍSTICAS POR SEPARADO

La forma de trabajar con datos de varias variables es muy parecida a cómo trabajamos con una sola variable. En primer lugar, se deben cargar el conjunto de datos asociados a este ejercicio.

```
> datos<-read.table("calles.txt", header=T)
```

En este caso como trabajamos con varias variables utilizaremos una nueva instrucción que permite referirnos a cada columna por su nombre. Esta instrucción es `attach()`:

```
> attach(datos)
```

En los datos hay tres columnas de datos: longitud, anchura y nombre. Se puede comprobar la utilidad de la función attach(), pidiendo que muestre la longitud de la calle 45 de la lista:

```
> longitud[45]
[1] 75
```

Para obtener los parámetros estadísticos más relevantes vasta con aplicar la función `summary()` a la tabla de datos:

```
> summary(datos)
   longitud          anchura                 nombre
Min.   :  25.0   Min.   : 4.000   ADUANA         :  1
1st Qu.:  89.5   1st Qu.: 7.375   ALAMO          :  1
Median : 135.0   Median : 9.000   ALFONSO XI     :  1
Mean   : 171.9   Mean   :10.054   ALMIRANTE      :  1
3rd Qu.: 210.0   3rd Qu.:11.250   AMOR DE DIOS   :  1
Max.   :1260.0   Max.   :40.000   ANDRES BORREGO:  1
                                  (Other)        :106
```

Haciendo caso omiso de la columna `nombre` (que utilizamos para identificar la calle), se observa en la variable `longitud` que la media es mayor que la mediana en 40 unidades. Esto indica cierta asimetría en los datos, que debe concordar con un coeficiente de asimetría grande y positivo (`> kurtosis(longitud) [1] 28.00447`).

Se puede ver con un diagrama de cajas la asimetría de los datos, así como la existencia de datos atípicos, para la longitud:

```
> boxplot(longitud, horizontal=T, col="red")
```

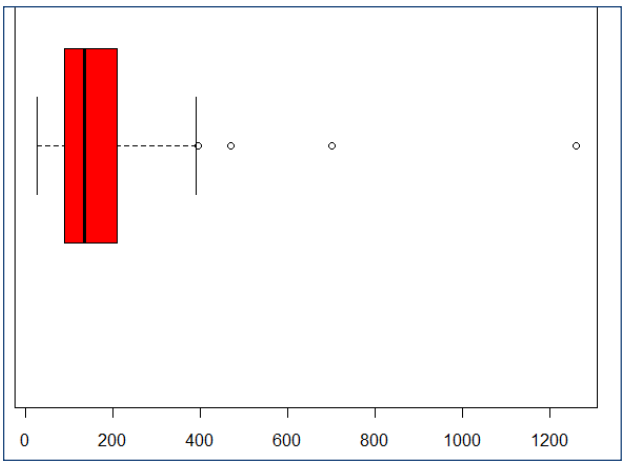


FIGURA 3.4. DIAGRAMA DE CAJAS DE LA VARIABLE LONGITUD

A la vista del gráfico se confirma la asimetría de los datos, con mayor concentración en la parte izquierda que en la derecha y se observa la existencia de 4 datos atípicos. Estos valores atípicos aparecen repartidos por la parte derecha del diagrama.

Con el histograma también se pueden ver estas características de los datos. Se recomienda al alumno su representación gráfica. Como sugerencia, se pueden cambiar el número de clases del histograma, para así observar las variaciones que se producen en éste.

Si se desea obtener una distribución simétrica y con posible desaparición de algunos valores atípicos es recomendable realizar una transformación de los datos:

Las cuatro transformaciones más habituales para resolver este tipo de problemas son: logaritmo neperiano (`log(x)`), raíz cuadrada (`sqrt(x)`), inversa (`1/x`) y cuadrado (`x*x`).

Para trabajar con una variable transformada, lo mejor es definir una nueva variable y trabajar sobre ella. Recuerda que se puede seguir utilizando el mismo nombre para la nueva variable, aunque no es recomendable si va a volver a ser usada. Así:

```
> longitud_inversa<-1/longitud
```

Se puede probar con las 4 transformaciones indicadas y ver cuál de ellas es la más adecuada para obtener unos datos repartidos simétricamente y con menor número de valores atípicos. De estas cuatro transformaciones, la que ofrece mejores propiedades es el logaritmo. Para confirmarlo, se debe repetir todo el análisis que se hizo anteriormente a la variable longitud.

```
> v1<-log(longitud)
> summary(v1)
Min.    1st Qu.  Median   Mean    3rd Qu.   Max.
3.219   4.494    4.905    4.914   5.347     7.139
> boxplot(v1,horizontal=T,col="grey")
```

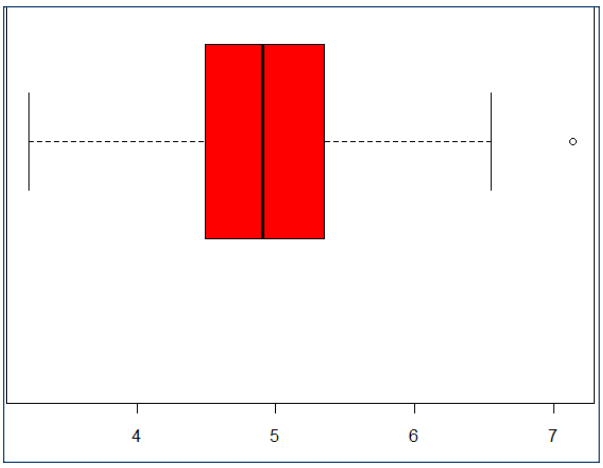


FIGURA 3.5. DIAGRAMA DE CAJAS DEL LOGARITMO DE LONGITUD

A pesar de la transformación, se detecta la presencia de un valor atípico correspondiente a la calle Atocha de 1260 metros. La presencia de este valor atípico se puede explicar acudiendo a razones históricas. Desde 1589, la calle Atocha está ubicada entre la plaza de Santa Cruz y el Paseo del Prado, de tal manera que ejercía la función de unión entre Madrid capital y el Hospital General. Se recuerda que las condiciones higiénicas de aquel entonces no eran las más adecuadas, por lo que los hospitales se solían situar a las afueras de las ciudades para evitar epidemias. Por esta razón, la calle Atocha es particularmente más larga que el resto de calles históricas de Madrid.

2.3.2 *RELACIÓN ENTRE LA VARIABLE LONGITUD Y LA VARIABLE ANCHURA*

Ahora se pretende estudiar si existe o no una relación lineal entre las variables longitud y anchura. Para ello, debemos estudiar el coeficiente de correlación entre la anchura y la longitud. La instrucción que permite obtener dicho coeficiente es `cor(x,y)`:

```
> cor(anchura,longitud)
[1] 0.1998042

> cor(datos[,c("anchura","longitud")])
           anchura  longitud
anchura  1.0000000 0.1998042
longitud 0.1998042 1.0000000

> cor(datos)
           anchura  longitud
anchura  1.0000000 0.1998042
longitud 0.1998042 1.0000000
```

A la vista de este dato podemos afirmar que la correlación entre ambos valores es muy pequeña. Esto quiere decir, que normalmente si una calle aumenta su longitud, su anchura no tiene por qué crecer en la misma medida, lo cual aparentemente es lógico.

Otra manera de observar la relación lineal entre dos puntos es a través de su diagrama de dispersión. La función que permite dibujar diagramas de dispersión es `plot(x,y)` que sirve para representar cualquier pareja de puntos, que en esencia es lo que es el diagrama de dispersión: una representación de todas las parejas de datos. Así:

```
> plot(anchura,longitud)
```

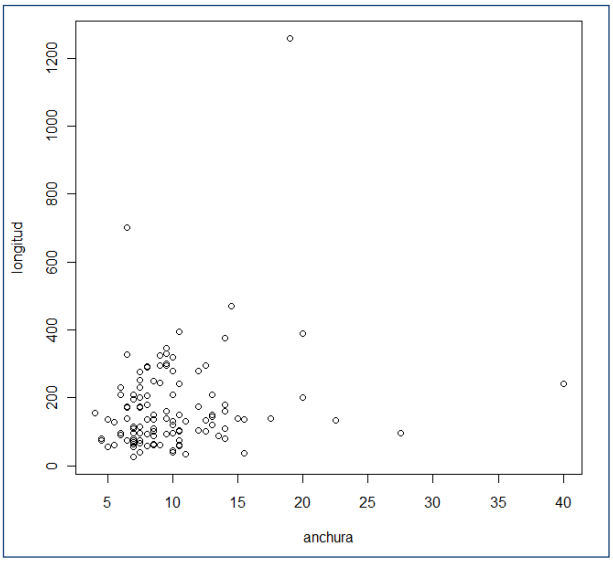


FIGURA 3.6. DIAGRAMA DE DISPERSIÓN LONGITUD-ANCHURA

Se observa en esta representación gráfica que los puntos se encuentran distribuidos sin ningún orden, y no se manifiesta ninguna relación lineal entre las variables.

También se puede pintar el diagrama de cajas conjunto de las variables anchura y longitud para ello basta utilizar la instrucción `boxplot()`. Aquí representaremos el diagrama de cajas de las variables "longitud" y "anchura" transformadas con el logaritmo neperiano.

```
> boxplot(log(Datos[,c("longitud","anchura")]),horizontal=T)
> boxplot(log(Datos[,1:2]),horizontal=T)
```

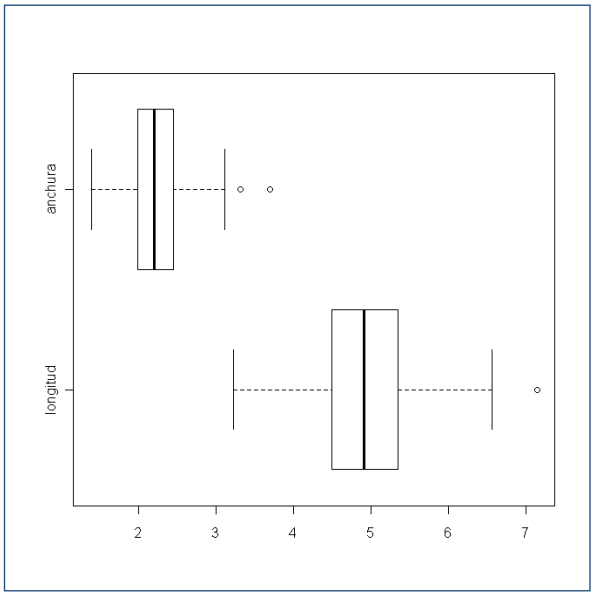


FIGURA 3.7. DIAGRAMA DE CAJAS DE LAS VARIABLES LOG(LONGITUD) Y LOG(ANCHURA)

R-Commander permite obtener fácilmente la matriz de correlaciones. Para ello se pulsa en **Estadísticos/Resúmenes/Matriz de correlaciones...** se deben seleccionar todas las variables que se quieren estudiar y elegir coeficiente de Pearson. Para hacer el diagrama de dispersión basta con pulsar en **Gráficas/Diagramas de dispersión...**

2.4. DESCRIPTIVA DE VARIAS VARIABLES: OLIMPIADAS

Para observar mejor la relación lineal que puede haber entre dos variables, vamos a realizar un nuevo ejercicio con las marcas obtenidas por varios deportistas en la Olimpiada de Seúl 1988, recogidas en el fichero `Olimpiada.txt`

En la Olimpiada de Seúl en 1988 participaron 34 atletas en la competición de decatlón. Las pruebas de la competición son 100 metros lisos (T100), 100 metros vallas (T110), 400 metros lisos (T400), 1500 metros lisos (T1500), lanzamiento de disco (DISCO),lanzamiento de peso (PESO), lanzamiento de jabalina (JABA), salto de longitud (LONG), salto de altura (ALTU) y salto con pértiga (PERT). Se trata de estudiar las correlaciones entre las marcas obtenidas en las diversas pruebas. Téngase en cuenta que estas pruebas se dividen en carreras (en segundos), lanzamientos y saltos (en metros). (archivo olimpiad.sf3)

Se comienza cargando los datos:

```
> datos<-read.table("Olimpiada.txt",header=T)
```

Antes de dibujar las gráficas correspondientes a cada par de variables, es recomendable observar la matriz de correlaciones para ver rápidamente que variables están más relacionadas.

```
> cor(datos)
```

	ALTU	DISCO	JABA	LONG	PERT	PESO	T100	T110	T400	T1500
ALTU	1.000000	0.376004	0.3377261	0.471393	0.471525	0.320513	-0.36374	-0.48684	-0.27549	-0.131551
DISCO	0.376004	1.000000	0.6178202	0.374849	0.619521	0.856184	-0.35260	-0.40345	-0.15439	-0.288190

JABA	0.337726	0.617820	1.0000000	0.446284	0.557130	0.702639	-0.34413	-0.35049	-0.14987	-0.044976
LONG	0.471393	0.374849	0.4462842	1.000000	0.632288	0.390585	-0.69050	-0.65449	-0.63558	-0.355977
PERT	0.471525	0.619521	0.5571306	0.632288	1.000000	0.642531	-0.62723	-0.70875	-0.52113	-0.070246
PESO	0.320513	0.856184	0.7026397	0.390585	0.642531	1.000000	-0.42016	-0.48884	-0.14227	-0.202015
T100	-0.36374	-0.35260	-0.344130	-0.69050	-0.62723	-0.42016	1.000000	0.751268	0.697732	0.2537741
T110	-0.48684	-0.40345	-0.350490	-0.65449	-0.70875	-0.48884	0.751268	1.000000	0.654716	0.1553156
T400	-0.27549	-0.15439	-0.149877	-0.63558	-0.52113	-0.14227	0.697732	0.654716	1.000000	0.5544555
T150	-0.13155	-0.28819	-0.044976	-0.35597	-0.07024	-0.20201	0.253774	0.155315	0.554455	1.0000000

Se observa que las mayores relaciones lineales se encuentran entre disciplinas similares. Por ejemplo, entre los deportes de lanzamiento la relación lineal, en general, es evidente. Lo mismo ocurre para las pruebas de velocidad. Sin embargo, también existen algunos coeficientes de correlación lineal negativos.

Como son muchas las variables que contiene este archivo, se pueden estudiar solo aquel grupo de variables que parecen más interesante. En primer lugar la variables lanzamiento de disco (DISCO), lanzamiento de peso (PESO) y lanzamiento de jabalina (JABA).

```
> cor(datos[,c("DISCO","JABA","PESO")])
           DISCO       JABA      PESO
DISCO  1.0000000  0.6178202 0.8561846
JABA   0.6178202  1.0000000 0.7026397
PESO   0.8561846  0.7026397 1.0000000
```

La más alta relación lineal la encontramos entre el lanzamiento de peso y el lanzamiento de disco, lo cual significa que el atleta que consiga buenas marcas en lanzamiento de disco también conseguirá buenas marcas en lanzamiento de peso.

Podemos verlo gráficamente. Para dibujar ahora todas las variables haremos uso de la función `par(mfrow=c(n,m))`, que permite dibujar varios gráficos en una misma ventana. Para ello escribimos:

```
> par(mfrow=c(3,3))
> plot(DISCO,DISCO)
> plot(DISCO,JABA)
> plot(DISCO,PESO)
> plot(JABA,DISCO)
> plot(JABA,JABA)
> plot(JABA,PESO)
> plot(PESO,DISCO)
> plot(PESO,JABA)
> plot(PESO,PESO)
```

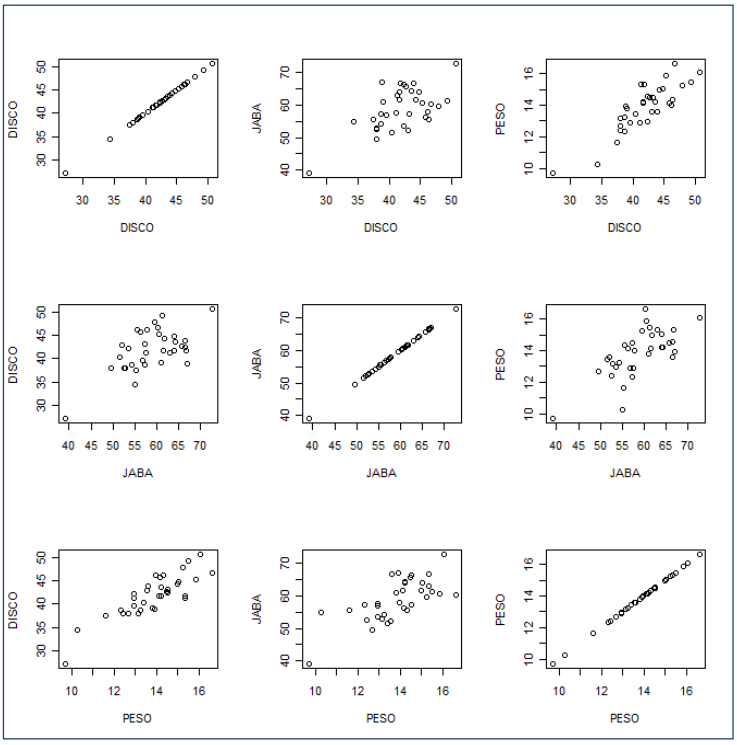


FIGURA 3.8. DIAGRAMAS DE DISPERSIÓN PARA LAS VARIABLES DISCO-JABALINA-PESO

El grafico anterior también puede hacerse directamente con la función `pairs()` que crea los gráficos de dispersión por variables.

```
> pairs(datos[,c("DISCO","JABA","PESO")]);
```

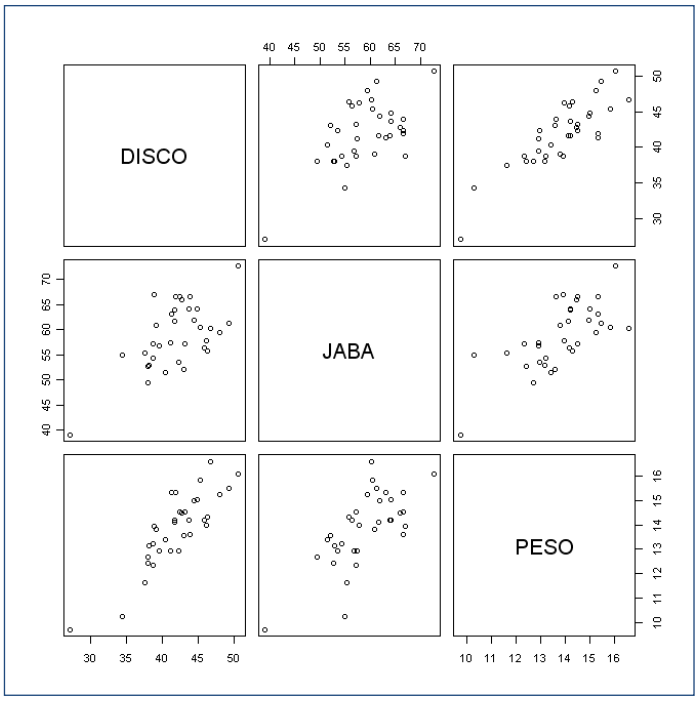


FIGURA 3.9. DIAGRAMAS DE DISPERSIÓN PARA LAS VARIABLES DISCO-JABALINA-PESO OBTENIDAS CON LA FUNCIÓN PAIRS()

En el diagrama de DISCO-PESO vemos como los puntos se distribuyen en el dibujo de forma muy lineal, y aunque el coeficiente de correlación entre las otras variables es alto, se observa cómo la distribución de los puntos no es tan lineal.

Pasemos a observar ahora las relaciones que existen entre las variables del salto de longitud (LONG), y las carreras de 100 y 110 metros lisos (T100 y T110)

```
> cor(datos[,c("LONG","LONG","LONG")])
             LONG       T100       T110
LONG    1.0000000 -0.6905079 -0.6544919
T100   -0.6905079  1.0000000  0.7512681
T110   -0.6544919  0.7512681  1.0000000
```

Obviamente, la correlación entre las carreras de 100 y 110 metros es muy alta y positiva, sin embargo la correlación entre el salto de longitud y la carreras es también alta pero sin embargo negativa. Gráficamente:

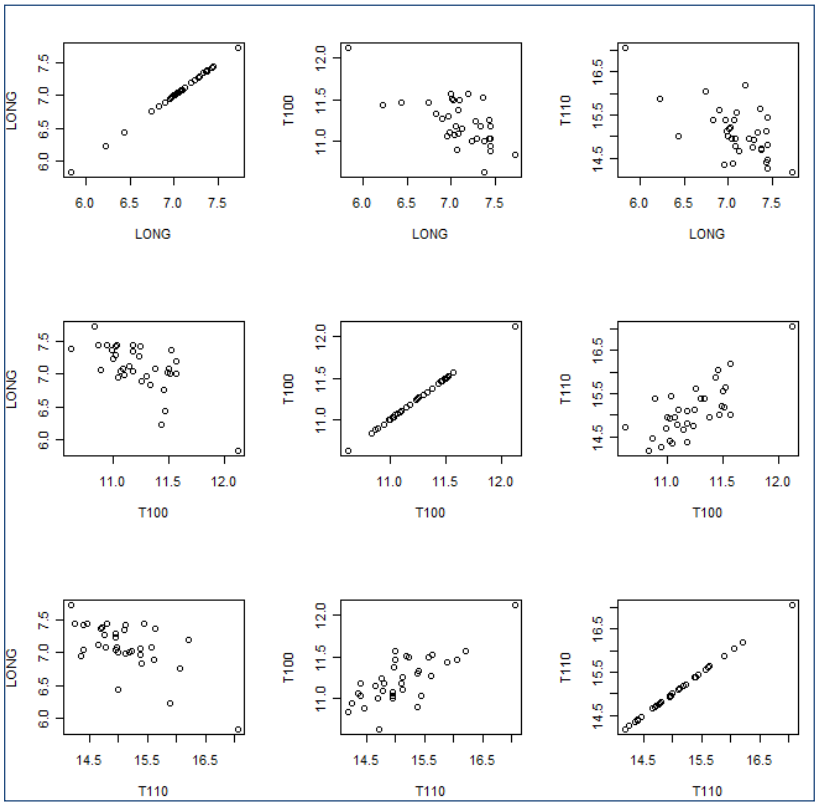


FIGURA 3.10. DIAGRAMA DE DISPERSIÓN DE LAS VARIABLES LONG-T100-T110

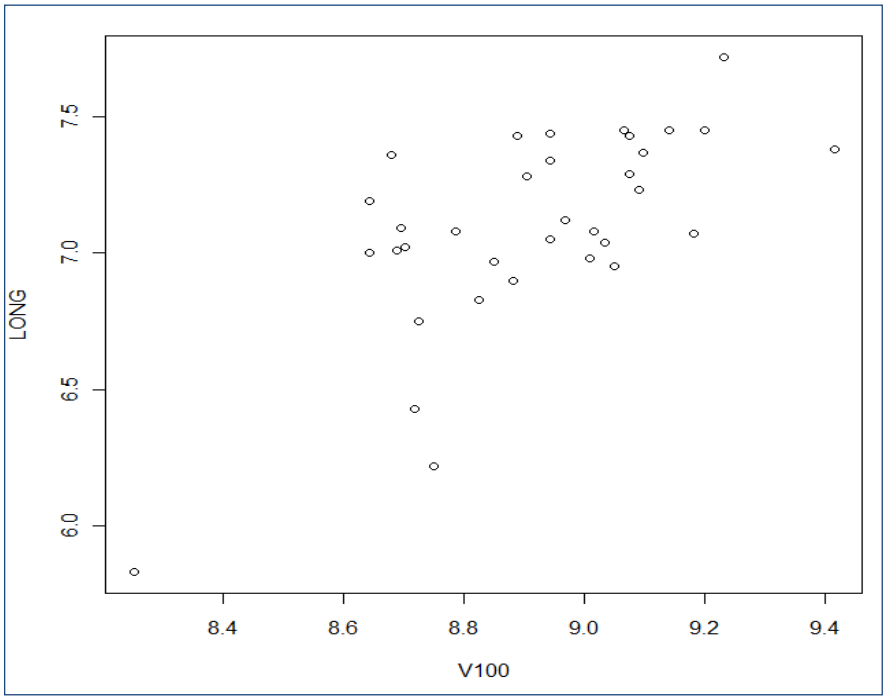
Ahora se observa lo que significa un coeficiente de correlación negativo: la pendiente de la posible recta que aproxima a los puntos es negativa. Pero, ¿esto significa que cuanto más salte, menos corre? Casos de grandes atletas que ganaban en una misma olimpiada la medalla de oro de 100 metro lisos y la de salto de longitud, ¿eran por simples excepciones de la naturaleza?

En primer lugar cuando evaluamos estos diagramas, tenemos que ser conscientes en que unidades estamos midiendo las variables. En este caso las pruebas de velocidad se miden en

segundos, mientras que el salto de longitud se hace en metros. Por lo tanto sí que es lógico afirmar que cuantos más metros salte, menos segundos tardará en recorrer 100 metros. Si transformamos las variables medidas en tiempo a unidades de velocidad media el signo de los coeficientes de correlación se debe invertir. Para ello, vamos a crear una estructura de datos con las velocidades medias de todas las pruebas de carreras:

```
> V100<-100/T100
> V110<-110/T110
> V400<-400/T400
> V1500<-1500/T1500
> V_MEDIAS<-data.frame(V100,V110,V400,V1500)
> cor(V_MEDIAS)
           V100      V110      V400     V1500
V100  1.0000000 0.7358233 0.6818714 0.2465959
V110  0.7358233 1.0000000 0.6384659 0.1511121
V400  0.6818714 0.6384659 1.0000000 0.5565242
V1500 0.2465959 0.1511121 0.5565242 1.0000000
```

Los coeficientes de correlación varían en las nuevas unidades. Las correlaciones entre las velocidades han variado ligeramente. Los coeficientes entre las velocidades y el resto de las pruebas han cambiado de sigo, debido al cambio de variable. Observa que es muy importante a la hora de interpretar los coeficientes, saber en qué unidades se está midiendo cada variable. En la figura inferior se observa que la correlación entre velocidad en la prueba de 100 metros y el salto de longitud es ahora positiva, los atletas más rápidos son los que más saltan.



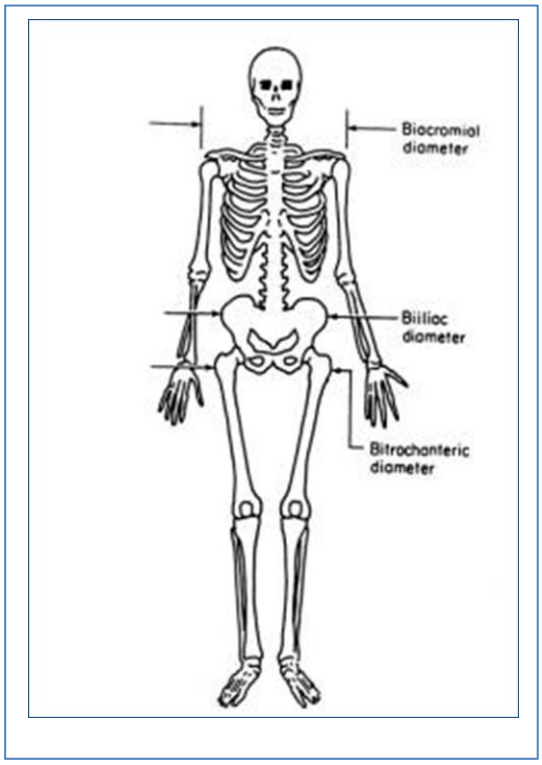


FIGURA 3.11. DIAGRAMA DE DISPERSIÓN DE LAS VARIABLES V100-LONG

La nube de puntos se distribuye en forma de una recta de pendiente positiva. En efecto, si comprobamos el valor del coeficiente de correlación entre ambas variables:

```
> cor(LONG,V100)
[1] 0.6816047
```

2.5. EL CUERPO HUMANO

Vamos a analizar los datos correspondientes a diferentes medidas del cuerpo humano (perímetros, medidas del esqueleto, edad, peso, estatura y sexo) de 507 personas (247 hombres y 260 mujeres) con el objetivo de estudiar las relaciones entre las diferentes medidas por un lado y encontrar las variables que mejor discriminan a hombres y mujeres, por otro.

Los datos corresponden a personas que acuden periódicamente a un gimnasio y se han obtenido de la revista electrónica Journal of Statistics Educations, Vol. 11, Nº 2, *"Exploring Relationships in Body Dimensions". (*www.amstat.org/jse/v11n2/datasets.heinz.html*)*

Se disponen de tres grupos de variables:

Medidas Esqueléticas:

1. Diámetro de Biacromial (véase figura)
2. Diámetro de Biiliac, o anchura pélvica (figura)
3. Diámetro de Bitrochanteric (Figura)
4. Profundidad del pecho entre la espina dorsal y el esternón en nivel del pezón, media expiración
5. Diámetro del pecho en el nivel del pezón (media expiración)
6. Diámetro del codo, suma de los dos codos
7. Diámetro de la muñeca, suma de las dos muñecas
8. Diámetro de la rodilla, suma de las dos rodillas
9. Diámetro del tobillo, suma de los dos tobillos

Medidas musculares:

1. Perímetro de los hombros sobre los músculos deltoides
2. Circunferencia del pecho, línea del pezón en varones y apenas sobre el pecho tejido fino en hembras, media expiración
3. Circunferencia de la cintura, la parte más estrecha del torso debajo de la caja torácica (costillas) promedio de contraído y posición relajada
4. Circunferencia del ombligo (o "abdominal") en ombligo y cresta ilíaca, cresta ilíaca como señal
5. Circunferencia de la cadera en el nivel del diámetro bitrochanteric
6. Circunferencia debajo del doblez glúteo, promedio del perímetro del muslo de la derecha e izquierda
7. Circunferencia de Bicep, doblada, promedio de la derecha e izquierda
8. Circunferencia del antebrazo, extendido, palma para arriba, media de la derecha e izquierda.
9. Circunferencia de la rodilla sobre la pantorrilla, doblada levemente, promedio
10. Circunferencia máxima en los gemelos, promedio
11. Circunferencia mínima del tobillo, promedio
12. Circunferencia mínima de la muñeca, promedio

Otras Medidas:

1. Edad (años)
2. Peso (kilogramo)
3. Altura (centímetro)
4. Sexo (1 - varón, 0 - mujer)

Todas las medidas están en centímetros excepto el peso que está en kilogramos y la edad en años. El primer grupo de medidas corresponden al esqueleto y junto con la altura proporcionan información de la *estructura* corporal de cada individuo. El resto de las variables están afectadas por la masa corporal y muscular del individuo y no son fijas en el tiempo en cada individuo excepto la circunferencia medida en muñeca, rodilla y tobillos.

En primer lugar cargamos las variables e identificamos el nombre de las 25 variables.

```
> cuerpo = read.table("CUERPO.TXT",header=TRUE)
> attach(cuerpo)
> names(cuerpo)
 [1] "A_Hombros" "A_Pelvis"  "A_Cade"  AP_Pecho"  "AD_Pecho"  "A_Codo"
 [7] "A_Muneca" "A_Rodilla" "A_Tobillo" "C_hombros" "C_Pecho" "C_Cintura"
[13] "C_abdomen" "C_Cadera" "C_Muslo"   "C_Bicep"  "C_Brazo" "C_Rodilla"
[19] "C_Gemelo" "C_Tobillo" "C_Muneca"  "Edad"     "Peso"    "Altura"
[25] "Sexo"
```

Vamos a trabajar solamente con las variables Peso, Altura y Sexo. El objetivo es mostrar las diferencias entre hombres y mujeres en relación a estas variables. Las siguientes instrucciones realizan nos permiten hacer el gráfico que mostamos a continuación

```
> par(mfrow=c(1,2)) #divide la figura en 1 fila y dos col
> etiq = 'Altura (0=Mujeres  1=Hombres)') #etiqueta del graf
> boxplot(Altura ~ Sexo, notch=T, col="blue", xlab=etiq)
> boxplot(Peso ~ Sexo, notch=T, col="blue", xlab=etiq)
```

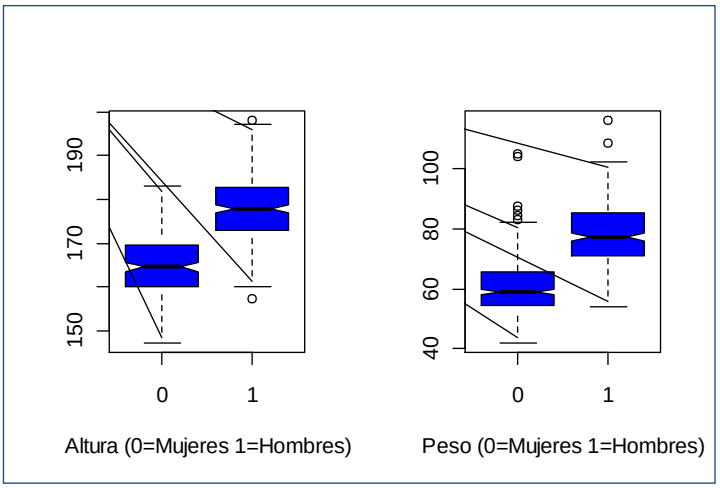


FIGURA 3.12 BOXPLOT DE ALTURA Y PESO PARA HOMBRES Y MUJERES

Los histogramas para las dos variables diferenciando hombres y mujeres aparecen en la figura 3.13 y se obtienen con las siguientes instrucciones:

```
> par(mfrow=c(2,2))
> hist(Altura[Sexo==0],xlim=c(140,200),col=2)
> hist(Peso[Sexo==0],xlim=c(40,120),col=3)
> hist(Altura[Sexo==1],xlim=c(140,200),col=2)
> hist(Peso[Sexo==1],xlim=c(40,120),col=3)
```

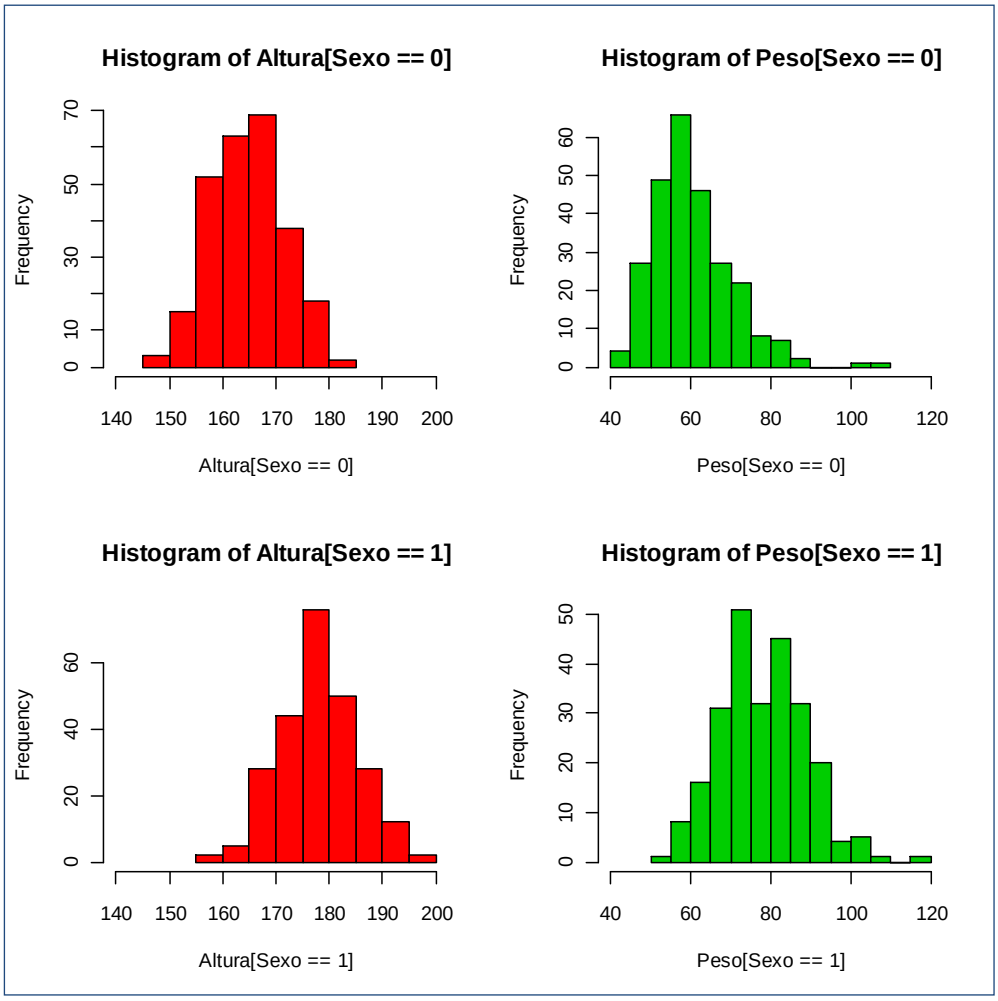


FIGURA 3.13. HISTOGRAMAS DE PESO Y ALTURA PARA HOMBRES Y MUJERES

Se observa que los hombres pesan y miden más que las mujeres. Los histogramas son bastante simétricos y con forma de campana. Antes de calcular las medias y desviaciones típicas para comparar las variables, es imprescindible realizar los histogramas, para comprobar que no existen anomalías en los datos. Vamos a calcular las medidas descriptivas básicas:

```
> c(mean(Altura[Sexo==1]),sd(Altura[Sexo==1]))
[1] 177.745344   7.183629
> c(mean(Altura[Sexo==0]),sd(Altura[Sexo==0]))
[1] 164.872308   6.544602
> c(mean(Peso[Sexo==1]),sd(Peso[Sexo==1]))
[1] 78.14453 10.51289
> c(mean(Peso[Sexo==0]),sd(Peso[Sexo==0]))
[1] 60.600385  9.615699
```

Estudiamos la relación entre las variables con un gráfico de dispersión, diferenciando a hombres de mujeres (ver figura 2.14)

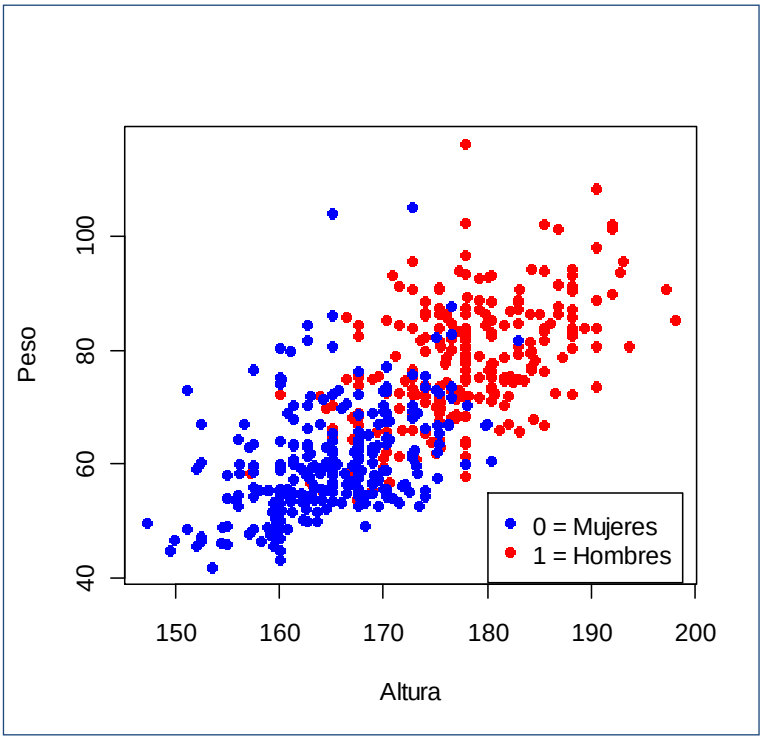


FIGURA 2.14 GRÁFICO DE DISPERSIÓN DE PESO FRENTE A ALTURA

Los coeficientes de correlación se obtienen de la siguiente forma

```
>cor(Altura[Sexo==0],Peso[Sexo==0])
[1] 0.4310593
> cor(Altura[Sexo==1],Peso[Sexo==1])
[1] 0.5347418
```

Observar que obtener el coeficiente de correlación sin distinguir entre hombres y mujeres es desaconsejable, dado que cada grupo de datos tiene media distinta.

2.6. EJERCICIOS PROPUESTOS

I. En 1893 Lord Rayleigh investigó la densidad del nitrógeno empleando en su obtención distintas fuentes. Previamente había comprobado la gran discrepancia existente entre la densidad del nitrógeno producido tras la eliminación del oxígeno del aire y el nitrógeno producido por la descomposición de ciertos compuestos químicos. Los datos de la tabla muestran esta diferencia de forma clara, esto llevó a Lord Rayleigh a investigar detenidamente la composición del aire libre de oxígeno y al descubrimiento, en colaboración con William Ramsay, de un nuevo elemento gaseoso, el argón. Este descubrimiento les valió un premio Nobel a cada uno, en física y química respectivamente.(archivo Nitrogeno.txt).

- Analice numérica y gráficamente estos datos. Preste especial atención al diagrama de tallo y hojas y al diagrama de cajas. ¿Hay alguna peculiaridad de la población de pesos que se manifieste en un diagrama y no en el otro?

- Realice diagramas de cajas dividiendo los datos en los pesos obtenidos a partir de aire y los obtenidos a partir de compuestos químicos del nitrógeno. ¿Qué se observa?

Comentarios:

Bimodalidad de los datos, implica que se están midiendo cosas distintas según sea a partir de aire o de compuestos de nitrógeno. Cuales son la causas de estas diferencias?

II. Una de las principales atracciones turísticas del Parque Nacional de Yellowstone (Estado de Wyoming, Estados Unidos de Norteamérica) es el geiser Old Faithful, cuyo nombre procede del hecho de que sus erupciones siguen una pauta bastante estable a lo largo del tiempo. En la siguiente tabla se proporcionan los lapsos de tiempo transcurridos entre sucesivas erupciones (variable Lapso) y las duraciones de esas erupciones (variable Duración). Ambas variables se dan en minutos. (archivo Oldfaithful.txt).

- Estudie numérica y gráficamente ambas variables ¿se observa alguna peculiaridad en ellas?
- Represente una variable frente a la otra y vea si existe alguna relación entre ellas.
- Resulta de interés conocer aproximadamente el intervalo de tiempo sin actividad entre dos erupciones ¿podría prever el tiempo de que dispondría para realizar otras visitas en el parque entre dos erupciones sucesivas?
- Una teoría geofísica indica que erupciones cortas deberían ser seguidas por cortos intervalos de espera hasta la siguiente erupción, y que erupciones largas deberían ser seguidas por largos intervalos de espera hasta la siguiente erupción. Algunos observadores de este fenómeno natural sugieren dividir las erupciones entre en dos grupos: Aquellas con duración inferior a tres minutos y aquellas con duración superior a ésta ¿Podría utilizarse esta información de modo provechoso para responder a la misma pregunta del apartado anterior?

Comentarios:

Bimodalidad para las dos variables

La gráfica `lapso=f(duración)` sugiere que hay dos subpoblaciones para el lapso , la correspondiente a duración para erupción anterior <3,1 y la correspondiente a duración >z3,1 (para que se pueda predecir está puesto cada lapso con la duración del anterior).

III. En la Olimpiada de Seúl en 1988 participaron 34 atletas en la competición de decatlón. Las pruebas de la competición son 100 metros lisos (T100), 100 metros vallas (T110), 400 metros lisos (T400), 1500 metros lisos (T1500), lanzamiento de disco (DISCO), lanzamiento de peso (PESO), lanzamiento de jabalina (JABA), salto de longitud (LONG), salto de altura (ALTU) y salto con pértiga (PERT). Se trata de estudiar las correlaciones entre las marcas obtenidas en las diversas pruebas. Téngase en cuenta que estas pruebas se dividen en carreras (en segundos), lanzamientos y saltos (en metros). (archivo Olimpiada.txt).

- Estudie la matriz de correlación indicando qué variables están más correlacionadas entre sí. Indique qué par de variables muestran mayor grado de correlación.
- Explique por qué algunos pares de variables presentan correlación negativa, como por ejemplo salto de longitud y 100 metros lisos.

- Transforme las cuatro variables correspondientes a carreras de tiempos a velocidades (por ejemplo, si X_i es el tiempo del atleta i en 100 metros lisos, $V_i=100/X_i$ será la velocidad media del atleta $_i$ en esa prueba) y obtenga la nueva matriz de correlaciones. ¿Qué cambios se aprecian en los coeficientes de correlación?

- Represente gráficamente los datos para comprobar que las relaciones son lineales.

2.7. INSTRUCCIONES UTILIZADAS EN ESTE CAPÍTULO

read.table("..."): sirve para cargar datos. Es importante tener en cuenta que la dirección del archivo que se quiere cargar debe de ir entre comillas y cada subcarpeta debe separarse por barras hacia delante y no hacia atrás, como es usual en Windows (esta barra se obtiene pulsando SHIFT+7). Un argumento muy útil de esta instrucción es header=T para que R reconozca los nombre de las columnas de los datos que introducimos. Si no estuviera este argumento, R interpretaría el nombre de la columna de datos como el primer dato, de tal manera que se pasaría a manejar datos cualitativos y cuantitativos.

`read.table("…")`	sirve para cargar datos
`mean(x)`	muestra por pantalla la media de un conjunto de datos x.
`median(x)`	muestra por pantalla la mediana de un conjunto de datos x
`quantile(x,p)`	muestra por pantalla el percentil 100*p% de un conjunto de datos x
`var(x)`	muestra por pantalla la varianza de un conjunto de datos x
`sd(x)`	muestra por pantalla la desviación típica de un conjunto de datos x
`summary`	muestra por pantalla un resumen estadístico de un conjunto de datos x. Fíjese que en el resumen estadístico no aparece ni la varianza, ni la desviación típica
`hist(x)`	muestra en una nueva pantalla, el histograma de un conjunto de datos x. Para imponer la cantidad de clases (por ejemplo 8), se modificará la instrucción con el argumento breaks=8. Si además se quiere cambiar el color, por ejemplo por el rojo, se escribe col="red".
`moment(x, order=p, central=FALSE)`	requiere haber cargado previamente el paquete "moments". Muestra por pantalla el momento de orden p de la variable x. Si central=TRUE calcula el momento respecto a la media.
`skewness(x)`	requiere haber cargado previamente el paquete "moments". Muestra por pantalla el valor del coeficiente de asimetría
`kurtosis(x)`	requiere haber cargado previamente el paquete "moments". Muestra por pantalla el valor del coeficiente de apuntamiento, también llamado kurtosis. El valor de una distribución normal de datos debería ser cercano a 3.
`boxplot(x)`	muestra en una nueva pantalla, el diagramas de cajas de un conjunto de datos x. Por defecto el diagrama aparecerá en vertical. Para facilitar su interpretación hay que hacer que aparezca en horizontal con el argumento horizontal=T. Es muy aconsejable darle un color al diagrama con el fin de obtener

	una mejor interpretación
`outline =FALSE`	es un argumento que se utiliza en distintas instrucciones para eliminar los datos atípicos de nuestros cálculos
`steam.leaf(x)`	muestra en una nueva pantalla, el diagrama de tallos y hojas de un conjunto de datos x. Para utilizar esta función hay que cargar el paquete "aplpack"
`attach(x)`	cuando tenemos una tabla de datos x, esta instrucción crea tantas variables como columnas tenga la tabla. Los nombres de las nuevas variables se corresponden con los nombre de cada columna
`cor(x,y)`	muestra por pantalla el coeficiente de correlación entre un conjunto de datos x y otro conjunto de datos y
`cor(x)`	muestra por pantalla la matriz de correlaciones de todas las variables que estén definidas en una estructura tipo tabla de nombre x. Si se quieren seleccionar solo algunas de las variables que contiene la tabla, se debe modificar la instrucción, rectificando el grupo de datos sobre el que se quiere trabajar: cor(x[c("a","b","c",…,"n")]); siendo a,b,c,…,n los nombres de las variables sobre las que se quiere trabajar
`plot(x,y)`	muestra en una nueva pantalla, la representación de las parejas de valores asociados a las variables x e y. Coloca x en el eje x e y en el eje y
`par(mfrow=c(m,n))`	permite dibujar n*m gráficos en una misma hoja. Los coloca según se vayan escribiendo a continuación de esta instrucción

3. PROBABILIDAD Y SIMULACIÓN

3.1. INTRODUCCIÓN

En este capítulo veremos funciones relacionadas con variables aleatorias. Conocer la función de densidad de probabilidad, función de distribución de probabilidad y generación de variables aleatorias serán los objetivos fundamentales aquí. También este tema está encaminado a ilustrar mediante simulación distintos teoremas del comportamiento de variables aleatorias. Para conseguir los objetivos que se muestran aquí, iremos resolviendo diferentes problemas que intentarán barrer todas las funciones y conceptos útiles de este capítulo. Se recuerda que al final del capítulo habrá un resumen con las funciones más interesantes.

3.2. EJEMPLO 1: SOBRE LA DISTRIBUCIÓN NORMAL, GRÁFICAS Y SIMULACIONES

Sean X_1, X_2, e X_3, variables aleatorias independientes que se distribuyen según una Normales con las siguientes medias y desviación típicas. $X_1 \rightarrow N(0,1)$, $X_2 \rightarrow N(3,1)$ y $X_2 \rightarrow N(0,2)$. Hacer un estudio de las funciones de densidad de probabilidad y distribución de probabilidad y compararlo con los histogramas de 1000 valores generados por simulación.

En primer lugar representaremos gráficamente las funciones de densidad de probabilidad de las variables aleatorias. Para ello definiremos el rango de valores con la función `seq()` para establecer los puntos en los que se van a evaluar o calcular las funciones de densidad `dnorm()`. A continuación escribiremos todas las sentencias necesarias para crear en una única gráfica las 3 funciones de densidad en colores.

```
> x<-seq(-5,7,0.1)
> plot(x,dnorm(x),col="red",type="l",main="Funciones de densidad",
  + xlab="x",ylab="f_densidad");
> lines(x,dnorm(x,mean=3),col="green");
> lines(x,dnorm(x,sd=2),col="blue");
> legend(4,0.4,c("X1->N(0,1)","X2->N(3,1)","X3->N(0,2)"),
  + col=c("red","green","blue"),lty=1);
```

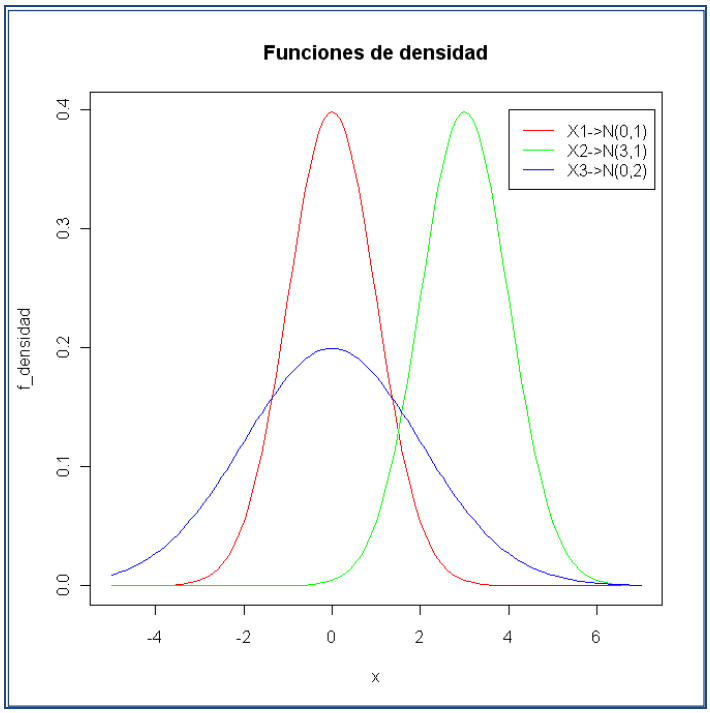


FIGURA 4.1. DIBUJO DE LAS FUNCIONES DE DENSIDAD DE PROBABILIDAD

De la misma forma se pueden representar las funciones de distribución de probabilidad de las variables aleatorias. Usaremos las mismas sentencias que antes, unicamente utilizaremos la función `pnorm()` que nos proporciona la función de distribución de probabilidad `pnorm(xo)`=P(X < = xo)

```
> x<-seq(-5,7,0.1)
> plot(x,pnorm(x),col="red",type="l",main="Funciones de distribución",
  + xlab="x",ylab="f_distribucion");
> lines(x,pnorm(x,mean=3),col="green");
> lines(x,pnorm(x,sd=2),col="blue");
> legend(4,0.4,c("X1->N(0,1)","X2->N(3,1)","X3->N(0,2)"),
  + col=c("red","green","blue"),lty=1);
```

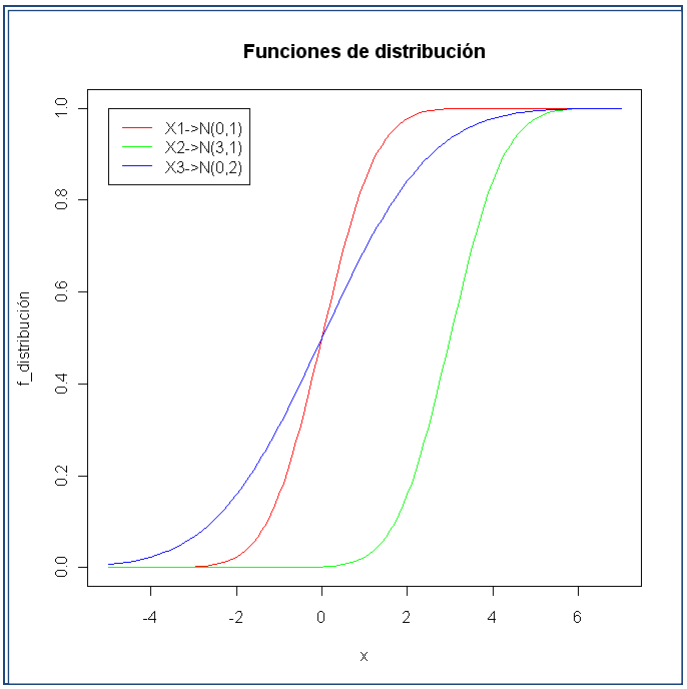


FIGURA 4.2. DIBUJO DE LAS FUNCIONES DE DISTRIBUCIÓN DE PROBABILIDAD

Ahora vamos a genera 1000 números aleatorios de cada una de las distribuciones de probabilidad, para ello utilizaremos la función `rnorm()`. Para luego definir una nueva variable aleatoria $Y=X_1+X_2+X_3$ de la cual diremos su función de densidad de probabilidad teorica y la compararemos con lo obtenido mediante simulación.

Generamos mediante simulación las variable X_1, X_2 y X_3 y luego definimos la variable Y.

```
> X1<-rnorm(1000)
> X2<-rnorm(1000,mean=3)
> X3<-rnorm(1000,sd=2)
> Y=X1+X2+X3;
```

Según la teoría la combinación de variables aleatorias normales es una normal, por lo tanto Y se distribuirá como una variable aleatoria normal de la cual solamente tenemos que calcular su media y su desviación típica

$E[Y]=E[X_1]+E[X_2]+E[X_3]=0+3+0=3$

$Var[Y]=Var[X_1]+Var[X_2]+Var[X_3]=1+1+4=6$

Vamos a calcular variables estadísticas en la variable Y que hemos generado con R comprobando que se distribuye según una normal $N(3,\sqrt{6})$:

```
# Calculamos la media muestral que debe ser similar a la poblacional 3
> mean(Y)
[1] 3.03093
# Calculamos la varianza muestral que debe ser similar a la poblacional 6
> var(Y)
[1] 5.957485
# Calculamos la Frecuencia absoluta de que Y<=4 y la comparamos con la
probabilidad
```

```
> length(Y[Y<=4])/length(Y)
[1] 0.671
> pnorm(4,mean=3,sd=sqrt(6))
[1] 0.6584543
```

Finalmente la última comparación que haremos será entre la función de densidad y el histograma teorico de la variable aleatoria.

```
> hist(Y,freq=F,ylim=c(0,0.2),xlim=c(-5,10))
> x<-seq(-5,10,0.1)
> lines(x,dnorm(x,mean=3,sd=sqrt(6)),col="blue")
```

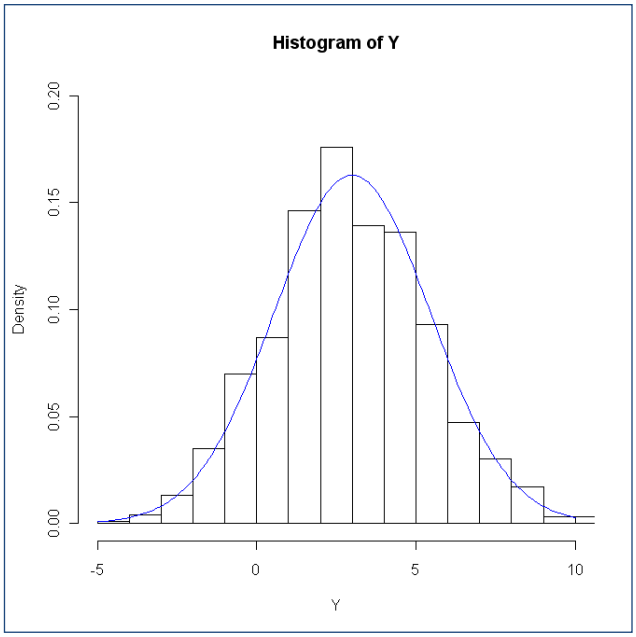


FIGURA 4.3. HISTROGRAMA Y FUNCIÓN DE DENSIDAD DE PROBABILIDAD TEÓRICA DE LA VARIABLE ALEATORIA Y

3.3. EJEMPLO 2: CÁLCULO DE PROBABILIDADES

En el primer curso de una facultad hay 5 asignaturas. Se permite pasar a 2º curso a todos los alumnos que hayan aprobado un mínimo de 3 asignaturas. La probabilidad de aprobar cada asignatura es del 60%. Vamos a contestar a las siguientes preguntas de forma teórica y haciendo uso de una muestra de 1000 de las variables aleatorias que nos interese generar.

a) ¿Cuál es la probabilidad de pasar de curso?

Para responder a esta pregunta hay que definir una variable aleatoria que sea X: número de asignaturas aprobadas de un total de n=5, sabiendo que la probabilidad de aprobar una asignatura en p=0.5. Por lo tanto X se distribuirá como una variable aleatoria Binomial.

$$X \to B(n=5, p=0.6)$$

Por lo tanto la probabilidad de pasar de curso debe ser la probabilidad de que el número de asignaturas aprobadas sea mayor o igual que 3, $P(X \geq 3)$.

Las probabilidades se calculan con las siguientes funciones:

- *P[X ≤ x]* se obtiene con la función *pbinom(x)* ó *pbinom(x,lower.tail=T)*.
- *P[X> x]* se obtiene con la función *pbinom(x,lower.tail=F)*.
- *P[X=x]* se obtiene con la función *dbinom(x)*. Nota recordar que esta probabilidad solamente tiene sentido para variables aleatorias discretas, para variables aleatorias continuas tomaría el valor nulo.

Entonces la probabilidad que nos dicen $P(X \geq 3)$ se puede calcular así:

```
> pbinom(2,size=5,prob=0.6,lower.tail=F)
[1] 0.68256
```

ó bien,

```
> dbinom(3,size=5,prob=0.6)+pbinom(3,size=5,prob=0.6,lower.tail=F)
[1] 0.68256
```

Si eligiésemos hacerlo mediante datos obtenidos por simulación (de forma aproximada) se tendría que hacer lo siguiente:

```
> X<-rbinom(1000,size=5,prob=0.6)
> length(X[X>=3])/length(X)
[1] 0.657
```

Estos ejercicios tienen carácter pedagógico y se han planteado preguntas sencillas. Muchas veces el cálculo de probabilidad es muy complejo y la única solución posible es mediante simulación. Los valores calculados mediante simulación mejoran aumentando el tamaño muestral (número de simulaciones). Con un millón de simulaciones, la probabilidad anterior mejora claramente

```
> X<-rbinom(1000000,size=5,prob=0.6)
>  length(X[X>=3])/length(X)
[1] 0.682582
```

b) Se quiere en la Universidad que al menos el 85% de los alumnos pase a 2º curso. ¿con cuantas asignaturas se debería dejar pasar de curso?

Esta pregunta se puede considerar como la inversa de la pregunta anterior, ya que nos dan la probabilidad y nos preguntan el número de asignaturas aprobadas $P(X \geq x) = 0.85$. Para realizar este ejercicio utilizamos la función cuantil **qbinom()**

```
> qbinom(0.85,size=5,prob=0.6,lower.tail=F)
[1] 2
```

Vamos a comprobar el resultado anterior calculando la probabilidad $P(X \geq 2)$:

```
> pbinom(1,size=5,prob=0.6,lower.tail=F) #P(X>=2)
[1] 0.91296
```

Si dejasemos pasar de curso con 2 asignaturas pasaría el 91.3% de los alumnos frente al 68.3% que pasaría con 3 asignaturas. Para hacer lo mismo (de forma aproximada) podemos

utilizar datos simulados, habría que ordenar el vector de mayor a menor y coger la componente 850 (equivalente al 85% de personas con mayor número de asignaturas).

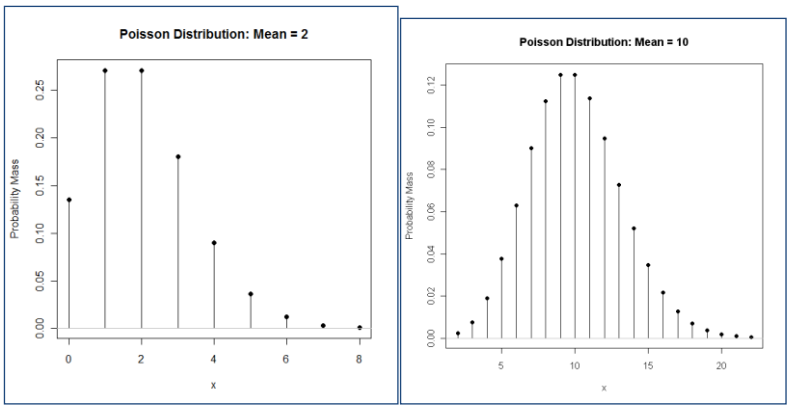
```
> Y=sort(X,decreasing=T);
> Y[850]
[1] 2
```

3.4. Ejemplo 3: Convergencia de la Poisson a la Normal

Sean X_1, X_2, X_3 variables aleatorias de Poisson con λ=2, 10, 50. ¿Qué comportamiento van teniendo las variables aleatorias a medida que el parámetro λ se hace más grande?

Para ver lo que sucede con la variable aleatoria cuando el parámetro λ aumenta vamos a dibujar la función de probabilidad. Para hacer ello podemos utilizar el paquete R-Commander que de forma rápida nos generará los gráficos que queremos. Una vez cargado el paquete tenemos que pulsar en **Distribuciones/Distribuciones discretas/Distribución de Poisson/Gráfica de la distribución de Poisson**. Hecho esto aparece una pantalla en la que seleccionamos la media, en este caso 2, 10, 50. El formato para el gráfico que nos muestra el paquete "Rcmdr" es muy intuitivo visualmente y se pueden ver las instrucciones que utiliza para obtener los gráficos:

```
> x<-0:14;
> plot(x,dpois(x,lambda=2),xlab="x",ylab="Probability Mass",
  + main="Poisson Distribution: Mean = 2", type="h");
> points(x, dpois(x, lambda=2), pch=16);
> abline(h=0, col="gray");
> remove(x);
```



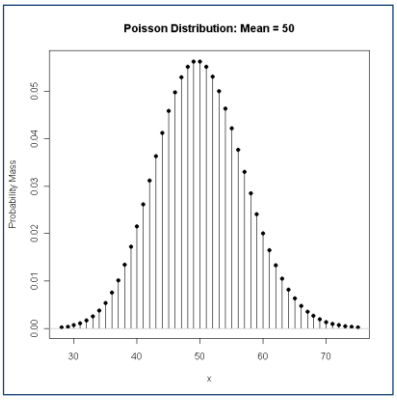


FIGURA 4.4. FUNCIONES DE DENSIDAD PARA VARIABLES DE POISSON CON Λ =2, 10, 50.

Cuando λ aumenta se observa que la función de probabilidad de una v.a. de Poisson tiende a parecerse mucho a la campana de Gauss que es la función de densidad de probabilidad de una v.a. Normal. Por lo tanto como dice la teoría, para λ grandes, Poisson(λ) → Normal(λ, √λ)

3.5. EJEMPLO 4: TEOREMA CENTRAL DEL LÍMITE

Se tienen 10 variables aleatorias independientes X_1, X_2, …, X_{10} de una distribución uniforme continua de probabilidad en el intervalo (0,2). Con ellas se define una nueva variable aleatoria $Y=(X_1+X_2+…+X_{10})/10$. ¿Qué distribución de probabilidad seguirá aproximadamente la variable aleatoria Y? ¿En que teorema se basa la respuesta? Comprobar mediante simulación con muestras de 1000 datos la veracidad de la respuesta anterior?

Por el teorema central del límite, si X_i son variables aleatorias con la misma distribución de probabilidad e independientes, de media μ y desviación típica σ, la distribución de la media se puede aproximar (muy bien) por una distribución normal de media μ y varianza σ^2/n, es decir

$$\frac{X_1 + X_2 + \cdots + X_n}{n} \rightarrow N(\mu, \sigma/\sqrt{n})$$

expresión anterior es conocida como teorema central del límite.

De una distribución Uniforme U(0,2) podemos calcular fácilmente la esperanza y varianza, quedando E[Xi]=1 y Var[Xi]=1/3. Por lo tanto la nueva variable aleatoria Y debe seguir una distribución de probabilidad parecida a la siguiente normal $Y \rightarrow Normal\left(\mu = 1, \sigma^2 = 1/30\right)$.

Para comprobar que lo dicho anteriormente es cierto se creará una matriz con 1000 filas y 10 columnas generada aleatoriamente de una distribución uniforme con la función **runif()**,

```
> X = array(runif(10000,min=0,max=2),dim=c(1000,10));
```

La instrucción `runif(10000,min=0,max=2)` genera 10000 valores con distribución uniforme entre 0 y 2. Con la instrucción `array` los colocamos en una matriz de 1000 filas y 10 columnas. Calculamos las medias de las filas mediante la instrucción `rowMeans()` (existen las instrucciones `colMeans()`, `rowSums()` y `colSums()` que realizan las operaciones correspondientes)

```
> Y = rowMeans(X);
```

El vector Y tiene 1000 valores, cada valor es la media de 10 valores generados de una distribución uniforme entre 0 y 2.

Finalmente para comprobar que el teorema central del limite está en lo cierto dibujamos el histogramas de la muestra Y obtenida mediante simulación junto a la función de densidad de probabilidad de una normal de media 1 y desviación típica 1/sqrt(30).

```
> hist(Y,freq=F,col="green");
> x<-seq(0.4,1.6,0.01);
> lines(x,dnorm(x,mean=1,sd=1/sqrt(30)),col="blue");
```

Calculamos la media y la varianza de la nueva variable para ver que está en consonancia con el resultado teórico.

```
> mean(Y)
[1] 0.9950531
> var(Y)
[1] 0.03538301
```

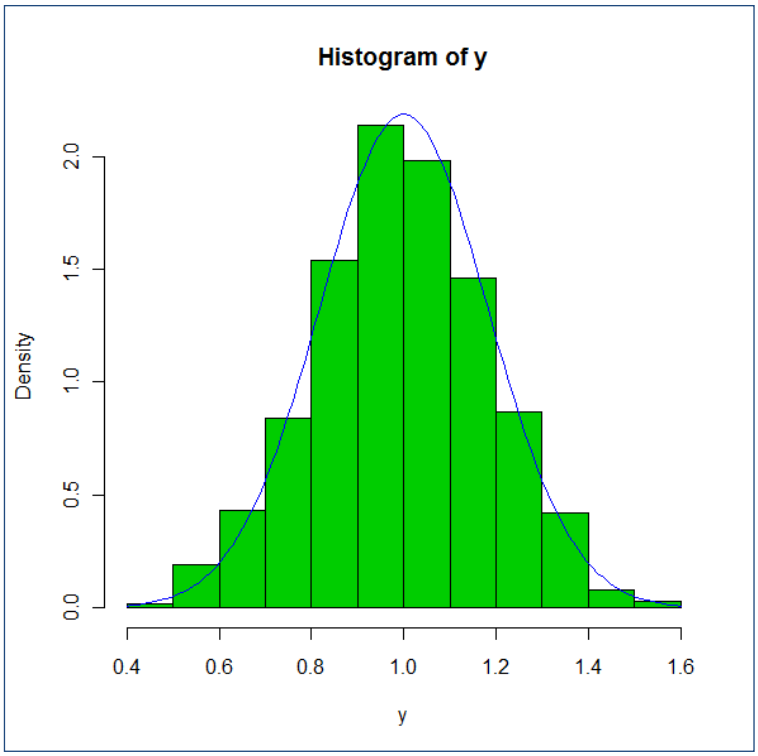


FIGURA 4.5. FUNCIONES DE DENSIDAD TEÓRICA FRENTE A HISTOGRAMA DE LA V.A. Y.

3.6. EJEMPLO 5: DIANA

Supóngase una diana circular con centro en el origen de coordenadas y radio r = 1. Sean X, Y las coordenadas de un punto elegido al azar dentro de la diana (por ejemplo, el lanzamiento de un dardo). Supóngase que cualquier punto de la diana tiene la misma probabilidad de ser elegido.

Responda a las siguientes preguntas de forma teórica y compruebe el resultado mediante simulación. Para realizar simulación se recomienda generar una muestra de tamaño n=1000 de 2 variables aleatorias uniformes entre -1 y 1, eliminando aquellos puntos que salgan fuera del circulo con el fin de obtener las coordenadas X e Y.

a) Área del círculo del problema

A esta pregunta es muy facil de responder de forma teorica, porque se sabe que el área de un circulo es $Area = \pi r^2$, en este caso $Area = \pi$.

Para contestar a esta pregunta por simulación generaremos una nueve de puntos de coordenadas X e Y independientes.

```
> X<-runif(1000,min=-1,max=1);
> Y<-runif(1000,min=-1,max=1);
> Z<-sqrt(X^2+Y^2);
```

Ahora queremos tener sólo los puntos que están dentro de nuestra diana para ello selecciono aquellos puntos cuya distancia al centro sea menor que el radio r=1:

```
> plot(X,Y,col="red")
> plot(X[Z<1],Y[Z<1],col="green")
```

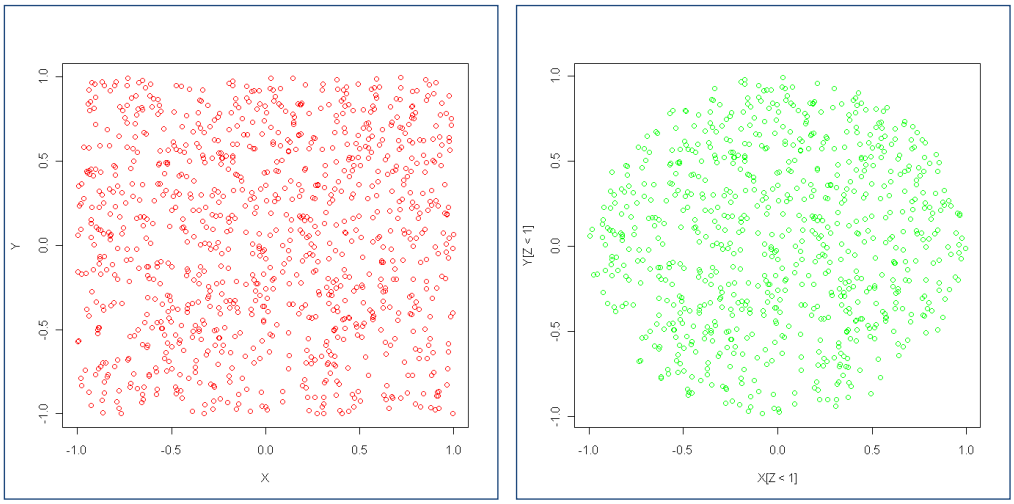


FIGURA 4.6. DIAGRAMA DE DISPERSIÓN DE LOS PUNTOS (X,Y) TOTALES Y VÁLIDOS DENTRO DE LA DIANA.

Para calcular de forma aproximada el Área del círculo podemos hacer uso de la Probabilidad, ya que la probabilidad de que un punto este dentro del círculo será proporcional a su área.

$$\frac{Area\_circulo}{Area\_cuadrado} = P(Z<1) \approx \frac{Puntos\ dentro\ del\ circulo}{1000}.$$

```
> Area=4*length(Z[Z<1])/1000
[1] 3.136
```

b) Considerando a partir de ahora solamente el dominio definido por el círculo de radio unidad y llamando Z a la variable aleatoria definida como la distancia entre el punto elegido y el centro de la diana, calcular la función de densidad de Z.

Para resolver esta cuestión de forma teórica volvemos hacer uso de la relación entre Probabilidades y área.

$$P(Z < z) = \frac{Area(r = z)}{Area(r = 1)} = \frac{\pi z^2}{\pi r^2} = z^2.$$

Esta probabilidad representa también la función de distribución de probabilidad de la variable aleatoria Z, $F_Z(z) = P(Z < z)$. Por lo tanto, se puede obtener la función de densidad de probabilidad de Z derivando la expresión anterior $f(z) = 2z$.

Para comprobar mediante simulación el resultado anterior, definimos Z para los puntos del interior al círculo, y comparamos su histograma con la función de densidad anterior.

```
> X=X[Z<1];
> Y=Y[Z<1];
> Z=Z[Z<1];
> hist(Z,freq=F,col="green");
> z<-seq(0,1,0.01);
> lines(z,2*z,col="blue");
```

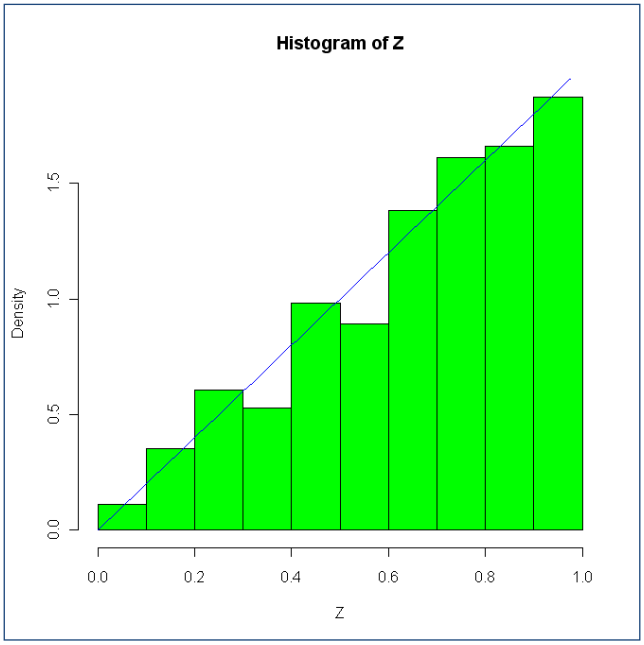


FIGURA 4.7. FUNCIONES DE DENSIDAD TEÓRICA FRENTE A HISTOGRAMA DE LA V.A. Z.

c) Calcular la función de densidad de probabilidad marginal de la coordenada X.

En primer lugar vamos a deducir la expresión teórica para la función de densidad de probabilidad marginal para X.

La función de densidad conjunta $f_{X,Y}(x,y)$ debe ser constante pues todo punto es igual de equiprobable en el interior de la circunferencia. Además la integral de la función de densidad en todo el dominio debe ser la unidad. Con todo esto:

$$f_{X,Y}(x,y)=\frac{1}{\pi}$$

Con la función de densidad conjunta podemos calcular la función de densidad marginal con solo integrar

$$f_X(x)=\int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} f_{X,Y}(x,y)dy=\int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}}\frac{1}{\pi}dy=\frac{2\sqrt{1-x^2}}{\pi}$$

Ahora compararemos mediante simulación el resultado anterior, para eso solamente hay que dibujar el histograma de X frente a la función de densidad anterior.

```
> hist(X,freq=F,col="green");
> x<-seq(-1,1,0.01);
> lines(x,2*sqrt(1-x^2)/pi,col="blue");
```

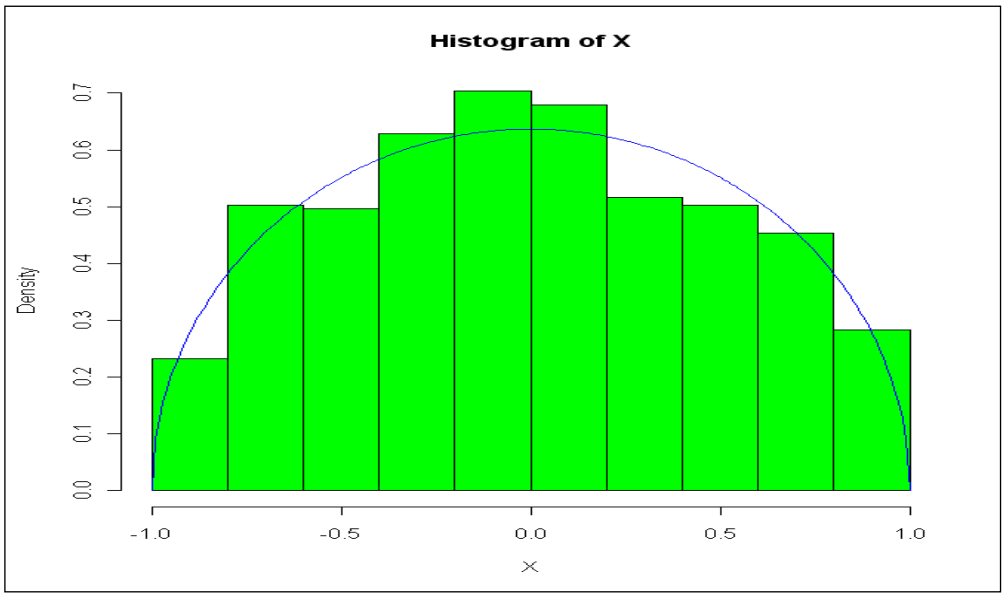


FIGURA 4.8. FUNCIONES DE DENSIDAD TEÓRICA FX(X) FRENTE A HISTOGRAMA DE LA V.A. X

3.7. RESUMEN DE INSTRUCCIONES

Las funciones básicas que hemos visto en este tema son las relacionadas con variables aleatorias. Las letras `d`, `p`, `r` y `q` precediendo al nombre de las distribuciones de probabilidad dist=(norm, unif, binom, pois…) nos darán los siguientes características de una variable aleatoria: (aquí usaremos los caracteres "dist" para referirnos a una función de distribución cualquiera, pero a la hora de utilizar R habría que cambiar los caracteres dist por (norm, unif, binom, pois…)).

- *ddist(x)*: función de densidad de probabilidad en X=x si la v.a. es continua, en el caso de que la variable aleatoria sea discreta obtendremos la probabilidad de que X=x.
- *pdist(x)*: función de distribución de probabilidad, nos da la probabilidad P(X≤x). Se puede obtener P(X>x) utilizando el argumento *lower.tail =F*.
- *rdist(n)*: Se generan n números aleatorios de la función de distribución de probabilidad elegida.
- *qdist(p)*: función de distribución de probabilidad inversa nos da x tal que P(X≤x)=p.

Además la funciones anteriores admiten argumentos, como son los parámetros de las funciones de distribuciones elegidas, por ejemplo en una v.a. hay que definir la media (mean=1) y la desviación típica (sd=2) . Todo esto se puede ver en los ejemplos o en la ayuda.
Las nombres de las funciones para las funciones de densidad, funciones de distribución, inversa de la función de distribución (función de *quatiles*) y el generador de números aleatorios se denominan con las siguientes claves `dxxx`, `pxxx`, `qxxx` and `rxxx` respectivamente. Por ejemplo las funciones de densidad de las distribuciones más conocidas son:

Distribución beta	`dbeta`
Distribución binomial (including Bernoulli)	`dbinom`
Distribución Cauchy	`dcauchy`
Distribución chi-squared	`dchisq`
Distribución exponential	`dexp`.
Distribución F	`df`
Distribución gamma	`dgamma`
Distribución geométrica	`dgeom`
Distribución hypergeometrica	`dhyper`
Distribución log-normal	`dlnorm`
Distribución multinomial	`dmultinom`
Distribución binomial	`dnbinom`
Distribución normal	`dnorm`
Distribución Poisson	`dpois`
Distribución Student's	`dt`
Distribución uniform	`dunif`
Distribución Weibull	`dweibull`

Sustituyendo d por p, q y r en la lista anterior se obtienen el resto de las funciones mencionadas.

4. INFERENCIA

4.1. INTRODUCCIÓN

El objetivo de este capítulo es como realizar la inferencia de un conjunto de datos utilizando R. Lo importante en este tema será realizar estimaciones puntuales, intervalos de confianza y contrastes de hipótesis, incluyendo contrastes de bondad de ajuste. Para ello, se van a realizar una serie de ejercicios que nos irán mostrando las distintas instrucciones que son útiles en este tema.

4.2. INTERVALO DE CONFIANZA PARA LA MEDIA

Vamos a comenzar a resolver un ejercicio que nos va a permitir introducirnos en el tema de la inferencia, y por lo tanto lo desarrollaremos de forma más detallada.

EJEMPLO 1. La resistencia a la compresión de 15 probetas de acero elegidas al azar es:

40,15	65,10	49,50	22,40	38,20
60,40	43,40	26,35	31,20	55,60
47,25	73,20	35,90	45,25	52,40

FIGURA 4.1 DATOS DE RESISTENCIA DE PROBETAS

Vamos a calcular la media y la varianza y a dar intervalos de confianza para los datos de la figura 4.1. Primero metemos los datos, como son pocos, los podemos introducir directamente en R con la instrucción `scan()`.

```
> x<-scan()
1: 40.15 65.10 49.50 22.40 38.20
6: 60.40 43.40 26.35 31.20 55.60
11: 47.25 73.20 35.90 45.25 52.40
16:
Read 15 items
```

La media y la desviación típica de estos datos es

```
> m = mean(x); m
[1] 45.75333
> s = sd(x); s
[1] 14.20399
```

Para calcular el intervalo de confianza para la media no existe ninguna función en R, podemos obtenerlo aplicando la fórmula directamente:

$$\bar{x} - t_{n-1,\alpha/2}\frac{\hat{s}}{\sqrt{n}} \le \mu \le \bar{x} + t_{n-1,\alpha/2}\frac{\hat{s}}{\sqrt{n}} \qquad (4.1)$$

Lógicamente en R no podemos utilizar esta notación, la media muestral es m y la desviación típica es s. Tenemos que obtener los valores de la t de *Student* de 14 grados de libertad, con la

instrucción `qt(alfa,gl)`. Si queremos dar el intervalo de confianza del 99%, teniendo en cuenta que hay 15 datos y por tanto 14 grados de libertada, los valores son

```
> qt(.005,14)
[1] -2.976843
> qt(.995,14)
[1] 2.976843
```

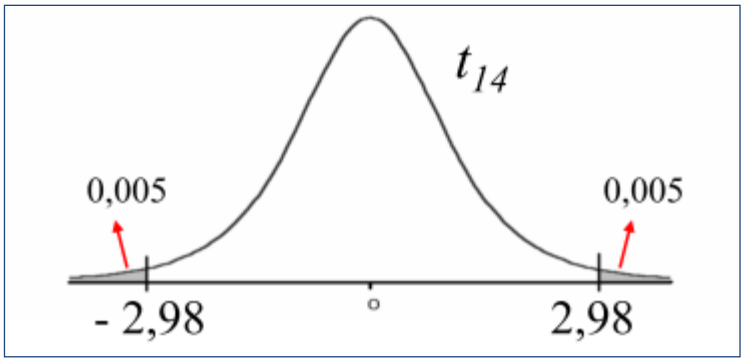


FIGURA 4.2. VALORES DE LA T DE STUDENT CON 14 GL

El intervalo de confianza es por tanto,

```
> n=length(x)
> m+qt(0.005,14)*s/sqrt(n)
[1] 34.8359
> m+qt(0.995,14)*s/sqrt(n)
[1] 56.67077
```

Observa que: `qt(0.005,14)=-qt(0.995,14)`,

en la fórmula (4.1) `tn-1,α/2 = 2.9768 (corresponde con qt(0.995,14))`

Podemos calcular los dos límites en una sola instrucción de la siguiente forma:

```
> c(m+qt(0.005,14)*s/sqrt(n),m+qt(0.995,14)*s/sqrt(n))
[1] 34.83590 56.67077
```

Que se corresponde con lo obtenido en la figura 4.3.

$$45{,}75 - 2{,}98\frac{14{,}2}{\sqrt{15}} \le \mu \le 45{,}75 + 2{,}98\frac{14{,}2}{\sqrt{15}}$$

99 % confianza: $34{,}8 \le \mu \le 56{,}7$

FIGURA 4.3 EJEMPLO DE INTERVALO PARA LA MEDIA

Con estas instrucciones podemos hacer un programa, que denominamos `media.conf()`, que dado unos datos x y un nivel de confianza alfa nos proporciona el intervalo de confianza para la media

```
media.conf <- function(x,alfa)

{n=length(x);

ts=qt(1-alfa/2,df=n-1);

c(mean(x)-ts*sd(x)/sqrt(n),mean(x)+ts*sd(x)/sqrt(n))}
```

Ahora es suficiente escribir la instrucción para conseguir el intervalo.

```
> media.conf(x,0.01)

[1] 34.83590 56.67077
```

4.3. EJEMPLO: VELOCIDAD DE LA LUZ

Vamos a seguir con otro ejercicio, con datos de interés histórico.

En 1879, el físico norteamericano Albert A. Michelson tomó 100 medidas de la velocidad de la luz en el aire empleando una modificación del método propuesto por el físico francés Foucault. Posteriormente, en el año 1882, e independientemente a Michelson, Simon Newcomb midió el tiempo que una señal luminosa tardaba en recorrer una distancia de 7.442 metros. Vamos a calcular los intervalos de confianza para la media de los datos de Michelson.

Todos estos datos pasados a velocidad de la luz en miles de km/s se encuentran en el fichero Michelson_Newcomb. Empecemos cargando los datos:

```
> datos<-read.table("Michelson_Newcomb.txt",header=T)
```

Si echamos un vistazo rápido a los datos, vemos que hay 166 observaciones, donde en la primera columna aparece la velocidad y en la siguiente columna el científico que realizó ese experimento

```
> datos
    Velocidad Cientifico
1      299.85  Michelson
2      299.74  Michelson
3      299.90  Michelson
....
164    298.04    Newcomb
165    298.19    Newcomb
166    298.11    Newcomb
```

Vamos a aceptar en un principio que los datos para cada uno de los científicos provienen de distribuciones de probabilidad normales independientes y tomaremos los datos de Michelson y Newcomb por separado.

```
> Michelson=datos$Velocidad[datos$Cientifico== "Michelson"]

> Newcomb=datos$Velocidad[datos$Cientifico== "Newcomb"]
```

El histograma de las medidas de Michelson se muestra en la figura 4.4

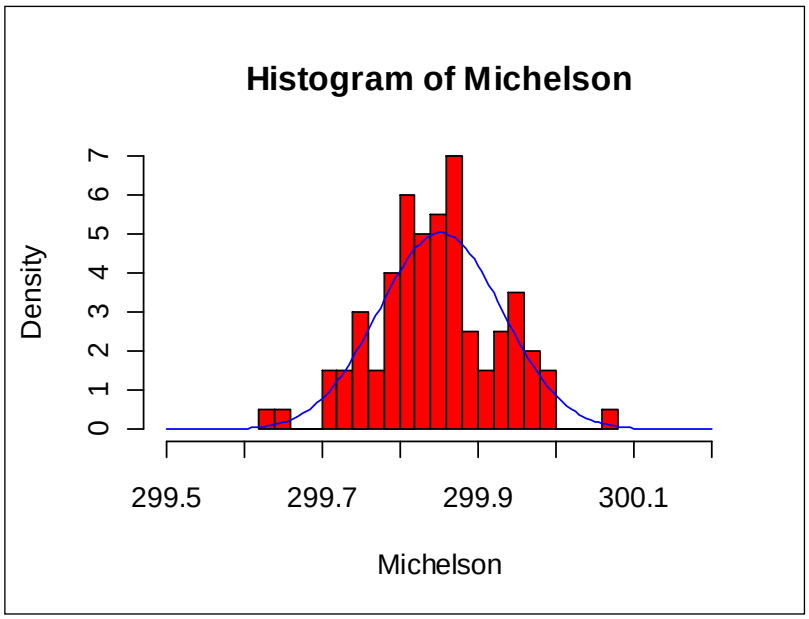


FIGURA 4.4 HISTOGRAMA DE DATOS DE MICHELSON

Y se ha obtenido de la siguiente forma

```
hist(Michelson,30,xlim = c(299.5,300.2),col=2,freq=F)

x<-seq(299.5,300.2,.005);

lines(x,dnorm(x,mean=mean(Michelson),sd=sd(Michelson)),col="blue");
```

La media y la desviación típica correspondientes se obtienen:

```
> c( mean(Michelson),sd(Michelson) )
[1] 299.85240000   0.07901055
```

Para calcular el intervalo de confianza para la media con los datos de Michelson basta con escribir:

```
> media.conf(Michelson,0.05)
[1] 299.8367 299.8681
```

Según se aprecia en la figura 4.4, el ajuste de normalidad no es excesivamente bueno, pero los datos son bastante simétricos y unimodales. El cálculo del intervalo de confianza para la media es válido. Obsérvese que el intervalo obtenido tiene una amplitud de 3 centésimas.

Veamos el mismo análisis para los datos de Newcomb. El histograma se muestra en la figura 4.5. Se observa claramente la presencia de dos medidas muy diferentes al resto, dos medidas que se denominan atípicas o outliers. Para realizar la figura hemos utilizado las siguientes instrucciones:

```
hist(Newcomb,30,xlim = c(297.8,299),col=2,freq=F)
x<-seq(297.8,299,0.05);
m = mean(Newcomb)
s = sd(Newcomb)
lines(x,dnorm(x,mean=m,sd=s),col="blue");
```

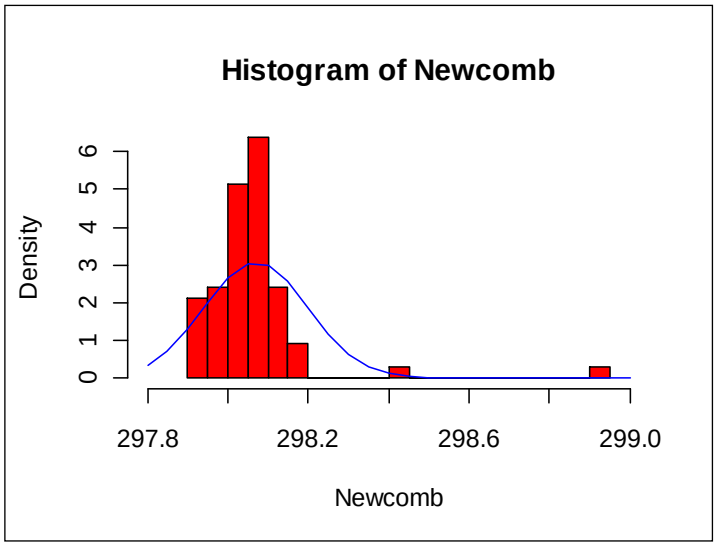


FIGURA 4.5 HISTOGRAMA DE LOS DATOS DE NEWCOMB

Los datos tienen como media y desviación tipica:

```
> c(mean(Newcomb),sd(Newcomb))
[1] 298.0712121   0.1296565
```

Vamos a recalcular todo lo anterior eliminando las dos observaciones atípicas. Para hacer esto, nos damos cuenta en el histograma que son valores superiores a 298.4, en las instrucciones nuestra variable sera Newcomb[Newcomb<298.4], que viene a decir que tome de los datos almacenados en la variable Newcomb aquellos que cumplen la condición Newcomb<298.4

```
hist(Newcomb[Newcomb<298.4],10,xlim = c(297.8,298.4),col=2,freq=F)
m = mean(Newcomb[Newcomb<298.3])
s = sd(Newcomb[Newcomb<298.3]);
x<-seq(297.8,299,0.005);
lines(x,dnorm(x,mean=m,sd=s),col="blue");
c(m,s)
[1] 298.05265625   0.06113327
```

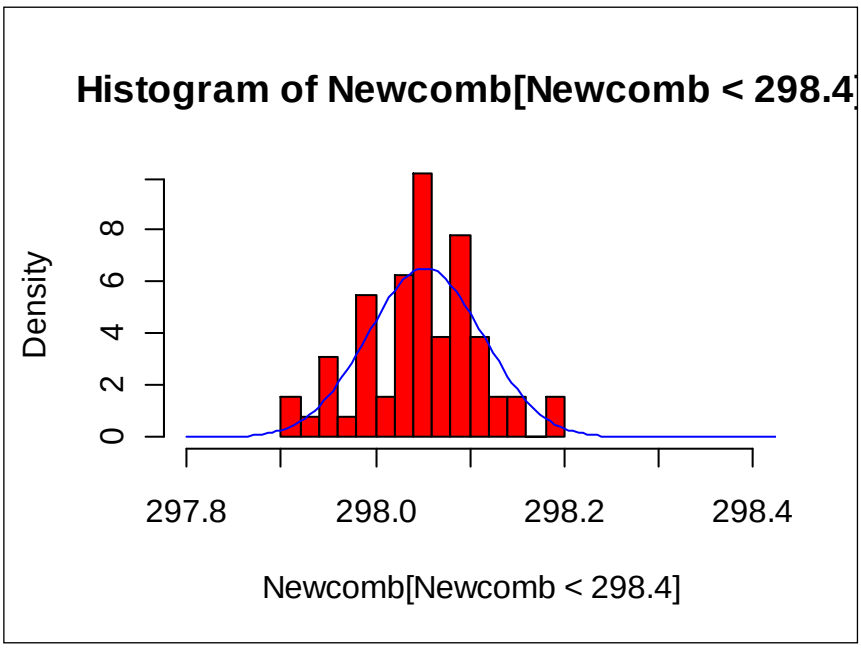


FIGURA 4.6 INSTOGRAMA MODIFICADO DE DATOS DE NEWCOMB

Para estos datos, tiene más sentido aplicar la fórmula del intervalo de confianza, pues el histograma es simétrico y sigue el perfil (aproximadamente) de la normal.

```
media.conf(Newcomb[Newcomb<298.4],0.05)
[1] 298.0374 298.0679
```

Se puede observar que existen grandes diferencias entre las medidas de Michelson y de Newcomb. La media de Michelson es 299.85 (con desviación típica de 0.079) y la de Newcomb 298.05 (con desviación 0.061). Las desviaciones típicas (si eliminamos los outliers de Newcomb) son similares. La diferencia entre las medias es 1.8, que independientemente que a efectos prácticos sea importante, como veremos más adelantes, en términos estadísticos es una diferencia muy significativa.

La mejor forma de apreciar las diferencias entre unas medidas y otras es mediante el gráfico boxplot:

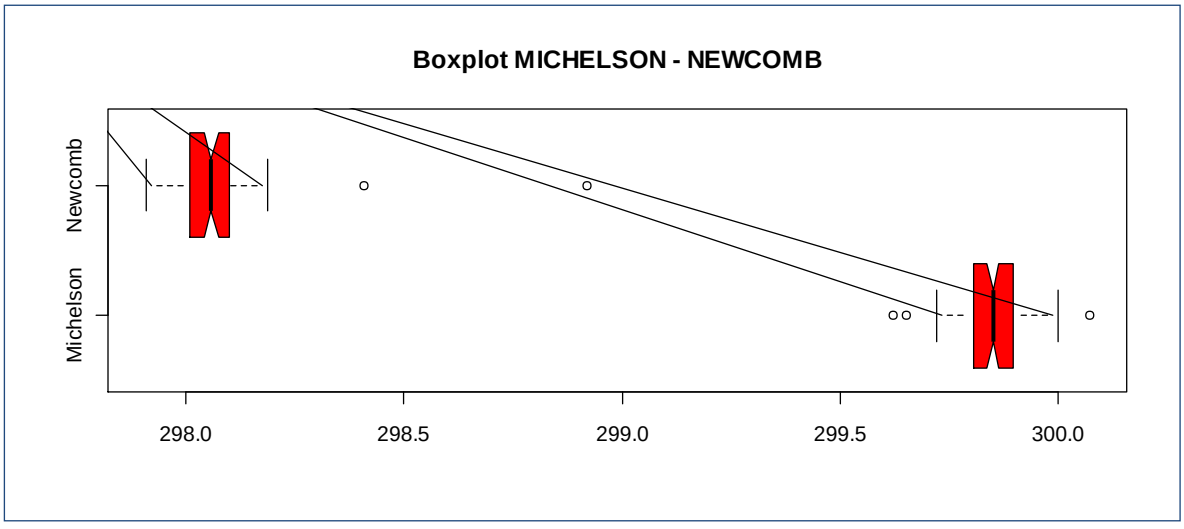


FIGURA 4.7 BOXPLOT COMPARANDO LAS MEDIDAS DE MICHELSON Y NEWCOMB

Observando el boxplot es evidente que las medidas tomadas por uno y otro son muy diferentes.

4.4. INTERVALO DE CONFIANZA PARA LA VARIANZA

Para realizar un intervalo de confianza para la varianza se utiliza la siguiente expresión

$$\frac{(n-1)s^2}{\chi^2_{n-1,1-\alpha/2}} \leq \sigma^2 \leq \frac{(n-1)s^2}{\chi^2_{n-1,\alpha/2}}$$

Directamente vamos a escribir una función que denominamos var.conf() para obtener el intervalo de confianza para la varianza de una distribución normal. La función qchisq(alpha/2,df=n-1) nos proporciona el cuantil de la distribición chi-cuadrado con grados de libertad n-1 (ver figura

```
var.conf <- function(x,alpha)
{
s2 = var(x); n = length(x);
ls = qchisq(alpha/2,df=n-1);
li = qchisq(1-alpha/2,df=n-1);
c((n-1)*s2/li,(n-1)*s2/ls)
}
```

Aplicando la función anterior a los datos proporcionados en la sección 4.1 (figura 4.1), para un nivel de confianza del 99%

```
> var.conf(x,0.01)
[1]  90.18538 693.19574
```

Los cálculos corresponden a la ilustración mostrada en la figura 5.4:

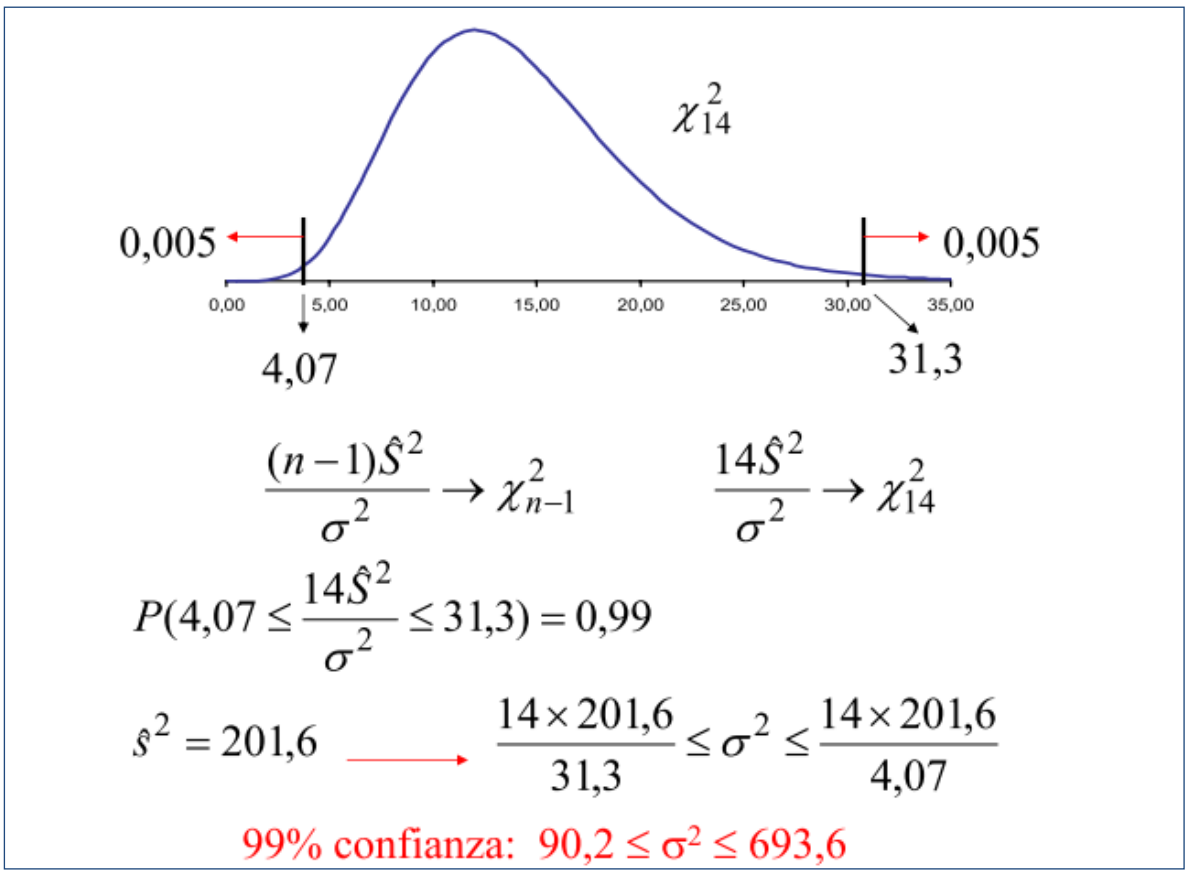


FIGURA 4.8. INTERVALO DE CONFIANZA PARA LA VARIANZA

Valores por defecto. Las funciones tienen unos argumentos o variables de entrada, en la función `var.conf`(x, alpha) es la variable `x` que contiene los datos y `alpha`, el nivel de significación. Es posile definir la función de tal forma que si no se especifica otra cosa, tome como valor de una de las variables de entrada un valor fijo. Por ejemplo, podemos hacer que alpha tome el valor 0.05 por defecto y solo se modifique si especificamente así se le indica. Basta en la definición de la función hacer la siguiente modificación

```
var.conf <- function (x , alpha = 0.05)
{
… (Las mismas instrucciones)
}
```

Cuando invoquemos la función si no asignamos valor al nivel de significación, tomará por defecto alfa=0.05.

4.5. INTERVALOS DE CONFIANZA PARA LA DIFERENCIA DE MEDIAS Y EL COCIENTE DE VARIANZAS CON DATOS NORMALES.

Vamos a ver como se comparan dos tratamientos con R. Aceptamos que los datos tienen distribución normal. Empezamos comparando las medias. Supongamos que deseamos comparar dos tratamientos para reducir el colesterol en la sangre. Se han elegido 20 pacientes al azar de un colectivo muy grande y se han asignado 10, también elegidos al azar, al

tratamiento A (dieta proteica baja en grasas) y a los otros 10 se les ha asignado al tratamiento B (dieta vegetal) . La reducción del nivel de colesterol en unidades adecuadas se proporciona en la figura 4.5. ¿Es mejor una dieta que otra? Aceptando que los datos siguen una distribución normal, vamos a ver si existen diferencias significativas entre las medias de los dos tratamientos.

A	B
51,3	29,6
39,4	47,0
26,3	25,9
39,0	13,0
48,1	33,1
34,2	22,1
69,8	34,1
31,3	19,5
45,2	43,8
46,4	24,9

- Sea desea comparar dos tratamientos para reducir el nivel de colesterol en la sangre. Se seleccionan 20 individuos y se asignan al azar a dos tipos de dietas A y B. La tabla muestra la reducción conseguida después de dos meses.

FIGURA 4.9. COMPARACIÓN DE DOS TRATAMIENTOS

El modelo se muestra en la figura 5.6, y nuestro objetivo es decidir si μ_1 es igual o diferente a μ_2.

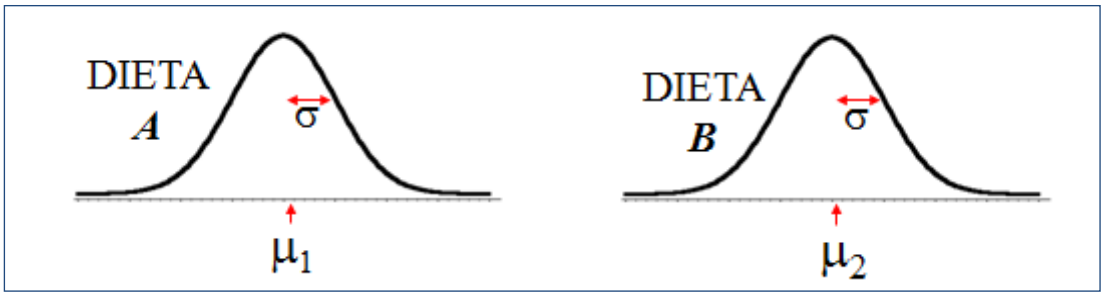


FIGURA 5.6. MODELO ESTÁNDAR PARA COMPARAR DOS TRATAMIENTOS

Introducimos los datos en R,

```
> ya<-c(51.3, 39.4, 26.3, 39.0, 48.1, 34.2, 69.8, 31.3, 45.2, +
46.4)
> yb<-c(29.6, 47.0, 25.9, 13.0, 33.1, 22.1, 34.1, 19.5, 43.8, +
24.9)
```

El contraste de medias y el intervalo de confianza se realiza mediante la función `t.test()` y se emplea de la siguiente manera

```
> t.test(ya,yb,var.equal=T)
Two Sample t-test

data:  ya and yb

t = 2.6965, df = 18, p-value = 0.01476

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

3.048013 24.551987

sample estimates:

mean of x mean of y

43.1      29.3
```

Es importante añadir la condición var.equal=T en la función sI se quiere realizar el test asumiendo homocedasticidad (igualdad de varianzas). En caso contrario realiza el test pero con una corrección (Welch) de los grados de libertad de la t de Student. Cuando el número de datos es el mismo en las dos muestras la varianza conjunta es la media aritmética de las varianzas de cada muestra. En nuestro caso 130.95 y su raíz cuadrada es 11.44 (la desviación típica) que se utiliza en las distintas fórmulas.

Se ha subrayado en los resultados lo más importante. El p-valor del contraste es 0.01476. Nos sirve para comparar con el nivel de significación, si tomamos α = 0.05, como p-valor = 0.01476 es menor que α = 0.05, se dice que existen diferencias significativas. Por el contrario si α = 0.01, la conclusión es la contraria, pues en este caso p-valor = 0.01476 > α = 0.01. Un resumen se muestra en la figura 5.7:

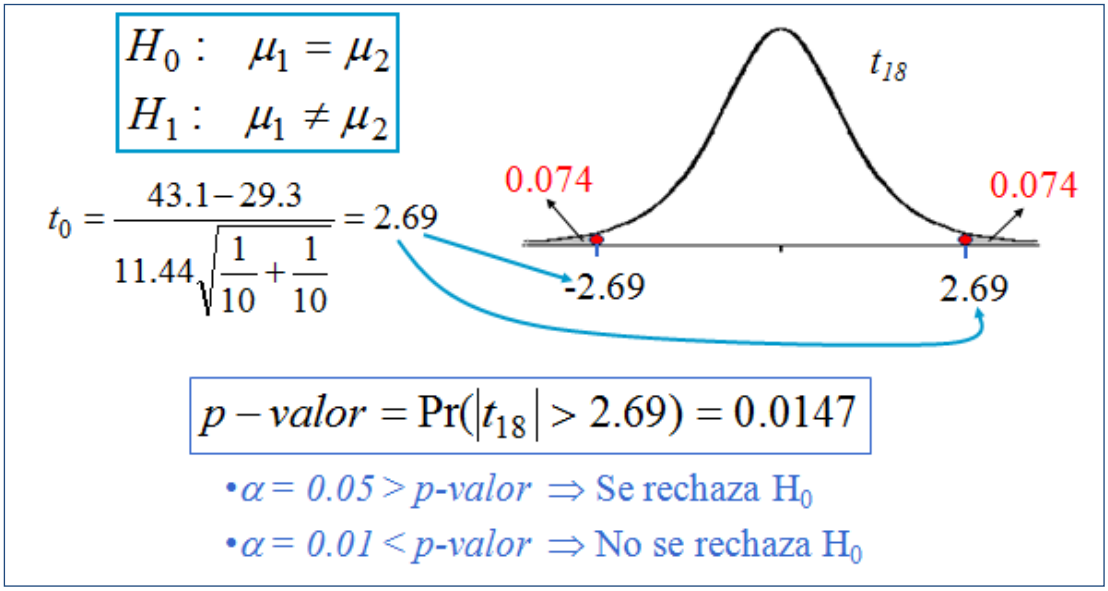


FIGURA 4.10. CÁLCULO DEL P-VALOR PARA LOS DATOS DEL EJEMPLO DE LAS DIETAS.

Otra forma de hacer el contraste es calculando el estadístico *t* que en nuestro caso es 2.69 y comparándolo con el valor crítico obtenido de la t de Student con 18 grados de libertad, que se obtiene con la función `qt()`

```
> qt(.975,18)

[1] 2.100922
```

La figura 5.8 ilustra esta forma de hacer el contraste.

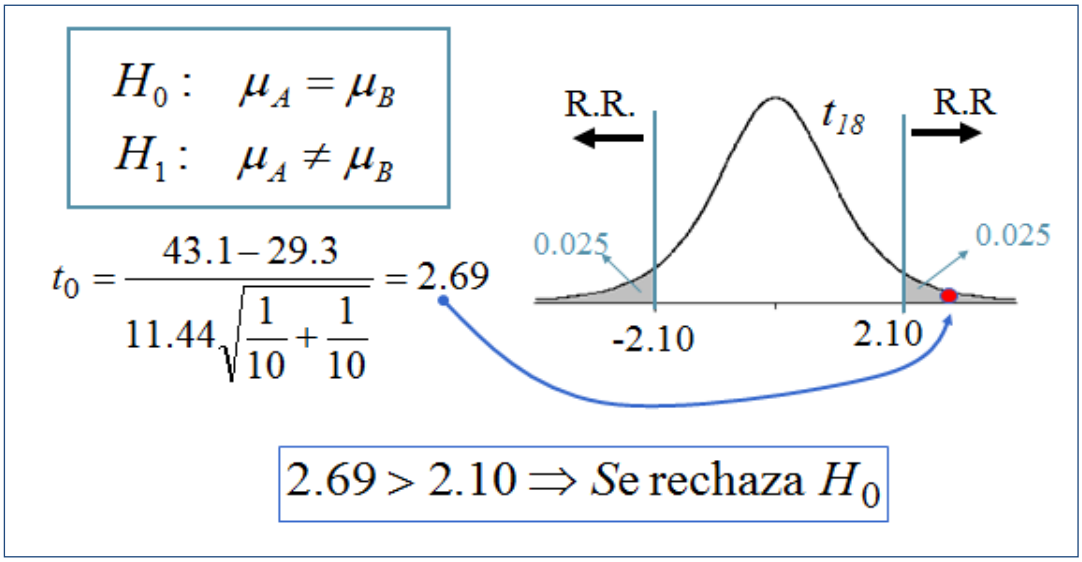


FIGURA 4.11 OTRA FORMA DE HACER UN CONTRASTE DE MEDIAS.

Otra información que proporciona la instrucción `t.test()` es el intervalo de confianza para la diferencia de medias. Las fórmulas aparecen en la figura 5.9:

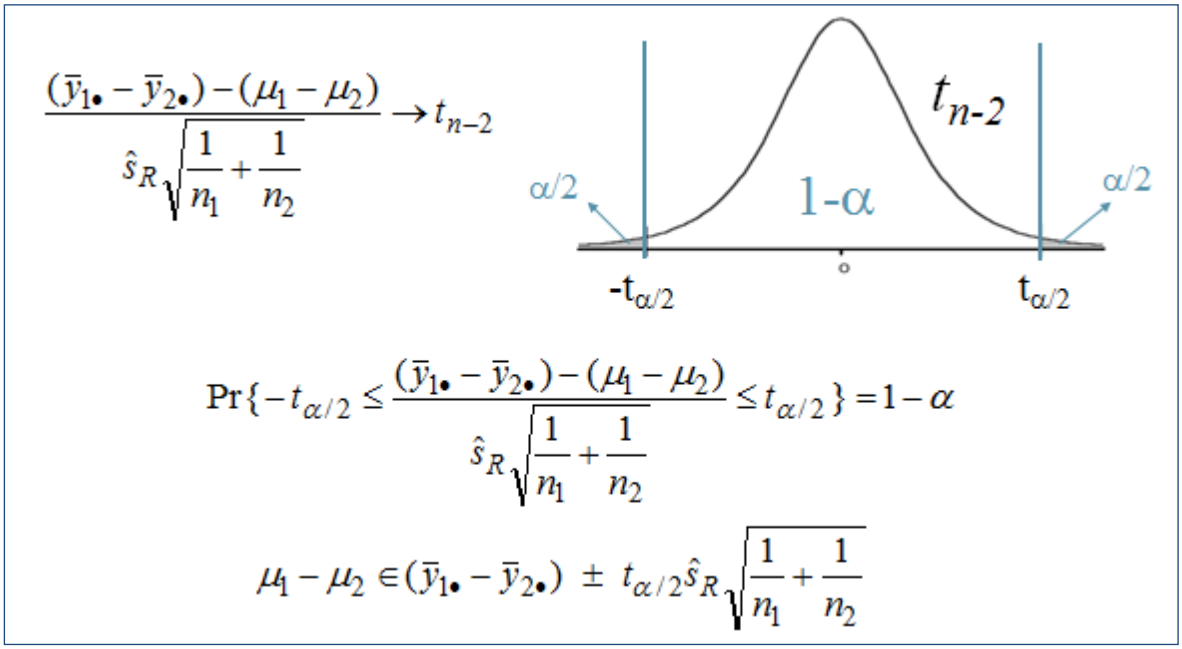


FIGURA 4.12 INTERVALO DE CONFIANZA PARA LA DIFERENCIA DE MEDIAS.

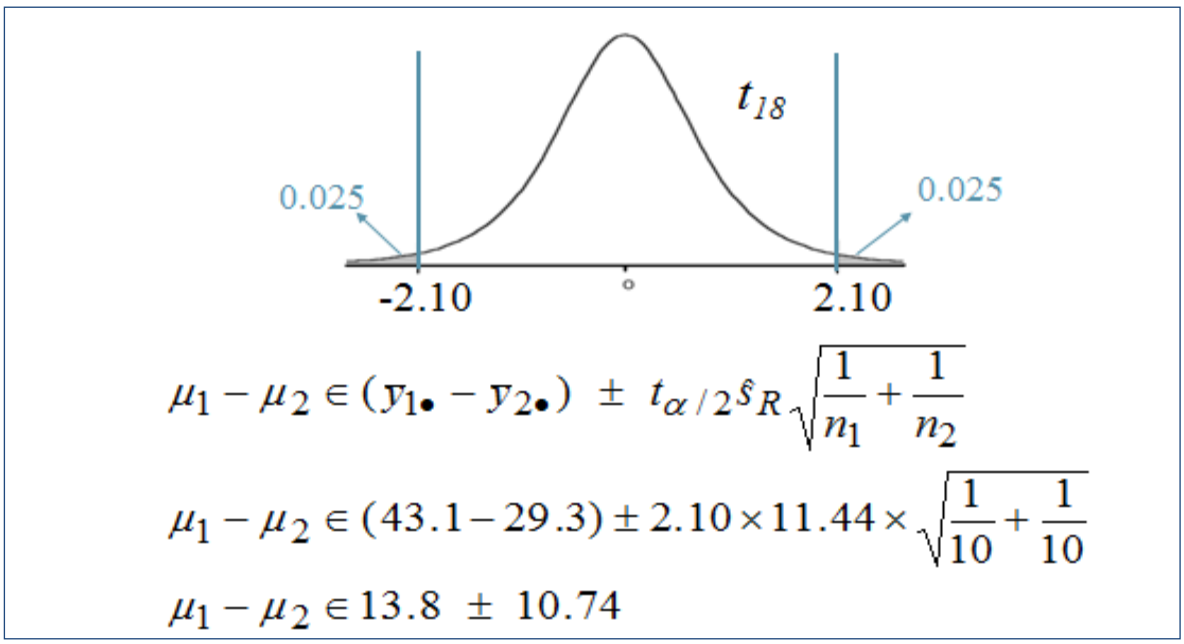


FIGURA 4.13 EJEMPLO DE INTERVALO DE CONFIANZA PARA LA DIFERENCIA DE MEDIAS

4.5.1 *CONTRASTES DE HIPÓTESIS PARA LAS MEDIAS*

Para realizar contrastes de hipótesis sobre las medias de la normal también se puede utilizar la función `t.test(x,mu,alt)` añadiendo como argumentos adicionales la hipótesis nula H_0 con mu= μ_0 y la hipótesis alternativa H_1 con `alt="two.sided"` ($\mu \neq \mu_0$), `alt="less"` ($\mu < \mu_0$), ó `alt="greater"` ($\mu > \mu_0$).

La velocidad de la luz en el vacío se conoce hoy con más precisión y se tiene un valor de 299.792x10³km/s. Vamos a contrastar con α=0.05 si la media obtenida por Michelson o por Newcomb son significativamente distintas a ese valor. Para ello hay que realizar el siguiente contraste

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

siendo la distribución de la media muestral $\frac{\bar{x} - \mu_0}{s/\sqrt{n}} \rightarrow t_{n-1}$ bajo la hipótesis nula.

```
> mu0=299.792
> t.test(Michelson,mu=mu0,alt="two.sided",conf.level=0.95)
One Sample t-test

data:  Michelson
t = 7.6445, df = 99, p-value = 1.374e-11
alternative hypothesis: true mean is not equal to 299.792
95 percent confidence interval:
299.8367 299.8681
```

```
sample estimates:
mean of x
299.8524
```

De toda esta información ahora nos interesan las líneas que están subrayadas. La hipótesis nula se rechaza y la conclusión es que la media de los datos es significativamente distinta a 299.792. Esto queda reflejado por el p-valor del contraste que es 1.374e-11. Si repito lo mismo para los datos de Newcomb también rechazaría la hipótesis nula

```
> t.test(Newcomb,mu=mu0,alt="two.sided",conf.level=0.95)
data:  Newcomb
t = -107.8214, df = 65, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 299.792
```

Para la diferencia de medias también se pueden hacer contrastes de hipótesis con la función t.test, por ejemplo si quisiéramos realizar el siguiente contraste de ver si la diferencia entre las medidas de Michelson y Newcomb son mayores de 1.76 unidades

$$\begin{cases} H_0 : \mu_x - \mu_y = d_0 \\ H_1 : \mu_x - \mu_y > d_0 \end{cases}$$

tendríamos que escribir:

```
> t.test(Michelson,Newcomb,mu=1.76,alt="greater",var.equal=FALSE,
+ conf.level=0.95)
Welch Two Sample t-test

data:  Michelson and Newcomb
t = 1.1898, df = 96.943, p-value = 0.1185
alternative hypothesis: true difference in means is greater than
1.76
95 percent confidence interval:
1.751613      Inf
sample estimates:
mean of x mean of y
299.8524  298.0712
```

4.6. COMPARACIÓN DE VARIANZAS

En el contraste de comparación de medias hemos utilizado la hipótesis de que las dos muestras provienen de distribuciones normales con la misma varianza. Para comprobar que esta hipótesis de igualdad de varianzas (también denominada hipótesis de homocedasticidad) se realiza un contraste de igualdad de varianzas. El problema se plantea en la figura xx

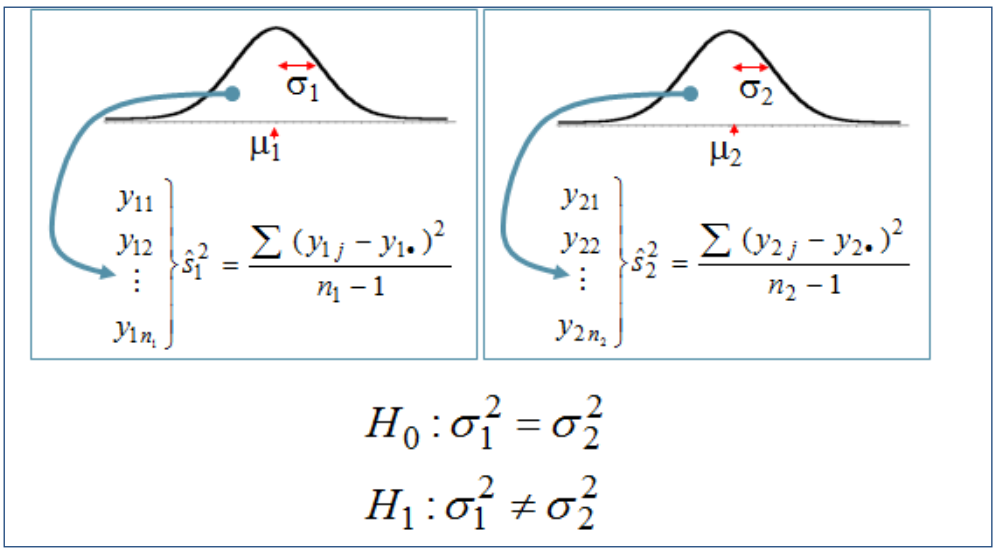


FIGURA 4.14 CONTRASTE DE IGUALDAD DE VARIANZAS

El procedimiento es muy sencillo, se calculan las varianzas de cada muestra. Si provienen de distribuciones con la misma varianza, las dos estimaciones deben ser parecidas y su cociente cercano a 1. Lógicamente los valores muestrales no van a ser idénticos, tenemos que estar dispuestos a aceptar ciertas diferencias. Los límites máximos y mínimos se obtienen mediante la distribución F con grados de libertad n_1-1 y n_2-1, siendo n_1 el número de datos de la varianza que ponemos en el numerador y n_2 el número de datos de la varianza que ponemos en el denominado. En nuestro ejemplo $n_1 = n_2 = 10$, por tanto la distribución de referencia es la F con 9 gl en el numerador y 9 gl en el denominador. El único dato que necesitamos es el nivel de confianza α con el que queremos hacer el contraste.

```
> var.test(ya,yb)
F test to compare two variances
data:  ya and yb
F = 1.3441, num df = 9, denom df = 9, p-value = 0.6667
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
0.3338537 5.4113109
sample estimates:
ratio of variances
1.344093
```

Los cálculos del contraste se ilustran en la figura X.X. El cociente de las varianzas muestrales es 1.34 y se encuentra entre los límites 0.248 y 4.03. Estos valores se pueden obtener en R con la función `qf()`

```
> qf(0.025,9,9)
[1] 0.2483859
> qf(0.975,9,9)
[1] 4.025994
```

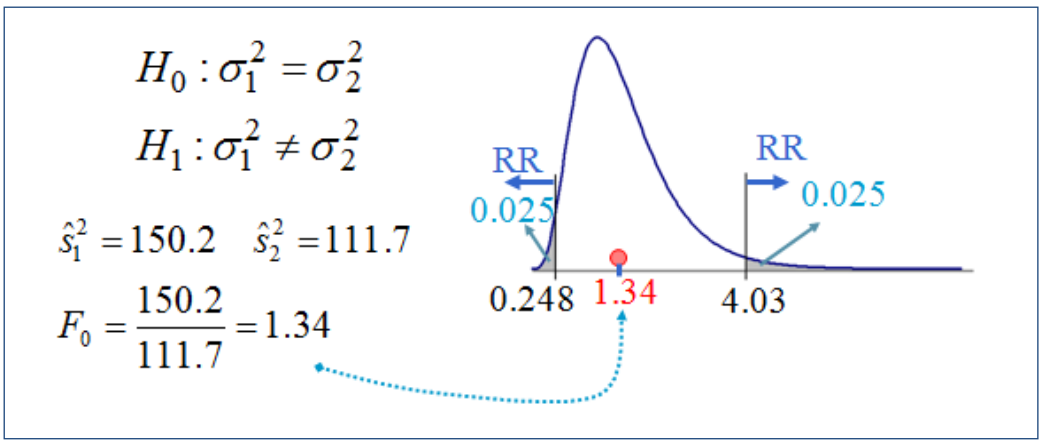


FIGURA 4.15 CONTRASTE DE VARIANZAS PARA LOS DATOS DE LAS DIETAS

El procedimiento var.test() también nos proporciona el intervalo para el cociente de varianzas. Las fórmulas necesarias se encuentran en la figura X.X.

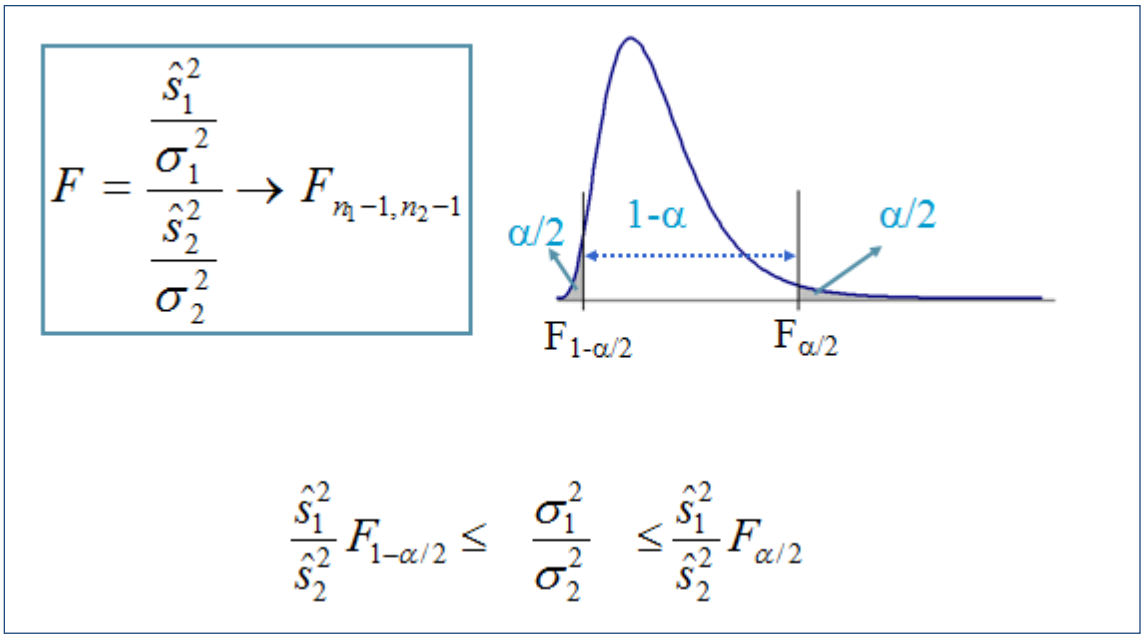


FIGURA 4.16 INTERVALO DE CONFIANZA PARA EL COCIENTE DE VARIANZAS.

4.7. CONTRASTES DE BONDAD DE AJUSTE

Por último vamos a comprobar si la hipótesis de normalidad que hemos utilizado en todo el ejercicio es cierta, para ello vamos a proceder de 2 formas, la primera gráfica, en la que mostraremos distintos gráficos que nos pueden ayudar a ver la semejanza y por último el contraste de bondad de ajuste.

1) Gráfico del histograma de los datos frente a la función de densidad de probabilidad

```
> X=Michelson
> hist(X,freq=F)
> x=seq(min(X),max(X),(max(X)-min(X))/100)
> lines(x,dnorm(x,mean=Mu_M,sd=Sigma_M),col="blue")
```

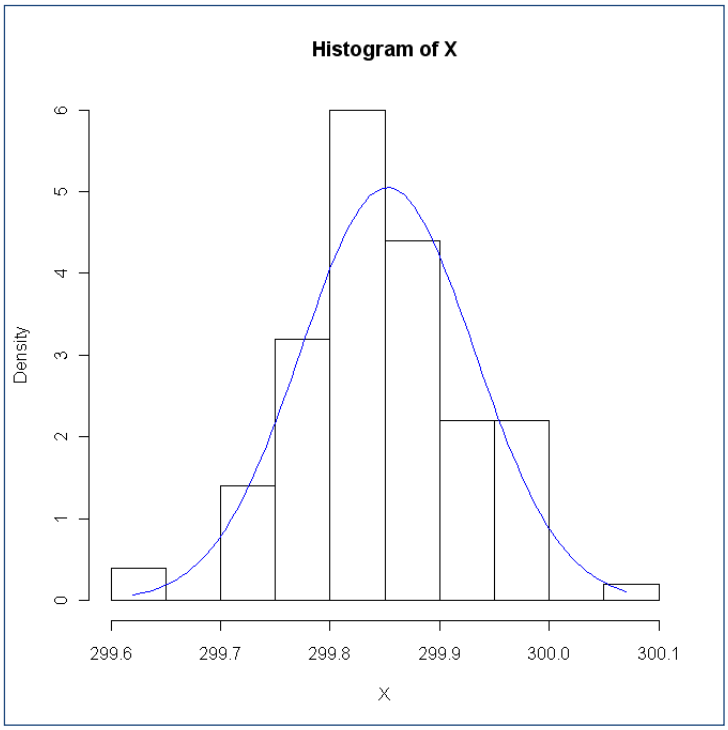


FIGURA 4-1. HISTOGRAMA DE VELOCIDAD DEL EXPERIMENTO DE MICHELSON

Aunque la hipótesis de que siga una distribución normal no tiene mala pinta, necesitamos nuevas y mejores herramientas para justificar que sigue esta distribución.

2) Gráfico de probabilidad acumulada (Gráfico probabilística Normal)

Para dibujar el gráfico probabilístico normal, definimos la variable "freq.obs" donde almacenamos las probabilidades empíricas acumuladas. La función *sort* ordena nuestros datos en orden creciente.

```
> X<-sort(X);
> freq.obs<-1/n*seq(0.5,n-0.5,1);
> plot(X,freq.obs,main="Probabilidad acumulada",xlab="X datos",
+ ylab="Probabilidad");
> lines(x,pnorm(x,mean=Mu_M,sd=Sigma_M),col="blue");
```

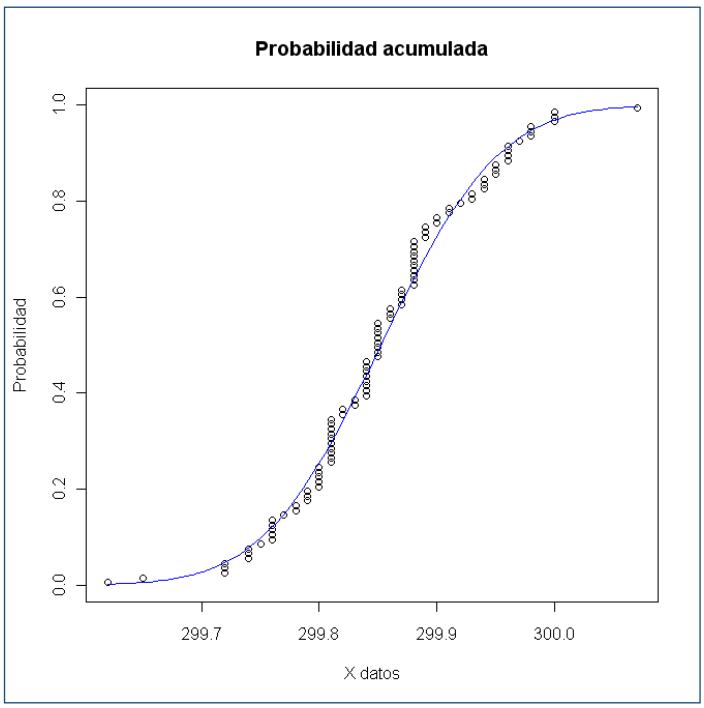


FIGURA 4.17. GRÁFICO PROBABILÍSTICO NORMAL DE VELOCIDAD DE MICHELSON

3) Gráfico Q-Q o Cuantil-Cuantil, utilizaremos dos funciones: qqnorm(x) y qqline(x)

```
> X.teorica=qnorm(freq.obs,mean=Mu_M,sd=Sigma_M);
> plot(X.teorica,X,main="Q-Q plot",xlab="X teorica",ylab="X datos");
> abline(0,1);
```

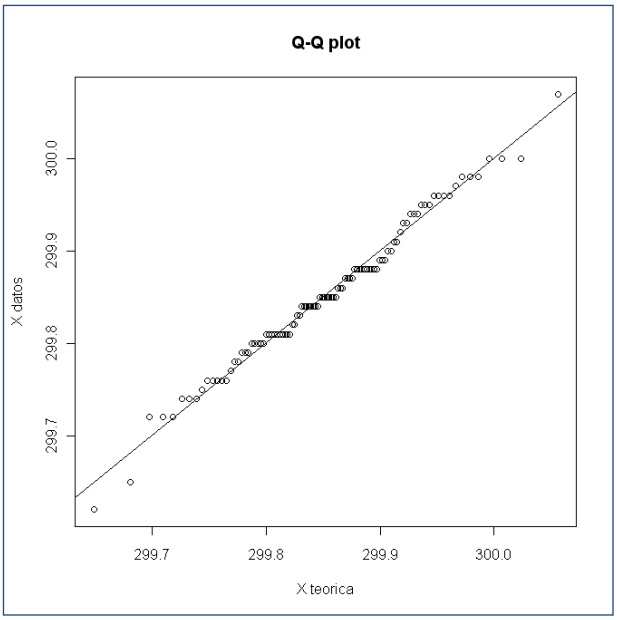


FIGURA 4.18. GRÁFICO Q-Q PLOT DE VELOCIDAD DE MICHELSON

Estos 3 gráficos son interesantes en los temas posteriores y por lo tanto todas las instrucciones van a quedar recogidas en la función qqplot.R que admite ajustes de las funciones de distribución Normal, Poisson, Binomial y Exponencial.

```
> source,"qqplot")
> qqplot(X,"norm")
```

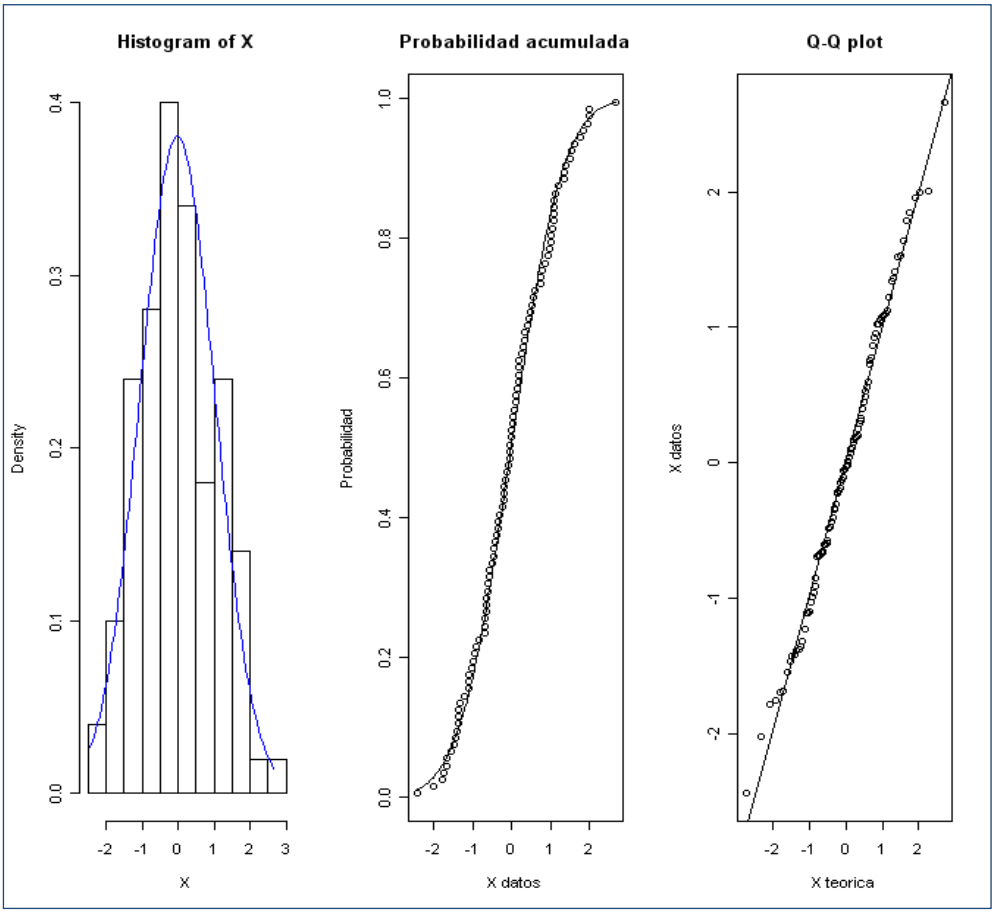


FIGURA 4.19. GRÁFICOS DE BONDAD DE AJUSTE PARA LA NORMAL

4) Contraste de Bondad de ajuste χ^2

El contraste χ^2 se basa en que la frecuencia observada de los datos y la frecuencia esperada siguiendo la distribución de probabilidad a la que se quieran ajustar los datos debe ser muy pequeña, en concreto:

$$\sum_{i=1}^{K} \frac{(E_i - O_i)^2}{E_i} \rightarrow \chi^2_{K-r-1}$$

donde K es el numero de clases en las que se organizan, r es el número de parámetros del modelo que han sido estimados con los datos, Oi es la frecuencia observada y Ei es la frecuencia observada siguiendo el modelo de probabilidad. Luego tenemos que ir calculando cada sumando.

```
> n<-length(X)
> K<-round(sqrt(n))  # Suponemos K aprox sqrt(n)
> r<-2  # en la normal se estiman 2 parámetros media y varianza
> Dx<-(max(X)-min(X))/K    # Anchura de cada clase
> xi=array(dim=K+1)
```

```
> xi[1]=-Inf
> for(i in 2:K) xi[i]=min(X)+(i-1)*Dx
> xi[K+1]=Inf
> Ei=array(dim=K)
> for (i in 1:K)
Ei[i]=n*diff(pnorm(xi[i:(i+1)],mean=mean(X),sd=sd(X)))
> Oi=array(dim=K)
> for (i in 1:K) Oi[i]=sum((X>xi[i])& (X<=xi[i+1]))
> chi2=sum((Ei-Oi)^2/Ei)
> chi2
[1] 14.25385
> pvalue=pchisq(chi2,df=K-r-1,lower.tail=FALSE)
> pvalue
[1] 0.04684626
```

Viendo el p-valor se demuestra que si consideraramos α=0.05 tendríamos que rechazar la hipótesis de Normalidad, mientras que si α=0.01 aceptaríamos la hipótesis.

Como el contraste χ^2 de bondad de ajuste lo vamos a utilizar mucho vamos a crear la función Chi2test.R que permita realizar el contraste de bondad de ajuste para la distribución Normal, Poisson, Binomial y Exponencial.

```
> source("Chi2test.R")
> pvalue=Chi2test(X,"norm")
-------------------------------------
El p-valor del contraste es: 0.046846259578708
-------------------------------------
```

Otro contraste de bondad de ajuste que se puede utilizar el contraste de Kolmogoroz-Smirnov, aunque este es solo válido para distribuciones normales y suele ser bastante bondadoso, es decir, tiende a aceptar la hipótesis nula de normalidad. Este contraste tiene la ventaja de estar programado en R y su sintaxis es la siguiente:

```
> ks.test(X,"pnorm",mean=mean(X),sd=sd(X))

One-sample Kolmogorov-Smirnov test
data:  vel
D = 0.0834, p-value = 0.4896
alternative hypothesis: two-sided
```

4.8. Aplicación para la Binomial

Se ha registrado el número de niñas en familias de 12 hijos nacidas entre 1879 y 1936 para unas comunidades de granjeros que habitaban en los Estados Unidos de Norteamérica y Canadá. Los datos se encuentran en el archivo "demog.txt".

Cargamos los datos:

```
> demog<-read.table("demog.txt",header=T)
```

Hay 103 datos ordenados en orden creciente. En primer lugar debemos de preguntarnos qué tipo de distribución siguen estos datos. Llamando a nuestra variable aleatoria X=”número de niñas en familias de 12 hijas”, está claro que los datos deberían ajustarse a una distribución binomial de probabilidad de niña p y número n=12. El único parámetro a estimar de esta distribución de datos es $\hat{p}$ que utilizando el método de máxima verosimilitud es:

```
>sum(demog$hijas)/(12*length(demog$hijas))
```

```
[1] 0.4846278
```

Vamos a realizar el test de bondad de ajuste a una distribución Binomial para ello utilizamos las funciones vistas en el primer apartado adaptadas a una distribución de datos binomial.

```
> qqplot(demog$hijas,"binom",12)
```

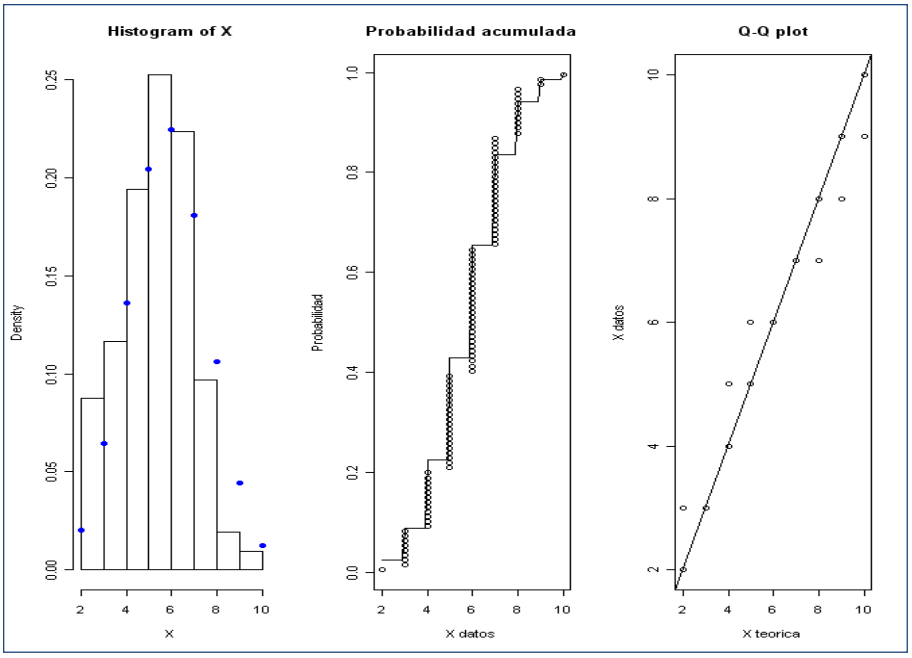


FIGURA 4.20. BONDAD DE AJUSTE PARA LA BINOMIAL

Parece que el ajuste mirando los gráficos es bastante bueno, finalmente vamos a realizar el contraste de bondad de ajuste χ^2 ,

```
> source("Chi2test.R")
> Chi2test(demog$hijas,"binom",12)
-------------------------------------
El p-valor del contraste es: 0.751086713644019
-------------------------------------
```

A la vista del pvarlor=0.75>α anterior, aceptamos la hipótesis de que los datos se distribuyen siguiendo una distribución Binomial. Para las distribuciones discretas como la Poisson, Binomial, y Binomial negativa la frecuencia Esperada y observada puede calcularse con la función goodfit() que para usarla hay que cargar previamente el paquete vcd.

```
> gf=goodfit(demog$hijas,"binomial","MinChisq",par=list(size=12))
> gf
Observed and fitted values for binomial distribution
with parameters estimated by `MinChisq'

 count observed      fitted
     0        0  0.03549453
     1        0  0.40181529
     2        1  2.08484081
     3        8  6.55594638
     4       12 13.91559111
     5       20 21.00416348
     6       26 23.11723711
     7       23 18.69273659
     8       10 11.02139713
     9        2  4.62102036
    10        1  1.30780467
    11        0  0.22431791
    12        0  0.01763463

> summary(gf)
Goodness-of-fit test for binomial distribution

X^2 df  P(> X^2)
Pearson 4.879252 11 0.9368729
Mensajes de aviso perdidos
In summary.goodfit(gf) : Chi-squared approximation may be
incorrect
```

A continuación vamos a realizar un contraste de hipótesis para ver si la probabilidad de tener hijas es más baja que la de tener hijos. Eso se traduce en el siguiente contraste:

$$\begin{cases} H_0 : p = p_0 \\ H_1 : p < p_0 \end{cases}$$

Utilizaremos para resolverlo la función binom.test(Xs,n,p,alt,conf.level) que realiza el contraste de hipótesis y construye intervalos de confianza para una serie de n datos de tipo Bernoulli donde Xs<n han salido favorables. P es la probabilidad de la hipótesis nula, alt representa la hipótesis alternativa y todo ello con un nivel de confianza marcado por conf.level

```
>
binom.test(sum(demog$hijas),n=12*length(demog$hijas),alt="greater"
,+ conf.level=0.95)
Exact binomial test
data:  sum(demog$hijas) and 12 * length(demog$hijas)
number of successes = 599, number of trials = 1236, p-value = 0.1463
alternative hypothesis: true probability of success is less than 0.5
95 percent confidence interval:
 0.0000000 0.5084233
sample estimates:
probability of success
0.4846278
```

Analizando el pvalor=0.1463 del contraste se aceptaría la hipótesis nula de diría: no hay evidencias estadísticas de que la probabilidad de nacimiento de niña sea más alta que la de varones.

4.9. Aplicación para la Poisson

Rutheford y Geiger registraron el número de centelleos en 2608 intervalos de 72 segundos de duración para una fuente de polonio. Los datos se encuentran en el archivo desint.txt.

Cargamos los datos:

```
> desint<-read.table("desint.txt",header=T)
> X=desint$numero
```

Si miramos los datos, vemos que existen 2608 experimentos donde la variable aleatoria es X: número de centelleos en 72 segundos. Una variable de Poisson cuyo parámetro $\hat{\lambda}$ puede ser estimado por máxima verosimilitud como

```
> mean(X)
[1] 3.871549
```

Vamos a ver gráficamente si el ajuste a una distribución de Poisson es bueno

```
> qqplot(X,"pois",12)
```

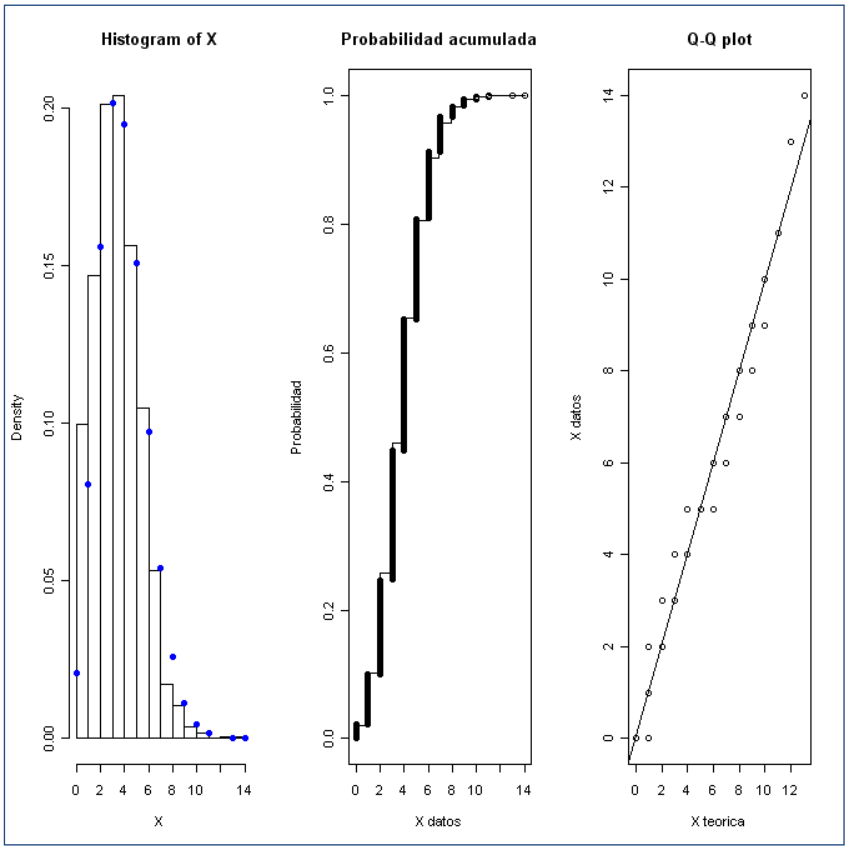


FIGURA 4.21 BONDAD DE AJUSTE PARA LA POISSON

Parece que el ajuste mirando los gráficos es bastante bueno, finalmente vamos a realizar el contraste de bondad de ajuste χ^2 ,

```
> source("Chi2test.R")
> Chi2test(X,"pois",12)
-------------------------------------
El p-valor del contraste es: 0.084986702772764
-------------------------------------
```

A la vista del pvalor=0.085>α anterior, aceptamos la hipótesis de que los datos se distribuyen siguiendo una distribución Poisson. A la misma conclusión llegamos utilizando la función goodfit() del paquete vcd.

```
> gf=goodfit(X,"poisson","MinChisq")
> summary(gf)
Goodness-of-fit test for poisson distribution

X^2 df  P(> X^2)
Pearson 20.14036 13 0.0917652
Mensajes de aviso perdidos
In summary.goodfit(gf) : Chi-squared approximation may be incorrect
```

A continuación vamos a realizar un intervalo de confianza y un contraste de hipótesis sobre el parámetro λ.

$$\frac{\hat{\lambda}-\lambda}{\sqrt{\lambda/n}} \to N(0,1)$$

Aunque no es necesario hacerlo manualmente ya que tenemos la función poisson.test(nX, n, r, alt, conf.level), que realiza el contraste de hipótesis y construye intervalos de confianza para un número de eventos nX que siguen una distribución de Poisson, n es el número de intervalos de tiempo de cada medida, r es el valor de la hipótesis nula, alt representa la hipótesis alternativa y todo ello con un nivel de confianza marcado por conf.level.

Un intervalo de confianza para λ del 95% se calcularía:

```
> poisson.test(sum(X),length(X),conf.level=0.95)
Exact Poisson test
data:  sum(X) time base: length(X)
number of events = 10097, time base = 2608, p-value <2.2e-16
alternative hypothesis: true event rate is not equal to 1
95 percent confidence interval:
 3.796397 3.947814
sample estimates:
event rate
3.871549
```

¿Se puede asegurar que la media de centelleos es mayor de 3 desintegraciones en 1 hora? Para responder a esta pregunta hay que hacer un contraste de hipótesis, (pero cuidado con las unidades, ahora λ tiene unidades desint/hora), así que habría que hacer el siguiente contraste:

$$\begin{cases} H_0 : \lambda = \lambda_0 \\ H_1 : \lambda > \lambda_0 \end{cases}$$

poniendo adecuadamente la unidad de tiempos

```
>
poisson.test(sum(X),length(X)*72/60,r=3,alt="greater",conf.level=0
.95)

Exact Poisson test


data:  sum(X) time base: length(X) * 72/60
number of events = 10097, time base = 3129.6, p-value =2.668e-13
alternative hypothesis: true event rate is greater than 3
95 percent confidence interval:
 3.173661      Inf
sample estimates:
```

```
event rate
3.226291
```

Se rechaza la hipótesis nula, hay evidencias estadísticas de que λ es significativamente mayor que 3.

4.10. APLICACIÓN PARA LA EXPONENCIAL

En el archivo "seísmo.txt" se recogió el tiempo transcurrido entre los sucesivos sismos acaecidos en el mundo con intensidad superior o igual a 7,5 en la escala de Richter o que causaron al menos 1000 víctimas mortales. Estos sismos acaecieron entre el 16 de Diciembre de 1902 y el 4 de Marzo de 1977. A la luz del estudio realizado entonces, trate de ver el modelo a que se ajustan aproximadamente estos datos y estime sus parámetros.

En primer lugar vamos a cargar los datos

```
> seismo<-read.table("seismo.txt",header=T)
> attach(seismo)
```

Son 62 datos que informan de los días que transcurrieron entre dos terremoto, luego la variable aleatoria a la que más pueden ajustarse los datos es una exponencial. El parámetro de la exponencial es $\hat{\lambda}=1/\bar{x}$:

```
> 1/mean(dias)
[1] 0.002287232
```

Luego $\hat{\lambda} = 0.002287 terremotos / dia$. Si vemos gráficamente si el ajuste es bueno a una variable aleatoria exponencial.

```
> qqplot(dias,"exp")
```

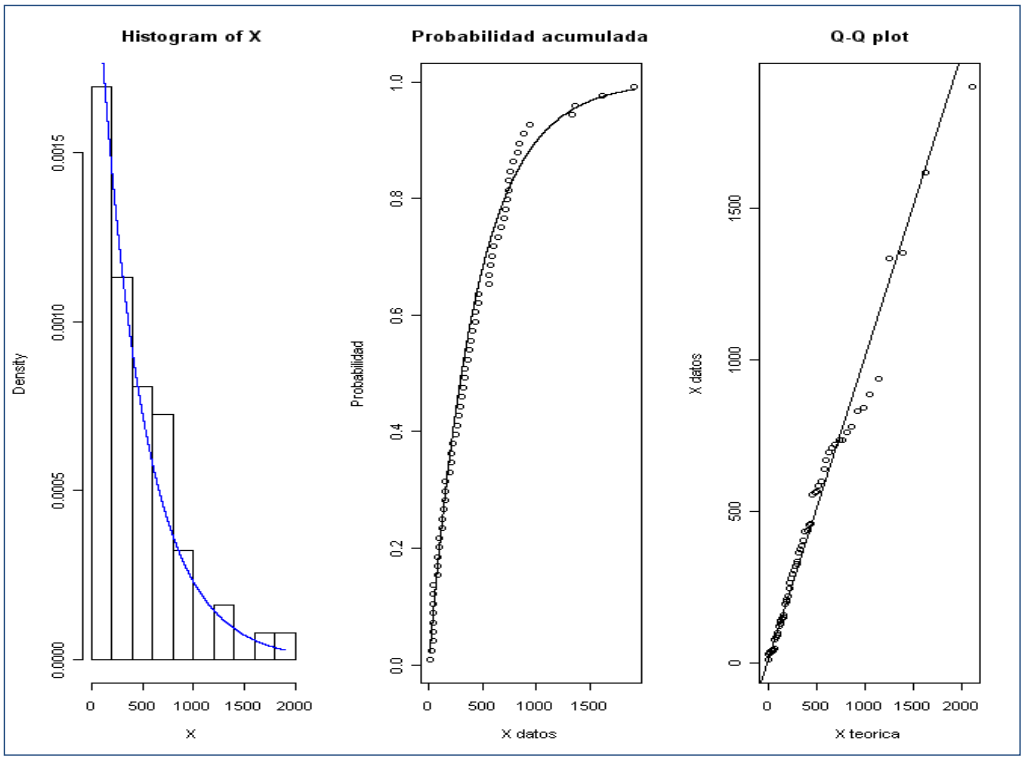


FIGURA 4.22. HISTOGRAMA DE LA VARIABLE DIAS ENTRE TERREMOTOS

El contraste de bondad de ajunste χ^2 también da un resultado favorable al ajuste

```
> Chi2test(dias,"exp")

--------------------------------------
El p-valor del contraste es: 0.355706425319352
--------------------------------------
```

Para realizar contrastes de hipótesis e intervalos de confianza sobre el parámetro λ hay que utilizar la función poisson.test igual que antes. Lo único que hay que tener precaución en escribir los argumentos de la función. Con un nivel de confianza del 90% el intervalo de confianza para λ se calcularía:

```
> poisson.test(length(dias),sum(dias),conf.level=0.90)

Exact Poisson test
data:  length(dias) time base: sum(dias)
number of events = 62, time base = 27107, p-value < 2.2e-16
alternative hypothesis: true event rate is not equal to 1
90 percent confidence interval:
0.001831310 0.002825800
sample estimates:
event rate
0.002287232
```

Se observa que el valor estimado es igual que antes $\hat{\lambda} = 0.002287 terremotos / dia$ y el intervalo de confianza del 90% sería:

$0.001831 < \lambda < 0.002826\ terremotos / dia$

4.11. INSTRUCCIONES UTILIZADAS

`sort(x)`	ordena los datos de la variable x
`length(x)`	da el número de datos de la variable x
`t.test(x,mu=mean(x), alternative"…", conf.level=(1-α)*100)`	nos da el intervalo de confianza de la media de una distribución normal de la cual no conocemos su desviación típica (T-Student). Además da el estadístico t que se utiliza para las hipótesis referidas a la distribución normal. El argumento alternative sirve para distinguir el tipo de hipótesis que queramos hacer: H_1: $\mu<\mu_0$ se utiliza "less"; H_1: $\mu>\mu_0$ se utiliza "greater"; H_1: $\mu\neq\mu_0$ se utiliza "two.sided". (1-α)*100 suele ser normalmente 95 o 99
`binom.test (X, n, p = 0.5, alternative = "…", conf.level = 0.95)`	nos realiza un informe sobre la adaptación de los datos según una distribución binomial. Para ello nos da un p-valor que informa sobre si es correcto tomar los datos como una binomial y una estimación del parámetro de la distribución: estimación de la probabilidad de éxito. X es el número de éxitos en una población n datos, mientras que p es la probabilidad real de los datos (no estimada). Si no se escribe el tercer argumento, R asume que p=0.5; igualmente si no escribimos el nivel de confianza asumirá que es del 95%. Por último, si queremos evaluar hipótesis podemos utilizar el argumento alernative= "two.sided", "less", "greater"
`poisson.test(x,n)`	nos realiza un informe sobre la adaptación de los datos según una distribución de Poisson. Para ello nos da una estimación del parámetro de la distribución. X es el número de sucesos en total (sum(vector)), mientras que n es el número de observaciones (length(vector)). Si no escribimos el nivel de confianza asumirá que es del 95%. Por último, si queremos evaluar hipótesis podemos utilizar el argumento alernative= "two.sided", "less", "greater".
`goodfit(X,dist_prob,metodo)`	Función que calcula la frecuencia observada y esperada de un vector de datos X que siguen un modelo de probabilidad "discreto"dado por dist_prob("binomial","poisson",...). Para utilizarlo

	hay que cargar el paquete vcd. Si se quiere realizar el contraste χ^2, después de utilizar esta función hay que llamar a summary(gf)
`pt(t,df=N-1)`	nos da el p-valor de un contrate de hipótesis de la T-Student, en el que la hipótesis alternativa hace referencia a que la media sea menor que la media de la hipótesis nula (H_1: $\mu<\mu_0$). Lo realiza sobre el estadístico t, siendo N el número de datos total de la muestra. Si le restamos 1 a este p-valor obtenemos el resultado de haber hecho el contraste para una hipótesis alternativa que considerara la media mayor que la media de la hipótesis nula (H_1: $\mu>\mu_0$). Para H_1: $\mu\neq\mu_0$ se puede obtener `p-valor=2*(pt(t,N-1)-1)`
`qt(α, df=N-1)`	nos da la $t_{N-1,\alpha/2}$, siendo N el número de datos total de la muestra
`qbinom( p,n,^p)`	nos da el número de éxitos que hay en una distribución binomial de probabilidad real p, en n datos y con una probabilidad estimada de ^p

5. DISEÑO DE EXPERIMENTOS

5.1. INTRODUCCIÓN

El objetivo de este tema es estudiar el efecto de un conjunto de factores sobre una variable respuesta. En este capítulo, en primer lugar estudiaremos modelos con un factor. Seguidamente, estudiaremos el modelo con dos factores con interacción y con bloques aleatorizados. Finalmente, estudiaremos los casos con tres factores: sin interacción, con replicaciones y cuadrado latino.

5.2. DISEÑO DE EXPERIMENTOS: UN FACTOR

En un laboratorio agrícula se lleva a cabo un estudio sobre la cantidad de grano producido por plantas de centeno. Se observa que el rendimiento de las plantas puede venir influenciado por el tipo de semilla utilizada.

Concretamente, los investigadores quieren estudiar cuatro tipos de semillas: A, B, C y D. Para ello, escogen de sus graneros aleatoriamente seis semillas de cada uno de los tipos. En la tabla siguiente se muestra el rendimiento obtenido por cada una de las semillas.

A

Sem	Rend
A	229.1
A	253.7
A	241.3
A	254.7
A	237.2
A	241.3

B

Sem	Rend
B	233.4
B	233.0
B	219.2
B	200.0
B	224.3
B	202.0

C

Sem	Rend
C	211.1
C	223.1
C	217.5
C	211.8
C	207.6
C	213.7

D

Sem	Rend
D	270.4
D	248.6
D	230.0
D	250.7
D	230.0
D	245.8

A continuación mostramos cómo se analizan estos resultados empleando el software R:

En primer lugar cargamos los datos situados en el fichero "centeno.txt", mediante el comando "read.table".

```
> centeno <- read.table('Centeno.txt', header = T)
```

Observación: en la cabecera del archivo "centeno.txt" se encuentra el nombre de las variables, por eso colocamos *header = TRUE*.

Seguidamente, comprobamos los nombres de la estructura de datos "centeno", y utilizamos el comando *attach* para acceder a los campos de manera más cómoda:

```
> names(centeno)
> attach(centeno)
```

Se observará que la estructura de datos "centeno" tiene los variables (columnas): "Rend" y "Sem".

El modelo matemático con el que trabajaremos es el siguiente:

$$y_{ij} = \mu_i + u_i \quad con\, u_i \rightarrow N(0, \sigma^2)$$

y la estimación por máxima verosimilitud de los parámetros del modelo se calcula como:

$$\hat{\mu}_i = \overline{y}_{i\bullet} \text{ y } \hat{\sigma}^2 = \hat{s}_R^2 = \frac{VNE}{gl}$$

El valor de estos parámetros los podemos obtener después de hacer la tabla ADEVA. Para construir la tabla de análisis de la varianza (tabla ADEVA en castellano, ANOVA en inglés), en primer lugar creamos el modelo con el comando *aov.*

```
> mod <- aov(Rend ~ Sem)
```

Con la instrucción anterior hemos creado un objeto que se denomina "mod" que contiene el modelo que emplearemos en este problema, donde la variable dependiente "Rend" se explica a través del factor "Sem".

Para mostrar la tabla ADEVA, utilizaremos la función *anova*, obteniendo los siguientes resultados:

```
> anova(mod)
Analysis of Variance Table

Response: Rend
          Df Sum Sq Mean Sq F value    Pr(>F)
Sem        3 4795.6 1598.53  11.217 0.0001559 ***
Residuals 20 2850.1  142.51
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Las medias para cada tratamiento se obtiene mediante el comando *model.tables*, con el segundo argumento *means*. Es decir:

```
> model.tables(mod, type= 'means')
Tables of means
Grand mean

230.3958

 Sem
Sem
     A      B      C      D
242.88 218.65 214.13 245.92
```

Se observa que la media de cada uno de los tratamientos es 242.88, 218.65, 214.13 y 245.92, para los tratamientos A, B, C, y D, respectivamente. La media de todos los tratamientos es 230.3958.

De la tabla ADEVA se observa que el valor de la varianza residual $\hat{S}_R^2$ es 142.51.

a) Contrastar la hipótesis de que con cualquiera de los tres tipos de semilla se obtiene el mismo rendimiento. (α=0.01)

H_0: $\mu_A = \mu_B = \mu_C = \mu_D$

H_1: algún μ_i es distinto

De la tabla ANOVA obtenemos que el contraste de la F para 3 grados de libertad en el numerador y 20 grados de libertad en el numerador da un p-valor de 0.00016 correspondiente a un F_0=11.21. Por lo tanto, como el p-valor es menor que el $\alpha = 0.01$ considerado, concluimos que existen diferencias significativas entre las medias de las semillas. Es decir, alguna semilla proporciona un rendimiento distinto a las demás.

b) Identifique qué semilla proporciona un rendimiento distinto.

Para hacer las comparaciones dos a dos existe la función *pairwise.t.test* que podemos usar de la siguiente manera:

```
> pairwise.t.test(Rend, Sem,p.adj="none")
        Pairwise comparisons using t tests with pooled SD

data:  Rend and Sem

  A       B       C
B 0.00217 -       -
C 0.00047 0.51972 -
D 0.66458 0.00078 0.00017

P value adjustment method: none
```

La tabla me proporciona el p_valor de los contrastes individuales

H_0: $\mu_i = \mu_j$

H_1: $\mu_i \neq \mu_j$

para todas las comparaciones posibles.

Por lo tanto tenemos que hay diferencias significativas entre las semillas A y B (p-valor=0.002), A y C (p-valor=0.0005), B y D, y C y D. Sin embargo, no hay diferencias significativas entre las semillas B y C, y A y D, ya que sus p-valores son mayores que el nivel de significación seleccionado α=0.01.

Los contrastes dos a dos pueden realizarse por varios métodos, como el método Holm (por defecto), Bonferroni, Hommel, etc. Para ver toda la información relativa a esta función ver la ayuda *help("pairwise.t.test")*.

c) Mostrar un intervalo de confianza para la media del rendimiento para cada uno de las semillas con α=0.05.

Para ver esta información de forma gráfica hay que usar los siguientes comandos:

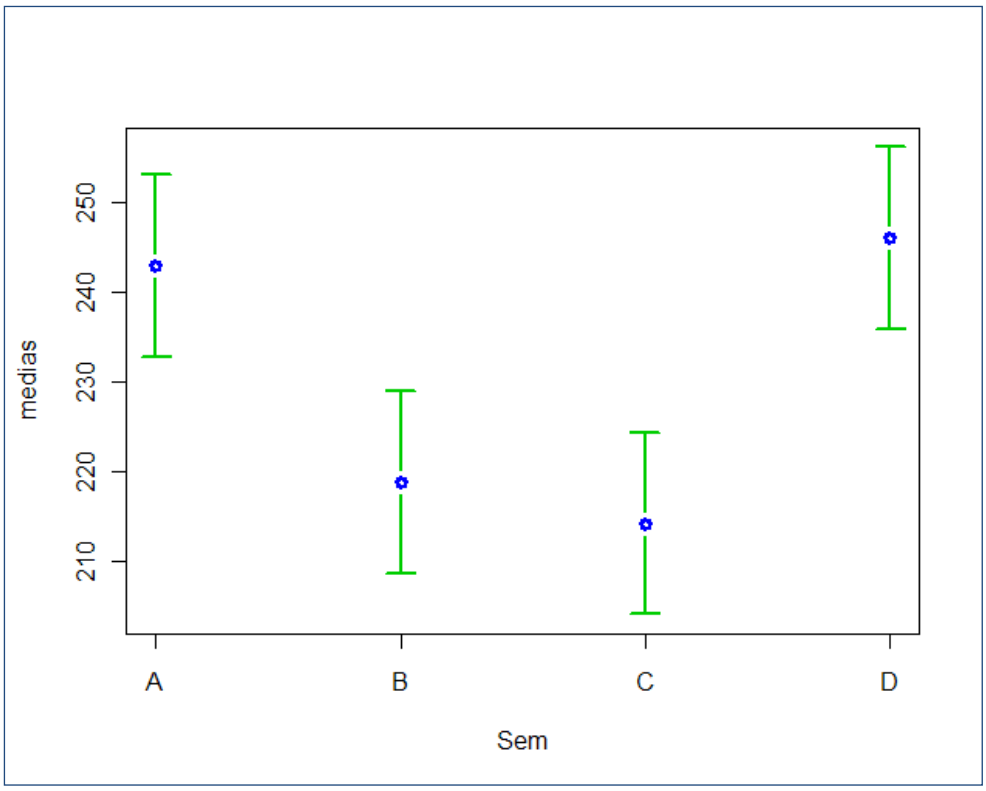
```
> source('ICplot.R')
> ICplot(mod, 'Sem')
```

El programa ICplot es una función desarrollada por los autores de esta documentación. La puedes encontrar al final del documento.

Importante: Para que esta función esté operativa, se utiliza la instrucción:

```
source('ICplot.R')
```

Por tanto, utilizando la función ICplot, se obtiene el siguiente gráfico:



Como podemos ver de las instrucciones, ha sido necesaria cargar la función "ICplot". Los argumentos que se requieren para esta función es el nombre de la estructura de datos del modelo, y en el segundo argumento el nombre del factor. Se puede incluir un tercer argumento (opcional) con el valor del nivel de confianza deseado. Por defecto, se emplea α=0.05.

Si quisiéramos emplear α=0.01, llamaríamos a la función de la siguiente manera.

```
> ICplot(mod, 'Sem',.01)
```

El gráfico muestra claramente que los intervalos de confianza de las semillas A y D se solapan, con lo que concluimos que estas semillas no muestran diferencias significativas entre sí. Lo mismo ocurre con B y C. Sin embargo, la comparación de A y C indica que las medias son diferentes, al no existir solapamiento. Lo mismo ocurre en las comparaciones A y C, By D, C y D.

De forma resumida podemos decir que no existen diferencias significativas entre A y D, ni entre B y C, y que el resto de comparaciones sí lo son. Si tenemos que determinar cuál es la semilla que proporciona un mayor rendimiento, diríamos que es la A o la D, y que las de menor rendimiento son la B y la C.

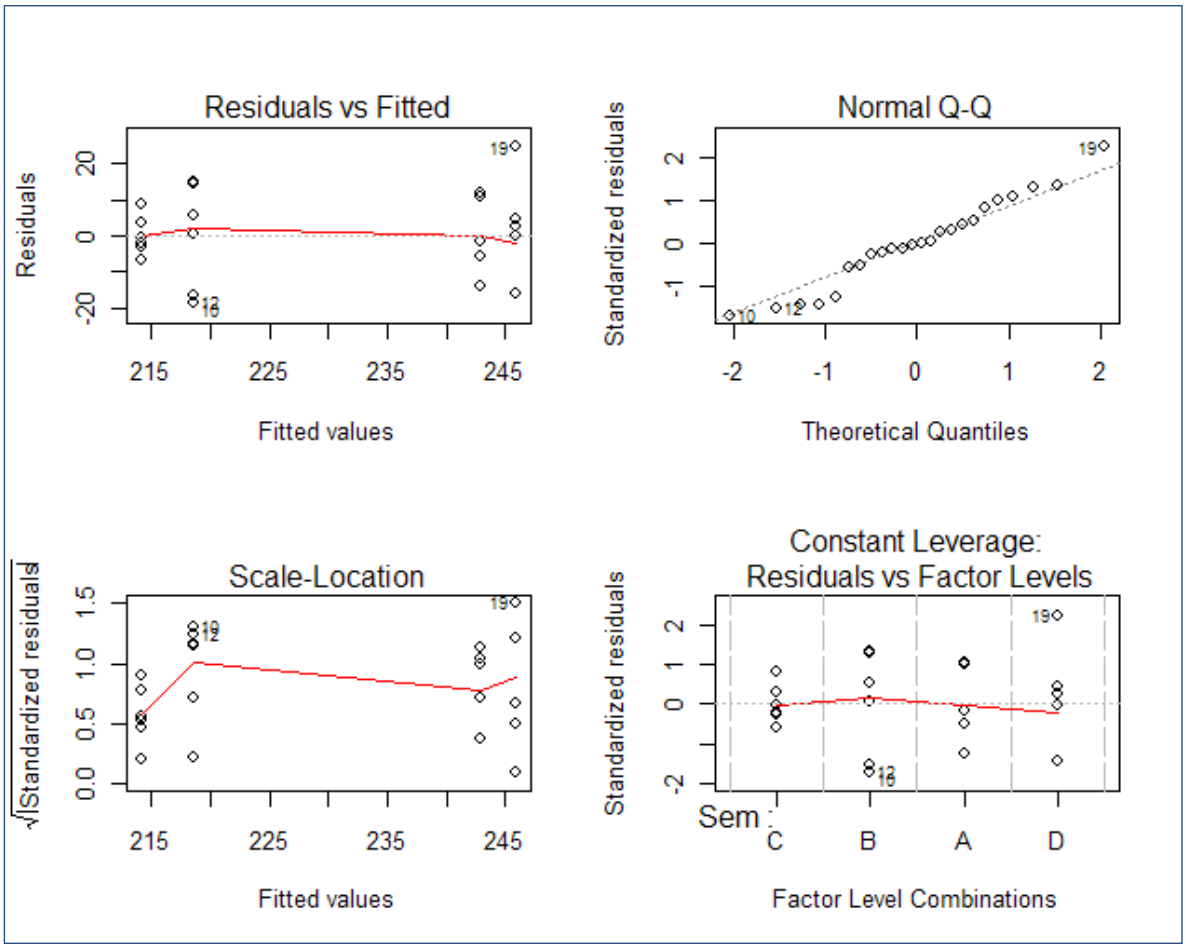
d) Diagnosis del modelo.

Para comprobar si las hipótesis del modelo lineal creado se satisfacen hay que realizar la diagnosis del mismo y comprobar la normalidad, homocedasticidad e independencia de los residuos.

Con el software R, esto se hace simplemente con las siguientes instrucciones:

```
> par(mfrow = c(2,2))
> plot(mod)
```

Obteniendo:



La primera instrucción la empleamos para dividir el panel gráfico en dos filas y dos columnas. Para ver un análisis de los residuos se emplea la función plot del modelo de regresión generado.

En la figura anterior se muestran los siguientes gráficos:

- Residuos en función de los valores estimados.
- Residuos estandarizados representados en papel probabilístico normal.
- La raíz cuadrada del valor absoluto de los residuos estandarizados en función de los valores estimados.
- Residuos estandarizados en función de los diferentes factores.

Observando estos gráficos, podemos aceptar que los residuos se distribuyen según una distribución normal (ver gráfico derecho superior). También se observa que los residuos pueden considerarse homocedásticos, es decir, se puede considerar que los residuos tienen la misma varianza para cada uno de los tipos de semilla empleados.

5.3. Diseño de Experimentos: Dos Factores con Interacción

Un grupo de investigadores pretenden combatir los efectos de ciertos agentes tóxicos. Para ello, se analiza el efecto de tres venenos y cuatro antídotos en el tiempo de supervivencia de unas ratas. Para el experimento se cogieron 48 animales y se asignaron al azar a cada uno de los doce tratamientos resultantes de combinar venenos y antídotos. A cada rata se le suministro una cierta cantidad de veneno y después de un cierto tiempo se les administró el antídoto. A partir de este momento se cuenta el tiempo de supervivencia del roedor. Los tiempos obtenidos en unidades de 10 horas para cada tratamiento se muestran en la tabla siguiente:

		ANTÍDOTO			
		A	B	C	D
VENENOS	I	0.31	0.82	0.43	0.45
		0.45	1.10	0.45	0.71
		0.46	0.88	0.63	0.66
		0.43	0.72	0.72	0.62
	II	0.36	0.92	0.44	0.56
		0.29	0.61	0.35	1.02
		0.40	0.49	0.31	0.71
		0.23	1.24	0.40	0.38
	III	0.22	0.30	0.23	0.30
		0.21	0.37	0.25	0.36
		0.18	0.38	0.24	0.31
		0.23	0.29	0.22	0.33

El objetivo de este experimento es (1) contrastar si existen diferencias entre los venenos, en el caso de que así sea se pretende decidir cual es el más perjudicial, (2) contrastar también si existen diferencias entre los distintos antídotos y (3) contrastar si el veneno y el antídoto interaccionan. Dos factores interaccionan si el efecto de uno de los factores depende del nivel en el que se encuentra, el otro. En este caso, la interacción implicaría que el efecto del antídoto será, distinto según el veneno ingerido.

En primer lugar cargamos los datos situados en el fichero "venenos.txt", mediante el comando "read.table", y comprobamos los nombres de la estructura de datos descargada:

```
> box <- read.table('venenos.txt', header = T)
> names(box)
[1] "Ven"    "Ant"    "Tiempo"
```

Se observa que los valores de las columnas "Ven" y "Ant" son valores numéricos, los convertimos en variables categóricas mediante las siguientes instrucciones:

```
> box$VEN <- factor(box$Ven,labels=c('I','II','III'))
> box$ANT <- factor(box$Ant,labels=c('A','B','C','D'))
> attach(box)
```

En muchas funciones en las que intervienen variables categóricas o cualitativas dependen de cómo estén definidas internamente. Por tanto, si la variable es de tipo cualitativo **es muy importante** realizar el paso anterior.

Debido a que en este apartado consideraremos la interacción entre los dos factores, el modelo que se empleará es el siguiente:

$$y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + u_{ijk}$$

Donde:

μ : Media global

α_i : Efecto del nivel i del primer factor.

β_j : Efecto del nivel j del segundo factor.

$(\alpha\beta)_{ij}$: Interacción entre el nivel i del primer factor y el nivel j del segundo.

u_{ijk} : Desviación (error) de la observación respecto a la media del tratamiento, tiene distribución normal de media cero y varianza 0-2.

A los términos α_i y β_j se llaman efectos principales y $(\alpha\beta)_{ij}$ es la interacción o efecto de segundo orden.

Teniendo en cuenta el modelo descrito anteriormente, utilizaremos los siguientes comandos para realizar el estudio de los resultados obtenidos:

```
> mod_box <- aov(Tiempo ~ANT*VEN)
```

Como se observa, hemos definido el modelo con la variable dependiente "Tiempo" que se explica a través de las variables explicativas "ANT", "VEN" y su interacción. A continuación calcularemos las medias para cada factor, así como también las medias para cada una de las interacciones.

Si se hubiera deseado estudiar el modelo **sin** interacción, la instrucción que habríamos tenido que emplear es la siguiente: `mod_box <- aov(Tiempo ~ANT + VEN)`.

El resultado de la estimación es el siguiente:

```
> model.tables(mod_box,'means')
Tables of means
Grand mean

0.479375

 ANT
ANT
     A      B      C      D
0.3142 0.6767 0.3925 0.5342

 VEN
VEN
     I     II    III
0.6175 0.5444 0.2762

 ANT:VEN
   VEN
ANT I      II     III
  A 0.4125 0.3200 0.2100
  B 0.8800 0.8150 0.3350
  C 0.5675 0.3750 0.2350
  D 0.6100 0.6675 0.3250
```

Nos proporciona las medias: la general, por antídotos, por venenos y por combinación de antídotos y venenos.

Para mostrar los valores de $\hat{\alpha}_i$, $\hat{\beta}_j$, y $\left(\alpha\beta\right)_{ij}$, se emplea el siguiente comando:

```
> model.tables(mod_box,'effects')
Tables of effects

 ANT
ANT
       A        B        C        D
-0.16521  0.19729 -0.08688  0.05479

 VEN
VEN
       I       II      III
 0.13812  0.06500 -0.20312

 ANT:VEN
   VEN
ANT I         II       III
  A -0.03979 -0.05917  0.09896
```

```
  B  0.06521  0.07333 -0.13854
  C  0.03688 -0.08250  0.04563
  D -0.06229  0.06833 -0.00604
```

Obsérvese que cumplen las condiciones impuestas, ya que suman cero.

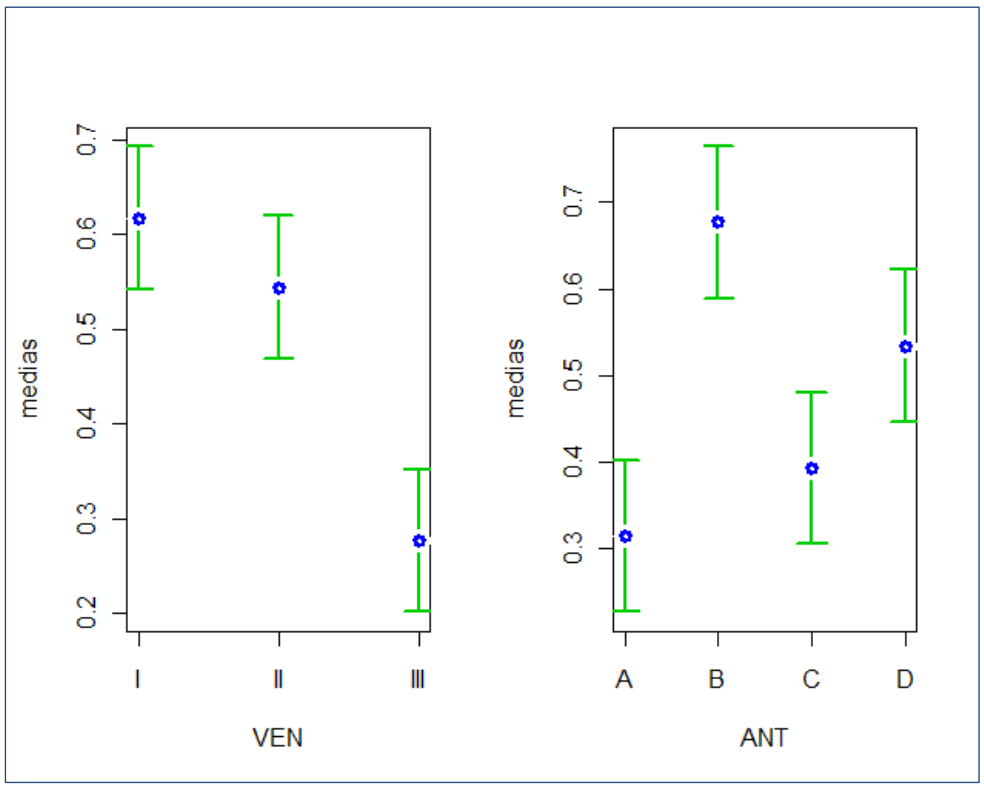
Podemos obtener la tabla ADEVA, se emplea el siguiente comando:

```
> anova(mod_box)
Analysis of Variance Table

Response: Tiempo
          Df  Sum Sq Mean Sq F value    Pr(>F)
ANT        3 0.92121 0.30707 13.8056 3.777e-06 ***
VEN        2 1.03301 0.51651 23.2217 3.331e-07 ***
ANT:VEN    6 0.25014 0.04169  1.8743    0.1123
Residuals 36 0.80073 0.02224
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

De la tabla anterior se observa como muy significativas las diferencias entre las medias de los venenos y antídotos, ya que sus p-valores son muy bajos. Es decir, que tanto los venenos como los antídotos tienen un efecto muy significativo sobre el tiempo de supervivencia de la rata. El nivel crítico de la interacción es 0.1123, por encima de los niveles habituales de significación, por tanto en este caso se concluye que no hay evidencia de interacción entre los factores.

Podemos comprobar gráficamente si las medias entre los venenos y antídotos son significativamente diferentes por medio del gráfico de los intervalos de confianza. A continuación mostramos las instrucciones necesarias para obtener el gráfico de los intervalos de confianza para las medias de los diferentes venenos y antídotos.

```
> par(mfrow=c(1,2))
> ICplot(mod_box, 'VEN')
> ICplot(mod_box, 'ANT')
```



NOTA: Los intervalos de confianza de estos gráficos corresponden a un nivel de confianza del 5%, ya que no hemos incluido ningún tercer argumento en la función "ICplot".

De los gráficos anteriores se observa que el veneno III es significativamente distinto a los venenos I y II (los cuales no son significativamente distintos entre sí). En cuanto a los antídotos, el antídoto A es significativamente distinto al B y D; y el B es significativamente distinto con el C.

Otra forma complementaria para hacer las comparaciones anteriores consiste en el método de Tukey.

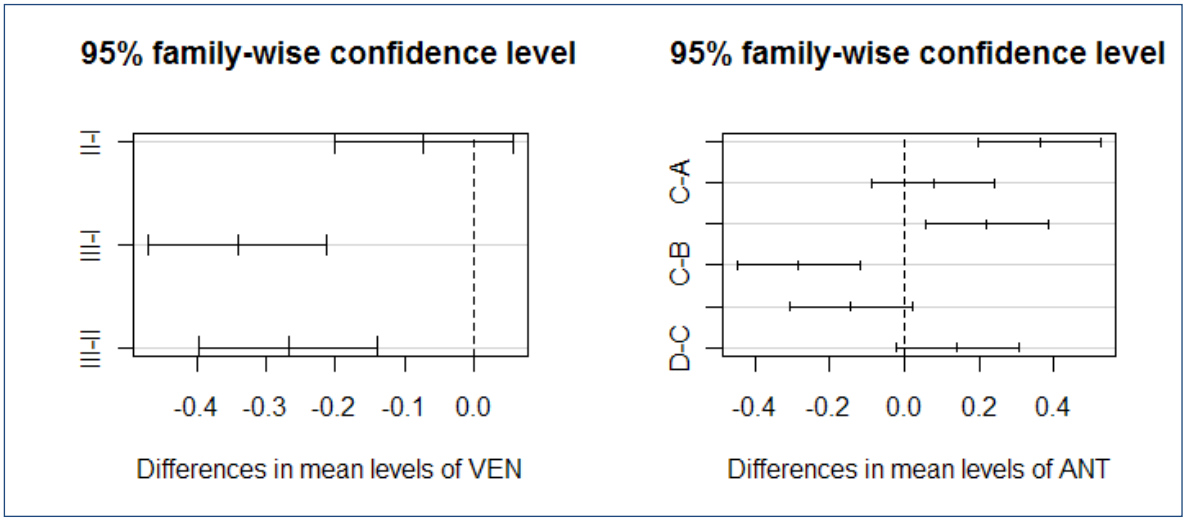
El método de Tukey es un procedimiento especial para realizar múltiples comparaciones. Es ideal cuando el número de comparaciones que se desea hacer es grande. Con este método se corrige la tendencia a rechazar la igualdad de tratamientos erróneamente motivado por el simple hecho de realizar múltiples comparaciones. Las comparaciones utilizando el criterio LSD tienen tendencia a encontrar diferencias significativas cuando las medias son iguales. Existen varios métodos para corregir esta tendencia. El método de Tukey es uno de los más utilizados.

El método de Tukey se emplea en R como se muestra a continuación:

```
> TukeyHSD(mod_box,'VEN')
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Tiempo ~ ANT * VEN)

$VEN
             diff        lwr         upr     p adj
II-I   -0.073125 -0.2020091  0.05575913 0.3583151
III-I  -0.341250 -0.4701341 -0.21236587 0.0000005
III-II -0.268125 -0.3970091 -0.13924087 0.0000339
```

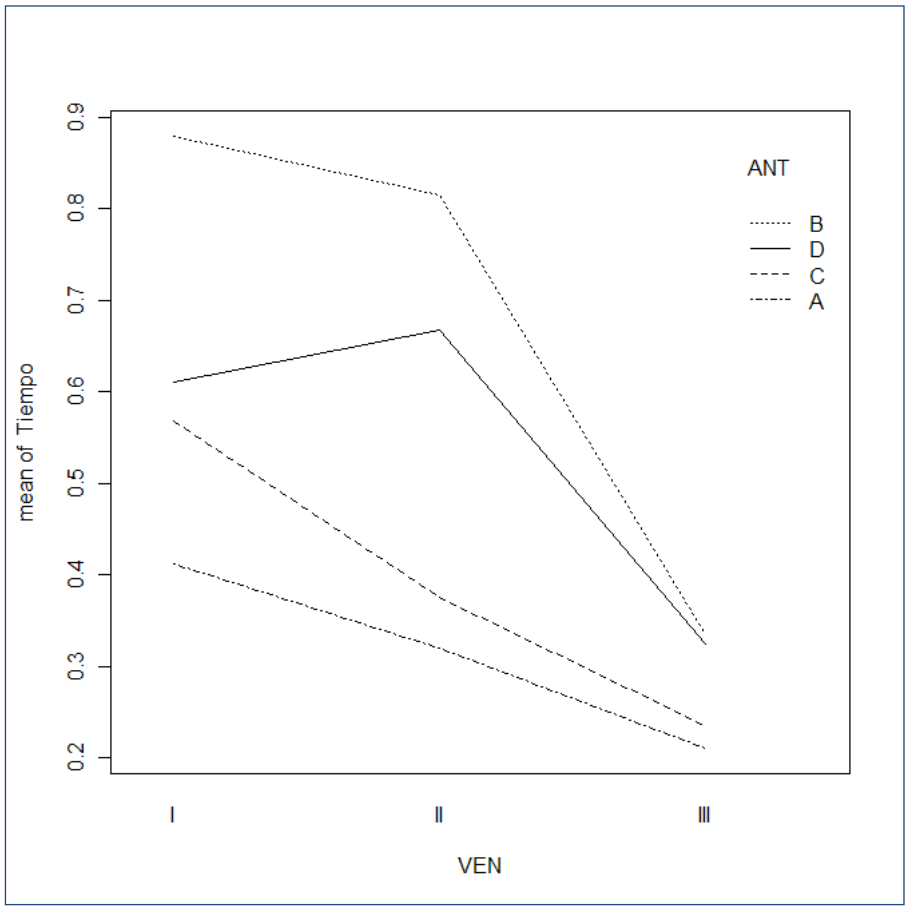
Según este criterio, existen diferencias significativas entre I y III, y entre II y III; sin embargo, no existen diferencias significativas entre I y II. En el gráfico de los intervalos de confianza para las diferencias de medias que se muestra a continuación, se observa que los intervalos no incluyen el valor cero, coinciden con los tratamientos (venenos) que son significativamente distintos. La misma comparación se puede hacer para los tratamientos.

```
> TukeyHSD(mod_box,'ANT')
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Tiempo ~ ANT * VEN)

$ANT
            diff         lwr         upr     p adj
B-A  0.36250000  0.19852116  0.52647884 0.0000047
C-A  0.07833333 -0.08564550  0.24231217 0.5772283
D-A  0.22000000  0.05602116  0.38397884 0.0048556
C-B -0.28416667 -0.44814550 -0.12018783 0.0002333
D-B -0.14250000 -0.30647884  0.02147884 0.1077087
D-C  0.14166667 -0.02231217  0.30564550 0.1107678

> plot(TukeyHSD(mod_box,'VEN'))
> plot(TukeyHSD(mod_box,'ANT'))
```



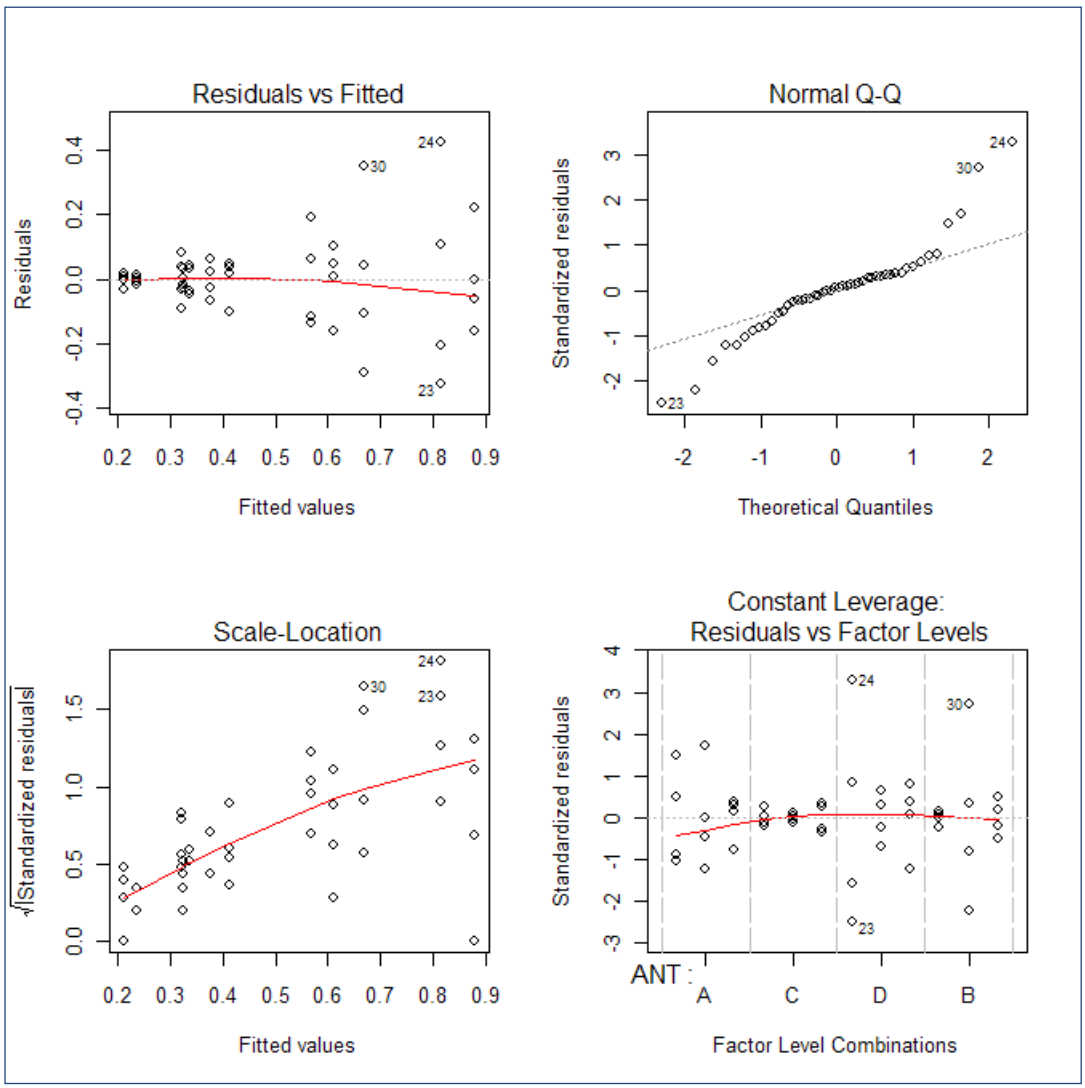
La interacción no es significativa. Mostramos el gráfico de interacción para ilustrar su utilización. El veneno III es el que tiene menor tiempo de supervivencia para los cuatro tratamientos.

```
> tapply(Tiempo,list(VEN,ANT),mean)
          A     B      C      D
I    0.4125 0.880 0.5675 0.6100
II   0.3200 0.815 0.3750 0.6675
III 0.2100 0.335 0.2350 0.3250
> interaction.plot(VEN,ANT,Tiempo)
```



Finalmente, vamos a hacer la diagnosis al modelo empleado, analizando los residuos. Para este análisis se procede de manera similar a la sección anterior:

```
> par(mfrow=c(2,2))
> plot(mod_box)
```



La falta de homocedasticidad se observa al representar los residuos frente a los valores previstos. De las figuras anteriores se aprecia como aumenta la dispersión de los residuos conforme el valor previsto crece. Esta deficiencia también justifica la falta de normalidad observada en el gráfico Q-Q.

Teniendo en cuenta que la diagnosis nos ha revelado que el modelo no se comportaba conforme habíamos asumido, y teniendo en cuenta el comportamiento de la varianza de los residuos frente a los valores esperados, optamos por aplicar una transformación.

El tratamiento habitual para resolver el inconveniente de la varianzas desiguales es la aplicación de una transformación que corrija la heterocedasticidad y posteriormente la repetición del análisis con los datos transformados.

Existen distintos métodos para buscar la transformación adecuada. Un método sencillo consiste en aplicar secuencialmente las transformaciones que se indican en la tabla siguiente, siguiendo el orden mostrado, y aplicar aquella transformación que corrija el problema de heterocedasticidad.

Transformaciones	
$\lambda = 1$	$y' = y$
$\lambda = 1/2$	$y' = \sqrt{y}$
$\lambda = 0$	$y' = \log(y)$
$\lambda = -1$	$y' = 1/y$
$\lambda = -2$	$y' = 1/y^2$

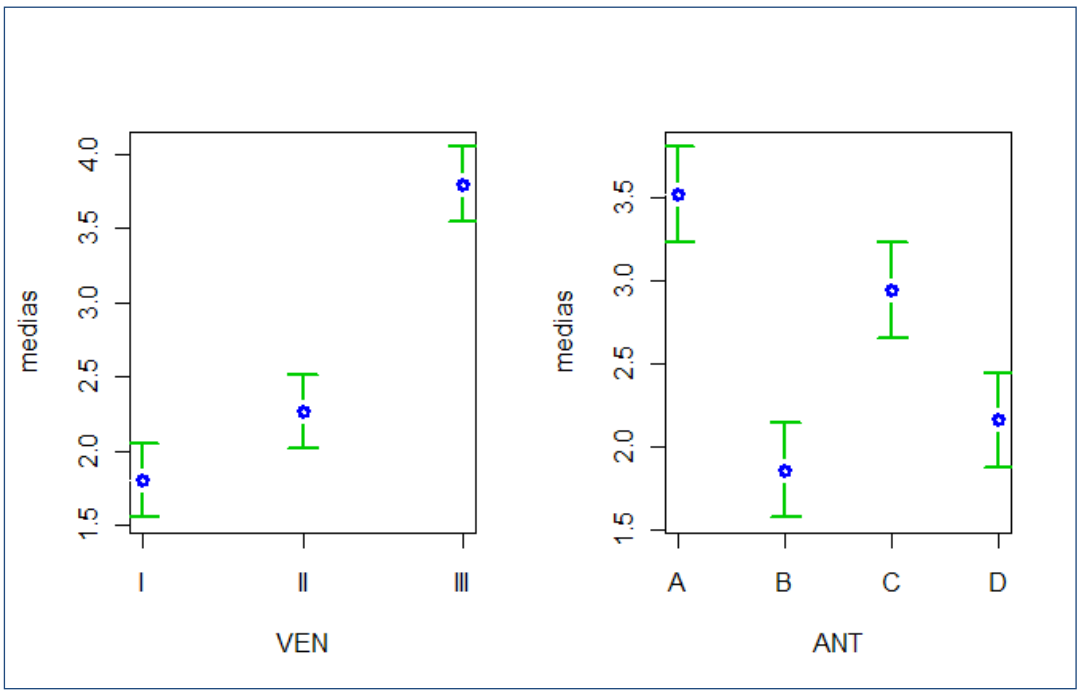
En nuestro caso particular, hemos aplicado diversas transformaciones, y comprobamos que la transformación *1/y* corrigen tanto los inconvenientes de normalidad como la heterocedasticidad.

Concretamente, aplicaremos la función inversa a la variable dependiente, tal y como se muestra en las siguientes instrucciones:

```
> mod_box2 <- aov(1/Tiempo ~ANT*VEN)
> anova(mod_box2)
Analysis of Variance Table

Response: 1/Tiempo
          Df Sum Sq Mean Sq F value    Pr(>F)
ANT        3 20.414  6.8048 28.3431 1.376e-09 ***
VEN        2 34.877 17.4386 72.6347 2.310e-13 ***
ANT:VEN    6  1.571  0.2618  1.0904    0.3867
Residuals 36  8.643  0.2401
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Los resultados de los contrastes son semejantes. Se aprecia que los valores de los estadísticos F para los efectos principales ahora son superiores. En las nuevas unidades las diferencias entre las filas y columnas son más claras. Por otra parte, el p-valor correspondiente a la interacción es mayor, en estas unidades es más evidente que los factores no interaccionan.

```
> source('ICplot.r')
>  par(mfrow=c(1,2))
>  ICplot(mod_box2, 'VEN')
>  ICplot(mod_box2, 'ANT')
```

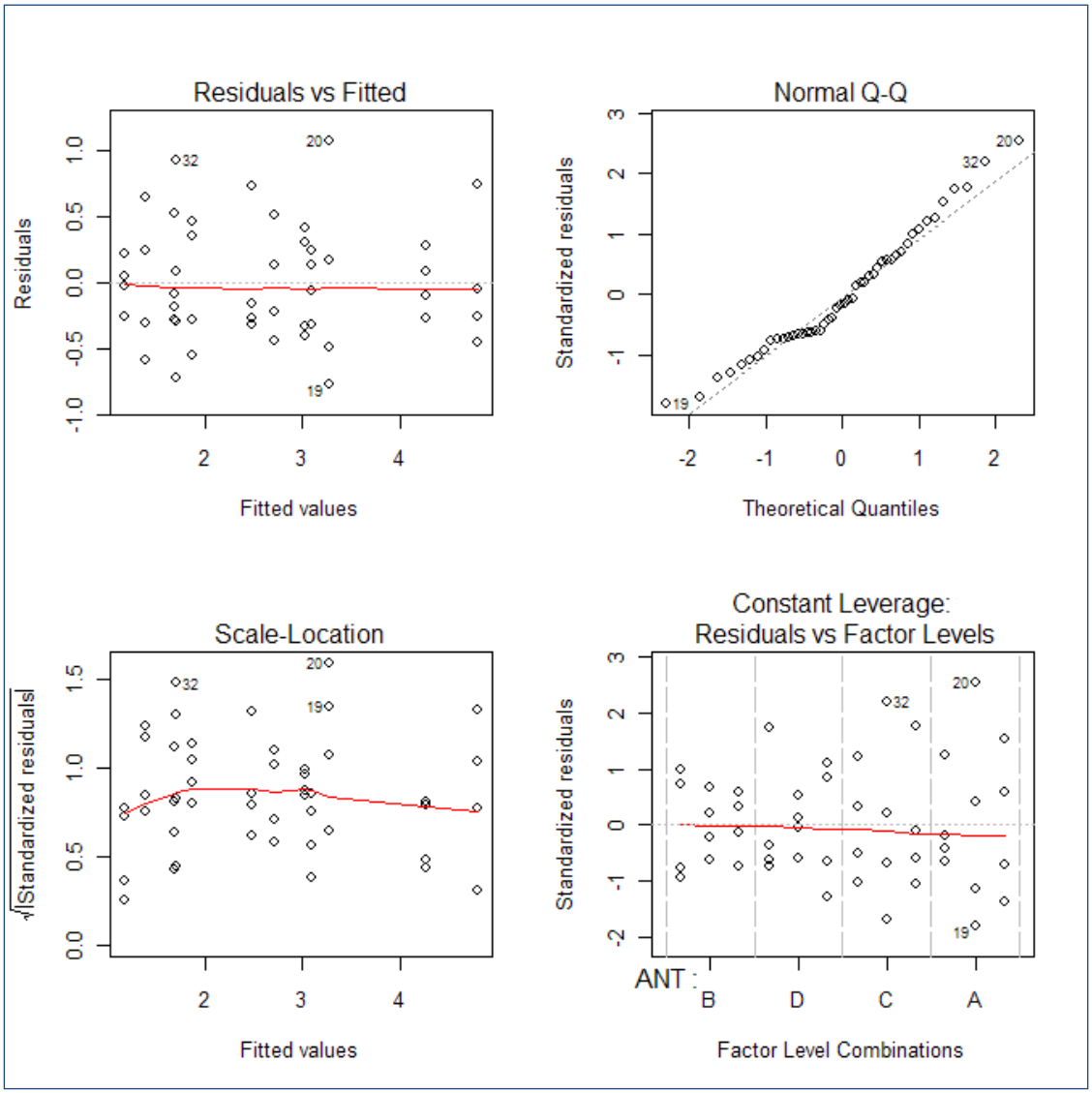


Los datos transformados muestran una mayor diferencia en el efecto de los venenos. El más rápido sigue siendo el veneno III. Obsérvese que ahora aparecen en orden inverso debido a la transformación.

Similarmente, las diferencias entre los antídotos son más importantes en esta nueva escala, aunque se mantienen las conclusiones extraídas del modelo original.

Finalmente hacemos la diagnosis de este nuevo modelo:

```
> par(mfrow=c(2,2))
> plot(mod_box2)
```



De los gráficos anteriores se observa que los residuos en este modelo ajusta más los residuos a distribuciones homocedásticas, normales e independientes.

5.4. Diseño de Experimentos: Bloques Aleatorizados

Se ha realizado el siguiente experimento para determinar si la presencia de fluorita reduce el coste de fabricación del cemento. Los resultados de cada experimento se evalúan en miles de euros por tonelada. Cada composición implica unos tiempos de proceso en las distintas etapas de fabricación y unos costes asociados por consumos de distintos recursos: materia prima, energía y otros. Se analizan cinco tratamientos distintos, según el porcentaje de fluorita, 0%, 1%, 2%, 3% y 4%. Antes de añadir la fluorita, se realiza la mezcla del resto de los componentes en un proceso complejo. Las características fisicoquímicas de esta mezcla inicial influyen en el resultado final y presenta una gran variabilidad.

Teniendo en cuenta esto, se ha realizado el experimento en dos fases, en la primera se consigue la mezcla de partida y, posteriormente, se divide esta en cinco partes iguales, añadiendo a cada una de ellas una proporción distinta de fluorita para conseguir los niveles

deseados en la comparación. El experimento se ha realizado con seis mezclas distintas y los resultados se muestran en la siguiente tabla:

		Fluorita				
		0%	1%	2%	3%	4%
Mezcla	1	15.02	11.86	9.94	12.45	13.23
	2	8.42	10.15	8.54	6.98	8.93
	3	18.31	16.84	15.86	14.64	15.96
	4	10.49	10.52	8.04	10.50	10.34
	5	9.78	9.59	6.96	8.15	9.24
	6	9.28	8.84	7.04	6.66	9.46

Como era de esperar por los expertos, existen grandes diferencias entre los valores para cada mezcla (bloques). El objetivo del análisis es decidir si existen diferencias significativas entre los tratamientos (i.e., entre las diferentes composiciones de fluorita).

Para analizarlo con R, en primer lugar cargamos el fichero:

```
> fluorita = read.table('fluorita.txt', header = T)
> attach(fluorita)
> FLUO = factor(fluo)
> MEZ  = factor(mez)
> mod_flu = aov(cost ~ FLUO + MEZ)
```

En la tercera y cuarta instrucción se convierten las columnas de valores numéricos ("fluo" y "mez") en variables categóricas o factores ("FLUO" y "MEZ"). En la última instrucción definimos el modelo, en el cual no se considera la interacción entre bloques y tratamientos.

Si generamos la tabla ADEVA, se observa lo siguiente:

```
> anova(mod_flu)
Analysis of Variance Table

Response: cost
          Df  Sum Sq Mean Sq F value    Pr(>F)
FLUO       4  26.051   6.513  7.4372 0.0007697 ***
MEZ        5 247.768  49.554 56.5862 4.068e-11 ***
Residuals 20  17.514   0.876
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Se observa que existen diferencias significativas tanto para el porcentaje de fluorita como para los bloques de mezcla. Los p-valores asociados son, en ambos casos, menores que el nivel de significación habitual.

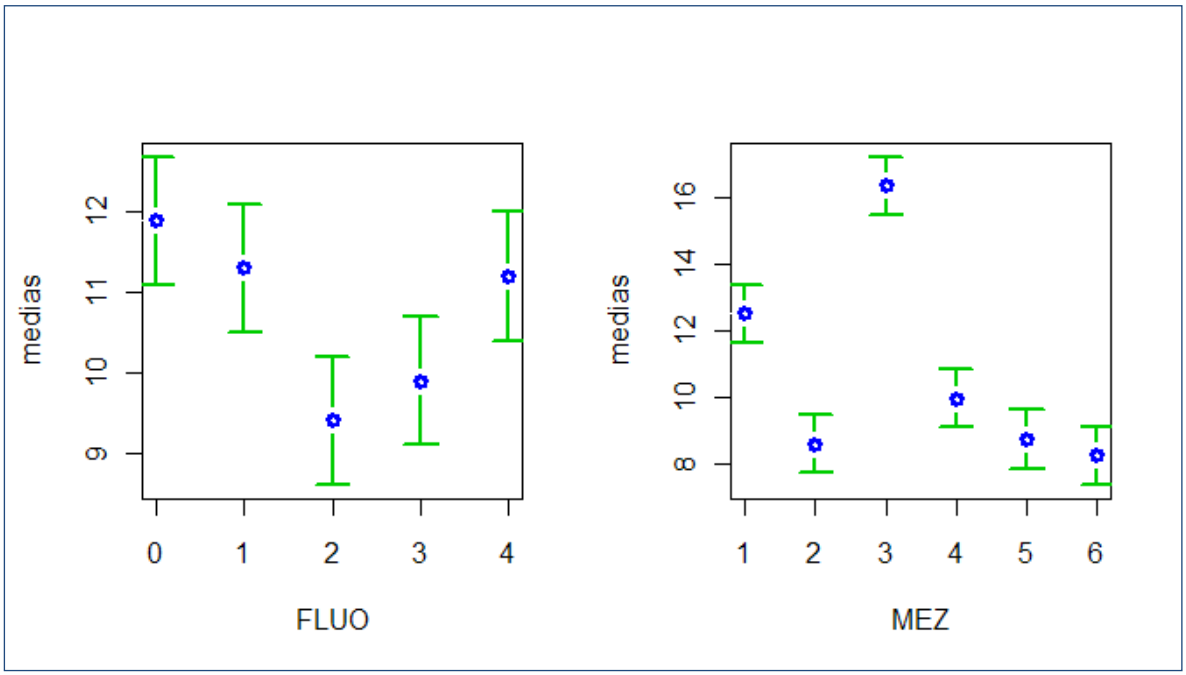
En caso de que no hubiéramos considerado los bloques, obtendríamos los siguientes resultados:

```
> mod_flu0 = aov(cost ~ FLUO)
> anova(mod_flu0)
Analysis of Variance Table

Response: cost
          Df  Sum Sq Mean Sq F value Pr(>F)
FLUO       4  26.051  6.5128  0.6138 0.6567
Residuals 25 265.282 10.6113
```

Donde se observa que, en este caso, los tratamientos no influyen significativamente. Esta conclusión errónea se ha derivado de menospreciar el efecto de los bloques.

A continuación generamos la representación de los intervalos de confianza para los tratamientos y para los bloques:

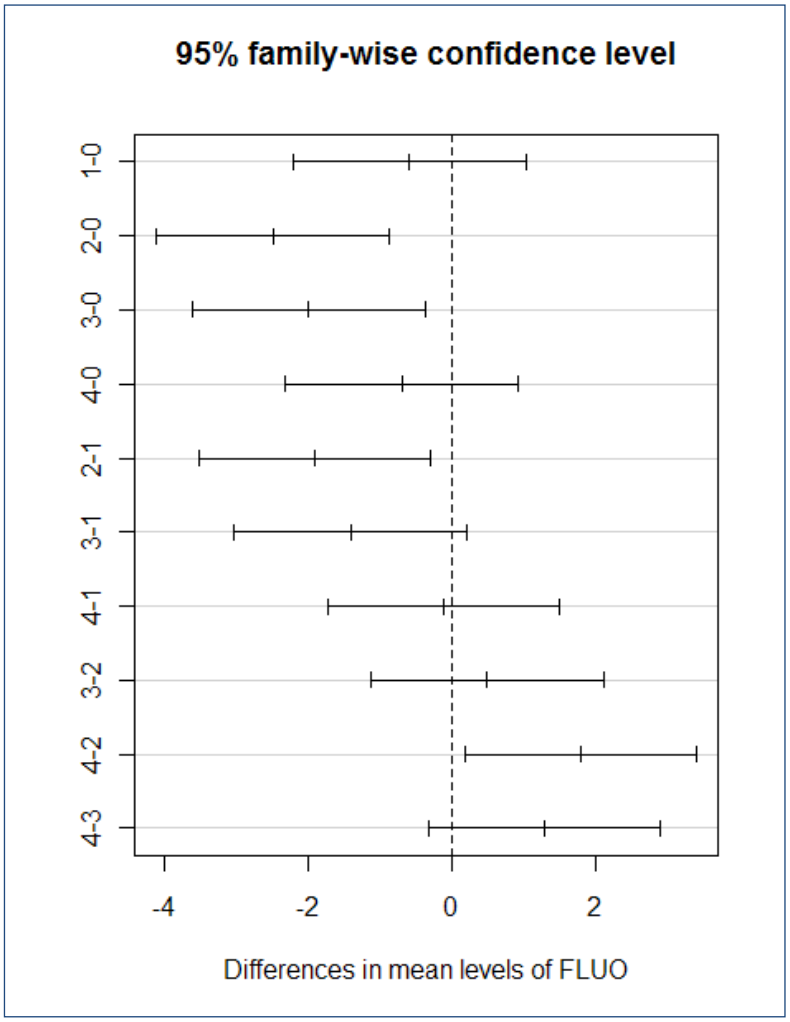
```
> par(mfrow = c(1,2))
> source('ICplot.R')
> ICplot(mod_flu, "FLUO")
> ICplot(mod_flu, "MEZ")
```



De la figura anterior se puede observar, por ejemplo, que los cementos de la mezcla tercera tienen una media que es significativamente superior al resto de medias. También se observa, por ejemplo, que los cementos con un 2% de fluorita tienen una media significantemente inferior que los cementos con porcentajes del 0, 1 y 4 de fluorita; sin embargo, no hay evidencia estadística de que las medias entre los cementos de 2% y 3% sean diferentes.

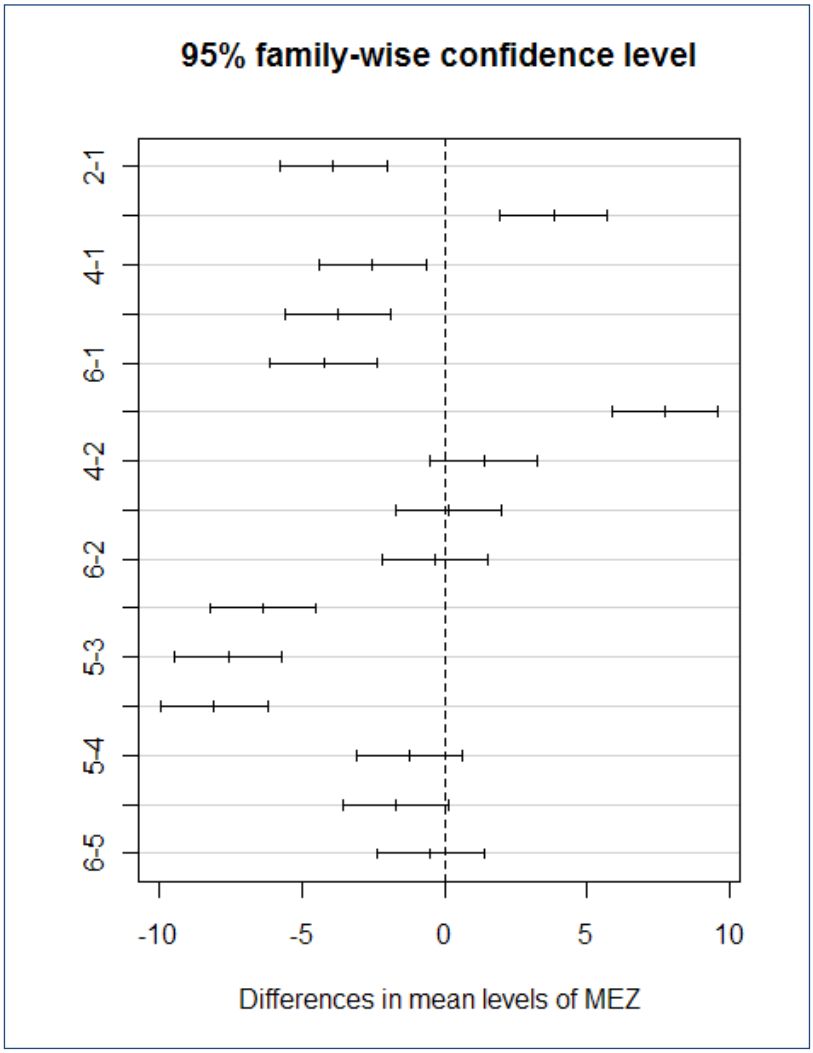
Estas comparaciones pueden hacerse dos a dos mediante el método Tukey, tal y como se muestra a continuación:

```
> plot(TukeyHSD(mod_flu, "FLUO"))
> TukeyHSD(mod_flu, "FLUO")

  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = cost ~ FLUO + MEZ)

$FLUO
          diff       lwr        upr     p adj
1-0 -0.5833333 -2.200062  1.0333953 0.8146785
2-0 -2.4866667 -4.103395 -0.8699380 0.0014460
3-0 -1.9866667 -3.603395 -0.3699380 0.0115672
4-0 -0.6900000 -2.306729  0.9267286 0.7075754
2-1 -1.9033333 -3.520062 -0.2866047 0.0162291
3-1 -1.4033333 -3.020062  0.2133953 0.1088135
4-1 -0.1066667 -1.723395  1.5100620 0.9996295
3-2  0.5000000 -1.116729  2.1167286 0.8836892
4-2  1.7966667  0.179938  3.4133953 0.0248733
4-3  1.2966667 -0.320062  2.9133953 0.1561637
```



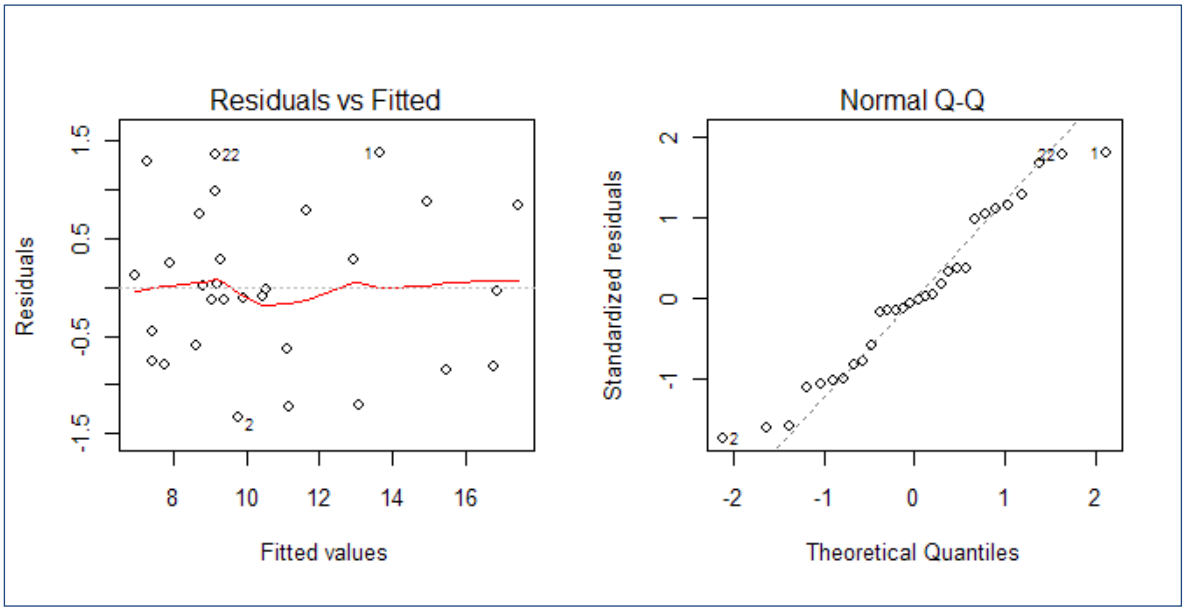
Los intervalos que no contienen el cero corresponden con los tratamientos cuyas medias son significativamente distintas. Lo mismo para el factor "mezcla":

```
> plot(TukeyHSD(mod_flu, "MEZ"))
> TukeyHSD(mod_flu, "MEZ")

  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = cost ~ FLUO + MEZ)

$MEZ
      diff        lwr        upr     p adj
2-1 -3.896 -5.7563373 -2.0356627 0.0000271
3-1  3.822  1.9616627  5.6823373 0.0000352
4-1 -2.522 -4.3823373 -0.6616627 0.0044376
5-1 -3.756 -5.6163373 -1.8956627 0.0000445
6-1 -4.244 -6.1043373 -2.3836627 0.0000081
3-2  7.718  5.8576627  9.5783373 0.0000000
4-2  1.374 -0.4863373  3.2343373 0.2313782
5-2  0.140 -1.7203373  2.0003373 0.9998810
6-2 -0.348 -2.2083373  1.5123373 0.9907137
4-3 -6.344 -8.2043373 -4.4836627 0.0000000
5-3 -7.578 -9.4383373 -5.7176627 0.0000000
6-3 -8.066 -9.9263373 -6.2056627 0.0000000
5-4 -1.234 -3.0943373  0.6263373 0.3336519
6-4 -1.722 -3.5823373  0.1383373 0.0798482
6-5 -0.488 -2.3483373  1.3723373 0.9594014
```



Finalmente hacemos la diagnosis del modelo:

```
par(mfrow = c(1,2))
plot(mod_flu)
```



De las figuras anteriores se observa que los residuos pueden considerarse homocedásticos y normales.

5.5. Diseño de Experimentos: Tres Factores

En un laboratorio científico se desea estudiar las características propias de un determinado proceso químico en función de diferentes factores. Concretamente, se pretende analizar el rendimiento de una reacción química dependiendo de los siguientes factores:

- Concentración
- Catalizador
- Temperatura

En las tablas siguientes se detallan los diferentes niveles que pueden tomar estos tres factores:

Concentración	
1	4%
2	6%
3	8%
4	10%

Catalizador	
C-1	Plata
C-2	Zinc
C-3	Mezcla

Temperatura	
T-1	300º C
T-2	320º C

Cada tratamiento viene determinado por cada nivel de los factores. Por tanto, el número posible de tratamientos distintos es: 6 x 3 x 2 = 90 tratamientos distintos.

Efectuamos en el laboratorio cada uno de los posibles tratamientos con tres replicaciones, aleatorizando adecuadamente, y obtenemos los siguientes resultados de rendimiento del proceso en función de los diferentes niveles de cada factor:

		CONCENTRACIÓN							
		1		2		3		4	
		T-1	T-2	T-1	T-2	T-1	T-2	T-1	T-2
CATALIZADOR	C-1	72,2	65,0	74,4	69,2	75,0	70,7	80,0	73,0
		74,4	71,6	66,3	71,8	78,9	80,6	65,0	74,4
		64,3	61,9	66,5	64,6	64,3	73,4	82,1	78,8
		T-1	T-2	T-1	T-2	T-1	T-2	T-1	T-2
	C-2	62,5	75,9	70,8	79,2	76,3	83,3	72,3	80,3
		65,8	72,9	63,9	80,1	79,1	88,0	72,4	86,9
		71,2	77,8	76,6	75,3	89,0	84,7	75,6	86,3
		T-1	T-2	T-1	T-2	T-1	T-2	T-1	T-2
	C-3	69,0	73,8	69,0	84,5	72,8	94,1	78,4	87,5
		70,3	59,2	68,2	93,7	73,7	87,3	79,9	79,7
		68,8	80,8	78,7	80,1	80,7	89,0	80,3	79,5

Para analizar estos datos con R, en primer lugar cargamos el fichero:

```
> quimico = read.table('quimico.txt', header = T)
> attach(quimico)
> CON = factor(con)
> TEMP = factor(temp)
> CAT = factor(cat)
```

Con la tercera, cuarta y quinta instrucción se convierten los números de las columnas "con", "temp" y "cat", en caracteres, y se almacenan en las nuevas variables "CON", "TEMP", y "CAT".

A continuación, se define el modelo a emplear

```
> mod_qui = aov(rend  ~ CON * TEMP * CAT  )
```

El modelo que se ha empleado considera los tres efectos principales (A, B, C), los tres efectos de segundo orden (AxB, AxC, BxC), y el efecto de tercer orden (AxBxC). El modelo que se emplea es el siguiente:

$$y_{ijkm} = \mu + \alpha_i + \beta_j + \gamma_k + \alpha\beta_{ij} + \alpha\gamma_{ik} + \beta\gamma_{jk} + \alpha\beta\gamma_{ijk} + u_{ijkm}$$

A continuación, generamos la tabla ADEVA a partir del modelo creado.

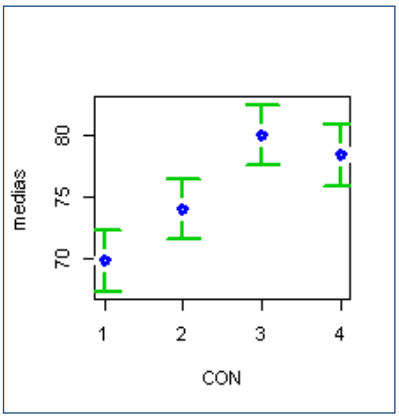
```
> anova(mod_qui)
Analysis of Variance Table

Response: rend
             Df  Sum Sq Mean Sq F value    Pr(>F)
CON           3 1141.58  380.53 13.7571 1.331e-06 ***
TEMP          1  481.53  481.53 17.4087 0.0001258 ***
CAT           2  600.01  300.01 10.8460 0.0001298 ***
CON:TEMP      3   69.29   23.10  0.8350 0.4812982
CON:CAT       6  177.87   29.64  1.0717 0.3925498
TEMP:CAT      2  310.71  155.36  5.6166 0.0064304 **
CON:TEMP:CAT  6  217.73   36.29  1.3119 0.2701401
Residuals    48 1327.71   27.66
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Se observa que el efecto principal del factor concentración influye significativamente (p-valor =0.0000) en el rendimiento. Este factor no interacciona con ningún otro.

También se observa que los efectos principales de catalizador y de la temperatura son significativos, además es muy significativa la interacción de los dos factores (p-valor 0.0064). La comparación de medias de estos factores debe ser conjunta.

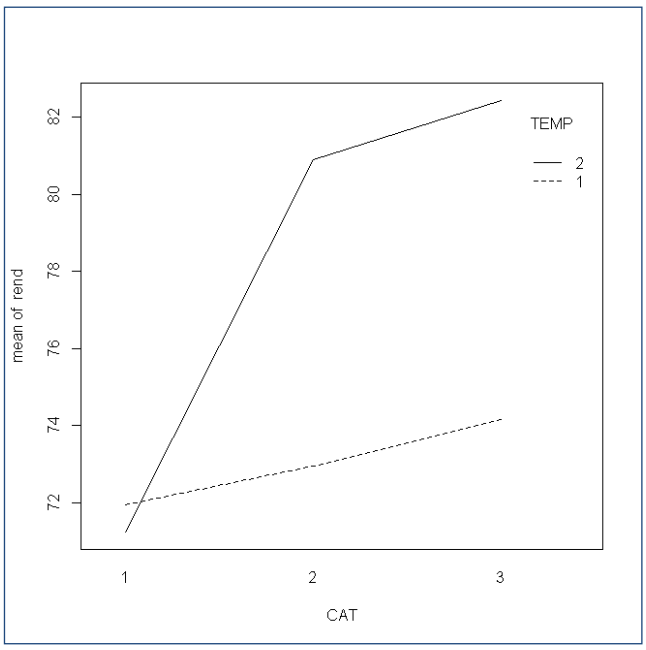
A continuación hacemos la comparación gráfica de los intervalos de confianza de las medias para la concentración, para un valor de $\alpha = 0.05$.

```
> source('ICplot.R')
> ICplot(mod_qui, "CON")
```



Como hemos observado, a partir de la tabla ADEVA, que la interacción "temperatura-catalizador" es significativa, a continuación presentamos un gráfico de la interacción de estos dos factores con respecto al rendimiento. Para ello utilizamos la función "*interaction.plot*".

```
> interaction.plot(CAT, TEMP,  rend)
```

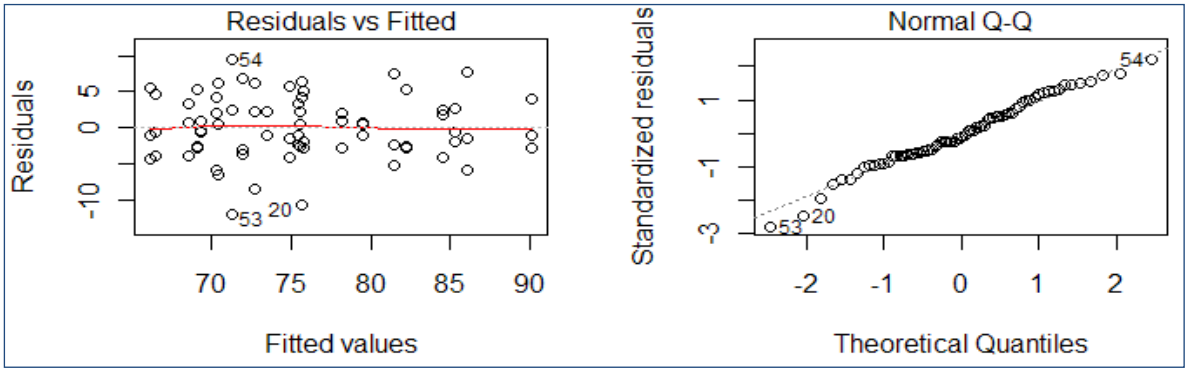


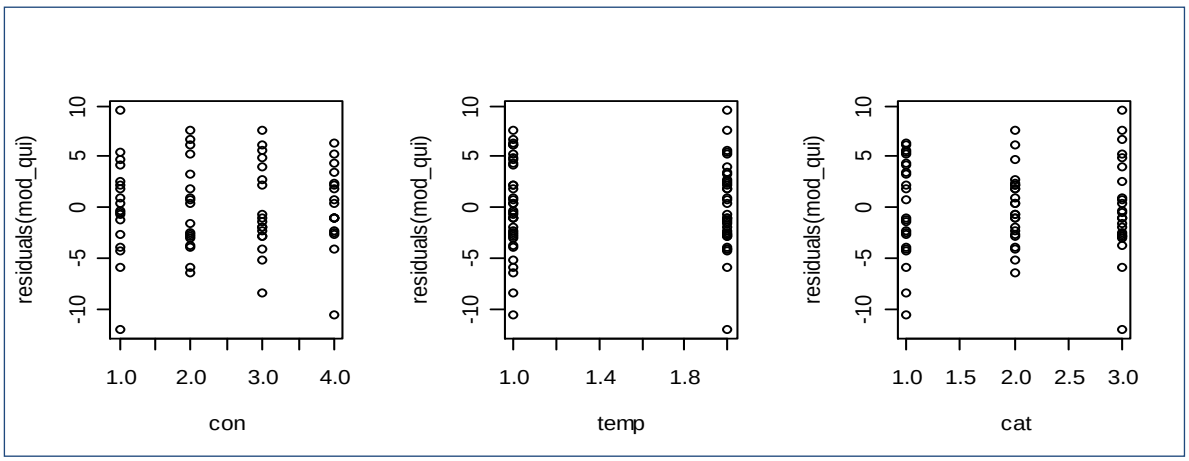
El efecto del catalizador depende de la temperatura. A la temperatura 2, los catalizadores 2 y 3 aumentan considerablemente el rendimiento. El catalizador 1 tiene el mismo comportamiento para las dos temperaturas.

Se observa que las mejores combinaciones corresponden a la temperatura 2, con el catalizador 2 o el 3.

Para finalizar, hacemos la diagnosis del modelo empleado:

```
> par(mfrow = c(1,2))
> plot(mod_qui)
> par(mfrow = c(1,3))
> plot( con, residuals(mod_qui))
> plot( temp, residuals(mod_qui))
> plot( cat, residuals(mod_qui))
```





Donde observamos que podemos asumir que se cumplen las hipótesis del modelo: homocedasticidad, normalidad e independencia.

5.6. Diseño de Experimentos: Tres Factores (sin replicaciones)

En una fábrica de obleas de silicio se desea estudiar la influencia de la temperatura y el acabado superficial en el espesor de óxido conseguido en las obleas de silicio fabricadas. El factor "temperatura" tiene tres niveles, y el factor "AS" (acabado superficial) tiene dos niveles.

Para realizar este experimento se utilizan cuatro hornos distintos: para cada horno se estudia el tratamiento de cada uno de los niveles de temperatura y de acabado superficial.

A continuación se muestran los resultados:

		Temperatura		
Horno	**AS**	**1**	**2**	**3**
1	**1**	122.2	103.2	115.8
	2	138.4	144.3	159.8
2	**1**	131.0	133.4	121.8
	2	147.4	138.0	147.5
3	**1**	120.5	102.8	120.0
	2	140.6	126.6	141.9
4	**1**	100.0	105.8	114.7
	2	117.0	134.4	131.7

Para analizar estos datos en R, empleamos la siguiente secuencia de comandos:

```
> espesor = read.table('espesor.txt', header = T)
> attach(espesor)
> TEMP = factor(temp)
> HORN = factor(horn)
> AS   = factor(as)
> mod_es = aov(esp  ~ TEMP + HORN + AS + TEMP:HORN + TEMP:AS + HORN:AS  )
```

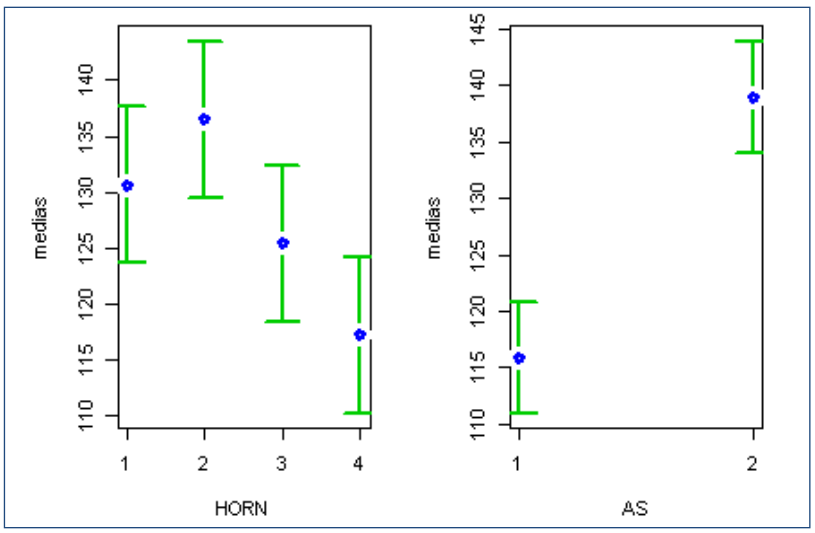
Como se observa de la última instrucción, el modelo que emplearemos considera los tres efectos principales y los tres efectos de segundo orden. El efecto de tercer orden no se considera en el análisis.

```
> summary(mod_es)
             Df Sum Sq Mean Sq F value    Pr(>F)
TEMP          2    263     131   2.696 0.146070
HORN          3   1201     400   8.214 0.015167 *
AS            1   3183    3183  65.321 0.000192 ***
TEMP:HORN     6    541      90   1.852 0.236218
TEMP:AS       2    101      51   1.039 0.409817
HORN:AS       3    265      88   1.815 0.244767
Residuals     6    292      49
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Se observa que el efecto del factor "Acabado Superficial" influye significativamente en el espesor de óxido de las obleas fabricadas. El factor "horno" también influye, pero con un p-valor mayor. Las interacciones de orden dos, sin embargo, no toman un valor significativo (ya que el p-valor es mayor que α).

A continuación mostramos gráficamente los intervalos de confianza para las medias, para los factores "HORN" y "AS".

```
> source('ICplot.R')
> par(mfrow = c(1,))
> ICplot(mod_es, "HORN")
> ICplot(mod_es, "AS")
```



Se observa que el acabado superficial que produce mayor espesor es el 2. También se observa que el horno que produce media mayor es el 2, aunque no es significativamente distinto del 1 ni del 3.

El diseño anterior permite analizar los efectos principales e interacciones de orden dos sin necesidad de replicación. Ello presenta la ventaja de un gran ahorro experimental. Para que el análisis sea correcto debe cumplirse la condición de que las interacciones de orden tres sean nulas.

Todavía es posible reducir más el número de observaciones a tomar en el estudio de tres factores si nuestro interés es estudiar los efectos principales exclusivamente y además se verifica que todas las interacciones (de cualquier orden) son nulas. El modelo experimental para analizar este problema es el cuadrado latino, y se ilustra mediante el ejemplo del siguiente apartado.

5.7. Diseño de Experimentos: Cuadrado Latino

Una organización de consumidores estudió la eficacia de cinco aditivos que según los fabricantes reducían el consumo de combustible. Se realiza un diseño experimental con cinco conductores, cinco vehículos y cinco aditivos, eligiendo las 25 combinaciones que se muestran en la tabla, junto con una medida del consumo.

Conductor \ Vehículo	1	2	3	4	5
1	C 71	A 64	D 68	B 78	E 82
2	D 65	C 64	B 81	E 82	A 82
3	E 63	B 68	A 74	D 77	C 85
4	B 66	E 77	C 79	A 88	D 74
5	A 73	D 70	E 78	C 80	B 88

Nota: los cinco aditivos son A, B, C, D y E.

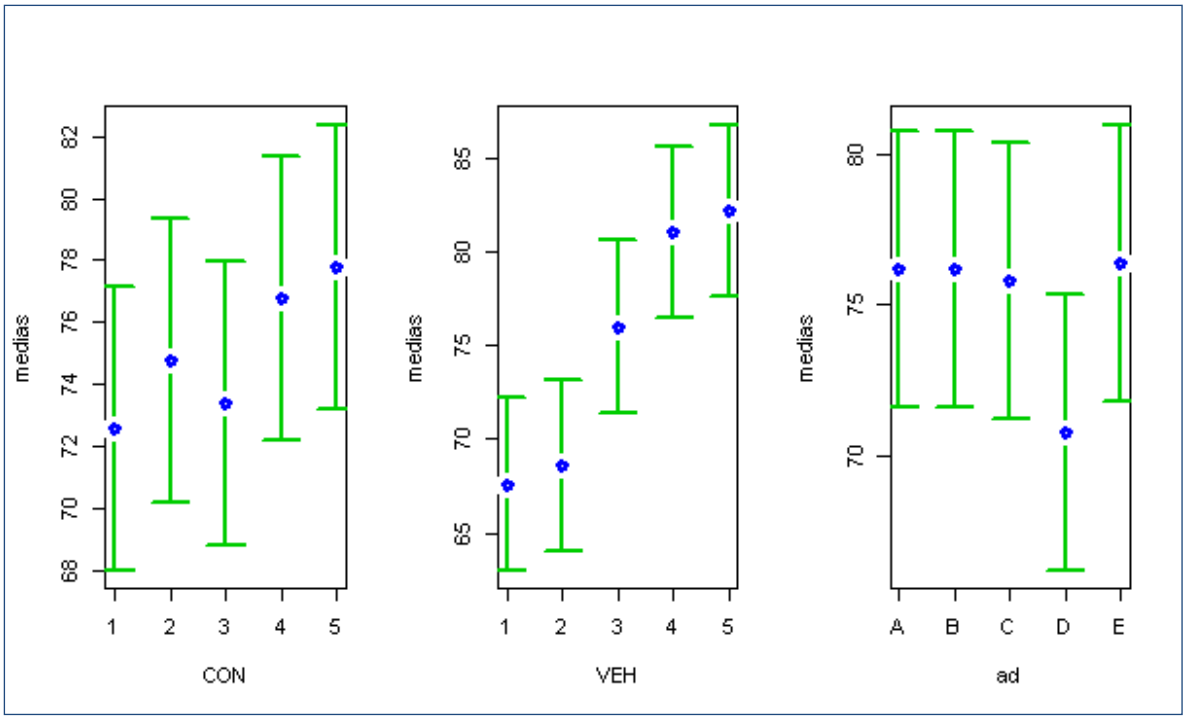
En primer lugar, cargamos los datos en R y creamos el modelo.

```
> aditivos = read.table('aditivos.txt', header = T)
> attach(aditivos)
> CON = factor(con)
> VEH = factor(veh)
> mod_ad = aov(cons ~ CON + VEH + ad)
```

Como se observa de la última instrucción, el modelo que emplearemos considera únicamente los tres efectos principales. El efecto de tercer orden y los tres efectos de segundo orden no se consideran en el análisis.

A continuación calculamos la tabla ADEVA y el gráfico de

```
> anova(mod_ad)
Analysis of Variance Table

Response: cons
          Df Sum Sq Mean Sq F value    Pr(>F)
CON        4  97.04   24.26  1.0997 0.4006348
VEH        4 922.64  230.66 10.4560 0.0006954 ***
ad         4 115.44   28.86  1.3083 0.3217173
Residuals 12 264.72   22.06
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> source('ICplot.R')
> par(mfrow = c(1,3))
> ICplot(mod_ad , "CON")
> ICplot(mod_ad , "VEH")
> ICplot(mod_ad , "ad")
```



De la tabla anterior se observa que el factor "vehículo" influye significativamente en el consumo de combustible. Sin embargo, el factor "aditivo" NO influye significativamente en el consumo de combustible: a pesar de que la media del aditivo "D" es menor que la del resto de aditivos, los intervalos de confianza se solapan.

5.8. INSTRUCCIONES UTILIZADAS

`aov()`	Ajusta nuestros datos a un modelo lineal. En su primer argumento se escriben la variable dependiente seguida de "~" y después las variables independientes. Las variables independientes se incluyen utilizando "+", "*" o ":".
`anova(aov())`	Nos da la tabla de análisis de la varianza del modelo lineal indicado.
`model.tables(, type='means')`	Genera unas tablas de resumen con las medias para cada factor.
`pairwise.t.test()`	Hace comparaciones dos a dos para los valores de la media de una de las variables según cada valor de otra variable y decide si sus medias son o no iguales. Su primer argumento es la variable a analizar y el segundo es el factor. Puede elegirse el método de ajuste con el argumento *p.adj*.
`TukeyHSD(,)`	Calcula las diferencias entre las medias de los niveles indicados en el segundo argumento, así como también el límite inferior, superior y probabilidad para cada una de las comparaciones. Una forma gráfica de ver este resultado es empleando: `plot(TukeyHSD(,))`
`tapply()`	Nos da el parámetro estadístico indicado de una variable por cada nivel del factor. Su primer argumento son los datos de los que hallar el parámetro y el segundo es el factor. El parámetro se indica en el tercer argumento y puede ser *length*, *mean*, *sd*, etc.
`interaction.plot(,,)`	Genera un gráfico de interacción de la respuesta para una combinación de dos factores, ilustrando por tanto posibles interacciones.
`ks.test()`	Es un test que evalúa la normalidad de unos datos. En su primer argumento indicamos los datos a analizar. En el segundo, escribimos *"pnorm".* Luego podemos o, mejor dicho, debemos elegir la media y la desviación típica escribiendo *mean=m,sd=n*, siendo m y n números cualesquiera.
`sort`	Esta función ordena los datos numéricos de un vector definido previamente en orden creciente por defecto. Para que ordene el vector de forma decreciente, modificamos su argumento *decreasing*, que por defecto es "*=F*" y escribiríamos *decreasing=T*.

6. REGRESIÓN

6.1. INTRODUCCIÓN

El objetivo de este tema es aprender a utilizar el modelo de regresión lineal que permite estudiar la relación de una o varias variables (regresores) con otra variable (variable respuesta). Con este fin se mostrarán varios ejemplos resueltos, uno de regresión simple y dos de regresión múltiple.

6.2. REGRESIÓN SIMPLE

Disponemos una base de datos referente a treinta coches. Para cada uno de los automóviles disponemos de su peso (kilogramos) y de su consumo (litros cada cien kilómetros).

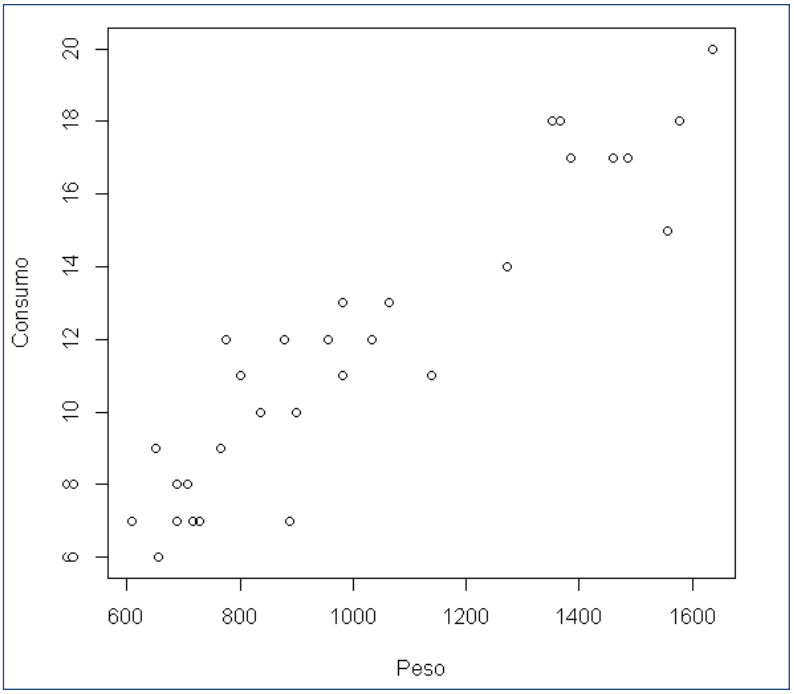
Peso	Consumo
981	11
878	12
708	8
1138	11
1064	13
655	6
1273	14
1485	17
1366	18
1351	18

Peso	Consumo
1635	20
900	10
888	7
766	9
981	13
729	7
1034	12
1384	17
776	12
835	10

Peso	Consumo
650	9
956	12
688	8
716	7
608	7
802	11
1578	18
688	7
1461	17
1556	15

Vamos a analizar estos datos con R. Para ello empleamos los siguientes comandos:

```
> coches <- read.table('coches.txt',header=T)
> attach(coches)
> plot(Peso,Consumo)
```



Mediante el gráfico de dispersión de los datos "consumo" en función de "peso", se observa que siguen una relación lineal. Para calcular los coeficientes del modelo lineal

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

se procede como sigue:

```
> mod_coches <- lm(Consumo ~ Peso)
> summary(mod_coches)

Call:
lm(formula = Consumo ~ Peso)

Residuals:
    Min      1Q  Median      3Q     Max
-3.3456 -0.7680 -0.0596  0.8829  2.9682

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.0712606  0.9451482  -0.075     0.94
Peso         0.0117307  0.0008865  13.232 1.44e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.543 on 28 degrees of freedom
Multiple R-squared: 0.8621,     Adjusted R-squared: 0.8572
F-statistic: 175.1 on 1 and 28 DF,  p-value: 1.436e-13
```

De estos resultados, observamos que:

1) El valor de $\hat{\beta}_0$ es de -0.0712606.

2) El valor de $\hat{\beta}_1$ es de 0.0117307.

3) Si queremos realizar los contrastes

$$\begin{cases} H_0 : \beta_i = 0 \\ H_1 : \beta_i \neq 0 \end{cases}$$

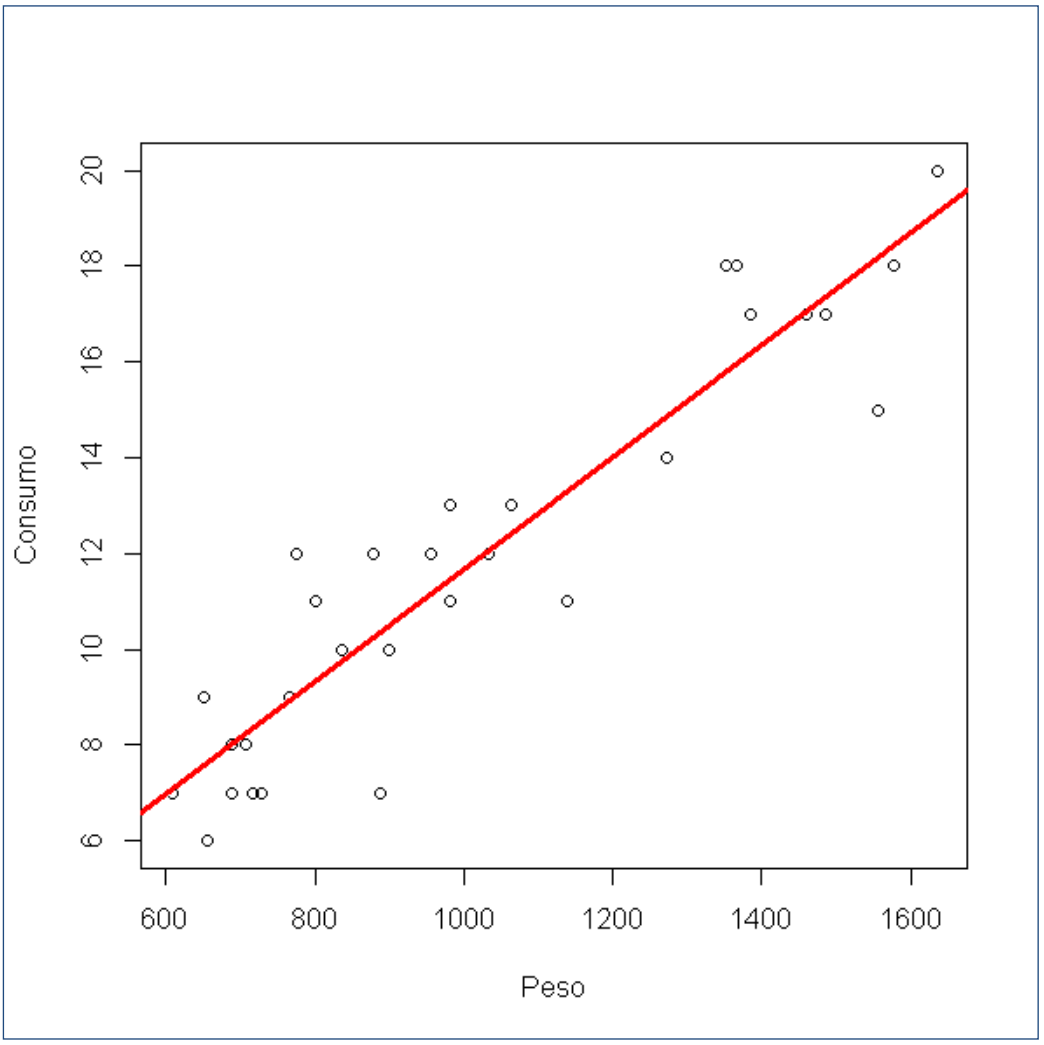
tanto para β_0 como β_1, se hace de forma muy sencilla observando los datos anteriores. Se observa que en el caso de β_0 no podemos rechazar la hipótesis nula, ya que el p-valor es muy elevado. Sin embargo, en el caso de β_1, se observa que el p-valor es muy pequeño y, por tanto, podemos rechazar la hipótesis nula.

4) Otro dato que podemos obtener de la tabla anterior es que el coeficiente de determinación R^2 es igual a 0.8621.

5) El contraste más importante es el relativo a β_1, con este modelo concluimos que el consumo depende muy significativamente (p-valor = 1.44E-13) del peso. Además, teniendo en cuenta las unidades que se ha estimado un aumento de 0.0117 litros/100km por kg.

Finalmente vamos a representar la recta de regresión sobre el gráfico, empleando la siguiente secuencia de comandos:

```
> plot(Peso, Consumo)
> abline(mod_coches, cex = 2, lty = 1, col = 2, lwd = 3)
```



6.3. Regresión Múltiple

Ahora analizaremos de nuevo el consumo de un conjunto de 394 coches. En este caso disponemos de la siguiente información:

- Consumo, medido en l/100Km
- Cilindrada, medido en centrímetros cúbicos
- Potencia medida en caballos
- Peso medido en kilogramos
- Aceleración medida en segundos

A continuación se muestra la tabla con los datos de los primeros seis coches:

Número	Y Consumo	X1 Cilindrada	X2 Potencia	X3 Peso	X4 Aceleración
	l/100Km	*cc*	*CV*	*kg*	*segundos*
1	15	4982	150	1144	12
2	16	6391	190	1283	9
3	24	5031	200	1458	15
4	9	1491	70	651	21
5	11	2294	72	802	19
6	17	5752	153	1384	14
...	...	...	...	...	...

Esta información se encuentra en el fichero "coches2.txt". Procedemos a cargar los datos en R:

```
> coches = read.table('coches2.txt', header = T)
> attach(coches)
> mod_coches = lm(Consumo ~ CC + CV + Peso + Acel)
> summary(mod_coches)

Call:
lm(formula = Consumo ~ CC + CV + Peso + Acel)

Residuals:
    Min      1Q  Median      3Q     Max
-5.1095 -1.0180  0.0322  0.9906  5.5775

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.6695833  0.9833053  -1.698   0.0903 .
CC           0.0003835  0.0001625   2.360   0.0188 *
CV           0.0402844  0.0065697   6.132 2.15e-09 ***
Peso         0.0057842  0.0009578   6.039 3.65e-09 ***
Acel         0.1115012  0.0496757   2.245   0.0254 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.662 on 386 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared: 0.8197,     Adjusted R-squared: 0.8178
F-statistic: 438.7 on 4 and 386 DF,  p-value: < 2.2e-16
```

Como se puede observar, el modelo que implementamos con R es el siguiente:

$$\text{Consumo} = \beta_0 + \beta_1\,\text{CC} + \beta_2\,\text{Pot} + \beta_3\,\text{Peso} + \beta_4\,\text{Acel} + \text{Error}$$

De los resultados obtenidos, podemos deducir que:

- Contraste F=438 (p-valor<0.00001): Alguno de los regresores influye significativamente en el consumo.
- Contrastes individuales:

- La potencia y el peso influyen significativamente (p-valor menor que 0.01)
- Para α=0.05, la cilindrada y la aceleración también tienen efecto significativo (p-valor < 0.05)

- El efecto de cualquier regresor es "positivo", al aumentar cualquiera de ellos aumenta la variable respuesta: consumo. A igualdad del resto de los regresores, el aumento de uno de ellos implica un aumento en el consumo.
- Los regresores explican el 82 % de la variabilidad del consumo ($R^2 = 0.8197$)

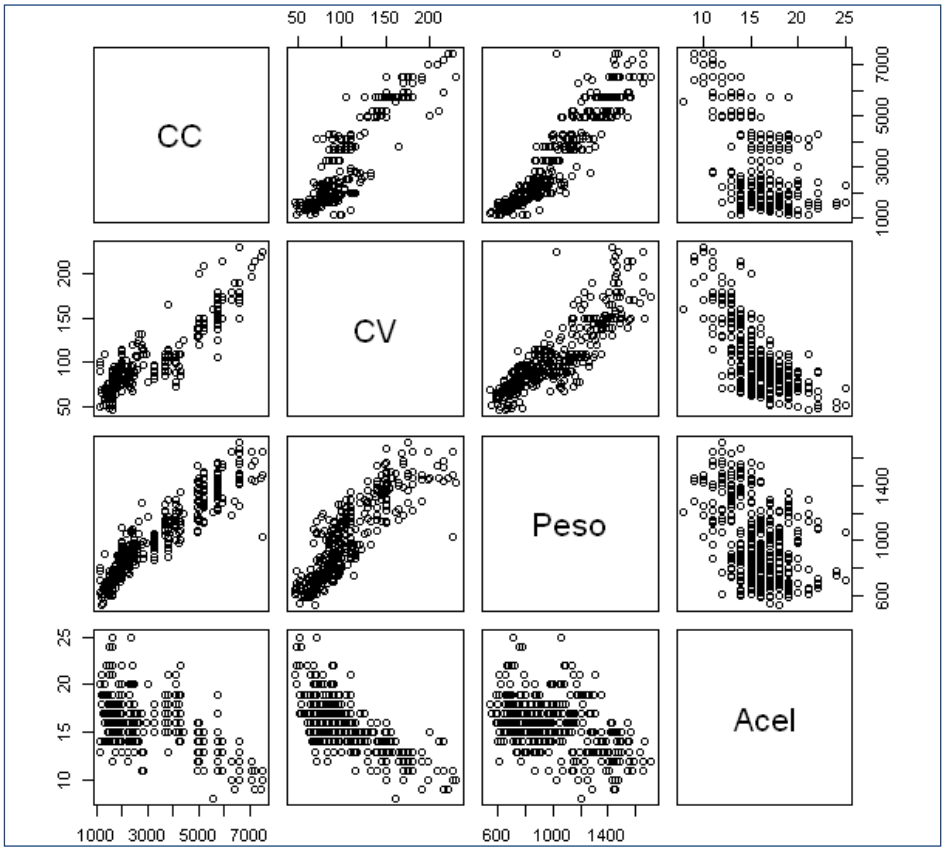
Para identificar una posible colinealidad, calculamos la matriz de correlación de los regresores:

```
> cor(coches[,2:5], use = 'pair')
             CC         CV       Peso       Acel
CC     1.0000000  0.8984444  0.9336283 -0.5474669
CV     0.8984444  1.0000000  0.8628927 -0.6962805
Peso   0.9336283  0.8628927  1.0000000 -0.4212948
Acel -0.5474669 -0.6962805 -0.4212948  1.0000000
```

Observamos que todos los regresores están muy correlacionados. Esto perjudica la estimación de los coeficientes del modelo y puede dificultar la interpretación. Sería interesante interpretar el efecto del tiempo de aceleración sobre el consumo.

En el gráfico siguiente se observan las relaciones lineales entre los regresores.

```
> pairs(coches[,2:5])
```



6.4. VARIABLES CUALITATIVAS

En esta sección introduciremos en el modelo variables cualitativas como regresores. En la base de datos anterior, incluiremos los datos de procedencia de los vehículos. La procedencia puede ser de Europa, de Japón, o de Estados Unidos.

Número	Y Consumo	X1 Cilindrada	X2 Potencia	X3 Peso	X4 Aceleración	Origen
	l/100Km	*cc*	*CV*	*kg*	*segundos*	
1	15	4982	150	1144	12	Europa
2	16	6391	190	1283	9	Japón
3	24	5031	200	1458	15	USA
4	9	1491	70	651	21	Europa
5	11	2294	72	802	19	Japón
6	17	5752	153	1384	14	USA
...	...	...	...	...	...	...

Para incluir las variables cualitativas, lo haremos mediante variables que toman el valor cero o uno.

$$Z_{JAP_i} = \begin{cases} 0 & \text{si } i \notin \text{JAPON} \\ 1 & \text{si } i \in \text{JAPON} \end{cases}$$

$$Z_{USA_i} = \begin{cases} 0 & \text{si } i \notin \text{USA} \\ 1 & \text{si } i \in \text{USA} \end{cases}$$

$$Z_{EUR_i} = \begin{cases} 0 & \text{si } i \notin \text{EUROPA} \\ 1 & \text{si } i \in \text{EUROPA} \end{cases}$$

El modelo que emplearemos en este caso será:

$$\text{Consumo} = \beta_0 + \beta_1 \text{CC} + \beta_2 \text{Pot} + \beta_3 \text{Peso} + \\ + \beta_4 \text{Acel} + \alpha_{JAP} Z_{JAP} + \alpha_{USA} Z_{USA} + \text{Error}$$

Insertamos la siguiente secuencia de comandos en R:

```
> ZUSA = Origen == 1
> ZEUR = Origen == 2
> ZJAP = Origen == 3
> mod_coches = lm(Consumo ~ CC + CV + Peso + Acel + ZJAP  + ZUSA)
```

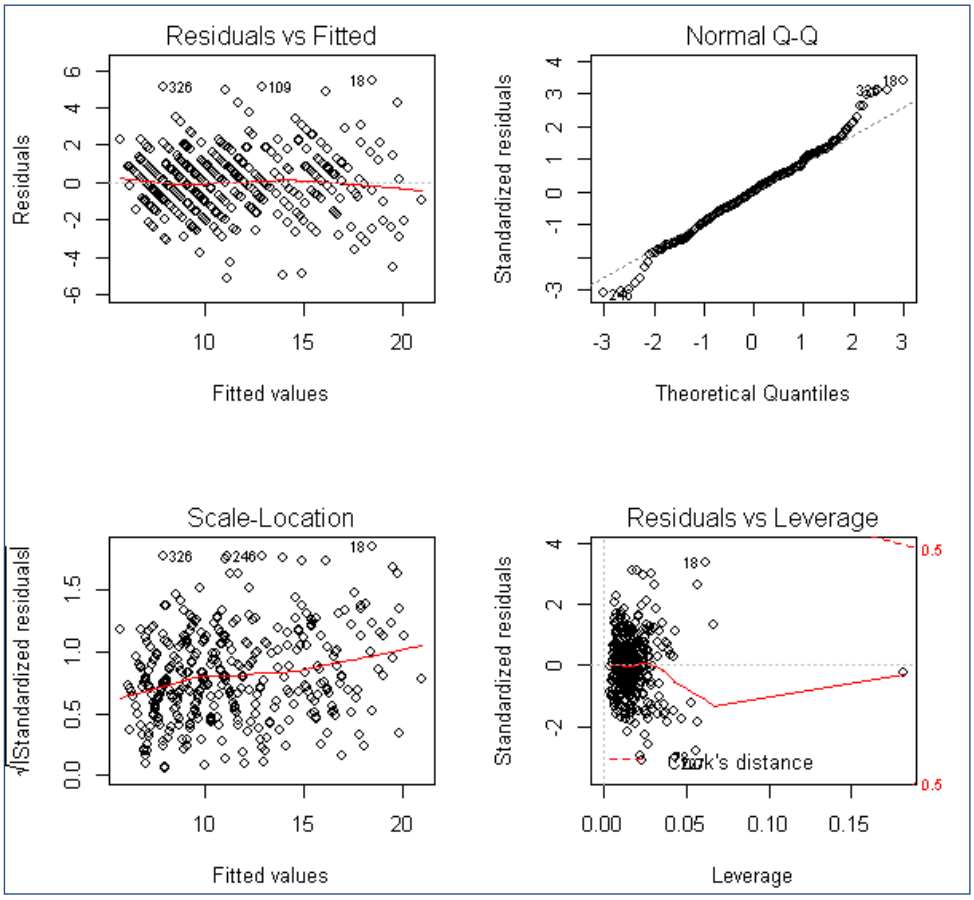
```
> summary(mod_coches)

Call:
lm(formula = Consumo ~ CC + CV + Peso + Acel + ZJAP + ZUSA)

Residuals:
    Min      1Q  Median      3Q     Max
-5.0965 -0.9687  0.0213  0.9496  5.4896

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.4550414  1.0172459  -1.430   0.1534
CC           0.0003228  0.0001792   1.801   0.0724 .
CV           0.0422677  0.0067890   6.226 1.26e-09 ***
Peso         0.0055996  0.0009655   5.799 1.39e-08 ***
Acel         0.1108411  0.0496919   2.231   0.0263 *
ZJAPTRUE    -0.3617622  0.2790488  -1.296   0.1956
ZUSATRUE     0.0611229  0.2802356   0.218   0.8275
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.66 on 384 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared: 0.821,      Adjusted R-squared: 0.8182
F-statistic: 293.5 on 6 and 384 DF,  p-value: < 2.2e-16
```

De los resultados anteriores se deduce lo siguiente:

- El p-valor del coeficiente asociado a Z_{JAP} es 0.1956 > 0.05, se concluye que no existe diferencia significativa entre el consumo de los coches Japoneses y Europeos (manteniendo constante el peso, cc, pot y acel.)
- La misma interpretación para Z_{USA}.
- Comparando R^2 =0.821 de este modelo con el anterior R^2=0.8197, se confirma que el modelo con las variables de Origen no suponen una mejora sensible.

Finalmente hacemos la diagnosis de los residuos, a partir de la cual concluimos que podemos asumir la hipótesis de homocedasticidad, normalidad e independencia.

```
> par(mfrow = c(2,2))
> plot(mod_coches)
```



6.5. Ejemplo de Regresión Múltiple

El objetivo de este problema es construir un modelo de regresión múltiple para explicar el volumen maderable de un árbol a partir de su diámetro y su altura. Se dispone de los datos correspondientes a la especie cerezo negro.

Se ha tomado una muestra de 31 árboles. A continuación se muestran los datos.

Posición	**altura** *pies*	**diámetro** *pies*	**volumen** *pies cúbicos*
1	70	8.3	10.3
2	65	8.6	10.3
3	63	8.8	10.2
4	72	10.5	16.4
5	81	10.7	18.8
6	83	10.8	19.7
7	66	11	15.6
8	75	11	18.2
9	80	11.1	22.6
10	75	11.2	19.9
11	79	11.3	24.2
12	76	11.4	21
13	76	11.4	21.4

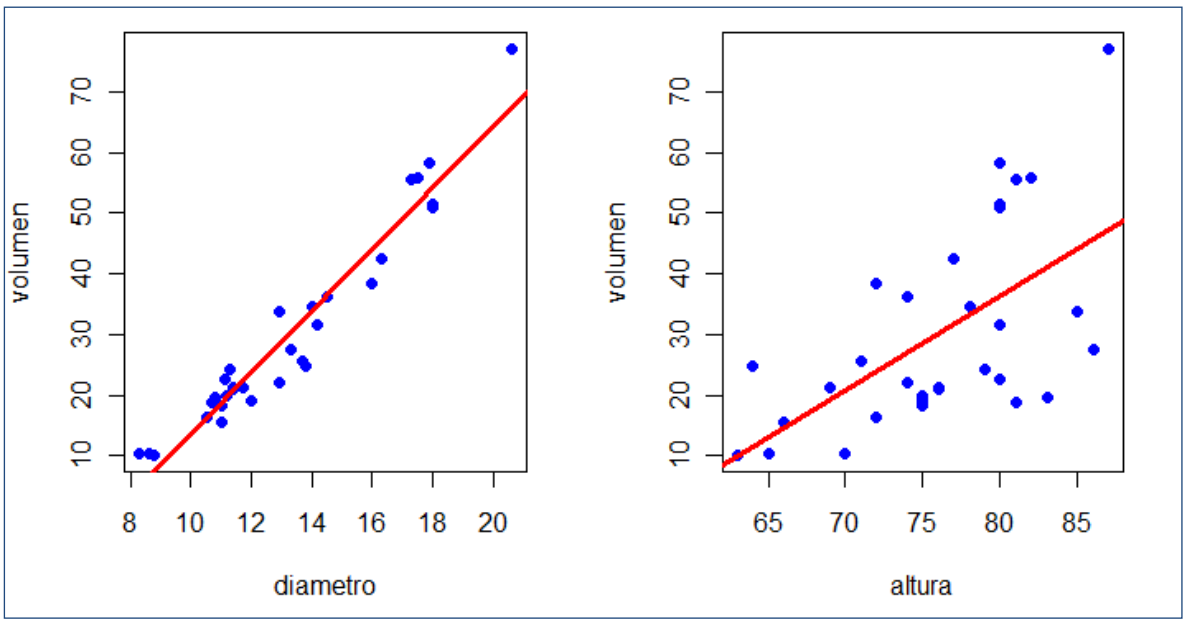
14	69	11.7	21.3
15	75	12	19.1
16	74	12.9	22.2
17	85	12.9	33.8
18	86	13.3	27.4
19	71	13.7	25.7
20	64	13.8	24.9
21	78	14	34.5
22	80	14.2	31.7
23	74	14.5	36.3
24	72	16	38.3
25	77	16.3	42.6
26	81	17.3	55.4
27	82	17.5	55.7
28	80	17.9	58.3
29	80	18	51.5
30	80	18	51
31	87	20.6	77

En primer lugar, cargamos los datos en R.

```
> datos<-read.table("cerezos.txt",header=T)
> attach(datos)
```

Y ahora vamos a generar los gráficos de dispersión “volumen” en función de “diámetro” y en función de “altura”. En los gráficos mostraremos las rectas de regresión resultantes.

```
> par(mfrow = c(1,2))
> plot(diametro, volumen, pch = 20, col='blue', cex = 1.5)
> m1 = lm(volumen ~ diametro)
> abline(m1, col = 'red', lwd = 3)
> plot(altura, volumen, pch = 20, col='blue', cex = 1.5)
> m2 = lm(volumen ~ altura)
> abline(m2, col='red', lwd = 3)
```



Emplearemos primero el siguiente modelo:

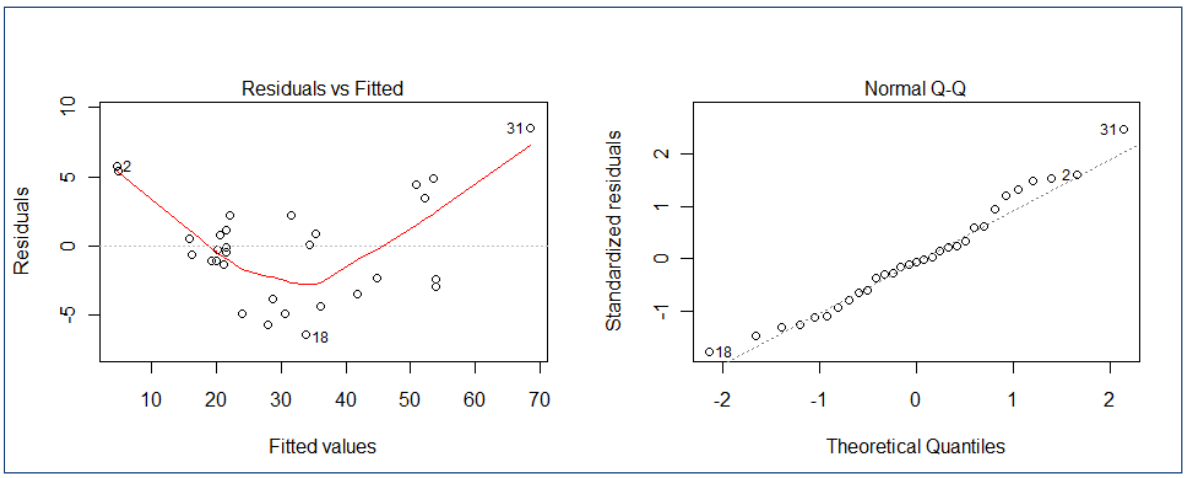
$$\text{Volumen} = \beta_0 + \beta_1 \text{ Diametro} + \beta_2 \text{ Altura} + \text{Error}$$

Mediante la siguiente secuencia de comandos se implementa el modelo anterior en R, obteniendo los siguientes resultados:

```
> mod1 = lm(volumen ~ diametro + altura)
> summary(mod1)

Call:
lm(formula = volumen ~ diametro + altura)

Residuals:
    Min      1Q  Median      3Q     Max
-6.4065 -2.6493 -0.2876  2.2003  8.4847

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -57.9877     8.6382  -6.713 2.75e-07 ***
diametro      4.7082     0.2643  17.816  < 2e-16 ***
altura        0.3393     0.1302   2.607   0.0145 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.882 on 28 degrees of freedom
Multiple R-squared: 0.948,      Adjusted R-squared: 0.9442
F-statistic:   255 on 2 and 28 DF,  p-value: < 2.2e-16
```

Hacemos la diagnosis del modelo:

```
> par(mfrow=c(1,2))
> plot(mod1)
```



Del gráfico superior se observa que no se cumple la hipótesis de homocedasticidad ni linealidad.

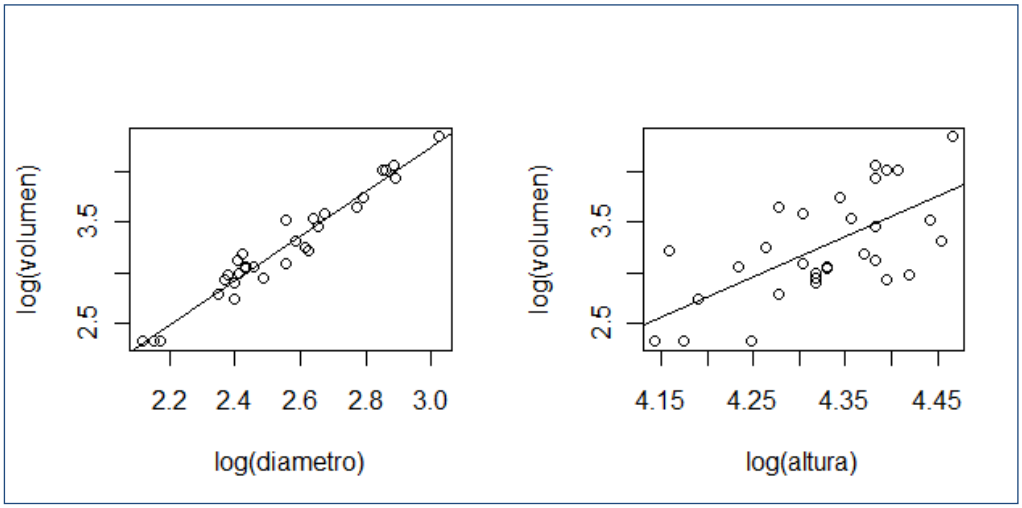
Por tanto, pasamos a hacer una transformación. Para seleccionar la transformación adecuada, nos basamos en el conocimiento de la fórmula del volumen:

$$\text{vol} \approx \text{k} \times \text{altura} \times \text{diámetro}^2$$

De donde, tomando logaritmos:

$$\log(\text{vol}) \approx \beta_0 + \beta_1 \log(\text{altura}) + \beta_2 \log(\text{diámetro}) + \text{error}$$

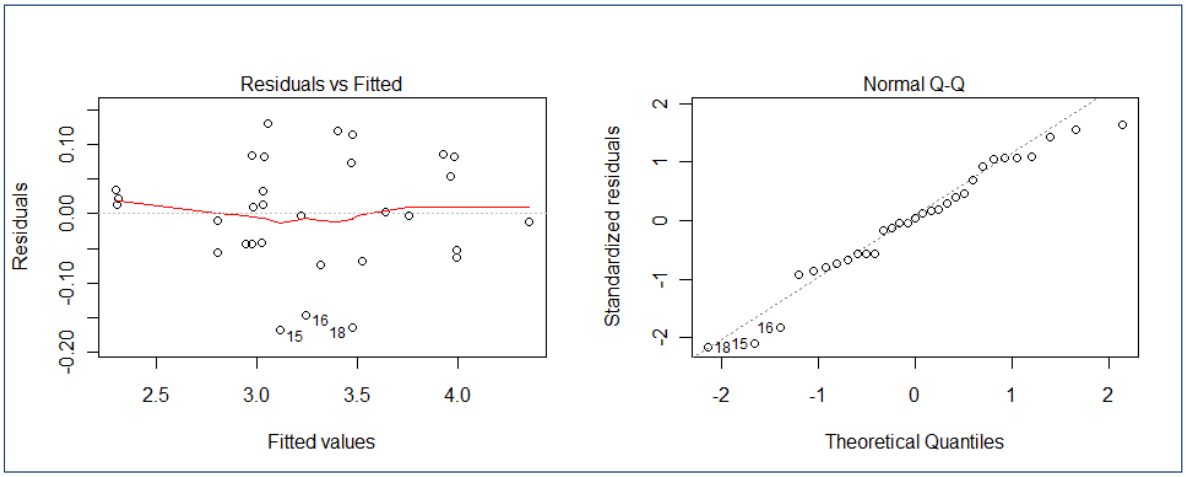
Así que tomamos logaritmos en todas las variables, obteniendo los siguientes gráficos de dispersión de la variable "volumen" en función de "diámetro" y en función de "altura".



A continuación, definimos el nuevo modelo en R, y mostramos la tabla ANOVA:

```
> mod2<-lm(log(volumen) ~ log(diametro) + log(altura))
> summary(mod2)

Call:
lm(formula = log(volumen) ~ log(diametro) + log(altura))

Residuals:
      Min        1Q    Median        3Q       Max
-0.168561 -0.048488  0.002431  0.063637  0.129223

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -6.63162    0.79979  -8.292 5.06e-09 ***
log(diametro)  1.98265    0.07501  26.432  < 2e-16 ***
log(altura)    1.11712    0.20444   5.464 7.81e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.08139 on 28 degrees of freedom
Multiple R-squared: 0.9777,     Adjusted R-squared: 0.9761
F-statistic: 613.2 on 2 and 28 DF,  p-value: < 2.2e-16
```

Y, finalmente, haciendo la diagnosis, vemos que los residuos pueden suponerse lineales y homocedásticos.



6.6. INSTRUCCIONES UTILIZADAS

Funciones que actúan sobre el objeto creado con la función lm()

`summary()`	Devuelve toda la información relevante acerca del modelo lineal.
`plot()`	Hace gráficos de diagnóstico.
`coef()`	Devuelve el valor de los coeficientes del modelo.
`residuals()`	Devuelve el valor de los residuos del modelo.
`fitted()`	Devuelve el valor de los valores estimados de la variable dependiente $\hat{y}_i$.
`deviance()`	Calcula el valor de la variabilidad no explicada.
`predict()`	Realiza predicciones.
`anova()`	Construye la tabla ANOVA.
`AIC()`	Criterio de ACAIKE utilizado para hacer selección de modelos.
`Cooks.distance()`	Calcula la distancia de Cooks muy útil para detectar puntos atípicos.

7. ANEXO I

A continuación se muestra el contenido de la función ICplot.

El contenido del archivo "ICplot.R" es el siguiente:

```
ICplot <- function(modelo,fac,alpha=0.05)
{
      tabla <- model.tables(modelo, type = "mean")
      xbar <- tabla$table[[fac]]
      num_dat <-tabla['n']
      num <- num_dat$n[fac]
      t <- qt((1-alpha/2),modelo$df.residual)
      sr2 <- sum((modelo$residuals)^2)/modelo$df.residual
      sr <- sqrt(sr2)
      ancho <- t*sr/sqrt(num)

      ncol  = length(xbar)
      xlabel =  modelo$xlevels[[fac]]

      plot(c(1:ncol, 1:ncol), c(xbar+ancho, xbar-ancho), col = 0,
      xlab = fac, xaxt = "n", ylab = "medias")
         axis(side=1, at=seq(1,ncol), paste(xlabel))

      arrows(1:ncol,xbar+ancho,1:ncol,xbar-
      ancho,angle=90,code=3,length=.1,
      lwd = 2, col = 259)
      points(1:ncol, xbar, lwd = 10, col = "white")
      points(1:ncol, xbar, lwd = 3, col = "blue")
}
```