Introducción al análisis de datos con R

Alfonso Urquía Moraleda
Carla Martín Villalba

Texto base de la actividad formativa transversal
GESTIÓN Y ANÁLISIS DE DATOS CIENTÍFICOS
del programa de doctorado en Ingeniería de Sistemas y de Control

Segunda Edición, septiembre de 2022

Dpto. Informática y Automática, ETS Ingeniería Informática, UNED
Juan del Rosal 16, 28040, Madrid, España

ÍNDICE

PREFACIO

ORGANIZACIÓN DE LA UNIDAD DIDÁCTICA

La presente Unidad Didáctica consta de un breve preámbulo y dos temas. En el preámbulo se describe el papel que juega el análisis de datos en el estudio de sistemas, tanto si se experimenta directamente sobre ellos, como si se plantea un modelo matemático que reproduzca el comportamiento de interés del sistema y se experimenta simulando el modelo en un ordenador. En el primer tema se explican los fundamentos del lenguaje R. En el segundo tema se describen algunas técnicas básicas de análisis y modelado de datos, y se muestra cómo pueden aplicarse usando R.

El análisis de datos es una disciplina amplia y compleja, con innumerables aplicaciones. El propósito de esta obra es servir como guía de iniciación, de modo que a partir de lo aprendido en ella el doctorando pueda profundizar por sí mismo en aquellos aspectos del análisis de datos que sean de utilidad para su investigación.

CÓMO UTILIZAR EL LIBRO

El contenido del breve preámbulo está autocontenido, no siendo su lectura imprescindible para poder comprender los restantes temas. Resulta aconsejable leer los dos temas en orden.

Asimismo, es recomendable que el doctorando instale en su propio ordenador un IDE para el lenguaje R y que ejecute por sí mismo el código mostrado en los Temas 1 y 2.

OBJETIVOS DOCENTES

Se plantea como objetivo docente que el doctorando adquiera la capacidad de:

1. Discutir qué papel desempeña del análisis de datos en el estudio de sistemas en el ámbito de la Ingeniería de Sistemas y de Control, tanto si la experimentación se realiza directamente sobre el sistema real, como si se realiza mediante modelado matemático y simulación por ordenador.
2. Discutir los conceptos fundamentales del lenguaje R.
3. Discutir técnicas para el análisis de la independencia y homogeneidad de los datos.
4. Seleccionar la familia de distribuciones teóricas que mejor se ajusta a un conjunto de datos empleando técnicas gráficas (histogramas y gráficas cuantil-cuantil).
5. Aplicar técnicas gráficas para medir la bondad del ajuste a los datos de una distribución de probabilidad.
6. Realizar análisis y modelos sencillos de los datos usando R.
7. Comparar grupos de datos usando R.

ANÁLISIS DE DATOS EN INGENIERÍA DE SISTEMAS Y DE CONTROL

El análisis de datos juega un papel central en la Ingeniería de Sistemas y de Control. En este contexto, la propia definición de qué se entiende por sistema comprende el concepto de datos. Así, puede considerarse que un **sistema** es cualquier objeto cuyas propiedades se desean estudiar. Por tanto, puede considerarse que *cualquier fuente potencial de datos* es un sistema.

Una manera de conocer el comportamiento de un sistema es experimentar con él. Éste ha sido el método empleado durante siglos para avanzar en el conocimiento: plantear preguntas acerca del comportamiento de los sistemas y responderlas mediante experimentación. Un **experimento** es *el proceso de extraer datos de un sistema sobre el cual se ha ejercido una acción externa.*

En la Figura 1 se muestran los pasos que típicamente se siguen en el diseño y realización de experimentos. El Paso 1 habitualmente se realiza aplicando técnicas de diseño de experimentos. El Paso 3, consistente en el análisis de las respuestas experimentales, normalmente se realiza aplicando técnicas y herramientas software para análisis de datos.

Experimentar directamente con el sistema real presenta indudables ventajas. Sin embargo, en ocasiones no es posible. Quizá el motivo más evidente es que el sistema real aun no exista físicamente. Esto sucede por ejemplo en la fase de diseño de nuevos sistemas, cuando se necesita predecir el comportamiento de los mismos antes de que sean construidos. Otro motivo es que la escala de tiempo del experimento lo haga inviable.

Aun siendo posible experimentar directamente con el sistema real, en ocasiones es desaconsejable debido a su elevado coste económico. Consideremos el problema de decidir si se realiza o no ciertos cambios en un proceso de fabricación. La resolución de este problema conlleva estimar si la ganancia potencial que supondrá el nuevo proceso justifica el coste que tendrá su realización. Experimentar directamente con

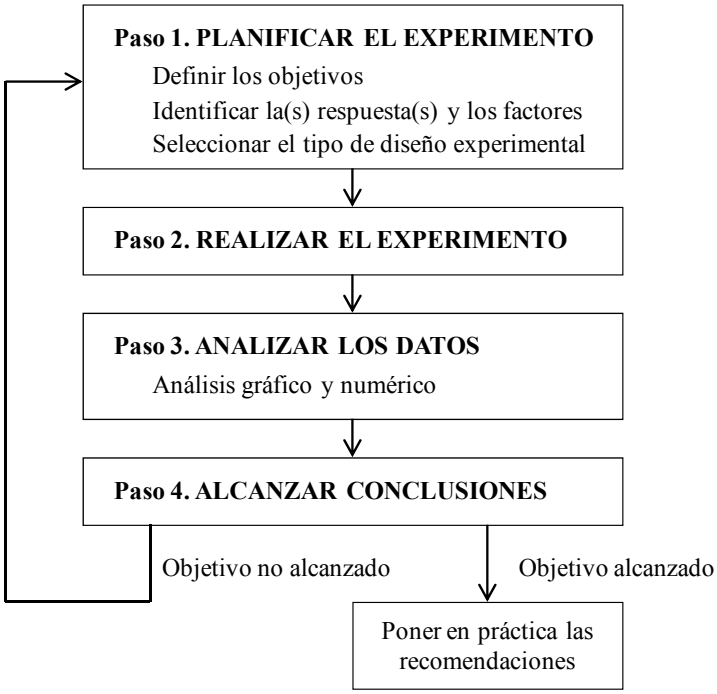

Figura 1: Pasos en el diseño y realización de experimentos.

el sistema real supondría realizar el cambio con el fin de evaluar su rendimiento económico, lo cual no parece razonable.

Otro posible motivo para no experimentar con el sistema real es que el experimento produzca perjuicio, incomodidad o sea peligroso. Así, experimentar con un nuevo sistema de facturación en un aeropuerto puede producir retrasos y problemas imprevisibles que perjudiquen al viajero. Un ejemplo de experimento peligroso sería usar el sistema real para entrenar a los operarios de una central nuclear acerca de cómo deben reaccionar ante situaciones de emergencia.

Una alternativa a la experimentación con el sistema real consiste en realizar un modelo del sistema y experimentar con el modelo. Experimentar con modelos resulta en ocasiones menos costoso y más seguro que experimentar directamente con el sistema real. Otra ventaja de la experimentación con modelos es que, con un modelo adecuado, se pueden ensayar condiciones de operación extremas que son impracticables en el sistema real. Este es el caso cuando el experimento requiere modificar variables que en el sistema real o bien no están accesibles, o bien no pueden ser modificadas en el rango requerido.

En la Ingeniería de Sistemas y de Control se emplean frecuentemente **modelos matemáticos**, que son resueltos mediante ordenador empleando algoritmos y métodos matemáticos numéricos. El análisis de datos desempeña un papel fundamental en los estudios mediante modelado y simulación. Así, el análisis de datos se aplica al modelar las entradas al modelo, al estimar el valor de los parámetros del modelo, al validar el modelo, al analizar los resultados de la simulación, y al documentar y divulgar los resultados del estudio.

Tema 1

INTRODUCCIÓN AL LENGUAJE R

1.1. INTRODUCCIÓN

El análisis de datos es parte esencial del estudio de sistemas. En particular, es una parte esencial en el planteamiento de los modelos matemáticos y también en la toma de decisiones basada en los resultados de la simulación de los modelos. Por ello, las herramientas de modelado y simulación empleadas en Ingeniería de Sistemas y de Control a menudo facilitan el análisis de datos. Algunas de estas herramientas, como por ejemplo Matlab/Simulink y Scilab/Scicos, permiten aplicar una gran variedad de técnicas para el análisis y modelado de los datos. Sin embargo, lo más habitual es que las capacidades para análisis de datos ofrecidas por las herramientas de simulación se limiten a la realización de unas determinadas representaciones gráficas básicas de los datos. En este último caso, el software de modelado y simulación debe usarse en combinación con alguna herramienta especializada en el análisis y modelado de los datos.

En este tema se introduce un lenguaje para el análisis de datos denominado R, así como una de las plataformas software que lo soporta, llamada RGui. Se trata de software gratuito y de código abierto, con versiones para los sistemas operativos Windows, Mac OS X y Linux. Puede descargarse del Comprehensive R Archive Network (CRAN), que está ubicado en `http://cran.r-project.org/`.

R es un lenguaje interpretado, en el cual se distingue entre letras mayúsculas y minúsculas. Es posible ir introduciendo los comandos uno a uno en la línea de comandos de la consola, tras el símbolo del sistema (>), o bien ejecutar un conjunto de comandos escritos en un fichero. En la Figura 1.1 se muestra un ejemplo de la consola de RGui en Windows.

El símbolo del sistema (>) que RGui muestra en la consola indica que está preparado para aceptar un comando. Por ejemplo, si escribimos `10 + 1.3` y pulsamos la tecla ENTER, se obtiene el resultado de la operación y vuelve a aparecer el símbolo del sistema, lo cual indica que RGui está listo para recibir un nuevo comando:

```
> 10 + 1.3
 [1] 11.3
>
```

Obsérvese que los comandos introducidos por el usuario se escriben en color rojo, mientras que el texto escrito por RGui aparece en color azul.

El lenguaje R tiene una gran variedad de tipos de datos, incluyendo vectores, matrices, data frames y listas. La mayor parte de la funcionalidad se consigue mediante el uso de funciones, tanto las proporcionadas por el lenguaje como las

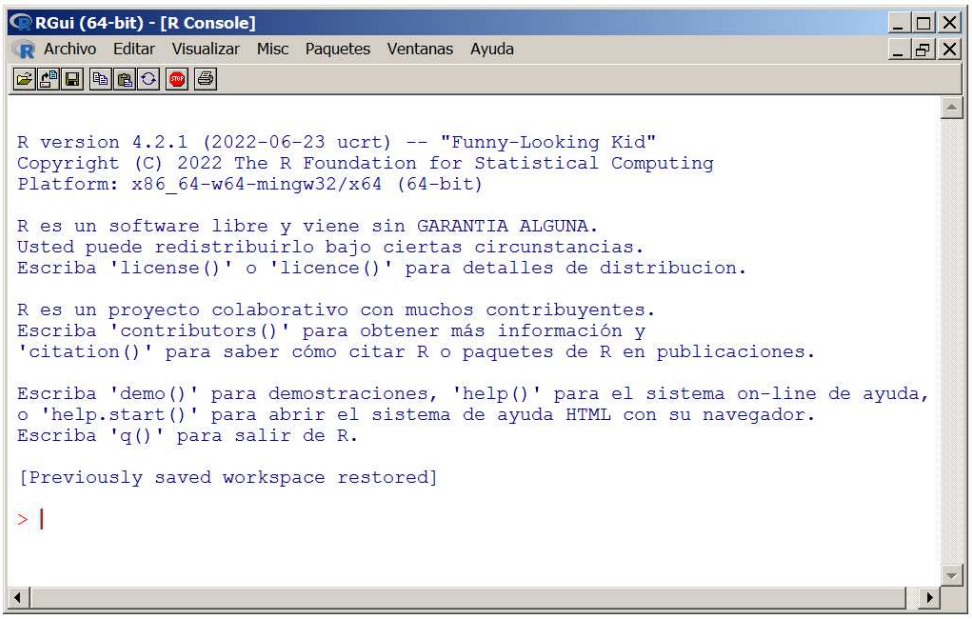

Figura 1.1: Ejemplo de la consola de RGui en Windows.

definidas por el propio usuario. Algunas funciones básicas están disponibles por defecto. Otras se encuentran en paquetes, que deben ser abiertos (esta acción se denomina *attach*) para poder usar las funciones que contienen.

Las sentencias consisten en funciones y asignaciones. R usa el símbolo <- para las asignaciones, en lugar del típico símbolo =. Por ejemplo,

```
> x <- rnorm(5)
```

crea un objeto de tipo vector llamado `x`, que contiene 5 observaciones independientes de la distribución normal estándar. Escribiendo el nombre del objeto, se muestra su contenido. Por ejemplo:

```
> x <- rnorm(5)
> x
[1] -0.1072172 -0.8227021  0.1917662 -0.1117749  0.2237148
```

Los comentarios son precedidos por el símbolo `#`. El intérprete de R ignora todo el texto que aparezca tras el símbolo `#`. Para finalizar una sesión debe ejecutarse la función `q()`.

La función `c()` convierte sus argumentos en un vector o una lista. Por ejemplo, supongamos que se realizan 5 réplicas independientes de la simulación del modelo de

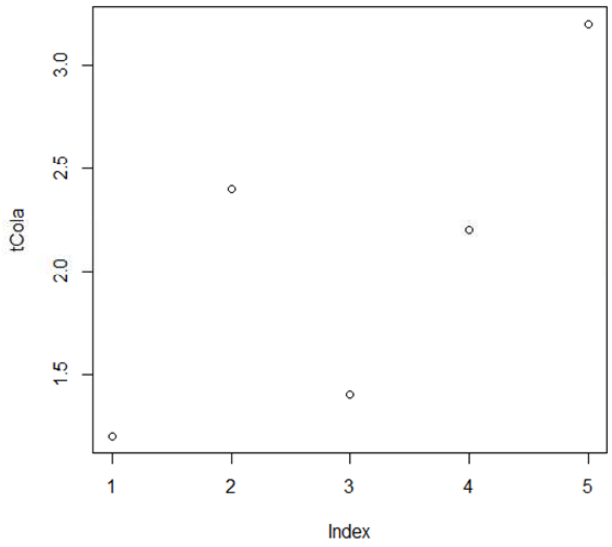

Figura 1.2: Ejemplo de uso de la función *plot*.

un sistema logístico, obteniéndose los siguientes 5 valores del tiempo medio de espera en una determinada cola: 1.2, 2.4, 1.4, 2.2 y 3.2 minutos. La primera de las siguientes dos sentencias crea un objeto del tipo vector llamado `tCola`, cuyos componentes son esos cinco valores. La segunda sentencia dibuja el valor del componente del vector frente al índice. En la Figura 1.2 se muestra el gráfico.

```
> tCola <- c(1.2, 2.4, 1.4, 2.2, 3.2)
> plot(tCola)
```

La función `help()` permite acceder a la documentación de las funciones. Por ejemplo,

```
> help(plot)
```

abre una ventana en la cual se muestra la documentación para la función `plot`.

RGui incluye demos que ilustran algunas de sus capacidades para la representación gráfica. Por ejemplo,

```
> demo(graphics)
```

arranca una de ellas. Otras dos demos relacionadas con las capacidades gráficas se arrancan mediante: `demo(persp)` y `demo(image)`. Puede obtenerse la lista completa de demos ejecutando:

```
> demo()
```

1.2. EL ESPACIO DE TRABAJO

El **espacio de trabajo** es el espacio de memoria en el cual se guardan todos los objetos creados durante la sesión. Es posible salvar el espacio de trabajo al finalizar la sesión (RGui pregunta antes de cerrarse si debe hacerlo), de modo que automáticamente sea vuelto a cargar la siguiente vez que se arranque RGui. Puede obtenerse una lista de todos los objetos del espacio de trabajo mediante:

```
> ls()
```

La función `rm()` elimina del espacio de trabajo los objetos que se le pasan como argumento. Por ejemplo,

```
> rm(x)
```

elimina del espacio de trabajo el objeto llamado `x`.

El **directorio de trabajo** es donde RGui salva por defecto los resultados y también de donde intenta por defecto leer los ficheros. Para saber cuál es el directorio de trabajo hay que ejecutar

```
> getwd()
```

Es recomendable separar en diferentes directorios los diferentes proyectos. La función `setwd()` permite modificar el directorio de trabajo, pasando como argumento a la función el nombre del nuevo directorio escrito entre comillas. Por ejemplo:

```
> setwd("F:/Simulacion")
```

La función `save.image` salva el espacio de trabajo completo a un fichero. Para salvar únicamente algunos objetos puede emplearse `save()`. La función `load()` carga en la sesión actual el espacio de trabajo almacenado en el fichero que se le pasa como argumento. Si se desea acceder a un fichero que no se encuentra en el directorio de trabajo, es necesario especificar el path completo en la llamada.

RGui guarda memoria de los comandos ejecutados durante la sesión. Con las flechas del teclado es posible moverse por la historia de comandos. Las funciones `savehistory()` y `loadhistory()` permiten salvar a fichero los comandos de la sesión y cargar dicha historia desde un fichero.

Es posible escribir los comandos en un fichero de texto y cargar este **fichero de comandos** desde RGui, de manera que se ejecuten en secuencia todos los comandos escritos en él. Por convenio, el nombre del fichero con los comandos (también llamado *fichero script*) se escribe con extensión .R. La función `source()` permite cargar y ejecutar el fichero. Por ejemplo,

```
> source("F:/Simulacion/script1.R")
```

carga y ejecuta el fichero de comandos llamado *script1.R*. En la práctica suele resultar más cómodo trabajar empleando ficheros de comandos que hacerlo escribiendo directamente las sentencias en la consola de RGui.

1.3. ESTRUCTURAS DE DATOS

El primer paso en el análisis de los datos es crear una estructura de datos que contenga los datos a estudiar. Los datos pueden ser cargados en la estructura de datos bien manualmente o bien pueden ser importados desde una fuente externa. R tiene básicamente cinco tipos diferentes de estructura de datos: vector, matriz, array, data frame y lista. Los vectores, matrices y arrays contienen números y tienen una, dos o más dimensiones.

Un data frame es una tabla bidimensional, pero más general que una matriz, dado que en el data frame el tipo de los datos almacenados en una columna puede diferir del tipo de los datos almacenados en otra. Los datos almacenados en una misma columna deben ser del mismo tipo: cadena de caracteres, numérico o lógico (TRUE, FALSE).

Las sentencias siguientes crean un data frame llamado `exp1` con tres columnas (las columnas del data frame son denominadas sus *variables*):

```
> numOper <- c(1,1,2,2)
> horario <- c("std", "nuevo", "std", "nuevo")
> tCola   <- c(12.3, 6.1, 5.2, 2.5)
> exp1    <- data.frame(numOper, horario, tCola)
> exp1
  numOper horario tCola
1       1     std  12.3
2       1   nuevo   6.1
3       2     std   5.2
4       2   nuevo   2.5
```

Otra forma de introducir o modificar los datos del data frame es mediante la función `edit()`. Para ello, en primer lugar debe crearse el objeto vacío, especificando

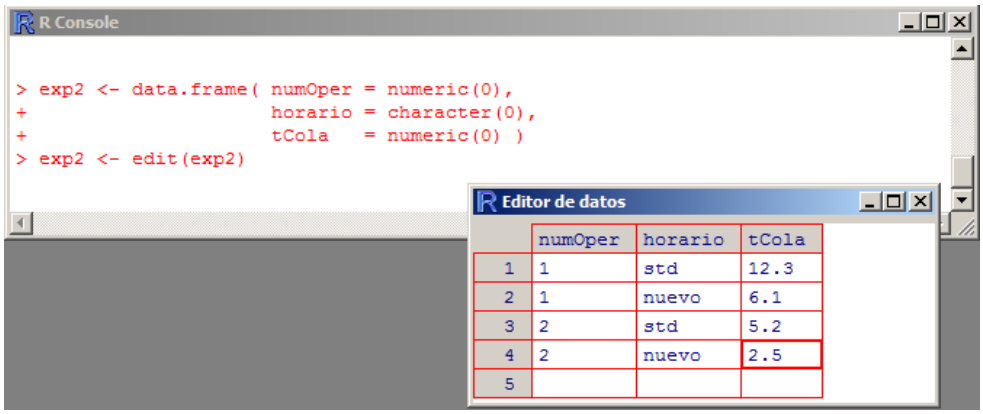

Figura 1.3: Ejemplo de uso de la función *edit* para editar el contenido de un data frame.

el nombre y tipo de las variables. A continuación, se invoca el editor de texto mediante la función `edit()` y se introducen los datos empleando dicho editor. Por ejemplo:

```
> exp2 <- data.frame( numOper = numeric(0),
                      horario = character(0),
                      tCola   = numeric(0) )
> exp2 <- edit(exp2)
```

En la Figura 1.3 se muestra el aspecto de la ventana de edición una vez se han introducido en ella los datos.

Otra forma de introducir datos en el data frame es importarlos desde un fichero de texto. Para ello puede emplearse la función `read.table()`. Supongamos que el contenido del fichero *data.txt* es:

```
numOper horario tCola
1       std     12.3
1       nuevo   6.1
2       std     5.2
2       nuevo   2.5
```

Obsérvese que en la primera fila se han escrito los nombres de las variables. En las siguientes filas están los valores separados por un delimitador, que en este caso es el espacio en blanco (puede usarse cualquier otro símbolo como delimitador).

Mediante las dos siguientes sentencias se crea un data frame llamado `exp3`, se cargan en él valores contenidos en el fichero *data.txt* y a continuación se muestra el contenido de `exp3`. Si el fichero con los datos no se encuentra en el directorio de trabajo, debe especificarse el path completo. Mediante `header=TRUE` se indica que la primera fila del fichero contiene los nombres de las variables.

```
> exp3 <- read.table("data.txt", header=TRUE)
> exp3
   numOper horario tCola
 1       1     std  12.3
 2       1   nuevo   6.1
 3       2     std   5.2
 4       2   nuevo   2.5
```

Puede accederse a una variable del data frame indicando su número de orden. La primera variable es la columna 1, la segunda la 2 y así sucesivamente. Otra posibilidad es escribir el nombre del data frame, seguido del símbolo `$` y del nombre de la variable. Por ejemplo, las dos siguientes sentencias son equivalentes:

```
> exp1[,3]
 [1] 12.3  6.1  5.2  2.5
> exp1$tCola
 [1] 12.3  6.1  5.2  2.5
```

De forma similar puede accederse a los elementos de cada variable. Por ejemplo, mediante estas dos sentencias se accede al cuarto elemento de la variable `tCola` (los índices en R comienzan en el valor 1):

```
> exp1$tCola[4]
 [1] 2.5
> exp1[4,3]
 [1] 2.5
```

Las siguientes funciones facilitan la manipulación de los datos. La función `pretty()` se emplea a menudo para dibujar gráficos.

`length(x)`	Devuelve la longitud del objeto `x`.
`seq(desde,hasta,paso)`	Devuelve una secuencia. El paso por defecto es uno.
`rep(x,n)`	Repite `x` `n` veces.
`pretty(x,n)`	Divide `x` en aproximadamente `n` intervalos iguales, de modo que los puntos de división tengan valores redondeados. Devuelve los puntos de división.

Los siguientes son algunos ejemplos de uso de estas funciones:

```
> seq(0, 10)     # El paso por defecto es uno
 [1]  0  1  2  3  4  5  6  7  8  9 10
> seq(1.5, 5)    # El paso por defecto es uno
 [1] 1.5 2.5 3.5 4.5
> seq(0, 10, by=2)
```

```
 [1]  0  2  4  6  8 10
> x <- seq(1, 10, by=2)
> x
 [1] 1 3 5 7 9
> length(x)
 [1] 5
> y <- rep(x, 2)
> y
 [1] 1 3 5 7 9 1 3 5 7 9
> pretty(c(1,5), 10)
 [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

Es posible generar la secuencia en orden decreciente asignando un valor negativo al parámetro `by`.

```
> seq(1, 10, by=2)
 [1] 1 3 5 7 9
> seq(10, 1, by=-2)
 [1] 10  8  6  4  2
```

Asimismo, en lugar de especificar el tamaño del paso, es posible especificar el número de elementos de la secuencia.

```
> seq(0, 1, length.out = 11)
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

La función `rep` admite varios argumentos opcionales, que modifican su comportamiento. A continuación se muestra un ejemplo de uso de los argumentos `each` (repite cada elemento de `x`) y `length.out` (repetición mediante ciclo incompleto). El número de repeticiones pasado como argumento a la función `rep` puede ser un vector, a fin de indicar que cada elemento se repita un determinado número de veces.

```
> x <- 1:4
> x
 [1] 1 2 3 4
> rep(x,2)
 [1] 1 2 3 4 1 2 3 4
> rep(x, each=2)
 [1] 1 1 2 2 3 3 4 4
> rep(x, c(2,3,1,5))
 [1] 1 1 2 2 2 3 4 4 4 4 4
> rep(x, 3, length.out=9)
 [1] 1 2 3 4 1 2 3 4 1
```

1.4. GRÁFICOS

La representación gráfica de los datos es una herramienta fundamental para su análisis. En ocasiones representar gráficamente los datos permite detectar patrones o anomalías que difícilmente pueden ser detectadas mediante análisis estadísticos numéricos. Por su potencia y sencillez, la mayoría de las técnicas de análisis descritas en este texto son técnicas gráficas.

R proporciona excelentes recursos para la construcción de gráficos. La construcción del gráfico se realiza ejecutando varias sentencias, cada una de las cuales va añadiendo nuevas características al gráfico y definiendo su aspecto. Se muestra un ejemplo a continuación. Supongamos un data frame llamado `datos` cuyo contenido es:

```
> datos
     x    y
1  1.2  2.4
2  2.3  6.4
3  1.4  2.8
4  3.7 10.1
5  1.2  2.0
6  0.6  1.0
```

La primera de las siguientes tres sentencias dibuja la variable `x` frente a `y`. En la sentencia se especifican las etiquetas de ambos ejes (`xlab`, `ylab`) y el rango de valores de cada eje (`xlim`, `ylim`). La segunda sentencia añade la línea correspondiente al ajuste de los datos y la tercera añade el título.

```
> plot( datos$x, datos$y,
        xlab = "x", ylab = "y",
        xlim = c(0,4), ylim = c(0,12) )
> abline( lm(datos$y ~ datos$x) )
> title( "Representación y vs x" )
```

La gráfica obtenida se muestra en la Figura 1.4. Es posible configurar otras características del gráfico, como son su tamaño, los símbolos usados para representar los valores y su color, el tipo y color de las líneas, el tamaño, color y fuente del texto de los ejes, del título y del subtítulo, las marcas de los ejes, etc. Asimismo, es posible añadir etiquetas al gráfico y superponer varios gráficos.

Otros tipos de gráficos útiles para el análisis de los datos son el histograma y el boxplot. Las funciones `hist()` y `boxplot()` dibujan el histograma y el boxplot de los datos pasados como argumento.

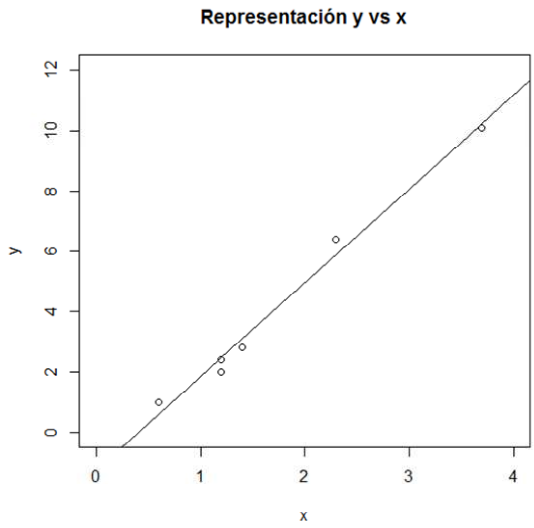

Figura 1.4: Ejemplo de construcción de un gráfico X-Y.

La función `dev.new()` abre una nueva ventana gráfica sobre la cual es posible construir un nuevo gráfico.

La función `par()` permite asignar valor a los parámetros del gráfico. Puede usarse para asignar valor al parámetro `mfrow`, que define el número de filas y columnas de gráficos que van a representarse en la ventana de dibujo. Por ejemplo,

```
> par( mfrow(2,3) )
```

prepara la ventana de dibujo para que aloje 6 gráficos, dispuestos formando una matriz de 2 filas y 3 columnas.

A continuación se muestra un ejemplo. Se crea un objeto de tipo vector llamado `x`, en el cual se guardan 500 observaciones independientes de una distribución exponencial de media 1, y se representa de tres maneras diferentes: gráfico X-Y, histograma y boxplot. Las tres gráficas obtenidas se muestran en la Figura 1.5.

```
> x <- rexp(500)
> par( mfrow = c(1,3) )
> plot(x)
> hist(x)
> boxplot(x)
```

El **boxplot** es una herramienta de análisis gráfico muy útil cuando se desea comparar la distribución de diferentes grupos de datos. En la Figura 1.6 se muestra nuevamente el boxplot que aparece en la Figura 1.5, indicando en esta ocasión cómo ha sido construido a partir de los datos.

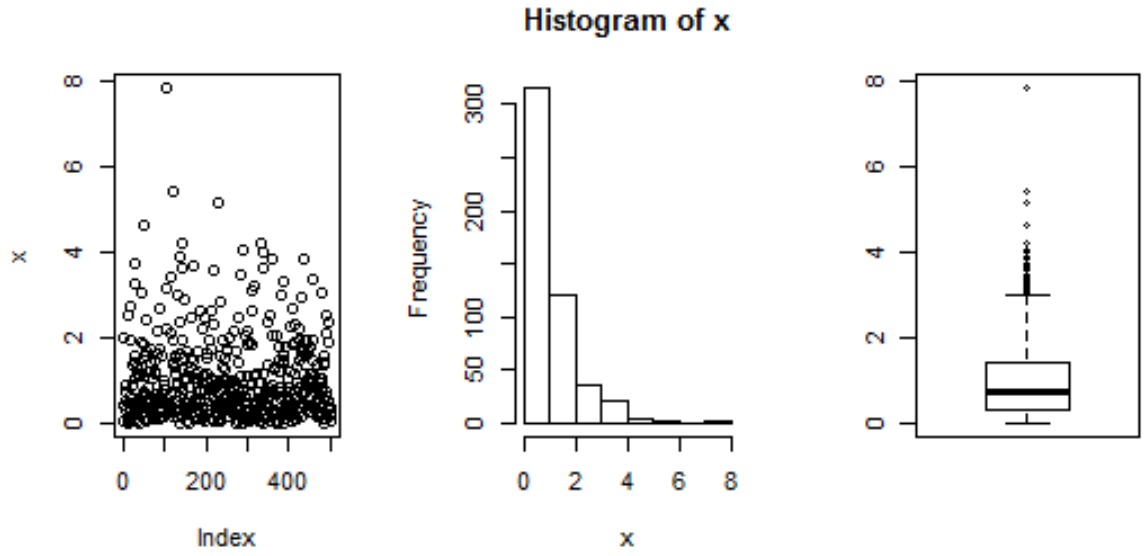

Figura 1.5: Ejemplo de construcción de un gráfico X-Y, un histograma y un boxplot.

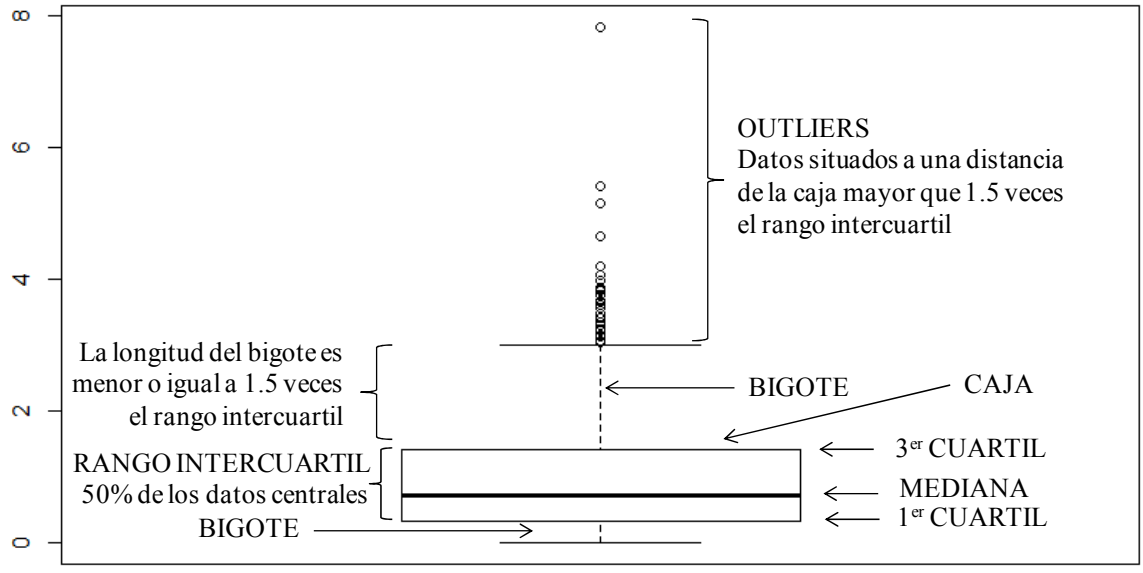

Figura 1.6: Construcción del boxplot de un grupo de datos.

1.5. MANEJO BÁSICO DE LOS DATOS

Comandos del tipo

```
> variable <- expresión
```

permiten crear nuevas variables y transformar las existentes. La expresión puede contener operadores y funciones.

Por ejemplo, los siguientes comandos asignan valor a los vectores `x1`, `x2` y `x3`, y a continuación asignan al vector `x` la concatenación de los anteriores.

```
> x1 <- c(1, 2, 3, 4)
> x1
 [1] 1 2 3 4
> x2 <- seq(10, 100, by=10)
> x2
 [1]  10  20  30  40  50  60  70  80  90 100
> x3 <- c(-1.1, -2, 2)
> x3
 [1] -1.1 -2.0  2.0
> x  <- c(x1, x2, x3)
> x
  [1]  1.0   2.0   3.0   4.0  10.0  20.0  30.0  40.0  50.0
 [10] 60.0  70.0  80.0  90.0 100.0  -1.1  -2.0   2.0
```

Un comentario respecto a la ejecución de este último comando. Cuando el listado de los elementos no cabe en una única línea, RGui los muestra en dos o más líneas. En este caso `[10]` indica simplemente que el elemento en la posición 10 es 60.0.

Por otra parte, obsérvese que los valores de los vectores `x1` y `x2` son de tipo entero, los valores del vector `x3` son de tipo real (debido a que uno de sus elementos es escrito como tal), y que al realizar la concatenación los valores enteros son convertidos al tipo real. Estas conversiones automáticas de tipo son debidas a que todos los valores de un vector deben ser del mismo tipo.

Un data frame puede tener variables de diferente tipo. En el ejemplo siguiente se muestra el data frame `data`, que tiene dos variables `x` e `y`, y al cual se añade posteriormente una nueva variable llamada `media`. Cada elemento de la nueva variable es la media de los correspondientes elementos de las otras dos variables.

```
> data <- data.frame( x=c(1,2), y=c(10,20) )
> data
   x  y
1  1 10
2  2 20
```

```
> data$media <- ( data$x + data$y ) / 2
> data
   x  y media
1 1 10   5.5
2 2 20  11.0
```

La función `by()` puede emplearse para aplicar una función (predefinida en el lenguaje R o definida por el usuario) a cada uno de los grupos de datos definidos mediante un factor. Supongamos que el data frame `df` es definido de la forma:

```
> df <- data.frame( conc = c(1.3, 2.4, 7.2, 3.5, 1.05),
                    prod = c("A", "B", "A", "A", "B") )
> df
   conc prod
1 1.30    A
2 2.40    B
3 7.20    A
4 3.50    A
5 1.05    B
```

Puede calcularse la media de la variable `conc` para cada uno de los valores de `prod` de la forma:

```
> by(df$conc, df$prod, mean)
df$prod: A
[1] 4
------------------------------------
df$prod: B
[1] 1.725
```

Un comando del tipo

```
> variable[condición] <- expresión
```

sólo realiza la asignación si la condición vale TRUE. Así, en el siguiente ejemplo se asigna el valor cero a los elementos del vector `x` que satisfacen la condición `x < 2.5`.

```
> x <- c(1:5, 5:1)
> x
 [1] 1 2 3 4 5 5 4 3 2 1
> x[ x < 2.5 ] <- 0
> x
 [1] 0 0 3 4 5 5 4 3 0 0
```

Por ejemplo, puede incluirse una nueva variable en el data frame que indique si la media es mayor, menor o igual que 10.

```
> data
    x  y media
  1 1 10   5.5
  2 2 20  11.0
> data$criterio10 [data$media < 10]  <- "menor"
> data
    x  y media criterio10
  1 1 10   5.5      menor
  2 2 20  11.0       <NA>
> data$criterio10 [data$media == 10] <- "igual"
> data
    x  y media criterio10
  1 1 10   5.5      menor
  2 2 20  11.0       <NA>
> data$criterio10 [data$media > 10]  <- "mayor"
> data
    x  y media criterio10
  1 1 10   5.5      menor
  2 2 20  11.0      mayor
```

El valor NA, que significa Not Available (No Disponible), se explicará en la Sección 1.6.

Análogamente, un comando del tipo

```
> variable1 <- variable2[condición]
```

crea el objeto `variable1` y copia en él los elementos de `variable2` que satisfacen la condición. Por ejemplo:

```
> x1 <- data$x[data$criterio10 == "mayor"]
> x1
 [1] 2
```

Las expresiones lógicas en R pueden tomar dos valores: TRUE y FALSE. Los operadores de comparación son:

$$< \qquad <= \qquad > \qquad >= \qquad == \qquad !=$$

Las operaciones lógicas se representan de la forma siguiente:

Operación	Significado
`!x`	not x
`x | y`	x or y
`x & y`	x and y

La función `order()` permite ordenar los elementos de una estructura de datos. Puede por ejemplo aplicarse a un data frame de la forma siguiente (se ordenan las filas de `exp3` tomando como criterio el valor de la variable `tCola`):

```
> exp3
  numOper horario tCola
1       1     std  12.3
2       1   nuevo   6.1
3       2     std   5.2
4       2   nuevo   2.5
> exp3[order(tCola),]
  numOper horario tCola
4       2   nuevo   2.5
3       2     std   5.2
2       1   nuevo   6.1
1       1     std  12.3
```

Las funciones `merge()`, `cbind()` y `rbind()` permiten realizar la unión de los datos de dos data frames.

1.6. VALOR NA (NOT AVAILABLE)

El valor `NA` significa Not Available (No Disponible). Se emplea para indicar que no se dispone de ese dato. Supongamos por ejemplo un vector de 5 elementos definido de la forma:

```
> x <- c(1:5)
> x
[1] 1 2 3 4 5
```

Si se asigna valor a un elemento del vector que no existe, R aumenta automáticamente el tamaño del vector rellenando con valores NA. Por ejemplo:

```
> x[8] <- 8
> x
[1]  1  2  3  4  5 NA NA  8
```

La función `is.na()` acepta un objeto como argumento y devuelve un objeto del mismo tamaño con las entradas reemplazadas por TRUE si el elemento es NA y por FALSE si no lo es. Por ejemplo:

```
> x
[1]  1  2  3  4  5 NA NA  8
```

```
> is.na(x)
 [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
> x[ !is.na(x) ]
 [1] 1 2 3 4 5 8
```

Si alguno de los operandos es NA, el resultado de la operación aritmética es también NA. Lo mismo sucede cuando se pasa el valor NA como argumento a una función. Esto debe ser tenido en cuenta al operar sobre conjuntos de datos: los valores NA deben ser excluidos del análisis. Con este fin, la mayor parte de las funciones tienen un parámetro llamado `na.rm`, tal que si se le asigna el valor TRUE los valores NA son eliminados de los datos pasados como argumento a la función. Por ejemplo, la función `sum` realiza la suma de los datos contenidos en el objeto que es pasado como argumento (puede consultarse su documentación ejecutando `help(sum)`):

```
> x
 [1]  1  2  3  4  5 NA NA  8
> sum(x)
 [1] NA
> sum(x, na.rm=TRUE)
 [1] 23
```

La función `na.omit()` devuelve un objeto en el cual se han eliminado todos los valores NA. Cuando se pasa como argumento un data frame, el objeto devuelto es un data frame en el cual se han eliminado todas las filas que contienen algún valor NA.

1.7. CONVERSIÓN DEL TIPO DE DATOS

R proporciona funciones para identificar el tipo de dato de un objeto y convertirlo a un tipo diferente. Las funciones cuyo nombre tiene la forma `is.tipodato()` devuelven TRUE o FALSE dependiendo de que el objeto pasado como argumento sea o no del tipo de dato. Por el contrario, las funciones cuyo nombre es de la forma `as.tipodato()` realizan la conversión del argumento al tipo de dato. En la Tabla 1.1 se muestran las funciones disponibles.

Tabla 1.1: Funciones para la comprobación y conversión de tipos de datos.

Comprobación	Conversión
`is.numeric()`	`as.numeric()`
`is.character()`	`as.character()`
`is.vector()`	`as.vector()`
`is.matrix()`	`as.matrix()`
`is.data.frame()`	`as.data.frame()`
`is.factor()`	`as.factor()`
`is.logical()`	`as.logical()`

1.8. CONTROL DEL FLUJO

R tiene sentencias para controlar el flujo de ejecución. La sentencias *for* y *while* permiten definir bucles, mientras que las sentencias *if-else* y *switch* permiten especificar que una sentencia se ejecute sólo cuando se satisface determinada condición.

La sentencia *for* tiene la siguiente sintaxis:

```
> for (var in secuencia) sentencia
```

Si el cuerpo del bucle consta de varias sentencias, debe escribirse entre llaves (`{ }`). La función `length()` devuelve el número de elementos del objeto que se le pasa como argumento. El siguiente ejemplo ilustra el uso de dicha función y del bucle *for*.

```
> x <- c(1, 2, 3, 5, 7, 11)
> y <- numeric(0)
> k <- numeric(0)
> for ( i in 1:length(x) ) {
      y[i] <- i * x[i]
      k[i] <- y[i] / 2
  }
> y
 [1]  1  4  9 20 35 66
> k
 [1]  0.5  2.0  4.5 10.0 17.5 33.0
```

La sentencia *if-else* permite definir que una o varias sentencias sean ejecutadas sólo si se satisface determinada condición. La sintaxis es:

```
> if (condición) sentencia
> if (condición) sentencia1 else sentencia2
```

1.9. DEFINICIÓN DE FUNCIONES

R proporciona las funciones matemáticas más habituales, tales como `abs(x)`, `sqrt(x)`, `ceiling(x)` (entero más pequeño no menor que `x`), `floor(x)` (mayor entero no mayor que `x`), `trunc(x)` (entero formado truncando a cero los decimales de `x`), `round(x,digits=n)` (redondea `x` al número de dígitos decimales especificado), `signif(x,digits=n)` (redondea `x` al número de dígitos especificado, incluyendo dígitos enteros más decimales), `sin(x)`, `cos(x)`, `tan(x)`, `asin(x)`, `acos(x)`, `atan(x)`, `sinh(x)`, `cosh(x)`, `tanh(x)`, `asinh()`, `acosh()`, `atanh()`, `log(x,base=n)`, `log(x)` (logaritmo en base e), `log10(x)` (logaritmo en base 10) y `exp(x)`.

El usuario puede definir sus propias funciones. La sintaxis para ello es la siguiente:

```
> nombreFunción <- function(arg1, arg2, ...) {
      sentencias
      return(objeto)
  }
```

Los objetos creados dentro del cuerpo de la función son locales a la función. El objeto devuelto por la función puede ser cualquier tipo de dato. La función se invoca de la misma forma que las demás funciones definidas en R: especificando su nombre y escribiendo a continuación los argumentos entre paréntesis, separados por coma.

Aquí concluye esta breve introducción al lenguaje R. En el siguiente tema se mostrarán algunos ejemplos de aplicación de R al modelado de datos.

1.10. LECTURAS RECOMENDADAS

Existe gran cantidad de documentación en Internet y excelentes referencias sobre el lenguaje R y también sobre el lenguaje S, en el cual se basa R. Excelentes textos sobre análisis estadístico en S y S-PLUS son (Venables & Ripley, 1997) y (Becker et al., 1988). Los dos siguientes textos sobre R son excepcionales por la claridad de sus explicaciones: (Kabacoff, 2011) y (Maindonald & Braun, 2010).

Tema 2

ANÁLISIS Y MODELADO DE DATOS

2.1. INTRODUCCIÓN

Cuando se trabaja con modelos estocásticos, surge la necesidad de estimar cómo están distribuidas las entradas aleatorias al modelo. Esto puede hacerse atendiendo a consideraciones teóricas y además, si se dispone de datos del sistema real, analizando dichos datos. Modelar adecuadamente las entradas aleatorias al modelo es tan importante para el éxito del estudio como lo son la construcción del modelo o el posterior análisis estadístico de los resultados.

La recogida de los datos experimentales del sistema es a menudo una de las tareas más costosas y delicadas en el proceso de modelado. Una vez se dispone de muestras experimentales de una variable aleatoria de entrada, pueden aplicarse básicamente las tres técnicas descritas a continuación.

- *Usar directamente los datos en la simulación.* Por ejemplo, si los datos representan el tiempo entre fallos de una máquina, entonces se usa uno de estos datos cada vez que se necesita un intervalo entre fallos en la simulación.
- *Definir una distribución de probabilidad empírica a partir de los datos experimentales.* Por ejemplo, si los datos representan el tiempo entre fallos, se muestreará de esta distribución cuando se necesite un tiempo entre fallos en la simulación.
- *Ajustar los datos experimentales a una distribución teórica.* El proceso de ajuste consta típicamente de las tres tareas siguientes:

  Tarea I — *Selección de la familia de distribuciones.*

  Tarea II — *Estimación de los parámetros de la distribución.*
  Se calcula qué valor de los parámetros de la familia de distribuciones proporciona el mejor ajuste a los datos. Con ello, se selecciona una única distribución de la familia de distribuciones.

  Tarea III — *Medida de la bondad del ajuste.*
  Pretende cuantificar en qué medida la distribución ajustada reproduce los datos experimentales.

El procedimiento comúnmente más aceptado consiste en tratar en primer lugar de modelar la probabilidad de la variable de entrada mediante una distribución teórica, recurriendo a las distribuciones empíricas sólo como último recurso ya que la forma de las distribuciones empíricas puede ser considerablemente irregular, particularmente cuando se dispone de un número pequeño de datos.

En aquellos casos en que existan fundamentos teóricos para modelar determinado fenómeno usando una determinada familia teórica de distribuciones, suele ser recomendable usar ese tipo de distribución teórica ajustada a los datos experimentales. No hay que olvidar que los datos experimentales recogidos son en sí aleatorios, con lo cual una distribución empírica obtenida a partir de un conjunto de observaciones puede diferir mucho de la obtenida a partir de otro conjunto de observaciones del mismo proceso.

Existen situaciones en que ninguna distribución teórica proporciona un ajuste adecuado a los datos. En estos casos suele ser preferible modelar la entrada aleatoria empleando una distribución empírica, construida a partir de los datos experimentales, en lugar de usar directamente los datos experimentales. Esta última técnica presenta fundamentalmente dos desventajas. La primera es que la simulación sólo puede reproducir lo que ha sucedido históricamente. La segunda es que raramente se dispone del número de datos suficiente para realizar todas las réplicas independientes de la simulación que requiere el estudio.

A pesar de lo anterior, en determinadas fases de la validación del modelo sí es aconsejable emplear directamente los datos experimentales obtenidos del sistema real. Con ello se consigue que las entradas del modelo simulado sean iguales a los correspondientes valores medidos en el sistema real. De este modo puede compararse más fielmente el comportamiento del modelo y del sistema real. La comparación puede realizarse empleando las técnicas estadísticas para la comparación de sistemas que se describirán en la Sección 2.7.

En este tema se ofrece una breve introducción al modelado de las entradas aleatorias, en la cual se asume que el lector posee ciertos conocimientos básicos de estadística. El orden seguido en la exposición es el siguiente.

1. Se describen métodos estadísticos para contrastar la independencia y la homogeneidad de los datos experimentales.

2. Las tres siguientes secciones están dedicadas a la descripción de las tareas propias del ajuste de distribuciones teóricas a los datos experimentales: la selección de la familia de distribuciones, la estimación de los parámetros y la medida de la bondad del ajuste realizado.

3. Se muestra cómo emplear el lenguaje R para seleccionar las distribuciones de probabilidad de entrada del modelo.

4. Finalmente, se muestra cómo emplear el lenguaje R para comparar grupos de datos.

2.2. INDEPENDENCIA Y HOMOGENEIDAD DE LOS DATOS

Algunas de las técnicas estadísticas que se aplican en el ajuste de distribuciones a los datos experimentales asumen que éstos son **independientes** entre sí (por ejemplo, la estimación de máxima verosimilitud y el test chi-cuadrado). Por consiguiente, si las muestras no son independientes, no es correcto aplicar estas técnicas (aunque siempre podrán aplicarse técnicas heurísticas como son los histogramas, etc.). Desde este punto de vista, resulta de gran importancia saber contrastar la independencia de las observaciones experimentales.

Por otra parte, en ocasiones se recogen diferentes conjuntos independientes de observaciones de determinado fenómeno aleatorio y es preciso saber si estos conjuntos de datos son **homogéneos** (*pueden considerarse idénticamente distribuidos*) y por tanto se pueden juntar en una única muestra. Para ello, es preciso conocer técnicas para contrastar la homogeneidad de varios conjuntos de observaciones.

2.2.1. Análisis de la independencia de los datos

Los datos experimentales presentan en ocasiones una correlación temporal. Por ejemplo, si se mide experimentalmente el tiempo que deben esperar los clientes en una cola, se comprobará que las observaciones están correlacionadas. Por ejemplo, si un cliente ha debido esperar mucho tiempo para ser atendido (el sistema está congestionado), cabe esperar que el cliente situado tras él en la cola también haya debido esperar mucho tiempo.

Considérese una secuencia de datos experimentales, $x_1, \ldots, x_n$, dispuestos en el mismo orden que han sido recogidos. Una técnica gráfica para analizar si los datos de esta secuencia son independientes consiste en representar las parejas de puntos (x_i, x_{i+1}), para $i = 1, \ldots, n-1$. Si los datos son independientes, los puntos estarán distribuidos aleatoriamente. Si los datos están correlacionados, los puntos tenderán a alinearse.

Como ilustración de aplicación de esta técnica, en la parte izquierda de la Figura 2.1 se han representado 200 observaciones independientes e idénticamente distribuidas (aproximadamente de forma exponencial con media 10). En la parte derecha de la Figura 2.1 se muestran 200 observaciones correlacionadas entre sí.

Esta técnica gráfica es sólo una de las múltiples técnicas existentes. Así, por ejemplo, se ha desarrollado una gran variedad de tests estadísticos de independencia.

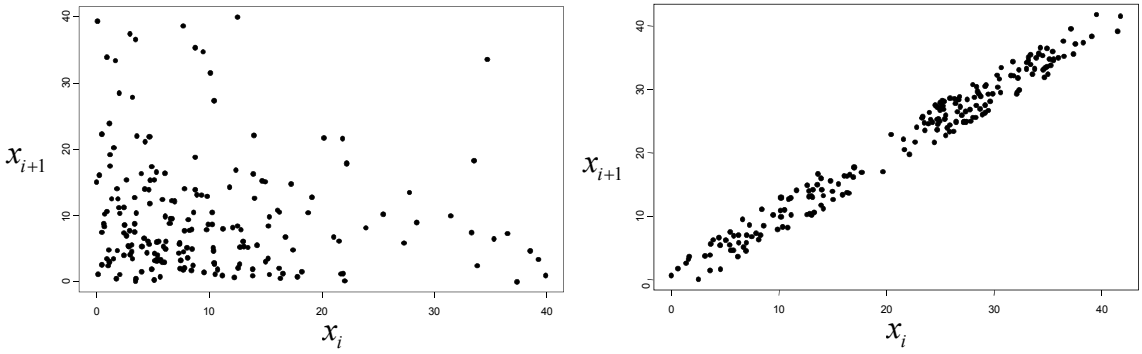

Figura 2.1: Ejemplo de evaluación de independencia de los datos.

Puede encontrarse abundante información sobre estos tests referente a su aplicación para contrastar la independencia de las secuencias de números pseudoaleatorios.

2.2.2. Análisis de la homogeneidad de los datos

Supóngase que se han recogido k conjuntos independientes de muestras y que las muestras de cada conjunto son independientes entre sí. Cada conjunto puede contener un número diferente de muestras.

$x_{1,1}, \ldots, x_{1,n_1}$	(conjunto 1, compuesto por n_1 muestras)
$x_{2,1}, \ldots, x_{2,n_2}$	(conjunto 2, compuesto por n_2 muestras)
$\ldots$	$\ldots$
$x_{k,1}, \ldots, x_{k,n_k}$	(conjunto k, compuesto por n_k muestras)

Con el fin de contrastar la homogeneidad de estos conjuntos de datos, se plantean las hipótesis nula (H_0) y alternativa (H_1) siguientes:

H_0: Las distribuciones de todos los conjuntos de muestras son idénticas.
H_1: Al menos la distribución de uno de los conjuntos de muestras tiende a producir valores mayores que al menos la distribución de otro de los conjuntos de muestras.

Uno de los tests que puede emplearse para contrastar la hipótesis nula H_0 es el **test de Kruskal-Wallis**. En la Sección 2.7 se mostrará cómo hacerlo usando el lenguaje R.

2.3. SELECCIÓN DE LA FAMILIA DE DISTRIBUCIONES

Supongamos que se pretende emplear una distribución de probabilidad teórica para modelar una entrada aleatoria de la simulación. El primer paso en la selección de la distribución teórica es decidir qué familia de distribuciones parece más apropiada. Esta decisión se toma en base al análisis de la forma de la distribución, sin tener en cuenta el valor de los parámetros de la distribución.

En ocasiones, se dispone del suficiente **conocimiento a priori** acerca del proceso a modelar como para poder seleccionar una familia de distribuciones, o al menos para poder descartar algunas otras. En la sección de lecturas recomendadas del final de la sección se cita bibliografía en la cual se ofrecen recomendaciones sobre el uso de las distribuciones teóricas.

Por otra parte, los valores de algunos **estadísticos** de la muestra experimental, tales como la media, la mediana y la varianza, pueden guiar en la elección de qué familia de distribuciones describe mejor la muestra. Sin embargo, debe tenerse presente que el valor del estadístico calculado de la muestra es una observación aleatoria. Por ello, el valor calculado generalmente no va a coincidir con el valor verdadero del estadístico de la distribución (desconocida) de la cual realmente han sido muestreados los datos.

Por ejemplo, si la media y la mediana de los datos difieren significativamente entre sí, es indicación de que probablemente la distribución de la cual se han muestreado los datos no sea simétrica. En la sección de lecturas recomendadas se indican referencias en las cuales se discute cómo extraer información útil acerca del tipo de distribución estimando otros estadísticos tales como el coeficiente de variación y el coeficiente de simetría (también denominado sesgo).

A continuación se describen dos representaciones gráficas que ayudan en la selección de la familia de distribuciones: los histogramas y las gráficas Q-Q. En la Sección 2.6 se mostrará como realizar estas representaciones gráficas usando R.

2.3.1. Histogramas

El histograma es básicamente una estimación gráfica de la densidad de probabilidad de los datos experimentales, $x_1, ..., x_n$. Las funciones de densidad de probabilidad de las distribuciones teóricas tienen formas reconocibles, con lo cual, el histograma sirve de ayuda para seleccionar qué familia o familias de distribuciones tiene sentido

emplear para modelar los datos. Para ello, debe compararse la forma del histograma con la forma de la distribución teórica, sin considerar la escala o la posición.

Puede construirse el histograma de una variable aleatoria continua de la forma siguiente:

1. Dividir el rango de valores de los datos experimentales en k intervalos disjuntos, adyacentes y de igual longitud: $[b_0, b_1), [b_1, b_2), \ldots, [b_{k-1}, b_k)$. Cada uno de estos intervalos se denomina una **clase** o **categoría**. A continuación, se agrupan los datos pertenecientes a cada clase.

2. Sea q_j la frecuencia de la clase j. Es decir, la proporción de los datos experimentales, $x_1, ..., x_n$, que se encuentran en el intervalo $[b_{j-1}, b_j)$. El histograma es la siguiente función constante a tramos:

$$h(x) = \begin{cases} 0 & \text{si } x < b_0 \\ q_j & \text{si } b_{j-1} \leq x < b_j, \text{ para } j = 1, ..., k \\ 0 & \text{si } b_k \leq x \end{cases} \tag{2.1}$$

Dado que al agrupar los datos se destruyen sus detalles originales, la elección del número de clases del histograma es importante. Tres de las recomendaciones que más comúnmente se encuentran en la literatura son las siguientes.

- Es recomendable escoger un número de clases entre 5 y 20 (lo más habitual es usar entre 8 y 12 clases), pero teniendo siempre al menos 5 observaciones en cada clase.

- La **regla de Sturges** proporciona el siguiente criterio: el número de clases, k, debe escogerse de acuerdo a la Ecuación 2.2, donde $\lfloor x \rfloor$ representa el mayor número entero que es menor o igual a x.

$$k = \left\lfloor 1 + \log_2(n) \right\rfloor = \left\lfloor 1 + 3.322 \cdot \log_{10}(n) \right\rfloor \tag{2.2}$$

- Otra recomendación es ir probando diferentes anchuras de intervalo y escoger aquella que proporciona un histograma "suave", con una forma similar a alguna distribución teórica.

Puede observarse que la construcción de los histogramas tiene cierta componente subjetiva, lo cual constituye su principal desventaja. La construcción del histograma

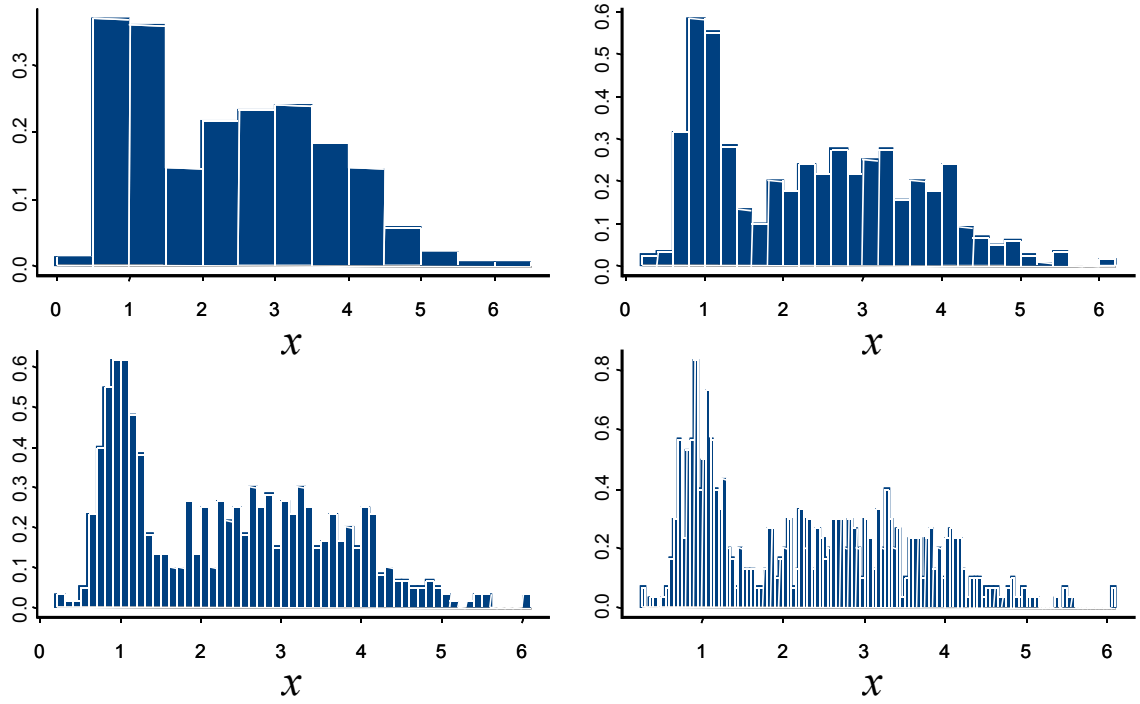

Figura 2.2: Histogramas de un mismo conjunto de datos.

de variables aleatorias discretas, también denominado **diagrama de barras**, es más sencilla, ya que no es preciso definir ni los intervalos ni agrupar los datos.

En la Figura 2.2 se muestran cuatro histogramas, con diferente número de clases, de los mismos 600 datos experimentales. Estos datos corresponden a medidas del tiempo necesario para atender a un cliente. Se han reescalado los histogramas, de modo que representen una densidad de probabilidad: la suma de los áreas de todas las barras es igual a uno. Se observa que el histograma tiene dos *modos* (máximos de la densidad de probabilidad) locales, uno entorno a 1 minuto y otro entorno a 3 minutos. Esto implica que existen esencialmente dos tipos de servicio, con distinta duración, que puede requerir el cliente.

Debe tenerse en cuenta esta **bimodalidad** al ajustar los datos experimentales. Las observaciones pueden separarse en dos grupos, siendo p_j la proporción de observaciones del grupo j (con $j = 1, 2$). Las observaciones del grupo j se usan para ajustar la distribución teórica $f_j(x)$. La densidad de probabilidad global, que considera ambos tipos de servicio, se calcula de la forma siguiente:

$$f(x) = p_1 \cdot f_1(x) + p_2 \cdot f_2(x) \tag{2.3}$$

2.3.2. Gráficas cuantil-cuantil

La función de distribución acumulada de una variable aleatoria X, también llamada probabilidad acumulada de X, se define:

$$F_X(x) = \text{Prob}\{X \leq x\} \tag{2.4}$$

Por ello, si los datos experimentales $x_1, \ldots, x_n$ están distribuidos de la misma forma que la variable aleatoria X, una aproximación razonable a $F_X(x)$ es la proporción de los datos experimentales que son menores o iguales que x.

Así pues, cabría plantearse comparar la distribución de probabilidad acumulada obtenida a partir de los datos experimentales, con las distribuciones de probabilidad acumulada de las distribuciones estándar. Sin embargo, éstas suelen tener forma de "S", con lo cual la comparación visual no suele ser demasiado esclarecedora.

Existen técnicas para reducir el problema de la comparación de funciones distribución de probabilidad acumulada a decidir cuál, de entre varias gráficas, se asemeja más a una recta. Una de estas técnicas, denominada gráfica cuantil-cuantil o abreviadamente **gráfica Q-Q**, está basada en la comparación de los cuantiles o puntos críticos de las distribuciones continuas.

Recuérdese que el cuantil q (con $0 < q < 1$) de una distribución $F_X(x)$ es un número x_q que satisface:

$$F_X(x_q) = q \tag{2.5}$$

Representando F_X^{-1} a la función inversa de la probabilidad acumulada, una definición equivalente del cuantil es:

$$x_q = F_X^{-1}(q) \tag{2.6}$$

La técnica gráfica Q-Q se basa en las dos propiedades siguientes:

– Si las variables aleatorias X e Y están igualmente distribuidas, entonces sus densidades de probabilidad y sus cuantiles son iguales: $y_q = x_q$. La representación gráfica de los puntos (x_q, y_q), para $q \in (0, 1)$, es una recta con pendiente unidad que pasa por el origen.

– Si las variables aleatorias X e Y pertenecen a la misma familia de distribuciones y sus densidades acumuladas difieren en el valor los parámetros de posición (γ) y escala (β), es decir, $F_Y(y) = F_X\left(\frac{y-\gamma}{\beta}\right)$, entonces la relación entre los cuantiles de las distribuciones es: $y_q = \gamma + \beta \cdot x_q$. Consiguientemente, la representación gráfica de los puntos (x_q, y_q), para $q \in (0,1)$, es una recta con pendiente β que no pasa por el origen.

A la vista de las dos propiedades anteriores, la representación de los pares de puntos (x_q, y_q) es una herramienta útil para comparar las distribuciones de X e Y. En el contexto que nos ocupa, dicha representación nos permitirá comparar la distribución empírica obtenida de los datos experimentales con una distribución teórica. A efectos de la aplicación de la técnica, sólo es relevante el parámetro de forma de la distribución teórica. Los parámetros de escala y posición son irrelevantes, con lo cual pueden escogerse de la forma que resulte más sencilla.

Los dos pasos a seguir para comparar la distribución empírica de un conjunto de datos experimentales, $x_1, ..., x_n$, con la distribución F_X son los siguientes:

1. **Construir una función de probabilidad acumulada empírica**, $\tilde{F}$, a partir de los datos experimentales. Para ello, es preciso ordenar crecientemente los datos experimentales. Sean $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$ los datos ordenados, donde $x_{(i)}$ representa el dato que ocupa la i-ésima posición. El valor de la distribución empírica en uno cualquiera de los datos experimentales, $x_{(i)}$, es igual al número de datos menor o igual que él mismo:

$$\tilde{F}\left(x_{(i)}\right) = \frac{i}{n} \qquad \text{para } i : 1, \ldots, n \tag{2.7}$$

   Esta definición presenta la desventaja de que la probabilidad acumulada vale uno para el valor experimental mayor, ya que $\tilde{F}\left(x_{(n)}\right) = \frac{n}{n} = 1$. Una forma de evitar este inconveniente es modificar de la forma siguiente la definición de la probabilidad acumulada empírica:

$$\tilde{F}\left(x_{(i)}\right) = \frac{i - 0.5}{n} \qquad \text{para } i : 1, \ldots, n \tag{2.8}$$

2. **Construir la gráfica Q-Q**. Puesto que $x_q = x_{(i)}$ es el cuantil $q = \frac{i-0.5}{n}$ de la distribución empírica, la gráfica consiste en la representación de los puntos:

$$\left(x_{(i)}, F_X^{-1}\left(\frac{i - 0.5}{n}\right)\right) \text{ para } i : 1, \ldots, n \tag{2.9}$$

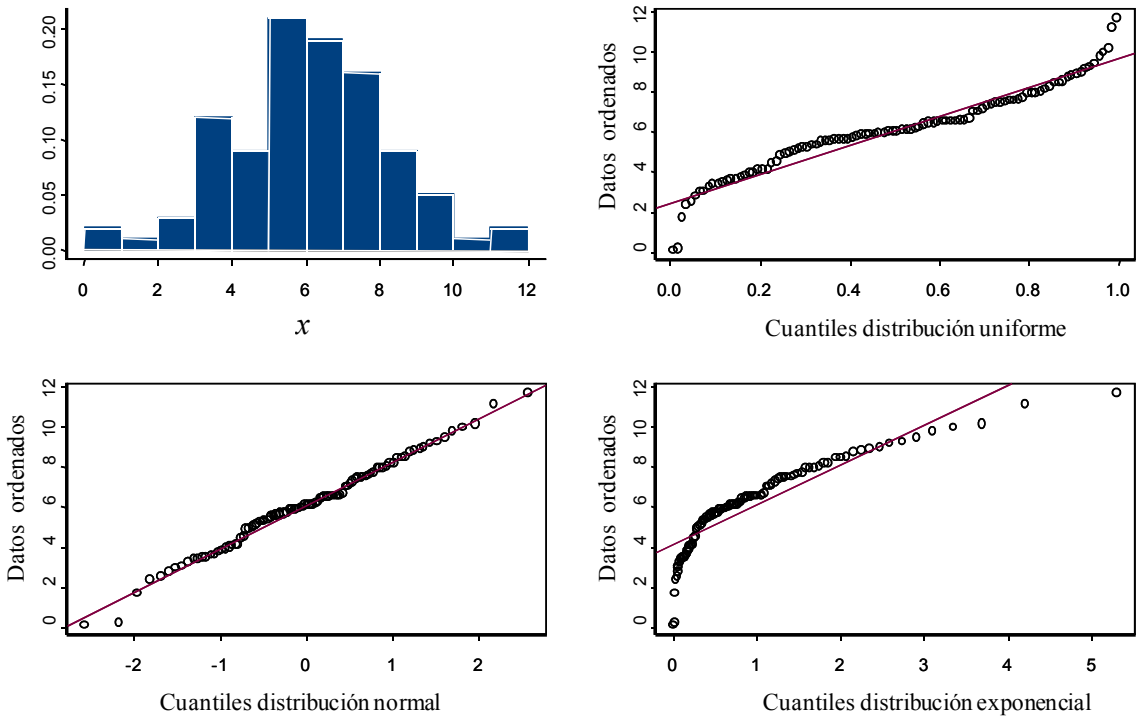

Figura 2.3: Ejemplo de aplicación de las gráficas Q-Q.

> Si estos puntos se ajustan razonablemente bien a una recta, con independencia de su pendiente o de si pasa o no por el origen, entonces está justificada la hipótesis de que la familia y el factor de forma de la distribución de la que están muestreados los datos experimentales y de F_X coinciden.

En la Figura 2.3 se muestra un ejemplo sencillo de aplicación de esta técnica. Se ha recogido una muestra de 100 datos experimentales. En primer lugar, se dibuja el histograma de los datos. Se observa que el histograma es aproximadamente simétrico. Esta observación puede contrastarse estimando el sesgo a partir de los datos experimentales. Por ello, tiene sentido realizar comparaciones con la distribución normal o con la distribución beta con $\alpha_1 = \alpha_2$. Como puede observarse de la segunda gráfica Q-Q, la comparación con la distribución normal es bastante satisfactoria. Sobre las gráficas Q-Q se ha dibujado la recta ajustada en cada caso. Como referencia del comportamiento de la gráfica Q-Q cuando la distribución empírica y teórica tienen una forma considerablemente diferente, se han comparado los datos experimentales con la distribución uniforme y exponencial (primera y tercera gráfica Q-Q, respectivamente).

2.4. ESTIMACIÓN DE LOS PARÁMETROS

Una vez seleccionada una familia de distribuciones, hay que escoger aquella distribución en concreto de la familia que mejor se ajusta a los datos experimentales. La forma de seleccionar la distribución es asignar valor a los parámetros de la misma. Existen diferentes métodos, uno de los cuales son los estimadores de máxima verosimilitud. Puede encontrarse información en las referencias citadas en la sección de lecturas recomendadas. En la Sección 2.6 se mostrará como realizar el ajuste usando el lenguaje R.

2.5. MEDIDA DE LA BONDAD DEL AJUSTE

Una vez estimados los parámetros de la distribución a partir de los datos experimentales $x_1, \ldots, x_n$, cabe preguntarse en qué medida los datos experimentales responden a la distribución ajustada, $\hat{F}_X(x)$. Se trata de responder a la pregunta: ¿cabría haber obtenido los datos experimentales muestreando la distribución $\hat{F}_X(x)$ o, por el contrario, existen grandes discrepancias entre la distribución ajustada y los datos experimentales? Para intentar responder a esta pregunta se usan técnicas gráficas y tests estadísticos.

Las **técnicas gráficas** son una herramienta muy potente de análisis. Comúnmente se usan el mismo tipo de representaciones que para seleccionar la familia de distribuciones. Es decir, histogramas y gráficas Q-Q.

En la Figura 2.4 se muestra un ejemplo. Se han tomado 100 muestras de un proceso de llegada (tiempos entre llegadas sucesivas). Apoyándose en razonamientos teóricos y en la gráfica Q-Q (véase la gráfica Q-Q de la parte izquierda de la Figura 2.4), se ha realizado la hipótesis de que los datos están distribuidos exponencialmente. El estimador del parámetro de la distribución se calcula de los datos experimentales: $\hat{\beta} = 4$. A continuación, se compara la distribución ajustada con los datos experimentales. Para ello, se representa el histograma de los datos experimentales escalado como una densidad de probabilidad (la suma del área de las barras vale uno) y la densidad de probabilidad de la distribución ajustada: expo (4). Este gráfico se muestra en la parte derecha de la Figura 2.4.

Los **tests de ajuste** pretenden contrastar la hipótesis nula de que las observaciones $x_1, \ldots, x_n$ están distribuidas $\hat{F}_X(x)$. La bibliografía sobre tests de ajuste es muy extensa. Dos de los tests más comúnmente empleados son el test chi-cuadrado

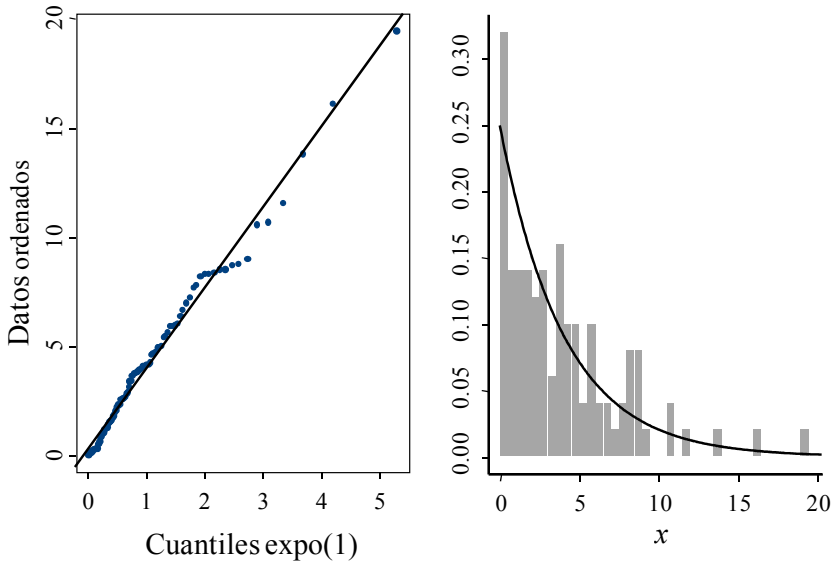

Figura 2.4: Ejemplo. Gráfica Q-Q (izqda). Histograma y densidad (drcha).

y el test de Kolmogorov-Smirnov. En la práctica es recomendable complementar el uso de los tests con el análisis visual del ajuste usando técnicas gráficas.

2.6. MODELADO DE LOS DATOS USANDO R

El lenguaje R proporciona muchas ayudas para el modelado de datos. A continuación se resumen algunas de ellas.

2.6.1. Funciones estadísticas

R proporciona las funciones estadísticas más habituales. Se muestran algunas de ellas en la Tabla 2.1.

Tabla 2.1: Algunas funciones estadísticas de R.

Función	Descripción
`mean(x)`	Media
`median(x)`	Mediana
`sd(x)`	Desviación estándar
`var(x)`	Varianza
`quantile(x,probs)`	Cuantiles
`min(x)`	Mínimo
`max(x)`	Máximo

La función `quantile(x,probs)` acepta como argumentos el vector numérico `x` del cual se desean calcular los cuantiles y un vector numérico `prob` con probabilidades en $[0, 1]$. Por ejemplo, los percentiles 0.25 y 0.75 de `x` pueden calcularse:

```
> y <- quantile(x, c(0.25,0.75))
```

Asimismo, R proporciona funciones de probabilidad que permiten generar datos y calcular valores de la probabilidad. El nombre de las funciones de probabilidad de R sigue el convenio siguiente. El nombre de la función comienza por una de las cuatro letras siguientes:

`d`	Densidad
`p`	Función de distribución
`q`	Función cuantil
`d`	Generación aleatoria de observaciones

y a continuación se escribe el nombre abreviado correspondiente a la distribución de probabilidad. En la Tabla 2.2 se muestran los nombres de algunas de las distribuciones de probabilidad de R.

Tabla 2.2: Algunas distribuciones de probabilidad.

Distribución	Nombre	Distribución	Nombre
Beta	`beta`	Geométrica	`geom`
Binomial	`binom`	Lognormal	`lnorm`
Binomial negativa	`nbinom`	Normal	`norm`
Cauchy	`cauchy`	Poisson	`pois`
Chi-cuadrado	`chisq`	T	`t`
Exponencial	`exp`	Uniforme	`unif`
Gamma	`gamma`	Weibull	`weibull`

Por ejemplo, las sentencias mostradas a continuación dibujan la densidad de probabilidad de la distribución normal, tal como se muestra en la Figura 2.5.

```
> x <- pretty(c(-3,3), 60)
> y <- dnorm(x)
> plot( x, y, type="l", xlab="x", ylab="f(x)", yaxs="i",
        main="Densidad de probabilidad N(0,1)" )
```

Veamos algunos otros ejemplos. El área que queda a la izquierda de $z = 2$, bajo la densidad de probabilidad N(0,1) (véase nuevamente la Figura 2.5), es:

```
> pnorm(2)
 [1] 0.9772499
```

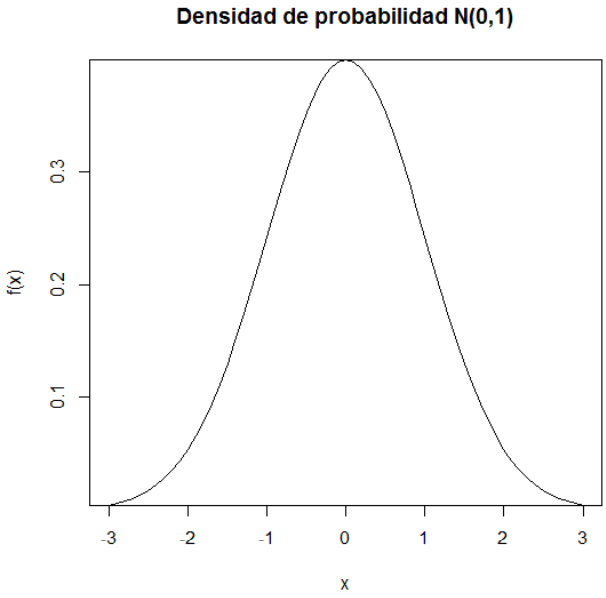

Figura 2.5: Densidad de probabilidad de la distribución normal estándar.

Los dos comandos siguientes calculan el valor del percentil 90 de la distribución normal con media 50 y desviación estándar 20, y 6 observaciones de la distribución:

```
> qnorm(0.9, mean = 50, sd = 20)
 [1] 75.63103
> rnorm(6, mean = 50, sd = 20)
 [1] 42.79969 60.27908 70.79518 31.82367 51.92393 50.23364
```

Cada vez que R genera observaciones de una distribución de probabilidad emplea una semilla diferente, con lo cual los valores generados son diferentes. Si se desea reproducir los mismos resultados, es necesario especificar la semilla mediante la función `set.seed()`. A continuación se muestra un ejemplo.

```
> rnorm(4, mean = 50, sd = 20)
 [1] 38.50520 39.06736 38.71096 32.19924
> rnorm(4, mean = 50, sd = 20)
 [1] 34.47492 51.28918 69.18988 47.79429
> set.seed(1234)
> rnorm(4, mean = 50, sd = 20)
 [1] 25.858685 55.548585 71.688824 3.086046
> set.seed(1234)
> rnorm(4, mean = 50, sd = 20)
 [1] 25.858685 55.548585 71.688824 3.086046
```

2.6.2. Histogramas

La función `hist()` permite crear un histograma. La opción `freq=FALSE` crea el histograma basándose en la densidad de probabilidad, en lugar de en la frecuencia relativa. La opción `breaks` controla el número de clases.

En el ejemplo siguiente se crea un objeto con 100 observaciones de una distribución normal con media 10 y desviación estándar 2. A continuación se dibuja el histograma basado en la densidad de probabilidad, con 10 clases, añadiendo pequeñas líneas bajo el histograma que señalan los datos. Finalmente, se dibuja sobre el histograma una estimación de la densidad de probabilidad de los datos y una caja que rodea el gráfico. Se emplea la función `lines()` para dibujar encima del gráfico (si se empleara `plot()`, se comenzaría un nuevo gráfico). El resultado se muestra en la Figura 2.6.

```
> x <- rnorm(100, mean = 10, sd = 2)
> hist(x, freq = FALSE, breaks = 10,
       xlab = "x distribuido N(10,2)",
       main = "Histograma de 100 observaciones N(10,2)")
> rug(jitter(x))
> lines(density(x), col = "blue", lwd = 2)
> box()
```

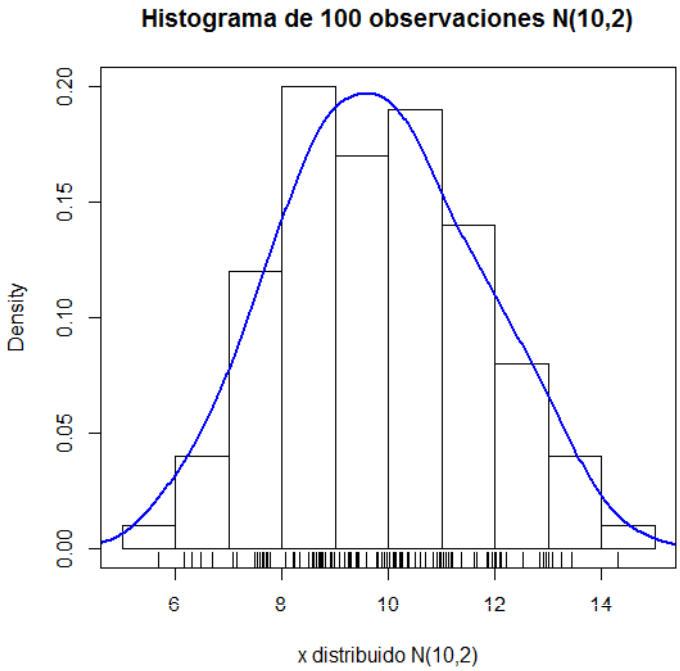

Figura 2.6: Histograma basado en la densidad de probabilidad.

2.6.3. Gráficas Q-Q

Las gráficas Q-Q son una herramienta muy útil para comparar un conjunto de datos con una distribución teórica. El objetivo de la comparación es determinar si aparentemente los datos podrían haber sido muestreados de dicha distribución.

Existen varias funciones en R para la representación de gráficas Q-Q. Una de ellas es `chart.QQPlot()`. Esta función se encuentra en el paquete PerformanceAnalytics, con lo cual para poder usarla es necesario primero instalar el paquete y a continuación cargarlo.

Esta función dibuja los cuantiles empíricos en el eje Y y el eje X contiene los valores del modelo teórico. Se dibuja también una línea recta con una inclinación de 45 grados. Si los datos provienen de la distribución teórica, los puntos caerán aproximadamente sobre esta línea. Cuanto mayor sea la distancia entre los datos y la línea, mayor será la evidencia de que los datos no provienen de la distribución teórica en cuestión. Por ejemplo, el resultado de la ejecución del código mostrado más abajo puede verse en la Figura 2.7.

```
> par( mfrow = c(1,3) )
> x <- rnorm(100, 20, 2)
> chart.QQPlot(x, envelope = 0.95,
                main = "NORMAL", distribution = 'norm')
> chart.QQPlot(x, envelope = 0.95,
                main = "EXPONENCIAL", distribution = 'exp')
> chart.QQPlot(x, envelope = 0.95,
                main = "UNIFORME", distribution = 'unif')
```

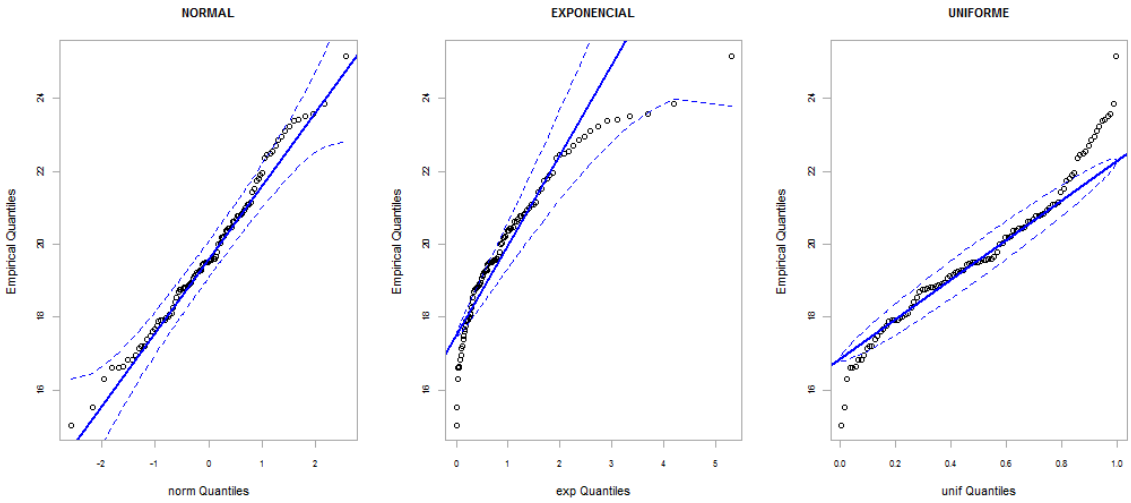

Figura 2.7: Gráficas Q-Q de un conjunto de datos para las distribuciones normal, exponencial y uniforme. Aparentemente los datos pueden haber sido muestreados de una distribución normal.

2.6.4. Ajuste de los datos

La función `fitdistr()`, que se encuentra en el paquete MASS, calcula estimadores de máxima verosimilitud para los parámetros de las distribuciones de probabilidad. Los parámetros que al menos deben pasarse a la función son el nombre del objeto donde se encuentran los datos y en nombre de la distribución a ajustar. Posibles nombre son: `"beta"`, `"cauchy"`, `"chi-squared"`, `"exponential"`, `"f"`, `"gamma"`, `"geometric"`, `"lognormal"`, `"logistic"`, `"negative binomial"`, `"normal"`, `"Poisson"`, `"t"` y `"weibull"`. A continuación se muestra un ejemplo.

```
> x <- rnorm(500,20,3)
> fitdistr(x,"normal")
      mean           sd
  20.05327880    2.87399628
 ( 0.12852902) ( 0.09088374)
 > fitdistr(x,"exponential")
      rate
  0.049867157
 (0.002230127)
```

2.6.5. Medida de la bondad del ajuste

Pueden emplearse técnicas gráficas para evaluar la bondad del ajuste. Una de ellas consiste en dibujar el histograma con los datos y sobre éste la densidad de probabilidad de la distribución teórica ajustada.

Además de las técnicas gráficas, pueden realizarse tests de ajuste para contrastar la hipótesis de que los datos provienen de la distribución ajustada. Entre los tests soportados por funciones en R están el test chi-cuadrado y el test de Kolmogorov-Smirnov.

2.7. COMPARACIÓN ENTRE GRUPOS DE DATOS USANDO R

R proporciona una gran variedad de métodos estadísticos para comparar grupos. Por otra parte, también facilita la representación gráfica de los datos de los grupos. El examen visual resulta crucial, ya que permite apreciar la magnitud de las diferencias e identificar peculiaridades de las distribuciones (por ejemplo, la bimodalidad) que pueden afectar a los resultados.

Una de las técnicas gráficas más eficaces para la comparación de grupos de datos es dibujar en paralelo los **boxplots** de los grupos. Consideremos un data frame en el cual hay una variable numérica que contiene los datos y una variable categórica que describe a qué grupo pertenece cada dato. En concreto, supongamos que el data frame se llama `datos`, que la variable que contiene los datos se llama `x` y que la variable que describe a qué grupo pertenece cada dato se llama `cat`. La siguiente sentencia dibuja un gráfico con un boxplot por cada grupo.

```
> boxplot(x ~ cat, data = datos, varwidth = TRUE,
          main = "Ejemplo boxplots en paralelo",
          xlab = "grupos", ylab = "valor a comparar")
```

La fórmula `x ~ cat` indica que se debe dibujar un boxplot de la variable numérica `x` por cada valor de la variable `cat`. Mediante `data=datos` se indica que ambas variables forman parte del data frame `datos`. Finalmente, `varwidth=TRUE` hace que la anchura del boxplot de cada categoría sea proporcional al número de datos de esa categoría.

Supongamos que el resultado de ejecutar la sentencia anterior es el mostrado en la Figura 2.8. Se observa que el grupo B tiene un comportamiento diferente que los grupos A y C. En general, cuando las cajas de dos grupos no solapan hay una fuerte evidencia de que sus medianas son diferentes. También se observa que la dispersión de los datos del grupo C es mayor que la de los datos de los otros dos grupos.

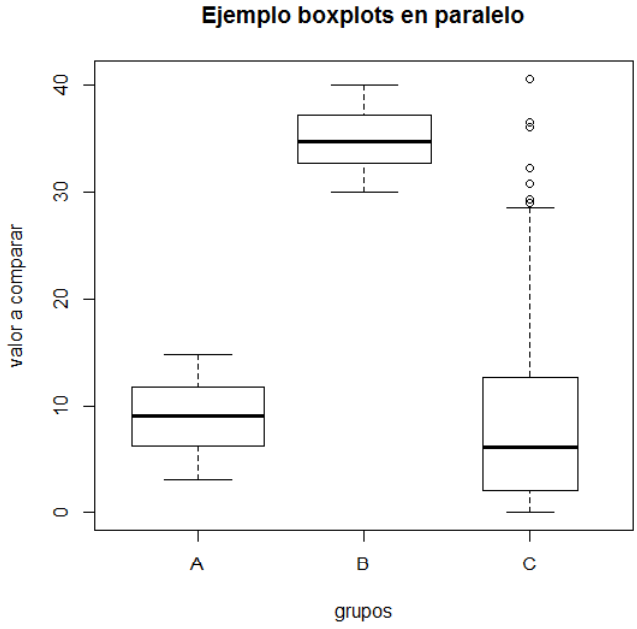

Figura 2.8: Análisis de las diferencias entre grupos mediante boxplots.

La anchura del boxplot del grupo C es menor (aproximadamente la mitad) que la de los grupos A y B. Esto indica que el número de datos pertenecientes al grupo C es menor: aproximadamente la mitad que el número en cada uno de los otros dos datos.

Un test estadístico que resulta adecuado para comparar varios grupos de datos independientes entre sí es el **test de Kruskal-Wallis**. La hipótesis nula del test es que los datos de los diferentes grupos han sido muestreados de una misma distribución de probabilidad. La siguiente sentencia aplica el test a los datos de los grupos A, B y C.

```
> kruskal.test(x ~ cat, data = datos)

        Kruskal-Wallis rank sum test

data:  x by cat
Kruskal-Wallis chi-squared = 349.0608, df = 2,
               p-value < 2.2e-16
```

Se obtiene un valor del p-value muy por debajo de 0.001, que es el valor por debajo del cual se considera comúnmente que el test rechaza la hipótesis nula. Así pues, el test rechaza la hipótesis de que los datos de los tres grupos han sido obtenidos muestreando una misma distribución de probabilidad.

La conclusión extraída del test de Kruskal-Wallis concuerda con la conclusión extraída de observar visualmente los boxplots de la Figura 2.8.

2.8. LECTURAS RECOMENDADAS

En (Bratley et al., 1987) se describen métodos para definir distribuciones empíricas de variables aleatorias continuas.

En los textos (Pedgen et al., 1995), (Banks et al., 1996), (Fishman, 2001), (Bratley et al., 1987) y (Hoover & Perry, 1989) se ofrecen explicaciones amplias y detalladas acerca del modelado de las entradas aleatorias a partir de consideraciones teóricas.

Puede encontrarse una discusión detallada en (Law & Kelton, 2000) acerca del uso de estadísticos para la selección de la familia de distribuciones que mejor modela unos datos experimentales.

La construcción de histogramas, en particular los criterios para la selección del número de clases, se aborda con detalle en (Pedgen et al., 1995) y en (Law & Kelton, 2000).

En (Law & Kelton, 2000) se discuten las ventajas del estimador de máxima verosimilitud frente a otras técnicas para la estimación de los parámetros de la distribución. También puede encontrarse en ese texto la explicación de propiedades importantes del estimador de máxima verosimilitud.

En (Pedgen et al., 1995) se discuten ampliamente diferentes técnicas para la medida de la bondad del ajuste. En (Law & Kelton, 2000) puede encontrarse una amplia discusión acerca de los tests estadísticos de ajuste.

En los textos (Pedgen et al., 1995) y (Law & Kelton, 2000) se ofrecen abundantes recomendaciones acerca de la selección de la distribución en ausencia de datos.

Índice alfabético

Bibliografía

Banks, J., Carson, J. S. & Nelson, B. L. (1996), *Discrete-Event System Simulation*, Prentice-Hall.

Becker, R., Chambers, J. & Wilks, A. (1988), *The New S Language*, Wadsworth & Brooks/Cole.

Bratley, P., Fox, B. L. & Schrage, L. E. (1987), *A Guide to Simulation*, Springer.

Fishman, G. (2001), *Discrete-Event Simulation*, Springer.

Hoover, S. V. & Perry, R. F. (1989), *Simulation. A Problem-Solving Approach*, Addison-Wesley Publishing.

Kabacoff, R. (2011), *R in Action*, Manning Publications Co.

Law, A. M. & Kelton, W. D. (2000), *Simulation Modeling and Analysis*, McGraw-Hill.

Maindonald, J. & Braun, W. (2010), *Data Analysis and Graphics using R - an Example-Based Approach*, Cambridge University Press.

Pedgen, C. D., Shannon, R. E. & Sadowsky, R. P. (1995), *Introduction to Simulation Using SIMAN*, McGraw-Hill.

Venables, W. & Ripley, B. (1997), *Modern Applied Statistics with S-PLUS*, Springer.