Introduccion a R

Universidad de Cantabria
Curso 2011–2012

Práctica 0: Introducción a R

El paquete R[1] es una colección de programas libres[2] diseñada para el análisis estadístico de datos, que permite desde los análisis descriptivos más sencillos (como tablas de frecuencias simples, cálculo de la media o correlaciones) a procedimientos inferenciales más complejos (como contraste de hipótesis, análisis de la varianza o el análisis de componentes principales). Solo unas pocas de las posibilidades de análisis que ofrece R serán abordadas en este curso.

R realiza tres funciones esenciales: (1) leer datos, (2) especificar el tipo de análisis que se quiere realizar con esos datos y (3) mostrar los resultados obtenidos tras los análisis. La selección de una metodología apropiada al caso de estudio, así como la interpretación de los resultados es tarea del investigador. Por este motivo resulta necesario que el investigador conozca el fundamento teórico de los distintos métodos estadísticos disponibles con el objeto de que la aplicación del análisis y la interpretación de los resultados se realice de forma satisfactoria.

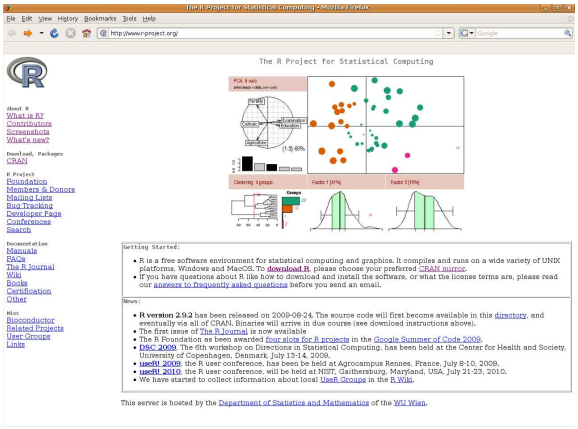

Figura 1: Página web del projecto R

[1]Sitio de referencia: `http://www.r-project.org`.
[2]En el sentido GNU: `http://gnu.org/philosophy/free-sw.es.html`.

1. Objetivos del curso

Con este manual se pretende que el alumno se familiarice con el paquete R y sea capaz de utilizarlo como una herramienta de aplicaciones estadísticas a través de las prácticas que se plantean en clase. Para ello se propone al alumno que ejecute cada una de las órdenes que se van introduciendo en el manual, así como los ejercicios propuestos en las secciones **Practica tu mismo** que servirán para asimilar contenidos.

2. Instalación de R

2.1. Microsoft Windows

Sigue los siguientes pasos para la instalación del software:

1. Descarga el fichero ejecutable desde la página del projecto R

   `http://cran.r-project.org/bin/windows/base/`

2. Ejecuta el fichero descargado, teniendo en cuenta:

   a) Cuando pregunta si deseamos establecer opciones de configuración, escoge *Sí*.

   b) Para el modo de presentación (MDI o SDI), escoge *SDI* (es conveniente por la implementación de la interfaz gráfica Rcommander que veremos después).

3. Ejecuta el programa R, ya instalado.

4. En el menú *Paquetes*, pincha en *Seleccionar espejo CRAN* para seleccionar un repositorio desde el cual se pueda descargar paquetes adicionales.

5. En el cuadro de diálogo, escoge *Spain (Madrid)*, *France (Toulouse)*, *Portugal* o algún otro cercano, y pulsa OK.

6. En el menú *Paquetes*, pincha en *Instalar paquete(s)*

7. Escoge *fBasics* y *Rcmdr* y acepta[3]. Estos son dos paquetes adicionales que necesitaremos tener instalados para algunas de las prácticas.

2.2. Linux

A través de la página inicial de R se accede a la descarga de R para diferentes versiones de Linux: Debian, Redhat, Suse y Ubuntu

`http://cran.r-project.org/bin/linux/`

Se recomienda seguir las instrucciones que se indican en la web en cada uno de los casos. Posteriormente instalar los paquetes `r-cran-rcmdr` y `r-cran-fbasics`.

[3]Si falta algún paquete aparecerá una ventana de aviso con los paquetes necesarios y la opción de instalarlos directamente. Los paquetes se instalan desde el espejo CRAN seleccionado anteriormente.

2.3. Mac Os X

En la siguiente dirección se accede a la instalación de R para Mac OS X:

```
http://cran.r-project.org/bin/macosx/
```

Seguid las instrucciones indicadas en la misma página web. Además en la siguiente web podeis encontrar los requisitos necesarios para Mac OS X antes de procecer a la instalación de R.

```
http://r.research.att.com/tools/
```

3. Estructura de R

El objetivo de esta primera práctica es que el alumno aprenda a manejar el programa R. Para ello, primero hablaremos de su estructura: sus ventanas y los elementos que las constituyen (barras de menús, de elementos activos, etcétera).

3.1. Comienzo de sesión

Tras arrancar el programa, aparece una ventana de órdenes titulada *Consola R* que indica la versión de R y cómo obtener información acerca de la licencia de uso.

```
R version 2.9.2 (2009-08-24)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R es un software libre y viene sin GARANTIA ALGUNA.
Usted puede redistribuirlo bajo ciertas circunstancias.
Escriba 'license()' o 'licence()' para detalles de distribucion.

R es un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener m\'as informaci\'on y
'citation()' para saber c\'omo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

>
```

3.2. Ventana de órdenes (consola)

Por debajo del título, esta ventana contiene una barra con los siguientes menús, cuyas opciones principales destacamos:

Archivo Operaciones básicas con los ficheros. Sólo usaremos:

- **Salir** Para salir del programa.

Editar Típico menú con opciones de edición (copiar, pegar, ...).

Misc Opciones avanzadas.

Paquetes Permite gestionar los paquetes adicionales de R. Nos interesará la opción:

Seleccionar espejo CRAN Para activar una URL de donde descargar distintos paquetes de R.

Instalar paquete Permite seleccionar los paquetes que se quieren instalar.

Cargar paquete Para activar un paquete en concreto.

Ayuda Información abundante sobre R.

3.3. R-Studio

RStudio (http://rstudio.org/) es una GUI, Graphical user interface para R programada en C, multiplataforma (Windows, Linux y Mac). Que aúna todos los entornos y guarda la filosofía de los comandos aportando algunas ayudas que hacen más llevadero el día a día. Otra definición de RStudio: es un entorno libre y de código abierto de desarrollo integrado (IDE) de R. Se puede ejecutar en el escritorio (Windows, Mac o Linux) o incluso a través de Internet mediante el servidor RStudio. Entre otras cosas encontramos que RStudio nos permite:

- Abrir varios scripts a la vez.
- Ejecutar pedazos de código con sólo marcarlo en los script.
- Mostrar el workspace.
- Mostrar los objetos del workspace.
- Integra la ayuda.
- Integra la librería.
- Etc.

4. Ayuda en R

>`help()` Muestra una ventana de ayuda general sobre R
>`help.start()` Arranca un manual de ayuda completo en formato html, utilizando el navegador del sistema.
>`help(mean)` Muestra una ventana de ayuda sobre la función "media aritmética".
>`?mean` Lo mismo que el ejemplo anterior.
>`help("[")` Muestra una ayuda sobre el carácter [, que es un carácter especial que forma parte del lenguaje R.
> `apropos("mean")` Muestra las funciones relacionadas con la función mean.
> `help.search("mean")` Busca ayuda sobre objetos o funciones que tengan nombre o título que contenga la cadena "mean".

5. Estructuras y tipos de datos

- Vectores: Son matrices de una dimensión que únicamente pueden contener valores numéricos, alfanuméricos y valores lógicos. Para formar vectores se emplea la función c() (contracción de la función combine). Todos los elementos deben ser del mismo tipo.

- Matrices: son un vectores con un atributo adicional (dim), que define el número de filas y columnas. dim es un vector numérico de longitud 2 . Se crean con la función matrix().
- Arrays: similar a las matrices pero pueden tener más de dos dimensiones.
- Factores: vectores de variables categóricas utilizados para agrupar las componentes de otro vector del mismo tamaño.
- Listas: colección ordenada de objetos donde los elementos almacenados pueden ser de diferente tipo.
- DataFrames: generalización de las matrices, donde cada columna puede almacenar un tipo de dato diferente, conservando la misma longitud.
- Funciones: objetos creados por el usario para realizar operaciones específicas.

5.1. Vectores

La estructura más simple es el vector, que es una coleccion ordenada de números. Existen varias formas de crear un vector. La forma más simple de crear un vector, por ejemplo x, consistente en cinco números, por ejemplo 10.4, 5.6, 3.1, 6.4 y 21.7, es usar la orden

```
> x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
```

Esta es una asignación en la que se utiliza la función *c()* que, en este contexto, puede tener un número arbitrario de vectores como argumento y cuyo valor es el vector obtenido mediante la concatenación de todos ellos. Un número, por si mismo, se considera un vector de longitud uno.

Advierta que el operador de asignación, (<-), no es el operador habitual en otros lenguajes, =, que se reserva para otro propósito, sino que consiste en dos caracteres, < (menor que) y '-' (guión), que obligatoriamente deben ir unidos y apuntan hacia el objeto que recibe el valor de la expresión. La asignación puede realizarse también mediante la función `assign()`. Una forma equivalente de realizar la asignación anterior es:

```
> assign('x', c(10.4, 5.6, 3.1, 6.4, 21.7))
```

El operador usual, <-, puede interpretarse como una abreviatura de la función `assign()`.

Si una expresión se utiliza como una orden por si misma, su valor se imprime y se pierde. Asi pues, la orden

```
> 1/x
```

simplemente imprime los inversos de los cinco valores anteriores en la pantalla pero al no haberle asignado ninguna variable al resultado este no queda guardado, (por supuesto, el valor de x no se modifica).
Si a continuación hace la asignación

```
> y <- c(x, 0, x)
```

creará un vector, y, con 11 elementos, consistentes en dos copias de x con un cero entre ambas, que no se imprime por pantalla.

Se puede estar interesado en seleccionar un determinado valor del vector, por ejemplo el valor 5.6 del vector `x`. Para ello únicamente tendremos que saber la posición de dicho elemento en el vector y escribir la orden

```
> x[2]
```

Practica tú mismo

1) Utiliza la función *seq* (consulta la ayuda para ver como se ejecuta esta función, recuerda: `> ?seq`) para crear los siguientes vectores:

- 55, 56, 57, 58, 59
- 1, 3, 5, 7, 9, 11

Practica tú mismo

2) Introduce el vector: `> x<-c(1,3,5,7,9)` e intenta adivinar antes de calcular los resultados de las siguientes instrucciones:

- `sum(x>5)`
- `sum(x[x>5])`
- `which(x>5)`
- `x>5`
- Extrae el elemento de x que ocupa la posición 2.
- Extrae utilizando un único comando, los 3 elementos de x que ocupan las posiciones de la 2 a la 4.

5.2. Matrices

Podemos crear una matriz de datos de la forma:

```
> M <- matrix(c(1,2,3,4,5,6),nrow = 2, ncol = 3)
```

y al igual que con vectores podemos seleccionar elementos de la matriz como la fila 2:

```
> M[2,]
```

la columna 3:

```
> M[,3]
```

o el elemento de la fila 1 y la columna 2:

```
> M[1,2]
```

Los vectores y matrices pueden usarse en expresiones aritméticas, en cuyo caso las operaciones se realizan elemento a elemento. Los operadores aritméticos elementales son los habituales +, -, *, / y ^ para elevar a una potencia. Además están disponibles muchas funciones matemáticas como *log, exp, sin, cos, tan o sqrt*. Las operaciones se realizan sobre cada uno de los elemntos del vector.

```
> 1/x
> M*M
> exp(x)
```

Existen muchas más funciones, entre otras, las siguientes: *max* y *min* que seleccionan respectivamente el mayor y el menor elemento de un vector; range cuyo valor es el vector de longitud dos conteniendo el mínimo y el máximo , c(min(x), max(x)); *length(x)* que es el número de elementos o longitud de x; *sum(x)* que es la suma de todos los elementos de x; *prod(x)* que es el producto de todos ellos , *sort(x)* que ordena los elementos de x de menor a mayor, *which.min(x)* obtiene la posición del menor elemento del vector x, y algunas variables estadísticas como *mean(x)*, *sd(x)*, *quantile(x, 0.25)*, etc. Como ya se comentó al principio se puede obtener más ayuda acerca de estas funciones con la orden `help()`

Practica tú mismo

---

3) Crea la siguiente matriz de datos: `> m<-matrix(1:15, nrow=3)` e intenta adivinar antes de calcular los resultados de las siguientes instrucciones:

- `dim(m)`
- `ncol(m)`
- `nrow(m)`
- `which(m>=5 \& m<=8)`
- Extrae la primera fila.
- Extrae el segundo elemento de la tercera columna.

Practica tú mismo

---

4) A lo largo de un año, los importes de las facturas mensuales de vuestro móvil han sido:

$$23, 33, 25, 45, 10, 28, 39, 27, 15, 38, 34, 29$$

1. ¿Cuanto habéis gastado en total en el año?
2. ¿Cuál ha sido el gasto mínimo?,
3. ¿y el máximo?
4. ¿Qué meses han supuesto un gasto menor que el gasto medio?
5. ¿ Existe mucha dispersión de unos meses a otros? ¿Cómo estimas esa dispersión?

5.3. Factores

Los factores son vectores que contienen información categórica útil para agrupar los valores de otros vectores. Para convertir un vector en un factor empleamos la función factor(). Por ejemplo, tenemos los datos de un grupo de personas,

```
> nombre <- c(ana", "luis", .elena", .amparo", "blanca","fran", çarlos", "luisa")"
>edad <- c(23, 24, 45, 67, 32, 56, 78, 45)
>fuma <- c(Si", "Si", "No", "No", "Si","Si", "No", "Si")"
actividad <- c(baja", "media", "media", "media", .alta",.alta", "baja", "media")"
```

Los vectores fuma y actividad son factores, con dos y tres factores respectivamente.

```
> factor(fuma)
> factor(actividad)
```

5.4. Data Frames

Un dataframe (o tabla) es una generalización de las matrices donde cada columna puede contener tipos de datos distintos al resto de columnas, manteniendo la misma longitud. Es lo que más se parece a una tabla de datos de cualquier paquete estadístico estándar. Se crean con la función data.frame(). En un data frame, los vectores de texto se transforman automáticamente en factores, el número de niveles vendrá determinado por el número de valores diferentes que contenga el vector. Vamos a definir un data frame con los datos del grupo de personas definido anteriormente.

```
> datos<- data.frame(nombre, edad, fuma, actividad)
> datos
```

Podemos seleccionar columnas concretas de un dataframe usando los corchetes `>[ ]` de forma similar a la usada para seleccionar elementos de una matriz. Para acceder al vector que forma una de las columnas de un dataframe usamos el operador `>$`. Para ampliar un dataframe unicamente es necesario definir un nuevo vector, del mismo tamaño que los anteriores, por ejemplo el peso, y usar la misma función.

```
> datos<- data.frame(datos,peso)
```

Para editar un dataframe definido podemos usar la función `edit(dataframe)`

Practica tú mismo

---

5) Crea un nuevo conjunto de datos, según las indicaciones anteriores, con los siguientes valores. Define el tipo de dato que consideres oportuno en cada caso (numérica o de caracteres).

Peso	Altura	Edad	Sexo
80	1.73	20	V
85	1.91	19	V
61	1.72	19	M
79	1.75	25	M
67	1.72	21	V
65	1.70	19	M
55	1.59	25	V
75	1.75	19	M

Practica tú mismo

6) Con el dataframe generado analiza los resultados de las siguientes instrucciones:

- `datos[2,]`
- `datos[3,4]`
- `datos[datos$edad>20,]`
- ¿Cuál es la diferencia entre estas dos sentencias?
  `attach(datos);datos[edad>20,]`
  `edad[edad>20]`
- `Altura[edad>20 & Sexo==`mujer"]"

Sin embargo la forma más común de deinir un dataframe es leyendo un fichero de datos, como se verá más adelante.

5.5. Funciones

Las funciones son objetos creados por el usuario para realizar una tarea específica. Además estas se pueden empaquetar en grupos y tener siempre disponibles para todas las sesiones. La sintaxis es sencilla:

```
nombre_funcion <- function (arg1, arg2,...){
#función de usuario
sentencias a ejecutar
return (objeto a devolver)
}
```

por ejemplo,

```
mi_primera_funcion <- function(x){
    f= sqrt(1-exp(-x^2))
    return (f)}
```

Una vez guardada la función creada por el usuario, el comando source() nos permitirá usarla cuando sea necesaria.

Practica tú mismo

7) Crear una función que obtenga las soluciones de una ecuación de segundo grado. Y probarla para la ecuación x2 + 5x - 16 = 0.

5.6. Instrucciones de control de flujo

La ejecución de una instrucción tras otra, se puede alterar con las instrucciones de control de flujo, que son de dos tipos:

1. Instrucciones de repetición o bucles. En este caso se repite la ejecución de un conjunto de intrucciones dentro de un lazo o blucle. Existen tres tipos de instrucciones de repetición *for*, *while* y *repeat*.
   - Los *for* ejecutan un número de predeterminado de veces el bucle
   - Los *while* ejecutan el bucle mientras una condición sea cierta
   - Los *repeat* ejecutan un bucle infinito
2. Instrucciones de selección o de ejecución condicional *if-elseif-else* testean si se cumple una condición o no.

Un bucle *for* necesita una variable que va iterando asignándola valores de una secuencia o de un vector. Su estructura básica será

```
for (i in 1:10){
print(i)
}
```

Estos cuatro bucles realizan la misma ejecución.

```
x <- c("a", "b", "c", "d")
for(i in 1:4) {
print(x[i])
}
for(i in seq_along(x)) {
print(x[i])
}
for(letter in x) {
print(letter)
}
for(i in 1:4) print(x[i])
```

Los bucles *for* pueden anidarse, es decir se puede poner uno dentro de otro, teniendo en cuenta que para seguir la ejecución del más externo antes se ejecuta totalmente el más interno

---

Practica tú mismo

---

8) Dado el vector (3 15 9 12 -1 0 -12 9 6 1), obtén el vector dx cuyo elemento i−ésimo es la diferencia entre el elemento i−ésimo y el elemento anterior, utilizando el bucle o lazo "for".

Los bucles while ejecutan repetidamente el bloque de comando mientras la condición lógica que sigue a *while* sea cierta. Esta condición depende de variables ajenas a *while* la cual no tiene ninguna variable asociada como ocurre con *for*. Su forma más básica es:

```
count <- 0
while(count < 10) {
print(count)
count <- count + 1
}
```

Por ejemplo,

```
z <- 5
while(z >= 3 && z <= 10) {
print(z)

}
```

Practica tú mismo

---

9) Define una función que calcule el factorial de un número, mediante un bucle while

Las estructuras de tipo condicional se ejecuta un conjunto de comandos en función del resultado de un test lógico. Comienzan con *if* a lo cual sigue una expresión a evaluar y puede tener instrucciones intermedias *else if* las cuales evalúan otra expresión. Puede tener también una instrucción *else* al final, y las instrucciones que sigan sólo se ejecutan si las expresiones evaluadas anteriormente han sido todas falsas. Cuando la expresión que sigue a *if* es cierta, se ejecuta la instrucción o instrucciones que siguen Cuando la expresión que sigue a *elseif* es cierta, se ejecutan las instrucciones que sigan. Sólo se ejecutan las instrucciones siguientes a la primera expresión cierta y sólo si no hay ninguna que lo sea se ejecutan las instrucciones que siguen a *else*. En cualquier caso, sólo se ejecuta un grupo de instrucciones, las que sigan a la primera expresión que sea cierta, el resto las salta y si no hay ninguna expresión que sea cierta, no se ejecuta ninguna instrucción o las que sigan a *else* si existe. Esquemáticamente se puede expresar :

```
if(<condition1>) {
## ejecuta las intrucciones correspondientes a la primera opcion
} else if(<condition2>) {
```

```
## ejecuta las intrucciones correspondientes a la segunda opcion
} else {
## ejecuta algo diferente
}
```

Una estructura *if* completa, puede ser, la siguiente. Ver como funciona, variando el valor de a :

```
a<-3
if (a==0){
    print('texto0')
}else if (a==2){
    print('texto2')
}else if (a==3){
    print('texto3')
}else {
    print('texto4')
    }
```

Las estructuras de tipo if se pueden anidar unas dentro de otras.

Practica tú mismo

---

10) Dado un valor de m, escribe una función que calcule el cuadrado de ese número sólo si m es mayor que 5. En caso contrario, que escriba un mensaje de error

---

Practica tú mismo

---

11) En un comercio se aplica una rebaja del 10 % a los precios comprendidos entre 50 y 100 euros y una rebaja del 20 % a los precios comprendidos entre 20 y 50 euros. Escribe una fucnión que scalcule la rebaja de un precio en caso de tenerla o indique que el producto no está rebajado.

6. Importar Datos

6.1. Importar datos desde un fichero de texto

La función más importante para importar datos, al menos una de las más usadas es read.table(), automáticamente te convierte los datos en un dataframe.

```
> santander<-read.table(Santander.dat")"
> santander<-read.table(Santander.dat", header=TRUE)"
```

El parámetro header será TRUE o FALSE según la primera fila del fichero represente el nombre de las variables, por defecto toma el valor FALSE. sep es el delimitador, es decir el separador de campos, por defecto es el espacio en blanco pero se pueden especificarse otros.

La función load nos permite leer un fichero de datos propio de R.

```
> Pulso<-load(Pulso.rda")"
```

Practica tú mismo

12) Utilizando los datos del fichero Pulso.rda, calcula los siguientes valores:

1. Calcular el número de personas que miden menos de 183 cm.
2. Calcular el porcentaje de personas que miden menos de 183 cm.
3. Calcular el porcentaje de mujeres que son fumadoras.
4. Determinar la frecuencia relativa de las personas que miden menos de 183 cm.
5. Determinar la frecuencia relativa de las personas que son mujeres.
6. Determinar la frecuencia relativa de las personas que son varones y su práctica deportiva habitual es `baja`.
7. Determinar la frecuencia relativa de las personas que son varones y fuman.
8. De entre las mujeres, determinar la frecuencia relativa de las que miden menos de 170 cm.
9. Determinar la frecuencia relativa de las personas cuyo pulso en reposo (Pulse1) es superior a 84.
10. Determinar la frecuencia relativa de las personas cuyo pulso en reposo es superior a 84 y cuya práctica deportiva habitual no es `alta`.
11. Determinar la frecuencia relativa de las personas tales que su pulso en reposo es superior de 84, fuman y son mujer.
12. De entre los personas que fuman, determinar la frecuencia relativa de los que registran el Pulse1 superior a 84.

7. Representaciones gráficas

Los gráficos básicos en 2-D en el paquete base de R (plot,hist,boxplot, etc.). Existen muchos parámetros de estas funciones que pueden ser sobreescritos usando la función par (consulta la ayuda ?par).

```
> plot(Altura, Peso)
> par("pch")
[1] 1
> par("lwd")
[1] 1
> par(lwd=4)
> par(pch=2)
```

```
> plot(Altura, Peso)
```

Las funciones básicas más importantes son:

1. plot: por defecto genera un gráfico de puntos
2. lines: añade lineas a un plot
3. points: añade puntos a un plot
4. text: añade etiquetas de texto a un plot
5. title: añade etiquetas a los ejes X e Y, a un título, subtítulo
6. axis: modifica los ticks de los ejes y sus etiquetas

Veamos un ejemplo usando el fichero de datos Pulso.rda.

```
> plot(Altura, Peso)
> plot(Altura, Peso,ylim=c(60,70))
> lines(c(160,190),c(65,65),col="red",lwd=2)
> points(c(175),c(65),pch=3,col="blue",lwd=3
> points(c(175,180),c(60,65),pch=3,col="green",lwd=4))
> text(170,68,"Fake point")
> title(main="Grafico de prueba",cex.main=1.5)
> axis(1,col="violet")
> axis(2,col="magenta")
```

Practica tú mismo

---

13) El fichero de datos santander.dat contiene los datos de temperatura media mensual medida en Santander desde 1950 hasta 2003. Se pide:

1. Representar las temperaturas medias registradas en febrero durante el periodo 1950-1960. Indicar en qué año se registró la temperatura más baja.
2. Representar las temperaturas medias de julio y agosto entre los años 1970 y 1980 y encontrar gráficamente en qué años ésta fue mayor en julio.
3. Calcule la mediana, media y desviación estándar de las temperaturas medias mensuales en enero y julio entre los años 1950 y 1980.
4. Determine los coeficientes de asimetría y curtosis de las temperaturas medias mensuales de junio, julio y agosto entre los años 1950 y 1980. ¿En que caso la serie presenta mayor asimetría?