Estructuras de Datos

José Enrique Martín García
Universidad Politécnica de Gimialcón

Clases de Objetos

- Los vectores son el tipo básico de objeto en R,
- Las matrices o, variables indexadas (Arrays) son generalizaciones multidimensionales de los vectores. De hecho, son vectores indexados por dos o más índices.
- Los factores sirven para representar datos categóricos.
- Las listas son una forma generalizada de vector en las cuales los elementos no tienen por qué ser del mismo tipo y a menudo son a su vez vectores o listas.
- Las hojas de datos (data frames) son estructuras similares a una matriz, en que cada columna puede ser de un tipo distinto a las otras.
- Las funciones son también objetos de R que pueden almacenarse en el espacio de trabajo, lo que permite extender las capacidades de R fácilmente.

Objetos

- Durante una sesión de trabajo con R los objetos que se crean se van almacenando por su nombre.
- R distingue entre mayúsculas y minúsculas, de tal modo que B y b son símbolos distintos y se referirán, por tanto, a objetos distintos.
- Los nombres de los objetos pueden contener sólo letras mayúsculas o minúsculas, junto con números y puntos (No admite blancos, __ , %,&,/, $, etc.).
- La función objects() se puede utilizar para obtener los nombres de los objetos almacenados en R. Es equivalente a la función ls().
- Los objetos creados durante una sesión de R pueden almacenarse en un archivo para su uso posterior (archivos .RData).
- Es posible eliminar objetos con el comando rm().
- Desde el menú Misc>remover todos los objetos se pueden eliminar todos los objetos a la vez.

Tipos y modos de los datos

Los objetos están compuestos de elementos. Los elementos más simples, las variables, pueden ser:

- numeric: número real con doble precisión. Se puede escribir como enteros (6, -7), como fracción decimal (9.64) o como notación científica (9.44E-08).
- complex: números complejos de la forma 7+5i.
- character: Cadenas alfanuméricas de texto. “pepito”
- logical: variables lógicas. (TRUE o FALSE).

Datos especiales: NA (Dato no conocido en Ingles “Not Available”); determinados cálculos llevan a expresiones que no son números (representados por R como NaN's, del inglés 'not a number'),o respuestas con valor infinito positivo (Inf) o infinito negativo (-Inf). NULL representa el valor nulo, y es útil para que una función no devuelva ningún valor o para establecer que un argumento no se pasa a una función.

Atributos de los objetos

Los objetos están compuestos de elementos que tienen una serie de atributos:

- Los atributos de un objeto suministran información específica sobre el propio objeto.
- El modo o tipo de un objeto es un caso especial de un atributo de un objeto. Con el modo de un objeto designamos el tipo básico de sus constituyentes fundamentales.
- Todos los objetos tienen dos atributos intrínsecos: el modo y su longitud.
- Las funciones mode(objeto) y length(objeto) se pueden utilizar para obtener el modo y longitud de cualquier estructura.
- Mediante attributes(objeto) podemos obtener una lista de los atributos no intrínsecos y con attr(objeto,atributo) podemos usar el atributo seleccionado (p.e. para asignarle un valor).

Asignación

- El operador de asignación (<-) permite almacenar valores
- La función principal para definir un objeto es a través de sus componentes, con la función c(),

```
a<-c(1,8)
```

- Las asignaciones pueden realizarse también con una flecha apuntando a la derecha, realizando el cambio obvio en la asignación.

```
c(1,8) ->a
```

- La asignación puede realizarse también mediante la función assign().

```
a<-c(1,8) # es equivalente a assign("a", c(1, 8))
```

- La letra c significa "concatenar", y de hecho es un acrónimo para dicha palabra. Vamos a crear y a concatenar dos vectores:

```
> x = c(1,3,5)
> y = c(2,4,6)
> c(x,y)

[1] 1 3 5 2 4 6
```

Generación de Datos

Secuencias Regulares.

R dispone de instrucciones para generar secuencias de números. Una de las más utilizadas es el operador **":"**

```
#Generamos un vector con los números 1, 2, 3, 4, …, 19, 20.
> 1:20 #Esto es equivalente al vector c(1, 2, …, 19, 20)
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20

#El operador ":" tiene la máxima preferencia
> n <- 10
> 1:n-1    #Aquí prevalece ":" sobre "-"
 [1] 0 1 2 3 4 5 6 7 8 9
> 1:(n-1) #Forzamos la prioridad del "-"
 [1] 1 2 3 4 5 6 7 8 9
```

Con la función **seq()** también se pueden generar secuencias de números

```
#Generamos una secuencia de 1 a 30 saltando dos números cada vez
> seq(1,30,by=2)
 [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29

#La función seq admite también la longitud de la secuencia que queremos generar,

> seq(1,15.5,length=8)
 [1]  1.000000  3.071429  5.142857  7.214286  9.285714 11.357143 13.428571 15.500000
```

La función **rep()** sirve para generar repeticiones de objetos (escalares o vectores)

```
> rep(1, 30)
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Si se quiere, también es posible introducir datos directamente desde el teclado usando la función **scan()** sin opciones:

Generación de Datos

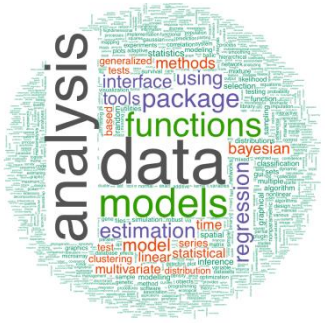
Secuencias Aleatorias.

R dispone de la posibilidad de generar datos aleatorios para un gran numero de funciones y distribuciones. Estas funciones son de la forma **rfunc(n, p1, p2, ...),** donde func indica la distribución, n es el numero de datos generado, y p1, p2, . . . son valores que toman los parámetros de la distribución.

Distribución/función	función
Gausse (normal)	`rnorm(n, mean=0, sd=1)`
exponencial	`rexp(n, rate=1)`
gamma	`rgamma(n, shape, scale=1)`
Poisson	`rpois(n, lambda)`
Weibull	`rweibull(n, shape, scale=1)`
Cauchy	`rcauchy(n, location=0, scale=1)`
beta	`rbeta(n, shape1, shape2)`
'Student' (t)	`rt(n, df)`
Fisher–Snedecor (F)	`rf(n, df1, df2)`
Pearson (χ^2)	`rchisq(n, df)`
binomial	`rbinom(n, size, prob)`
geométrica	`rgeom(n, prob)`
hypergeométrica	`rhyper(nn, m, n, k)`
logística	`rlogis(n, location=0, scale=1)`
lognormal	`rlnorm(n, meanlog=0, sdlog=1)`
binomial negativa	`rnbinom(n, size, prob)`
uniforme	`runif(n, min=0, max=1)`
Estadístico de Wilcoxon's	`rwilcox(nn, m, n), rsignrank(nn, n)`

Vectores

- R trabaja con estructuras de datos. La estructura más simple es el vector numérico, que consiste en una colección ordenada de números (1 ó más).
- La función vector(), que tiene dos argumentos mode y length, crea un vector cuyos elementos pueden ser de tipo numérico, lógico o carácter dependiendo del argumento especificado en mode (0, FALSE o " " respectivamente).
- Las siguientes funciones tienen exactamente el mismo efecto y tienen un solo argumento (la longitud del vector): numeric(), logical(), y character().



Extracción de elementos de un Vector

- Especificar los índices de los elementos a extraer:

```
> x = c(4,3,6,5,7,6,8)
> x[c(1,3,6)]
[1] 4  6  6
```

La orden anterior extrae los elementos 1, 3 y 6 del vector. Un número negativo precediendo al índice significa exclusión. Con el vector x creado anteriormente:

```
> x[-3]
[1] 4 3 5 7 6 8
> x[-c(1,2)]
[1] 6 5 7 6 8
```

- Especificar una condición lógica. En el caso del vector x creado arriba:

```
> x>6
[1]  FALSE FALSE FALSE FALSE TRUE FALSE TRUE
> x[x>6]
[1] 7 8
```

- En el caso de un vector de variables, podemos utilizar los nombres de las variables para extraer los elementos:

```
> A = 1
> B = 3
> C = 5
> y = c(A,B,C)
> y
[1] 1 3 5
> y[B]
[1] 5
```

Operadores

Aritmética de vectores.

R tiene aritmética vectorial, por lo que los vectores pueden aparecer en las expresiones que generamos.

En caso que los vectores que aparecen en una expresión no sean de la misma longitud, el más corto se "recicla" hasta que alcanza la longitud del más largo.

#Generamos dos vectores.

```
> x <- c(1., 2., 4.5, 7.6, 6.4)
> y <- c(x,0,x)
> x
[1] 1.0 2.0 4.5 7.6 6.4

> y
[1] 1.0 2.0 4.5 7.6 6.4 0.0 1.0 2.0 4.5 7.6 6.4

> xz=2*x+y+1

Warning message: In 2 * x + y : longer object length is
not a multiple of shorter object length

> xz
[1] 4.0 7.0 14.5 23.8 20.2 3.0 6.0 12.0 20.7 21.4 9.4
```

Operador/función	Símbolo/instrucción
suma	+
resta	-
multiplicación	*
división	/
módulo	%%
división entera	%/%
raíz cuadrada	sqrt
logaritmo nep.	log
log gen	logb
exponencial	exp
seno	sin
coseno	cos
tangente	tan
máximo	max
mínimo	min
rango	range
longitud	length
sumatorio	sum
producto	prod
media	mean
desv. estándar	sd
varianza	var

Vectores Lógicos

Los valores de un vector lógico pueden ser TRUE o T (cierto), FALSE o F (falso) y NA/NaN.

Los vectores lógicos se generan mediante condiciones:

```
#Generamos un vector de 1 a 10
> x <- 1:10
#cond1 vector lógico, de la misma longitud que x, donde cada casilla
#nos dice si la correspondiente casilla de x cumple la condición x>7
> cond1 <- x > 7
> cond1

[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
```

En R los vectores lógicos se pueden utilizar en aritmética ordinaria, siendo substituído (coercionado) el FALSE por 0 y el TRUE por 1.

```
> cond2 <- x >= 9   #Generamos otra condición
> cond1 & cond2   #Hacemos una and lógica de las dos condiciones
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
> !cond1   #Negación lógica del vector cond1
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

Operador	Símbolo
igualdad	==
desigualdad	!=
menor	<
menor igual	<=
mayor	>
mayor igual	>=
and lógica	&
or lógica	|
negación lógica	!

Factores/Variables Categóricas

- Un factor es un vector utilizado para especificar una clasificación discreta de los elementos de otro vector de la misma longitud.
- En R existen dos tipos de factores:
  - No ordenados (nominales): No existe jerarquía entre ellos (p.e., colores)
  - Ordenados (ordinales): Existe jerarquía entre ellos (p.e., grupos de edad)
- Se pueden crear a partir de un vector numérico con las funciones as.factor(), as.ordered() o con la función gl().
- También a partir de un vector de caracteres utilizando factor().
- Las etiquetas se asignan con levels().
- La función factor crea un factor con las siguientes opciones:

```
factor(x, levels = sort(unique(x), na.last = TRUE), labels = levels, exclude = NA,
ordered = is.ordered(x))
```

levels especifica los posibles niveles del factor (por defecto los valores únicos de x), labels define los nombres de los niveles, exclude especifica los valores x que se deben excluir de los niveles, y ordered es un argumento lógico que especifica si los niveles del factor están ordenados.

Variables Indexadas (Arrays)

Un *array* es un conjunto de datos de k dimensiones. El caso más sencillo se da con k=2, lo que llamaremos matriz (*matrix*). Todos los elementos de un *array* han de ser del mismo tipo.

En R cualquier *array* ha de tener asociado un atributo llamado *dim* que indique los límites superiores de cada una de las dimensiones. Por definición el límite inferior es 1.

```
> a <- 1:42 # Creamos un vector de 42 posiciones

# Lo transformamos en un array añadiéndole el límite superior de cada dimensión. En este caso en un array de # tres dimensiones de longitudes 3, 7 y 2. Nótese que las dimensiones que se "mueven" más rápido son las de # más a la izquierda.

> dim(a) <- c(3,7,2)
> a
, , 1

     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    1    4    7   10   13   16   19
[2,]    2    5    8   11   14   17   20
[3,]    3    6    9   12   15   18   21

, , 2

     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   22   25   28   31   34   37   40
[2,]   23   26   29   32   35   38   41
[3,]   24   27   30   33   36   39   42
```

Variables Indexadas (Arrays)

Existen dos maneras de crear una variable indexada:

- Un vector puede transformarse en una variable indexada cuando se asigna un vector de dimensiones al atributo dim

```
z<-numeric(800); dim(z) <- c(2,4,100)
```

- Utilizando la función array(vector de datos,vector de dimensiones)

```
h<-numeric(24); Z <- array(h, dim=c(3,4,2))
```

- La creación de la variable indexada sigue la regla de que el primer índice es el que se mueve más rápido y el último es el más lento.

```
Si se define una variable indexada, a, con vector de dimensiones c(3,4,2), la variable indexada
tendrá 3x4x2 = 24 elementos que se formarán a partir de los elementos originales en el orden
a[1,1,1], a[2,1,1], ..., a[2,4,2], a[3,4,2]
```

- Para referirnos a un elemento concreto de una variable indexada (array) daremos el nombre de la variable y, entre corchetes, los índices que lo refieren, separados por comas

```
a[1,1,1], a[2,1,1], ..., a[2,4,2], a[3,4,2]
```

Variables Indexadas (Arrays)

Operaciones con variables indexadas y vectores:

Cuando se realizan operaciones que mezclan variables indexadas y vectores, se siguen los siguientes criterios:

- La expresión se analiza de izquierda a derecha.
- Si un vector es más corto que otro, se extiende repitiendo sus elementos (lo que se denomina reciclado) hasta alcanzar el tamaño del vector más largo.
- Si solo hay variables indexadas y vectores más cortos, las variables indexadas deben tener el mismo atributo dim, o se producirá un error.
- Si hay un vector más largo que una variable indexada anterior, se produce un mensaje de error
- Si hay variables indexadas y no se produce error, el resultado es una variable indexada del mismo atributo dim que las variables indexadas que intervienen en la operación.

Matrices

- Una matriz es realmente un vector con un atributo adicional (dim) el cual a su vez es un vector numérico de longitud 2, que define el número de filas y columnas de la matriz.
- Las matrices son un caso particular de array con dos dimensiones
- Todos los elementos deben ser del mismo tipo.
- Una matriz se define con el comando matrix() especificando el número de filas y columnas o asignando la dim a un vector.
- Recordar que la matriz se crea por columnas, aunque con la opción byrow=TRUE lo hace por filas.
- Podemos asignar nombres a las filas y columnas con el atributo dimnames.
- Las funciones is.matrix() y as.matrix() comprueban o fuerzan el carácter de matriz de un objeto.

Matrices

Funciones útiles para trabajar con matrices

Función	Utilidad
ncol(x)	Número de columnas de x.
nrow(x)	Número de filas de x.
t(x)	Transpuesta de x
cbind(...)	Combina secuencias de vectores/matrices por col's.
rbind(...)	Combina secuencias de vectores/matrices por filas.
diag(x)	Extrae diagonal de matriz o crea matriz diagonal.
col(x)	Crea una matriz con elemento ij igual al valor j
row(x)	Crea una matriz con elemento ij igual al valor i
apply(x,margin,FUN,)	Aplica la función FUN a la dimensión especificada en margin 1 indica filas, 2 indica columnas. NB.
outer(x,y,fun="*") otra forma x%o%y	Para dos vectores x e y , crea una matriz A[i,j]=FUN(x[i],y[j]). Por defecto crea el producto externo.

Matrices

Operaciones con matrices

Función	Utilidad
x %*% y	Multiplicación de matrices
crossprod(x,y=x)	Idem que t(x) %*% y, pero más rápida
cov(x,y=x,use="all.obs")	Matriz de varianzas-covarianzas
cor(x,y=x,use="all.obs")	Matriz de correlaciones
scale(x,center=,scale=)	Resta a las columnas la media si center=TRUE, Si center es un vector, resta dichos valores. Idem para scale ,luego de centrar, pero divide por la desviación típica si scale=TRUE o los valores asignados si es un vector.
chol(x)	Descomposición de Choleski.
solve(a,b,tol=1e-7)	Resolución de la ecuación a %*% x=b. tol=tolerancia para detectar dependencias lineales en las columnas de a
eigen(x)	Cálculo de valores y vectores propios.
sdv(x)	Descomposición en valores singulares.

Matrices

Operaciones con matrices

```
> matrix(1:6)
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
[5,]    5
[6,]    6
```

Crea una matriz con 6 elementos. Al no especificarse nada, se entiende que se desea crear un vector columna

```
> matrix(1:6,nrow=2)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Crea una matriz con 6 elementos y dos filas. Los elementos, que son los números 1,2,3,4,5,6 se van leyendo por columnas.

```
> matrix(1:6,nrow=2,byrow=T)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

Igual que en el caso anterior, pero se lee por filas, al especificar que la lectura por filas está activada.

Matrices

Asignación de nombres a filas y columnas de matrices

```
>datos = matrix(c(20,65,174,22,70,180,19,68,170),
nrow=3,byrow=T)
```

Se crea una matrix 3x3

```
> datos
     [,1] [,2] [,3]
[1,]   20   65  174
[2,]   22   70  180
[3,]   19   68  170
```

Se muestra la matriz

```
> colnames(datos) = c("edad","peso","altura")
```

Se asignan nombres a las columnas

```
> datos
     edad peso altura
[1,]   20   65    174
[2,]   22   70    180
[3,]   19   68    170
```

Se vuelve a mostrar la matriz

```
> rownames(datos) = c("paco","pepe","kiko")
```

Se asignana nombres a las columnas

```
> datos
     edad peso altura
paco   20   65    174
pepe   22   70    180
kiko   19   68    170
```

Se vuelve a mostrar la matriz. Ya se ven los nombres asignados.

Listas

- Una lista es un objeto consistente en una colección ordenada de objetos que se suelen llamar componentes.
- Estos componentes no tienen que ser necesariamente del mismo tipo o *mode*, y pueden ser vectores, *arrays* o nuevas listas.
- Una lista puede estar compuesta de, por ejemplo, un vector numérico de tamaño 2, un valor lógico, un vector de tamaño 3, una matriz y una función.
- Se construyen con la función list() o concatenando otras listas.
- Son una parte importante de la programación de funciones en R.
- Los componentes siempre están numerados y pueden ser referidos por dicho número, o por su nombre (si lo tiene, por defecto no lo tiene).
- Debemos ir con mucho cuidado al seleccionar partes de la lista:
  - La selección de elementos se hace con doble corchete o con el nombre del elemento precedido del símbolo del dólar.
  - Utilizamos el corchete simple estamos considerando una sublista (de menos componentes) de la lista.

Listas

- Las listas sirven para concatenar objetos donde cada uno puede tener una estructura distinta. Esto no ocurre, por ejemplo, en los arrays, donde todos los elementos deben ser del mismo tipo (todos números, o todos carácter digamos).
- Una lista tiene una serie de componentes, a los que deberemos asignar un nombre.
- Para acceder a componentes concretos se usa el operador $ seguido del nombre de la componente de la lista, o bien el número de la componente entre corchetes dobles [[]]:

```
> familia$padre
[1] "jose“
> familia$numero.hijos
[1] 3
> familia[[1]]
[1] “jose”
> familia[[3]]
[1] 3
```

Listas

Creamos una lista

```
> lista1 <- list(padre="José", madre="María", num.hijos=4,  edad.hijos=c(4,7,9,13))
> lista1
$padre
[1] "José"

$madre
[1] "María"

$num.hijos
[1] 4

$edad.hijos
[1] 4 7 9 13
```

Los elementos de una lista siempre están numerados
Los elementos de la lista se indexan mediante dobles claudators ([[]]). Si lista1[[4]] es un vector, entonces lista1[[4]][1] es su primer elemento.

```
> lista1[[1]]
[1] "José"

> lst[[3]]
[1] 3

> lst[[4]][1]
[1] 4
```

Listas

La función *length*, aplicada a una lista, devuelve el número de componentes "de primer nivel" que contiene.

```
> length(lista1)
[1] 4
```

Los componentes de las listas también pueden tener nombres. En este caso, también nos podremos referir a ellos por su nombre además por su posición, ayudados del símbolo del dólar ($). Esto es útil para no tener que recordar en qué posición está cada componente de la lista.

```
> lista1$madre #Equivalente a lista1[[2]]
[1] "María"

> lista1$edad.hijos[2] #Equivalente a lista1[[4]][2]
[1] 7

# Hay que distinguir entre lista1[1] y lista1[[1]]. El primer comando devuelve una sublista.
# Mientras que el segundo devuelve el primer componente de la lista.

> lista1[1] #Si la lista tiene nombres, éstos se transfieren a la sublista.
$padre
[1] “José"

> lista1[[1]] #Se devuelve el primer elemento de la lista respetando su tipo.
[1] “José"
```

Data Frames

- La forma más habitual de almacenar datos es hacer uso de tablas (data.frames en R). Esto es como una matriz, formada por filas y columnas, con la diferencia que cada columna puede ser una variable de tipo diferente.
- En una tabla pueden coexistir columnas con información numérica, entera, decimal, otras con información cualitativa de caracteres, otras lógicas, etc. Lo más frecuente es que estas tablas tengan dos dimensiones (filas y columnas), pero en algún caso puede tener más de dos dimensiones.
- Los data frames son una estructura de datos que generaliza a las matrices, en el sentido en que las columnas (variables a menudo) pueden ser de diferente tipo entre sí (no todas numéricas, por ejemplo). Sin embargo, todos los elementos de una misma columna deben ser del mismo tipo. Al igual que las filas y columnas de una matriz, todos los elementos de un data frame deben ser de la misma longitud.
- De este modo, pueden usarse funciones tales como dimnames, dim, nrow sobre un data frame como si se tratara de una matriz. Los datos de un data frame pueden ser accedidos como elementos de una matriz o de una lista.

Data Frames

Ejemplos de construcción de DataFrames.

```
> datos = matrix(c(20,65,174,22,70,180,19,68,170),nrow=3,byrow=T)

> dimnames(datos)<-list(c("paco","pepe","kiko"),   c("edad","peso","altura"))
```

Vamos a añadir una columna a la matriz datos para que contenga la provincia de origen de cada persona:

```
> provincia = c("madrid","malaga","murcia")
> datos2 = cbind(datos,provincia)
> datos2
     edad peso altura provincia
paco "20" "65" "174"  "madrid"
pepe "22" "70" "180"  "malaga"
kiko "19" "68" "170"  "murcia"
```

Para construir una estructura de tipo data.frame se utilizará la función data.frame()

```
> dfra<-data.frame(datos,provincia)
> dfra
     edad peso altura provincia
paco   20   65    174    madrid
pepe   22   70    180    malaga
kiko   19   68    170    murcia
```