Vectores y Factores

Sitio:	Plataforma de Formación On line del Instituto Andaluz de Administración Pública	Imprimido por:	ALFONSO LUIS MONTEJO RAEZ
Curso:	(I22F-PT05) Entorno de Programación R	Día:	lunes, 4 de abril de 2022, 10:26
Libro:	Vectores y Factores		

Tabla de contenidos

1. Que es un vector

Un **vector** es una secuencia ordenada de datos que presenta las siguientes características:

- Todos los elementos almacenados en un vector deben ser del mismo tipo, normalmente numéricos o carácter, es decir, no podemos almacenar números y letras en un mismo vector.
- Si se mezclan distintos tipos, todos se convierten a un tipo que sea común a todos ellos, normalmente a carácter.
- Las órdenes para crear un vector son **c()** y **scan()**.

Ejemplos de vectores pueden ser:

(1,2,3,4,5) una sucesión de números o (AL, CA, CO, GR, HU, JA, MA, SE) una sucesión de caracteres con las iniciales de las provincias de Andalucía.

2. Creando vectores

Vamos a empezar a crear un vector usando las órdenes **c()** o **scan()**.

La diferencia entre las dos es que la orden scan() crea el vector de manera interactiva y la orden c() hay que escribir el contenido del vector de una vez.

Cosas a tener en cuenta al crear un vector:

- Tenemos que recordar que el vector solo puede contener un tipo de dato (o numérico o caracter) y no se pueden mezclar.
- Si mezclamos tipos el vector se convertirá en caracter.
- Si vamos a usar texto dentro del vector debemos usar comillas para indicarlo.
- Recordar que mayúsculas y minúsculas no son lo mismo.
- Para trabajar con los vectores la orden siempre tiene que estar asignada a un objeto para que se almacene.

2.1. Concatenar c()

La orden **c()** realmente lo que hace es concatenar cosas.

Para usarla simplemente escribimos la orden y vamos separando sus valores por comas.

c(1,2,3,4,5) -> datos

En este ejemplo se crearía un vector llamado datos y con los valores del 1 al 5.

Ejemplo

```
> prueba<-c(4,Almería)
Error: object 'Almería' not found
> prueba<-c(4,'Almería')
> prueba
[1] "4"       "Almería"
> class(prueba)
[1] "character"
```

En este ejemplo se crea un vector llamado prueba que va a tener los valores 4 y Almería.

En primer lugar nos da un error ya que Almería es un texto y debe ir entrecomillado (si no lo hacemos así R piensa que es un objeto).

Después, al mezclar dos tipos de datos distintos (número y texto) los trata a los dos como si fueran texto y crea un vector de texto (character).

2.2. Escanear scan()

Con la orden **scan()** se crea un vector de manera interactiva, es decir, introduciendo valores desde teclado.

scan()->datos

Ejemplo

```
> prueba<-scan()
1: 54
2: 19
3: 478
4:
Read 3 items
> prueba
[1]  54  19 478
```

En este ejemplo creamos un vector llamado prueba y vamos tecleando los valores que va a contener el vector. Los valores aparecen numerados.

Cuando queremos terminar de meter los datos, simplemente pulsamos enter (como si dejáramos un valor en blanco) y se cierra el vector.

El vector prueba tiene los valores 54, 19 y 478.

La orden, por defecto, solo crea vectores numéricos. Si queremos almacenar texto o caracteres hay que indicarlo como un parámetro en la orden.

```
> prueba<-scan(what='character')
1: Almería
2: Cádiz
3: Lebrija
4:
Read 3 items
> prueba
[1] "Almería" "Cádiz"   "Lebrija"
```

Ojo: Recordad que si usamos un objeto que ya existe y le asignamos otro contenido estamos machando el contenido original.

2.3. Otras formas

Otras órdenes útiles para la creación de los vectores son:

- **rep(x,n)**. Repite n veces el valor de x.
- **seq(a,b,by=c**). Crea una secuencia desde a hasta b de paso c.
- **Dos puntos (:)**. También se puede crear una sucesión con los dos puntos.

Ejemplo

```
> rep('Almería',5)
[1] "Almería" "Almería" "Almería" "Almería" "Almería"
> seq(1975,1990,by=5)
[1] 1975 1980 1985 1990
> 2010:2018
[1] 2010 2011 2012 2013 2014 2015 2016 2017 2018
```

En el primer caso se repite el valor Almería cinco veces, en el segundo creamos una secuencia desde 1975 a 1990 saltando de cinco en cinco y por último creamos una secuencia desde el 2010 al 2018 usando los dos puntos.

En estos ejemplos no hemos usado la asignación por lo que no se han creado los objetos correspondientes, simplemente R nos muestra el resultado directamente en la consola.

3. Trabajando con vectores

Ya sabemos como crear los vectores ahora vamos a ver como podemos trabajar con ellos. Lo principal que vamos a ver será:

- Acceder al contenido de un vector.
- Filtrar el contenido de un vector.
- Operaciones con un vector.

3.1. Acceso

Una vez creado un vector, se puede acceder a cualquier valor dentro de él indicando su posición con corchetes.

nombre_del_vector[posición]

Ejemplo

Vamos a ver un ejemplo para tenerlo más claro.

```
> provincias<-c('AL','CA','CO','GR','HU','JA','MA','SE')
> str(provincias)
 chr [1:8] "AL" "CA" "CO" "GR" "HU" "JA" "MA" "SE"
```

Creamos un vector llamado provincias donde almacenamos las iniciales de las ocho provincias de Andalucía. Por lo tanto tenemos un vector de tipo caracter con ocho valores.

Para acceder al contenido del vector, usamos el nombre del vector y, entre corchetes, las posiciones a las que queremos acceder.

```
> provincias[4]
[1] "GR"
```

Accede a la cuarta posición

```
> provincias[c(1,7)]
[1] "AL" "MA"
```

Accede a la primera y séptima posición

```
> provincias[2:4]
[1] "CA" "CO" "GR"
```

Accede a todas las posiciones entre la segunda y cuarta inclusive

```
> provincias[-2]
[1] "AL" "CO" "GR" "HU" "JA" "MA" "SE"
```

Accede a todas las posiciones salvo a la segunda

Podemos acceder de la siguiente forma:

- **Usando un solo número** se accede a la posición deseada.
- Más de un número, debemos usar la orden de concatenar **c()** separando los valores con comas.
- **Usando los dos puntos** creamos una sucesión entre los valores que indiquemos.
- **Usando el negativo** accedemos a todas las posiciones salvo la que indiquemos con el negativo.

3.2. Filtrado

Otra de las operaciones que podemos hacer es poner condiciones para filtrar y quedarnos con aquellos valores del vector que cumplan determinadas condiciones.

El filtrado de elementos de un vector vuelve a crear otro vector con el conjunto de valores que hayan cumplido la condición establecida.

Los operadores que podemos usar para filtrar un vector son los siguientes (estos operadores se usan para filtrar cualquier elemento de R):

Operador	$=$	$\neq$	$<$	$>$	$\leqslant$	$\geqslant$	negación	conjunción	disjunción
Signo	==	!=	<	>	<=	>=	!	&	|

Lo más importante a tener en cuenta es que, para determinar la igualdad entre dos valores hay que usar el signo ==. Con esto tendréis muchas confusiones siempre que estéis programando en R, pero es cuestión de acostumbrarse.

Ejemplo

Vamos a crear un vector llamado notas donde almacenaremos la calificación de 11 personas (recordad que el separador decimal es el punto).

```
> notas<-c(4.7,5,6,8.2,9.4,2.1,6.2,3.7,7.2,8,5.4)
```

Si quiero filtrar y quedarme con las personas que han aprobado podría escribir la siguiente orden.

```
> notas>=5
 [1] FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
> notas[notas>=5]
[1] 5.0 6.0 8.2 9.4 6.2 7.2 8.0 5.4
```

Hay que tener cuidado a la hora de escribir la condición. Si nos fijamos en las dos líneas de código, la primera nos devuelve el listado de valores del vector diciéndonos si cumplen (TRUE) o no cumplen (FALSE) la condición que hemos puesto.

Sin embargo, la segunda línea de código nos devuelve un vector con los valores que cumplen la condición.

Recordad que, para almacenar lo que hemos creado, hay que asignar la orden a un objeto. En nuestro caso, al no haber hecho la asignación, los valores simplemente aparecen en la consola pero no se guardan.

3.3. Operaciones con un vector

También podemos hacer operaciones con los vectores. Vamos a ver una serie de órdenes o funciones que nos sirven para trabajar con los vectores.

Función	longitud	máximo	mínimo	suma	producto
Signo	`length`	`max`	`min`	`sum`	`prod`
Función	media	sumas acumuladas	diferencias	ordenar	invertir el orden
Signo	`mean`	`cumsum`	`diff`	`sort`	`rev`

Cada una de estas funciones nos devuelve información sobre el vector.

Ejemplo

```
> notas<-c(4.7,5,6,8.2,9.4,2.1,6.2,3.7,7.2,8,5.4)
```

Si seguimos con el vector de notas anterior, podemos ver cuantos elementos tiene el vector (orden length), obtener el valor máximo y mínimo (órdenes max y min) o calcular la nota media (orden mean).

```
> length(notas) # Cuantos elementos hay en notas
[1] 11
> max(notas) # Máxima nota
[1] 9.4
> mean(notas) # Nota media
[1] 5.990909
> sort(notas) # Notas ordenadas
 [1] 2.1 3.7 4.7 5.0 5.4 6.0 6.2 7.2 8.0 8.2 9.4
```

Otras funciones útiles cuando trabajamos con vectores son aquellas que nos permiten localizar elementos dentro del vector. Estas son las funciones **which:**

- **which()**. Nos sirve para buscar un valor dentro del vector.
- **which.max()**. Localizar el máximo valor del vector.
- **which.min()**. Localizar el mínimo valor del vector.

Lo que nos devuelven estas funciones no son los valores en sí, sino cuál es la posición que ocupan dentro del vector. Vamos a ver varios ejemplos para saber como trabajan estas funciones.

```
> which.max(notas)
[1] 5
> max(notas)
[1] 9.4
> which.min(notas)
[1] 6
> which(notas>=5)
[1]  2  3  4  5  7  9 10 11
```

La primera orden nos devuelve la posición que ocupa en el vector el valor máximo. La mayor nota era un 9,4 y está en la quinta posición del vector.
La tercera línea nos devuelve en que posición se encuentra el menor valor del vector, que en este caso es en la sexta posición y por último, la cuarta línea, nos indica cuales son las posiciones que ocupan en el vector todas aquellas notas que son superiores o iguales a cinco.

4. Factores

Los **factores** son un tipo especial de vector que sirve para poder categorizar los valores contenidos en él. Es un constructo que se utiliza principalmente en estadística para trabajar con variables de tipo cualitativo como por ejemplo: Sexo, provincia de nacimiento, categoría profesional, actividad económica, etc.

Normalmente se utiliza con vectores de tipo caracter y se trata de codificar esta información.

Para crear un factor se utiliza la orden **factor()**.

Para ver como funciona un factor y la diferencia entre vector y factor vamos a ver el siguiente ejemplo.

Ejemplo

Vamos a crear un vector con el sexo de siete personas.

```
> Sexo<-rep(c('Hombre','Mujer'),c(3,4))
> Sexo
[1] "Hombre" "Hombre" "Hombre" "Mujer"  "Mujer"  "Mujer"  "Mujer"
> class(Sexo)
[1] "character"
```

La forma más rápida es usando la función rep() que crea un repetición de las palabras hombre y mujer, repitiéndolas 3 y 4 veces respectivamente. Con esta acción hemos creado un vector de tipo caracter llamado sexo cuyos valores son Hombre y Mujer de manera repetida.

```
> factor(Sexo)->Sexo2
> str(Sexo2)
 Factor w/ 2 levels "Hombre","Mujer": 1 1 1 2 2 2 2
> class(Sexo2)
[1] "factor"
```

Sin embargo, si creamos un factor a partir del vector sexo que ya teniamos creado (usando la función **factor()**), observamos que lo que ha hecho es crear dos categorías que son Hombre y Mujer a las que les ha asignado el valor 1 y 2 respectivamente.

Ya no es un vector normal y corriente cuyos valores son Hombre y Mujer, ahora es un factor que contiene los valores 1 y 2, donde 1 significa Hombre y 2 significa mujer, es decir, ha codificado los valores del vector sustituyéndolos por sus códigos y etiquetas.

```
> Sexo2
[1] Hombre Hombre Hombre Mujer  Mujer  Mujer  Mujer
Levels: Hombre Mujer
> levels(Sexo2)
[1] "Hombre" "Mujer"
```

Realmente lo que hemos hecho es categorizar el vector sexo convirtiéndolo en un factor.

4.1. Trabajando con factores

Cuando trabajamos con factores hay varias órdenes que son muy útiles para poder definir bien los valores del factor. Estas órdenes son:

- **levels()**. Acceder o cambiar las categorías.
- **ordered()**. Crear un factor con categorías ordenadas.

Ejemplo 1

Partiendo del factor Sexo2 vemos con la orden **levels()** que tiene como categorías los valores Hombre y Mujer. Si quiero cambiarlos por H y M simplemente le asigno las nuevas categorías en la segunda línea de código.

```
> Sexo2
[1] Hombre Hombre Hombre Mujer  Mujer  Mujer  Mujer
Levels: Hombre Mujer
> levels(Sexo2)
[1] "Hombre" "Mujer"
> levels(Sexo2)<-c('H','M')
> Sexo2
[1] H H H M M M M
Levels: H M
```

Ejemplo 2

Si queremos crear un factor donde las categorías guarden un orden entre sí usamos la orden **ordered()**.

Si creamos un factor con los estudios que tiene una persona usando ordered(), las categorías tienen un orden donde los estudios de primaria son los menores, seguidos de secundarios y por último los superiores. Esto es un factor pero con categorias ordenadas.

```
> Estudios<-c('Sec','Prim','Sup','Sup','Sec','Prim','Sec')
> ordered(Estudios)
[1] Sec  Prim Sup  Sup  Sec  Prim Sec
Levels: Prim < Sec < Sup
```