Programación científica en R

Introducción a R

Marcos Ehekatzin García Guzmán

Agosto de 2024

1. Modos y atributos
2. Factores Nominales
3. Variables indexadas (Arrays)
4. Listas y hojas de datos
5. Data Frames

Modos y atributos

Atributos intrínsecos: Modo y atributos

- Como ya vimos, las entidades que manipula R se conocen como *objetos*.
- Hasta ahora el tipo de objetos que hemos visto se denominan *atómicos* debido a que todos sus elementos son del mismo tipo o *modo* (numérico, lógico, caracteres).
- Así el vector será del mismo modo que sus elementos.
- Solo hay una excepción, que surge cuando un vector contiene valores faltantes (`NA`).
- Cabe destacar que, incluso si un vector es vacío éste tendra un modo.

Atributos intrínsecos: Modo y atributos

- Con el modo de un objeto podemos designar el tipo básico de sus elementos. Con la función `mode()` podemos obtener el modo de un objeto.
- Con `R` podemos modificar el modo de cualquier objeto.
- Por ejemplo, considere el vector `z`

```
z <- 0:9
mode(z)
```

```
## [1] "numeric"
```

```
#Para hacer z un vector de caracteres
digits <- as.character(z)
#Para hacer digits un vector numérico
d <- as.integer(digits)
```

- Se puede utilizar la función `as.lo que sea` para cambiar el modo.

Modificación de la longitud de un objeto

- Recordemos que aunque un objeto esté vacío, tiene modo. Por ejemplo:

```
#Ejemplo 1
mode(x[x<0])
```

```
## [1] "numeric"
```

```
#Ejemplo 2
v <- numeric()
mode(v)
```

```
## [1] "numeric"
```

Modificación de la longitud de un objeto

- Una vez creado un objeto pueden añadirse nuevos elementos simplemente asignándolos a un índice que esté fuera del rango original:

```
v[3] <- 17
```

- ¿Cuál será la longitud del vector `v`? ¿Cuáles serán sis elementos?

- De la misma manera, puede reducirse la longitud de un objeto simplemente volviendo a hacer una asignación

```
alfa <- 1:10
alfa <- alfa[2*1:5]
```

- ¿Cómo se modificó el objeto `alfa`?

Modificación de atributos

- Ademas de un modo, los objetos tienen atributos. Con la función `attributes()`podemos ver la lista de todos los atributos de un objeto que han sido definidos o asignar atributos nuevos.
- Por su parte, con la función `attr(nombre, objeto)` podemos asignar un atributo nuevo. Por ejemplo, podemos asignarle una nueva dimensión a un objeto

```
attr(z,"dim") <- c(5,2)
z
```

```
##      [,1] [,2]
## [1,]    0    5
## [2,]    1    6
## [3,]    2    7
## [4,]    3    8
## [5,]    4    9
```

Factores Nominales

Factores Nominales

- Un *factor* es un vector utilizado para especificar una clasificación discreta de los elementos de otro vector de igual longitud.
- Ejemplo:

```
estado <- c("tas", "sa", "qld", "nsw", "nsw", "nt",
            "wa", "wa","qld", "vic", "nsw", "vic",
            "qld", "qld", "sa", "tas","sa", "nt",
            "wa", "vic", "qld", "nsw", "nsw", "wa",
            "sa", "act", "nsw", "vic", "vic", "act")
```

- Para hacer que el vector `estado` se haga un factor utilizaremos la función factor:

```
FactorEstado <- factor(estado)
levels(FactorEstado)
```

```
## [1] "act" "nsw" "nt"  "qld" "sa"  "tas" "vic" "wa"
```

Función `tapply()`

- Sigamos el ejemplo anterior y tomemos en cuenta otro vector de la misma longitud:

```
ingresos <- c(60, 49, 40, 61, 64, 60, 59, 54, 62, 69,
              70, 42, 56,61, 61, 61, 58, 51, 48, 65,
              49, 49, 41, 48, 52, 46, 59, 46, 58, 43)
```

- Para calcular la media de cada estado podemos utilizar la función `taply()`:

```
tapply(ingresos,FactorEstado, mean)
```

```
##       act       nsw        nt       qld        sa       tas
## 44.50000 57.33333 55.50000 53.60000 55.00000 60.50000 56
```

- Ejercicio:
  a. Hacer un vector llamado `tiempo` que contenga el tiempo de traslado desde su casa al Colmex.
  b. Hacer un **factor** que contenga el centro de estudios al que pertenece cada uno y otro con la maestría/doctorado que estudian.
  c. Calcular el número y porcentaje de personas en cada centro
  d. Calcular el promedio de tiempo de traslado por centro de estudios
  e. Repetir los cálculos pero por maestría/doctorado

Hint: Para calcular y el número y porcentaje de personas crearemos una función: `F <- function(x) expresión` en donde expresión es el código para calcular el número y el porcentaje de personas.

Variables indexadas (Arrays)

Variables indexadas (Arrays)

- Una variarable indexada o `array` es una colección de datos que está indexada por varios índices.
- Un vector puede transformarse en una variable indexada cuando se asigna un vector de dimensiones al atributo `dim`(como ya lo vimos).
- Un elemento de una variable indexada puede referirse dando el nombre de la variable y, entre corchetes, los índices que lo refieren separados por comas.
- Ejercicio:

a. Hagamos un vector con 1500 elementos con las dimensiones `(3,5,100)`.

b. Imprima todos los elementos del vector `s` que estén en el índice 1 y 2 de la tercera dimensión.

Uso de una variable indexada

- Los elementos de un vector, al formar un array, se ordenan siguiendo una regla: el primer índice siempre se mueve más rápido, y el último es el más lento.

```
#Ejemplo
a <- 1:8
dim(a) <- c(2,4)
a
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    5    7
## [2,]    2    4    6    8
```

- Pregunta:¿Cómo se ordena `a` si le damos las dimensiones $2 \times 2 \times 2$?

Uso de una variable indexadas como índices

- Una variable indexada puede utilizar no solo un vector de índices, sino también una *variable indexada de ínidices*.
- Por ejemplo, si tenemos un array `A <- 1:4` de dimensión 2×2 y queremos imprimir los elementos [1,1] y [2,2]:

```
i <- c(1,2,1,2)
dim(i)<- c(2,2)
A[i]
```

```
## [1] 1 4
```

Uso de una variable indexadas como índices

- Ejercicio:
  a. Hacer un Array `X` con dimensiones 4×5.
  b. Extraer los elementos `X[1,3]`, `X[2,2]` y `X[3,1]`con una array `i`
  c. Reemplazar los elementos con 0

La función `array()`

- Para construir una variable indexada con mayor facilidad podemos utilizar la función `array`, que tiene la siguiente forma:

```
> Z <- array(vector_de_datos, vector_de_dimensión)
```

- Ejercicio: tomen los vectores `w <- 2*1:10` y `q<- 3*1:9` y hagan dos arrays con cada uno. El primero con dimensión 4×3 y el segundo con dimensión 5×2
- ¿Cómo son esas array?
- ¿Qué pasaría si el vector fuera `q<-1`?

Las funciones `cbind()` y `rbind()`

- Con las funciones `cbind()` y `rbind()` podemos construir matrices nuevas uniendo matrices o vectores horizontal o verticalmente, respectivamente. La sintaxis para utilizar estas funciones es de la forma:

```
> x <- cbind(arg_1,arg_2,...)
```

- Nota `cbind()`: Los argumentos pueden ser de cualquier longitud en el caso de que sean vectores.
  - Si los argumentos son matrices entones **deben** tener el mismo número de filas.
  - Si uno de los argumentos es un vector y otro es una matriz, entonces el vector no puede ser más largo que el número de filas de la matriz.
- ¿Qué pasa si el vector es más corto?
- La función `rbind()` realiza el mismo papel pero tomando en cuneta el número de columnas.

Listas y hojas de datos

- Una *lista* es un objeto que consiste en una colección ordenada de objetos o *componentes*.
- Los componentes no tienen que tener el mismo modo, por ejemplo

```
Lst <- list(nombre="Pedro", esposa = "María", hijos =3,
            edad =c(4,7,9))
```

- Los componentes siempre están numerados. Por ejemplo:

```
#Esposa
Lst[[2]]
```

```
## [1] "María"
```

```
#Edad del segundo hijo
Lst[[4]][2]
```

```
## [1] 7
```

- Cuando son listas más grandes puede ser complicado recordar el número y la dimensión de cada uno de sus componentes.
- Para facilitar el acceso a los componentes podemos utilizar `Lst$nombre`:
- Ejercicio: Repita los ejemplos de la slide anterior utilizando esta nueva estructura.

Data Frames

Data Frames

- Un `dataframe` es una clase de objeto que tiene las siguientes particularidades:
  - Los componentes deben ser vectores (numéricos, caracteres o lógicos), factores, matrices numéricas, listas u otras ojas de datos.
  - Las matrices, listas y hojas de datos contribuyen a la nueva hoja de datos con tantas variables como columnas, elementos o variables que posean, respectivamente.
  - Los vectores numéricos y los factores se incluyen sin modificar, los vectores no numéricos se fuerzan a factores, cuyos niveles son los únicos valores que pertenecen al vector.
  - Los vectores que constituyen la hoja de datos deben tener todos *la misma longitud* y las matrices deben tener el mismo *tamaño de filas*

Construcción de Data Frames

- Se puede construir una hoja de datos utilizando la función `data.frame`.
- Con `as.data.frame` podemos forzar algunos objetos para que se conviertan en una hoja de datos.

```
cont <- data.frame(dom=FactorEstado, bot=ingresos)
```

Construcción de Data Frames

- Para acceder a un elemento del data frame utilizaremos el mismo método que con las listas (`data.frame$elemento`)[1]:
  - Ejemplo: ¿Qué valor obtenemos si escribimos `cont$bot[3]`?
- Ejercicio, tome los vectores de tiempo de traslado y los factores de centro de estudios y maestría/doctorado y haga un data frame.
- Seleccione los datos de tiempo de traslado que:
  a. Son de su centro
  b. Son de su maestría/doctorado
  c. Los datos que **no** son ni de su centro ni de su maestría/doctorado.
  d. Obtenga la media y la desviación estándar utilizando el método de su elección.

[1] También podemos agregar un nuevo elemento usando este método.

Ahora hagamos un ejercicio con datos reales.

1. Descarguen el archivo "homicidios" del `Github` de la clase y ábranlo. Este archivo contiene el número de homicidios por estado durante el periodo 1990-2023.
2. Obtengan el promedio de homicidios por año.
3. El número de homicidios por estado durante cada periodo
4. El número de homicidios total durante este sexenio.
5. La tasa de crecimiento del número de homicidios de mujeres en este sexenio (año 2018 vs año 2023) y la tasa de crecimiento de los homicidios de mujeres en el sexenio pasado (año 2012 vs 2017).