Estructuras de datos

Software Estadístico

Dra. Eva Romero Ramos

Dpto. Estadística e Investiación Operativa

Estructuras de datos

- Ya conocemos los tipos básicos de datos, ahora veremos algunas estructuras en las que se pueden almacenar los datos.
- Las estructuras de datos en R son tipos de objetos que nos permitirán almacenar y operar con los datos.
- Las estructuras de datos pueden contener datos el mismo tipo (**homogéneas**) o de distinto tipo (**heterogéneas**).
- Conoceremos en este tema las siguientes estructuras de datos: Vector, Matriz, Array, Lista y Data Frame.

Vectores

- Un vector es una estructura homogénea de una dimensión, cuyo tipo de datos hace referencia al tipo de los datos que contiene.
- Podemos crear un vector usando la función c().
  Ejemplo.- Vector <- c(1,2,5,7,9)
  CharVector <- c("Luis","Lola","Marcos")
  LogicalVector <- c(TRUE, TRUE,FALSE)
- Para ver el tamaño de un vector podemos usar la función length().
  Ejemplo.- length(Vector)

Vectores

- Podemos crear vectores formados por secuencias números.
  Ejemplo.- $v3 <- c(1:10)$
- También se pueden crear vectores por secuencias números con la función seq().
  Ejemplo.- $v4 <- seq(0, 100, by = 5)$
- Se pueden realizar operaciones aritméticas entre vectores.
  Ejemplo.- $v3 + v4$
- Se pueden agregar elementos nuevos a un vector mediante:
  $v3 <- c(v3,15)$
- Se pueden combinar vectores mediante:
  $c(v3,v4)$
- Accederemos a los elementos del vector, indicando el índice del elemento u elementos que nos interesen entre corchetes.
  $v3[2]$

Vectores

- Podemos crear una variable tipo factor a partir de un vector.
  Ejemplo.-
  colores $<-$ c(‘azul‘, ‘verde‘, ‘verde‘,‘rojo‘,‘azul‘, ‘verde‘, ‘rojo)
  colores; class(colores)
  coloresfactor $<-$ as.factor(colores)
  coloresfactor; class(coloresfactor)
- Una variable tipo factor, es una variable que toma un pequeño rango de valores alfanuméricos para cada item.

Matrices y Arrays

1. Los **arrays** y las **matrices** son estructuras homogéneas de más de una dimensión.
2. La principal diferencia entre ambos es que los arrays pueden tener cualquier número de dimensiones (incluido una) y las matrices tienen siempre dos dimensiones.
3. Para crear un array usaremos la función array():
   Ejemplo.- $A1 <- array(0, dim = c(2, 3, 4))$
4. Para crear una matrix usaremos la función matrix():
   Ejemplo.- $Mat1 <- matrix(0, nrow = 3, ncol = 4)$
5. Otra diferencia importante es que **arrays** y **matrices** tienen funciones diferentes.

Matrices

- El uso de arrays no es tan común como el uso de matrices así nos centraremos un poco más en estas.
- Otra forma de crear matrices es haciendo uso de la funciones rbind() y cbind().
- **rbind()** une filas.
  Ejemplo.- Mat4 <− rbind(2,3,5, NA)
- **cbind()** une columnas.
  Ejemplo.- Mat7 <− cbind(1:3, c(2,7,NA), c(9,6,3))
- Podemos usar el comando dim() para conocer las dimesiones de una matriz.
  Ejemplo.- dim(Mat7)

Matrices

- Accederemos a los elementos de una matriz mediante índices, indicando las posiciones del valor que nos interese entre corchetes.

  Ejemplo.- Mat7[2,3]
  Mat7[2,]
  Mat7[,3]

- Al igual que en el caso de los vectores, se pueden utilizar operadores aritméticos entre matrices.

Data Frames

- Los Data Frames son estructuras de datos bidimensionales y heterogéneas.
- Es la estructura más utilizada en análisis de datos.
- Cada columna de un data frame se refiere a una variable o atributo y cada fila corresponde con una observación.
- Para crear un data frame se utiliza el comando data.frame(). **Ejemplo**.- df_personas $<-$ data.frame(Nombre $=$ c('Juan', 'María', 'Luis'), Edad $=$ c(25,23,19), Genero $=$ c('M','F','M'))

Data Frames

- Podemos usar el comando dim() para ver la dimensión del data frame.
  Ejemplo.- dim(df_personas)
- El comando length() nos indica el número de columnas del data frame.
- names() nos da el nombre de la columas y también nos permite cambiarlo si lo usamos en una asignación.
  Ejemplo.- names(df_personas) $<-$ c('Name', 'Age','Gender')
- La función as.data.frame() permite convertir una matriz en un dataframe.
- Como un data frame está compuesto por vectores les podremos aplicar los mismos operadores.

Data Frames

- Podemos acceder a los elementos de un data frame mediante los nombres de las variables y mediante índices.
- Si lo hacemos mediante los nombres de las variables debemos escribir el nombre de data frame seguido de $ y seguido del nombre de la columna a la que queremos acceder.
  Ejemplo.- df_personas$Age
- Esta acción nos permite acceder a la columna completa del data frame.
- Si queremos acceder a un dato, es decir a una celda, debemos usar índices o combinar los nombres con índices:
  Ejemplo.- df_personas$Age[2]
  df_personas[2,2]

Listas

- Las listas son estructuras de datos unidimensionales y heterogéneas.
- Las listas pueden incluir cualquier tipo de datos, incluidos vectores, matrices e incluso dataframes.
- Para crear listas usaremos el comando list().
  Ejemplo.- lista <- list(nombre = 'Juan', edad = 21, calificaciones = c(9.5, 8.7, 7.8), aprobado = TRUE)
- Observamos en el ejemplo que cada elemento de la lista puede tener un nombre.
- Accederemos a los elementos de la lista con el nombre de la lista seguido de $.
  Ejemplo.- lista$aprobado
- Podemos obtener el tamaño de la lista con la función length().
- No se pueden aplicar operadores aritméticos a los elementos de una lista.

Coerción

- Ya hemos visto que en ocasiones nos interesa cambiar el tipo de datos para poder operar con los datos con las funciones que nos interesen.
- Para coercionar los datos tenemos las funciones del tipo as().
- En el caso de los datos estudiados las funciones de coerción son:
  - **as.vector()** → Coerciona a vector las matrices.
  - **as.matrix()** → Coerciona a matrices, vectores y dataframes.
  - **as.data.frame()** → Coerciona a dataframes los vectores y matrices.
  - **as.list()** → Coerciona a lista los vectores y dataframes.