**Introducción**

Existen numerosos tipos de estructuras de datos en todos los lenguajes de programación, cada uno con fortalezas y debilidades, los cuales son adecuados para tareas particulares.

Dado que R es un lenguaje de programación utilizado ampliamente para el análisis de datos, machine learning y depp learning, las estructuras de datos que utiliza fueron diseñadas con este tipo de trabajo en mente.

**¿Qué tipo de tipo de estructuras utiliza R?**

R puede adoptar otras estructuras de distintos lenguajes, pero nos centraremos en las que vamos a utilizar con mayor frecuencia teniendo como fin alcanzable poder generar código de aprendizaje automático, profundo e inteligencia artificial.

Las cuales son:

* vectores
* factores
* listas
* matrices
* marcos de datos o dataframes

R contempla un tipo de dataframe con mayores propiedades, a mi me gusta verlo como un dataframe con esteroides el cual se puede generar a través de una librería especifica y se le llama “tibble” o “tbl”, lo veremos próximamente, aunque por experiencia hay algunos algoritmos que no son capaces de trabajar con este tipo de dataframe y también hay algunas librerías estadísticas que no soportan este formato, el formato tbl es muy potente si tenemos en mente raspados web, importación de datos o los típicos ETL (Extracción, Transformación y Carga de Datos por sus siglas en ingles).

Cada una de estas estructuras se adaptarán a una tarea determinada en la gestión de los datos, por lo que es muy importante que podamos entender como interactúan en un proyecto típico de R.

**Vectores**

El vector es la estructura de datos fundamental en R, este se encarga de almacenar un conjunto de valores ordenados llamados elementos.

Un vector puede contener cualquier numero de elementos, sin embargo, todos los elementos deben de ser del mismo tipo de valor.

Es decir, si un vector es numérico todos los elementos deben de ser numéricos, al igual si el vector fuera texto o lógico.

En R podemos determinar el tipo de vector con el comando typeof(nombre del vector).

Creemos un nuevo documento con R, (CTRL+SHIFT+N) y guardémoslo con el nombre 001\_vectores.

Tomemos el siguiente ejemplo:

w **<-** c**(TRUE**, **FALSE**, **TRUE)**

x **<-** c**(**1**:**10**)**

y **<-** c**(**"a", "b", "c"**)**

z **<-** c**(**1**-**9**)**

typeof**(**w**)**

typeof**(**x**)**

typeof**(**y**)**

typeof**(**z**)**

Los resultados deben de ser los siguientes:

**>** typeof**(**w**)**

**[**1**]** "logical"

**>** typeof**(**x**)**

**[**1**]** "integer"

**>** typeof**(**y**)**

**[**1**]** "character"

**>** typeof**(**z**)**

**[**1**]** "double"

Como observamos tenemos un tipo de valor lógico, entero, carácter y doble.

En ninguno de los objetos w,x,y,z podemos meter tipos de datos que no sean los que corresponden a cada uno.

**Tipos de Vectores**

Varios tipos de vectores se utilizan comúnmente en el aprendizaje automático entre los cuales podemos nombrar los siguientes:

* Enteros (números sin decimales)
* Dobles (números con decimales)
* Caracteres (datos de texto)
* Lógicos (valores VERDADEROS o FALSOS).

También hay dos valores especiales: NULL, que se utiliza para indicar la ausencia de cualquier valor, y NA, que indica un valor faltante.

**Función de Combinación**

Es tedioso ingresar grandes cantidades de datos manualmente, pero se pueden crear pequeños vectores usando la función de combinación c (). También se le puede dar un nombre al vector mediante el operador de flecha <, que es la forma en que R asigna valores, al igual que el operador de asignación = se usa en muchos otros lenguajes de programación.

**Motivación**

Entre la documentación conforme se desarrolla mostraremos de a poco algunas pequeñas cosas que podemos ir haciendo ya en un preproceso de análisis de datos.

En esta ocasión crearemos un pequeño vector con números del 1 al 10 y extraeremos 5 datos, es decir un 50% de las observaciones del objeto y que las muestras no se repitan.

Para ello escribimos el siguiente código:

x **<-** c**(**1**:**10**)**

w **<-** sample**(**x, 5, replace **=** **FALSE**, prob **=** **NULL)**

w

En el objeto w obtenemos una muestra aleatoria de datos del vector x.

**Ejemplo Practico**

Crearemos 3 vectores de esta forma:

subject\_name **<-** c**(**"John Doe", "Jane Doe", "Steve Graves"**)**

temperature **<-** c**(**98.1, 98.6, 101.4**)**

flu\_status **<-** c**(FALSE**, **FALSE**, **TRUE)**

Podemos utilizar el comando typeof() con el nombre de cada uno de los vectores para ver su tipo.

**Orden Inherente**

Los vectores de R están ordenados de forma inherente es decir que es esencial y permanente en un ser o en una cosa o no se puede separar de él por formar parte de su naturaleza y no depender de algo externo.

Por ello se puede acceder a los registros contando por el numero de elementos en el conjunto, comenzando por el uno. Para ello rodeamos el numero entre corchetes [] después del nombre del vector.

Para obtener la temperatura del segundo dato del vector de “temperature” tendríamos que escribir el siguiente código:

temperature**[**2**]**

**Métodos**

R ofrece una variedad de métodos convenientes para extraer datos de vectores. Podemos obtener un rango de valores o guardarlos utilizando el operador de dos puntos (:). Veamos el código de cómo obtener la temperatura del vector 2 y 3.

temperature**[**2**:**3**]**

**Exclusiones**

Se puede excluir información especificando la posición con numero negativo, por ejemplo, excluyamos y guardemos en otro objeto la temperatura con posición 2.

temperature**[-**2**]**

temperatura2 **<-** temperature**[-**2**]**

temperatura2

**Vector Lógico**

Finalmente, a veces también es útil especificar un vector lógico que indique si se debe incluir cada elemento. Por ejemplo, para incluir las dos primeras lecturas de temperatura, pero excluir la tercera, escriba:

temperature**[**c**(TRUE**, **TRUE**, **FALSE)]**

**Finalizando**

Como verá en breve, el vector proporciona la base para muchas otras estructuras de datos R. Por lo tanto, el conocimiento de las diversas operaciones vectoriales es crucial para trabajar con datos en R.

**Practica**

**Vamos a hacer unos ejercicios para poder practicar y entender mejor lo que es un vector.**

Introducción: Un vector es la forma más elemental de almacenar y estructurar datos en R. Por ahora, piense en él como una lista de números, que puede ser tan corta como un solo número, o tan larga como aproximadamente 2 mil millones de números.

**Ejercicio 1**

Considere el siguiente vector:

* x <- c(4,6,5,7,10,9,4,15)

Cuál es el valor de:

c(4,6,5,7,10,9,4,15) < 7

a. TRUE, FALSE, TRUE, FALSE, FALSE, FALSE, TRUE, FALSE

b. TRUE, TRUE, TRUE, FALSE, FALSE, FALSE, TRUE, FALSE

c. FALSE, TRUE, TRUE, FALSE, FALSE, FALSE, TRUE, FALSE

d. TRUE, TRUE, TRUE, TRUE, TRUE, FALSE, TRUE, FALSE

e. TRUE, TRUE, TRUE, FALSE, FALSE, FALSE, TRUE, FALSE

**Ejercicio 2.**

Considere 2 vectores “p” y “q”:

p <- c (3, 5, 6, 8)

q <- c (3, 3, 3)

¿Cuál es el valor de p+q?

a. 6, 8, 6, 8

b. 6, 8, 0, 0

c. 6, 8, NA, NA

d. 3, 5, 6, 8

e. 6, 8, 9, 11 Warning message: In p+q : longer object length is not a multiple of shorter object length

**Ejercicio 3.**

Si:

Age <- c(22, 25, 18, 20)

Name <- c("James", "Mathew", "Olivia", "Stella")

Gender <- c("M", "M", "F", "F")

Cuál es el código que usaría en R para la siguiente salida:

## Age Name Gender

## 1 22 James M

## 2 25 Mathew M

**a.**DataFrame <- data.frame(c(Age), c(Name), c(Gender))

subset(DataFrame, Gender == "M")

**b.**DataFrame <- data.frame(c(Age),c(Name),c(Gender))

subset(Gender=="M"), eval=FALSE

**c.**DataFrame <- data.frame(Age,Name,Gender)

subset(DataFrame,Gender=="M")

**d.**DataFrame <- data.frame(c(Age,Name,Gender))

subset(DataFrame,Gender=="M")

**Ejercicio 4.**

Si

z <- 0:9

entonces, ¿cuál es el resultado de las siguientes declaraciones R:

digits <- as.character(z)

as.integer(digits)

**a.** Error in subset. object 'z' not found

**b.** 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

**c.** "NA", "NA", "NA", "NA", "NA", "NA", "NA", "NA", "NA"

**d.** "0", "1", "2", "3", "4", "5", "6", "7", "8", "9"

**e.** 0, 0, 0, 0, 0, 0, 0, 0, 0

**Ejercicio 5.**

Considere el vector:

x <- c(1,2,3,4)

Cuál es el valor de K para:

(x+2)[(!is.na(x)) & x > 0] -> k

k

**a.** 1, 2, 3, 4

**b.** 1, 4, 9, 16

**c.** Error: object 'k' not found

**d.** 3, 4, 5, 6

**e.** numeric(0)

**Ejercicio 6.**

Considere el data set AirPassenger incluido en R por defecto.

data(AirPassengers)

¿Cual declaración producirá la siguiente salida?

## [1] 112 118 132 129 121 135 148 148 136 119 104 118

**a.** AirPassengers[time(AirPassengers) >= 1949 & time(AirPassengers) < 1950, 12]

**b.**AirPassengers[AirPassengers >= 1949 & AirPassengers < 1950]

**c.**AirPassengers[time(AirPassengers) >= 1949 & time(AirPassengers) < 1950]

**d.**AirPassengers[AirPassengers >= 1949 & AirPassengers < 1950, 12]

**Ejercicio 7.**

Si

x <- c(2, 4, 6, 8)

y además

y <- c(TRUE, TRUE, FALSE, TRUE)

¿Cuál es el valor del siguiente enunciado?

sum(x[y])

**a.** 20

**b.** 8

**c.** 14

**d.** NA

**Ejercicio No. 8**

Considere el siguiente vector:

x <- c(34, 56, 55, 87, NA, 4, 77, NA, 21, NA, 39)

¿Qué enunciado R contará el número de valores de NA en x?

count(is.na(X))

length(is.na(x))

sum(is.na(x))

count(!is.na(x))

sum(!is.na(x))