Data Science

ESTRUCTURAS DE DATOS CON RSTUDIO

www.datascience.pe

Data Science

Las estructuras de datos son objetos que contienen datos.
Cuando trabajamos con R, lo que estamos haciendo es manipular estas estructuras.

Las estructuras tienen diferentes características. Entre ellas, las que distinguen a una estructura de otra es el número de dimensiones y si son homogeneas o hereterogeneas.

La siguiente tabla muestra las principales estructuras de control que encontrarás en R.

Dimensiones	Homogeneas	Heterogeneas
1	Vector	Lista
2	Matriz	Data frame
n	Array	

Adaptado de Wickham (2016).

https://bookdown.org/jboscomendoza/r-principiantes4/estructuras-de-datos.html

Data Science

CÓMO CREAR VECTORES CON RSTUDIO

Data Science

En esta sección abordaremos los siguientes tópico con relación a las Vectores en R usando Rstudio:

- ❑ Crear vectores en un script en R,
- ❑ Seleccionar elementos de un vector.
- ❑ Realizar operaciones aritméticas con vectores utilizando Rstudio.

Usaremos vectores para almacenar múltiples datos.

Data Science

¿QUÉ ES UN VECTOR EN R?*

En Matemática un vector es un ente matemático que posee modulo, dirección y sentido.
Un vector en R es un conjunto de objetos del mismo tipo concatenados con la función c, donde los términos del vector son conocidos como componentes , se puede crear vectores con componentes lógicos, de caracteres, numéricos, entre otros.

- Variable

- Vector

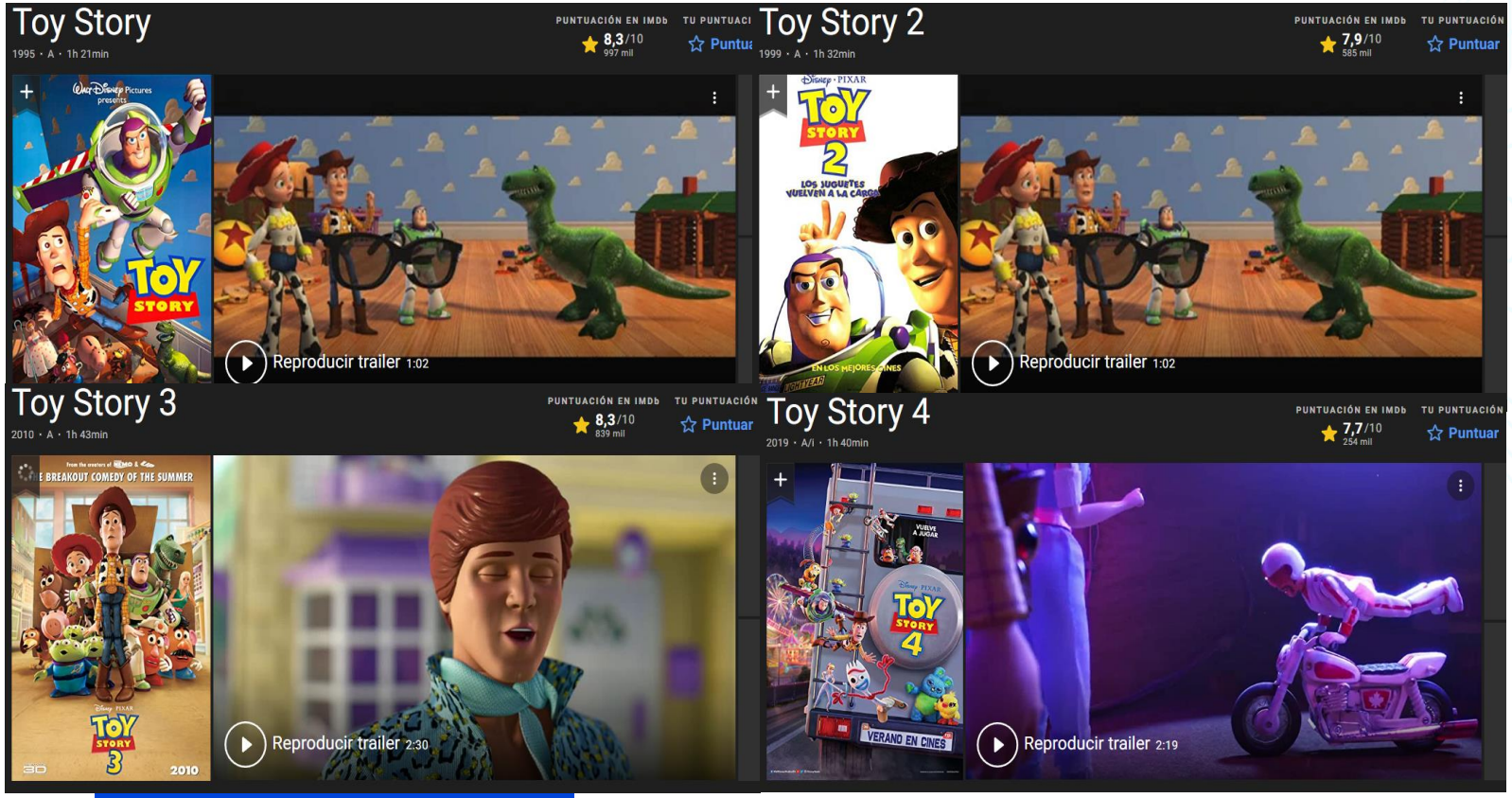
En R un vector es un espacio de memoria en la computadora que se usa para guardar información de cualquier cosa.

Es importante destacar que puedes verificar la clase de un vector con la función class y el tipo de elementos del mismo con la función typeof.

Data Science

EJEMPLO DE VECTOR EN R?

Tenemos esta saga de películas animadas, Tenemos que guardar sus nombres, la puntuación que le da el internet y el año de lanzamiento sea posterior al año 2000.



Data Science

Primero vamos a crear un vector para cada una de estas características o atributos de las películas .
Para esto, utilizamos variables numéricas, carácter y tipo lógico.

Vector 1	Vector 2	Vector 3
Nombre	**Puntuacion**	**¿Es posterior a 2000?**
Toy Story 1	8.3	FALSE
Toy Story 2	7.9	FALSE
Toy Story 3	8.3	TRUE
Toy Story 4	7.7	TRUE

Los vectores pueden representar las columnas de una tabla de datos. En este caso, las columnas serían el nombre, la puntuación y si es una película posterior a 2000. Es importante que, en R, cada vector debe tener un solo un tipo de dato.

Data Science

¿CÓMO CREAR VECTORES EN R?*

Podemos crear un vector de manera similar a como creó una variable en la sección anterior, pero la forma mas sencilla de crear un vector en R es usando la función c() que se utiliza para la concatenación de objetos que deseamos almacenar en el vector. Lo más común, es que los vectores sean de tipo numérico, carácter o lógico.

Entonces vamos a ver el ejemplo de las películas; podemos escribir sus nombres, puntuación y que sea posterior al año 2000, en un vector llamado "name", "punctuation" y "posterior_2000" respectivamente.

```
# crear vector carácter con nombre de las películas
name <- c("Toy story1", "Toy story 2: los juguetes vuelven a la carga", "Toy story 3", "Toy story 4")
# crear vector numérico con puntuación de las películas
Puntuación <- c(8.3, 7.9, 8.3, 7.7)

# crear vector lógico sobre si la película es posterior a 2000
posterior_2000 <- c(FALSE, FALSE, TRUE, TRUE)
```

Si queremos agregar los números 1, -5, 15, 8 y 10 al vector x:

```
# Vector numérico
c(1, 2, 3, 5, 8, 13)

# Vector de cadena de texto
c("arbol", "casa", "persona")

# Vector lógico
c(TRUE, TRUE, FALSE, FALSE, TRUE)
```

Data Science

ELEMENTOS VECTORIALES

Nos referimos a cada pieza de datos dentro de un vector como un elemento. Es importante tener en cuenta que todos los datos deben ser del mismo tipo dentro de un vector. Si no lo son, RStudio convertirá automáticamente los datos en el mismo tipo, el que sea más flexible.
Por ejemplo, si tengo una mezcla de números, lógicos y caracteres dentro de mi vector.

```
x <- c(2, -1, 3, "cafés", "chocolates", TRUE)
print(x)
```

RStudio convertirá todos los elementos en caracteres porque el carácter es el tipo de datos más flexible.
Si solo tengo números y lógicos, RStudio convertirá los lógicos en numéricos porque lo contrario no siempre es posible. Por ejemplo, VERDADERO siempre se convertirá en 1 y también es posible lo contrario; sin embargo, 2 no se convertirá en VERDADERO. Aquí, los datos numéricos se convierten en el tipo de datos más flexible porque todos los datos lógicos se pueden convertir en datos numéricos.

```
x <- c(2, -1, TRUE)
print(x)
```

Data Science

SELECCIÓN DE ELEMENTOS VECTORIALES

Para seleccionar e imprimir un elemento de vector en particular, puede usar corchetes []. Por ejemplo, si quiero seleccionar e imprimir el primer elemento en el vector, puedo usar:

```
x <- c(2, -1, 3, "cafés", "chocolates", TRUE)
print(x[1])
```

Para imprimir un rango de elementos vectoriales, especifique el primer elemento que desea seleccionar seguido de dos puntos: y, por último, especifique el último elemento que desea seleccionar. RStudio luego seleccionará todos los elementos entre e incluyendo el primer y el último elemento. Por ejemplo:

```
x <- c(2, -1, 3, "cafés", "chocolates", TRUE)
print(x[1:4])
print(x[c(1,4)])
```

Data Science

+ EJEMPLO DE SELECCIÓN DE VECTORES EN R

En algunos casos, estamos interesados en seleccionar elementos específicos de un vector. Así, podemos seleccionar elementos basados en la posición del elemento, como la tercera, o la primera y la última. Por otro lado, también podemos seleccionar basados en una condición como los que sean mayores a 7.

Digamos que queremos seleccionar el nombre de la tercera película. Luego, queremos escoger la primera y la cuarta película. Por último, seleccionamos todos los nombres menos el de la segunda película.

```
# Seleccionar la 2da pelicula
name[2]
#seleccionar la segunda y la última
name[c(2,4)]

#Seleccionamos todos excepto la 2da
name[-2]
```


```
x <- c(2, -5, "apple", "banana", TRUE)
x[c(1,4)]
```

Ahora vamos a crear una condición para las películas que tengan una puntuación menor a 7.5. Las condiciones en este caso van a ser vectores tipo lógico. Luego, seleccionamos las puntuaciones bajas. Por último, pedimos los nombres que tengan puntuaciones bajas.

```
# Creamos una condicion logica puntuacionBaja=
puntuacion<7.5

#Mostramos la condicion para ver TRUE o FALSO
PuntuacionBaja
#Mostramos nombre de las peliculas con puntuación baja
name[puntuacionBaja]
```

Data Science

¿CÓMO OPERAR VECTORES EN R?

Podemos operar vectores contra valores numéricos o contra otros vectores. Las operaciones aritméticas básicas son la suma, la multiplicación, la resta y la división.
En R, supongamos que quiero sumarle dos puntos a todas las películas porque me gusta mucho la saga. Pero digamos que otra persona quiere ponerle la mitad. El código para esto sería:

```
# Sumar 2 a la puntuacion
puntuacion +2
#Dividir la puntuacion entre 2
puntuacion/2
```

Por otro lado, si genero mi propia puntuación en un vector, podría ver la diferencia de lo que pienso contra lo que dice internet con una resta de vectores. Como ejercicio puedes hacer tu propia puntuación para hacer pruebas.

```
# Crear puntuacion de los
asistentes al bootcamp
puntuacion1 =c()
#Calcularemos las diferencias entre
estas puntuaciones
puntuacion1 - puntuacion
```

Data Science

USO DE FUNCIONES
ESTADÍSTICAS EN VECTORES

Las funciones son extremadamente importantes en R porque ayudan a calcular cierta información rápidamente. Comenzaremos con la función mean().

Mean

La función mean() calcula la media o el promedio de los datos proporcionados. El promedio se calcula sumando todos los datos y luego dividiéndolos por el número de datos utilizados.

```
x <- c(2, -2, 30, 4, 8, 12)
print(mean(x))
```


```
y <- c(12, 80, -10, -76)
```

Data Science

USO DE FUNCIONES *
ESTADÍSTICAS EN VECTORES

Median

```
x <- c(2, -2, 30, 4, 8, 12)
print(median(x))
```

```
y <- c(12, 80, -10, -76)
```

Data Science

Mode

A diferencia de la media y la mediana, no existe una función de modo que nos ayude a calcular los datos que aparecen más dentro de un conjunto de datos o vector. Sin embargo, podemos construir nuestra propia función get_mode() usando la siguiente sintaxis:

```
get_mode <- function(f) {
  uf <- unique(f)
  tab <- tabulate(match(f, uf))
  uf[tab == max(tab)]
}
```

Notarás que la función de palabra clave se usa en el código anterior, lo que nos ayuda a crear la función get_mode(). Dentro de las llaves {} hay líneas de código que contienen comandos para indicarle al programa qué hacer cuando se llama a la función get_mode(). Por ahora, se le proporcionarán funciones, así que no se preocupe por crearlas. Sin embargo, si tiene curiosidad acerca de cómo crear funciones R, puede intentar e investigar.
Ahora que tiene la función get_mode(), puede usarla para encontrar el modo dentro de su vector x. Asegúrese de continuar incluyendo las instrucciones para la función get_mode() dentro de su código, de lo contrario, no funcionará.

Data Science

```
get_mode <- function(f) {
  uf <- unique(f)
  tab <- tabulate(match(f, uf))
  uf[tab == max(tab)]
}

x <- c(2, -5, 45, 13, -5)
print(get_mode(x))
```

Lo cual tiene sentido porque -5 ocurre más dentro del vector.
Si agrego datos adicionales al vector, como 13 y 45, la función ahora devolverá múltiples elementos porque tengo múltiples modos dentro de mis datos.

```
get_mode <- function(f) {
  uf <- unique(f)
  tab <- tabulate(match(f, uf))
  uf[tab == max(tab)]
}

x <- c(2, -5, 45, 13, -5, 13, 45)
print(get_mode(x))
```

A diferencia de mean() y mediana(), podemos usar get_mode() en datos de caracteres. Por ejemplo,

```
get_mode <- function(f) {
  uf <- unique(f)
  tab <- tabulate(match(f, uf))
  uf[tab == max(tab)]
}

y <- c("cat", "dog", "cat", "pig",
"pig")
print(get_mode(y))
```

Maximum

Para encontrar el máximo de datos dentro de un vector, podemos usar la función max().

```
x <- c(2, -5, 45, 13, -5, 13, 45)
print(max(x))
```

Data Science

Minimum

Para encontrar los datos mínimos dentro de un vector, podemos usar la función min().

```
x <- c(2, -5, 45, 13, -5, 13, 45)
print(min(x))
```

Range

Para encontrar el rango, o los valores de datos más grandes y más pequeños dentro de un vector, podemos usar la función range(). Devolverá el valor de datos más pequeño seguido del más grande.

```
x <- c(2, -5, 45, 13, -5, 13, 45)
print(range(x))
```

porque -5 es el valor más pequeño mientras que 45 es el valor más grande en el vector.
Para calcular realmente el rango en sí, que es la diferencia entre los valores de datos más grandes y más pequeños, use max() - min().

```
x <- c(2, -5, 45, 13, -5, 13, 45)
print(max(x) - min(x))
```

Data Science

GRACIAS

www.datascience.pe