`which(), which.min(), which.max(),'%in%'`

`which()`

La función `which()` proporciona la indexación (posición en el `length()`) de los valores `TRUE` de una sentencia lógica. La sentencia lógica es lo que va dentro de los paréntesis de la función.

Por defecto `which()` devuelve un vector con datos de tipo `logical`.

Ejemplo:

Creo un vector para probar distintas funciones. El vector lo voy a crear con datos aleatorios sacando 50 datos de una muestra de valores del 1 al 1000. Para que siempre salgan los mismos datos es que planto una semilla (`set.seed()`). Esto hace que el proceso aleatorio siempre produzca el mismo resultado.

```
set.seed(2018)
mi_vector <- sample(1:1000, 50)

# imprimo en consola el vector para verlo
mi_vector
```

```
##  [1] 337 464  61 197 473 300 604 130 951 542 392 658 971 670 795 625 267
## [18] 544 725 813 255 557 149  93 744 523 320 900 582  63 138 836  33 475
## [35] 489 736  45  51 559 169 861 742 574 958 576   3 919  66 106 737
```

Consulto el largo del vector para verificar que efectivamente contenga 50 datos.

```
length(mi_vector)
```

```
## [1] 50
```

Ahora voy a verificar si el número 473 aparece en el vector llamado `mi_vector`. Esto lo voy a hacer con la operación lógica `'=='` que significa "idéntico". Y se debe leer de la siguiente manera: lo que está a la izquierda del `'=='` es idéntico a lo que esta a la derecha del `'=='`.

```
mi_vector == "473"
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [45] FALSE FALSE FALSE FALSE FALSE FALSE
```

Esto me va a devolver un vector de TRUE y FALSE de la misma longitud del vector (`mi_vector`). Lo que hace la operación es ir consultando dato por dato y verifica si cada dato es idéntico a 473.

Si quiero saber cuantas veces aparece el número 473 en el vector llamado `mi_vector` una posibilidad es sumar la operación lógica previamente hecha. Al sumar un vector lógico vamos a tener la cantidad de valores `TRUE` que hay. En este caso, un valor de `TRUE` indica que el dato de `mi_vector` es idéntico (`'=='`) al número 473.

```
sum(mi_vector == "473")

## [1] 1
```

La función `which()` proporciona la ubicación en donde la operación lógica que se haga tiene por resultado un `TRUE`. En nuestro caso, nos va a dar la ubicación del valor 473 en el vector llamado `mi_vector`.

```
which(mi_vector == "473")

## [1] 5
```

Como se puede ver, la función `which()` devuelve un vector numérico de longitud igual a la cantidad de valores `TRUE` producto de la operación lógica hecha.

Hagamos una prueba, voy a cambiar el valor que está en la posición 30 de `mi_vector` por el número 473.

```
mi_vector[30]           # imprimo el valor de la posición 30

## [1] 63

mi_vector[30] <- 473    # modifico el valor de la posición 30 por 473

mi_vector[30]           # consulto si lo modificó

## [1] 473
```

Volvemos a usar la función `which()` sobre el vector que ahora está modificado y tiene dos valores idénticos a 473.

```
which(mi_vector == "473")

## [1]  5 30
```

Ahora vamos a usar la la función `which()` para modificar todos los valores del vector llamado `mi_vector` que sean idénticos a 473 por el valor `'50'`

```
mi_vector[which(mi_vector == "473")] <- 50
mi_vector

##  [1] 337 464  61 197  50 300 604 130 951 542 392 658 971 670 795 625 267
## [18] 544 725 813 255 557 149  93 744 523 320 900 582  50 138 836  33 475
## [35] 489 736  45  51 559 169 861 742 574 958 576   3 919  66 106 737
```

which.min()

La función `which.min()` devuelve la posición en la que se ubica el valor máximo.

```
min(mi_vector)

## [1] 3

which.min(mi_vector)

## [1] 46

mi_vector[which.min(mi_vector)]              # esto es igual que poner mi_vector[46]

## [1] 3

mi_vector[which(mi_vector == min(mi_vector))]

## [1] 3
```

¿Las dos lineas de código que están a continuación hacen lo mismo?:

```
mi_vector[which.min(mi_vector)]

## [1] 3

mi_vector[which(mi_vector == min(mi_vector))]

## [1] 3
```

La respuesta es, conceptualmente NO, pero posiblemente SI en alguna circunstancia como en el ejemplo. Veamos un ejemplo paso a paso de lo que hace cada una de las operaciones:

mi_vector[which(mi_vector == min(mi_vector))]

PASO 1. Consulta lógica

```
mi_vector == min(mi_vector)

##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [45] FALSE  TRUE FALSE FALSE FALSE FALSE
```

PASO 2. Ubicación donde la consulta lógica es TRUE

```
which(mi_vector == min(mi_vector))
```

```
## [1] 46
```

PASO 3. Indexación '[ ]'

```
mi_vector[which(mi_vector == min(mi_vector))]
```

```
## [1] 3
```

PASO 4. Modificación

```
mi_vector[which(mi_vector == min(mi_vector))] <- 1000
```

PASO 5. Consulto operación realizada

```
mi_vector[46]
```

```
## [1] 1000
```

⇒ Distinción importante!

La función `which.min()` ofrece el mismo resultado que `which(x == min(x)` siempre que el valor mínimo no se repita. El valor mínimo de un vector es un solo número con independencia de si ese número se repite en el vector. Veamos un ejemplo:

```
mi_vector2 <- c(3, 1, 5, 3, 1, 8)      # creo un vector con un valor mínimo repetido

min(mi_vector2)                        # consulto el valor mínimo

## [1] 1

which.min(mi_vector2)                  # posición del valor mínimo

## [1] 2

which(mi_vector2 == min(mi_vector2))   # acá aparece la diferencia!

## [1] 2 5

mi_vector2[which(mi_vector2 == min(mi_vector2))]

## [1] 1 1
```

Esto es de este modo ya que la función `which.min()` devuelve la primera (o única) posición en donde se ubica el valor mínimo. Cuando se indexa de este modo:

```
mi_vector2[which(mi_vector2 == min(mi_vector2))]
```

...le estamos pidiendo a R que haga otra operación. Le estamos pidiendo que nos diga la ubicación de los valores del vector en los que el vector es igual al valor que devuelve la función `min()`, que es el valor mínimo.

La función `which.max()` es igual a `which.min()` solo que con el valor máximo.

%in%

El operador `%in%` es similar a la función `match()`[1] Este operador devuelve un vector lógico que indica si el o los valores de un vector se encuentran en otro vector.

Veamos un ejemplo, creamos dos vectores de distinto tamaño (uno de 7 y otro de 10 valores):

```
set.seed(201)
vector1 <- sample(1:10, 7, replace = TRUE)
set.seed(202)
vector2 <- sample(1:10, 10, replace = TRUE)

vector1

## [1] 7 2 4 1 7 2 1

vector2

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

Ahora vamos a usar el operador `%in%` para consultar que valores del `vector1` aparecen en el `vector2`:

```
vector1 %in% vector2

## [1]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
```

Este operador nos devuelve un vector lógico de igual longitud del vector sobre el que queremos saber algo (en este ejemplo el `vector1`).

[1] Consulten la ayuda en R: `?match`.

Esto significa que no es lo mismo hacer:

```
vector1 %in% vector2

## [1]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE

vector2 %in% vector1

##  [1]  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE
```

Funciones complementarias de %in%

`match`	Devuelve un vector numérico que indica la posición de cada valor del `vector1` en el `vector2`. Si no existe un `match` de valores la respuesta es `NA`.
`intersect`	Devuelve los valores que dos vectores tienen en común.
`setdiff`	Devuelve los valores que aparecen en el `vector1` pero no aparecen en el `vector2`.

```
vector1

## [1] 7 2 4 1 7 2 1

vector2

##  [1] 2 7 4 3 4 8 2 3 5 7

match(vector1, vector2)

## [1]  2  1  3 NA  2  1 NA

intersect(vector1, vector2)

## [1] 7 2 4

setdiff(vector1, vector2)

## [1] 1

setdiff(vector2, vector1)

## [1] 3 8 5
```