

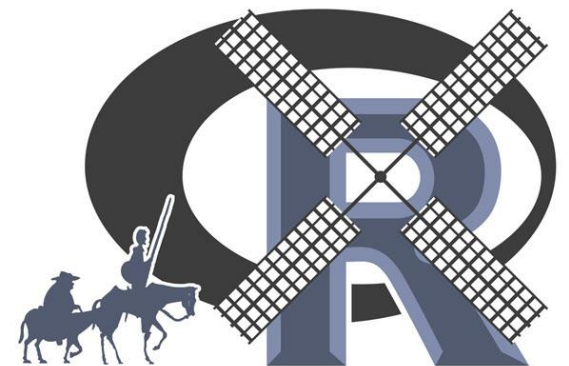
Curso: Computación y Programación Básica

Billy Ernst & Diego Narvaez

Departamento de Oceanografía

Cabina 10 / Edificio Oceanografía (2do Piso)

biernst@udec.cl / diegonarvaez@udec.cl



Fono: 4012 / 1028



Contenido

- Al final de esta sesión Ud debiera reconocer los tipos de variables disponibles en R
- Podrá construir series de datos con distintos procedimientos.
- Realizar operaciones básicas en R



Tipos de datos

R tiene cinco clases básicas de variables:

- **Character** (texto o alfa numérico) **Auto, Z2n1**
- **Numeric** (numérico, número reales) **3.14**
- **Integer** (entero) **1, 100**
- **Logical** (lógico, verdadero-falso) **TRUE, FALSE**
- **Complex** (complejos) **3+2i**

Números en R

- Números en R por defecto son reales
- Si queremos números enteros hay que especificarlo explícitamente usando el carácter **L**.

```
v<-1
```

```
class(v)
```

[1]"numeric" El resultado será un vector de clase "numeric"

```
>v1<-1L ; v2<-1:12L
```

```
class(v1), class(v2),
```

[1]"integer" El resultado será un vector de clase "integer"

En el caso de v1 es vector de largo 1 y v2 uno de largo 12, pero ambos de clase enteros.

Vector

El objeto más básico de R es el vector

- Un vector es un “contenedor” que puede almacenar 1 o más datos, pero de una misma clase.
- Ejemplos

[1 55 21 -100] Vector de números enteros con 4 elementos

[peso largo alto ancho espesor materialidad] Vector de caracter con 6 elementos

No es vector de enteros en R

[1 peso 21 -100]

Crear un vector sin asignarle valores

- Se puede utilizar la función *vector* para generar un vector para almacenar luego elementos de esa clase.
- Por ejemplo si nuestro vector debe ser de número reales, utilizaremos el argumento “mode” especificando la clase, es este caso “numeric”.

```
> mi.vector<-vector(mode = "numeric", length = 10)
```

```
> class(mi.vector)
```

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

```
> mi.vector
```

```
[1] 0 0 0 0 0 0 0 0 0 0
```

Pruebe Uds mismo en R.



Creando una secuencia de valores a:b

>La forma más fácil de crear una secuencia de valores enteros que se incrementan en 1 es utilizando

a:b, donde **a** y **b** pueden ser positivos o negativos y **a** menor o mayor a **b**.

1:5

```
[1] 1 2 3 4 5
```

5:-1

```
[1] 5 4 3 2 1 0 -1
```

2

Creando una secuencia de valores `seq()`

Una segunda forma de crear una secuencia de valores enteros o reales que se incrementan en un valor especificado es utilizando la función **seq**(from = a, to = b, by=c)

```
> seq(from = 1, to = 5, by=1)
```

```
[1] 1 2 3 4 5
```

```
> seq(from = 2000, to = -2000, by=-500)
```

```
[1]
```


3

Creando un conjunto de datos en un vector

- `c()` -- concatenar

```
> Carbon <- c(8, 54, 534, 1630, 6611)
```

```
[1] 8 54 534 1630 6611
```

Usar operador de asignación en vez de Signo =

```
Carbon2 <- c("casa", "edificio", 101)
```

```
class(Carbon2)
```

```
[1] "character"
```

```
[1] "casa" "edificio" "101"
```

Fijese que al combinar texto con números el vector resultante es de clase character, dado que el texto no puede ser convertido en número, pero los número pueden ser considerados caracteres alfanuméricos.

```
Carbon3 <- c(Carbon, Carbon2, -5:1)
```

```
[1] "8" "54" "534" "1630" "6611" "casa" "edificio" "101"  
[9] "-5" "-4" "-3" "-2" "-1" "0" "1"
```

Interpretemos este resultado

```
[1] "8" "54" "534" "1630" "6611" "casa" "edificio" "101"  
[9] "-5" "-4" "-3" "-2" "-1" "0" "1"
```

Qué significa el [1] y [9] en la consola de resultados?

El resultado de utilizar la función de concatenación es un vector carácter de largo 15, es decir tiene 15 elementos. Entonces al desplegarlos [1] indica que el “8” es el primer elemento del vector y [9] indica que “-5” es el noveno.

4

Creando un vector con elementos repetidos `rep()`

```
rep(a, times=b)
```

a= número entero o real, vector,
character

```
[1] 61 61 61 61 61 61 61 61 61 61 0)
```

Combinando lo aprendido

```
rep (c(1,2,3), times = 3)
```

```
[1] 1 2 3 1 2 3 1 2 3
```

```
rep (seq(from=-1,to=2.5,by=0.5), times = 3)
```

```
[1] -1.0 -0.5 0.0 0.5 1.0 1.5 2.0 2.5 -1.0 -0.5 0.0 0.5 1.0 1.5 2.0 2.5 -1.0 -0.5 0.0  
[20] 0.5 1.0 1.5 2.0 2.5
```

```
rep (c("Juan","Arno","Karen",1:2), times = 3)
```

```
[1] "Juan" "Arno" "Karen" "1" "2" "Juan" "Arno" "Karen" "1" "2" "Juan" "Arno" "Karen" "1" "2"
```

INDEXANDO VECTORES

- A menudo queremos seleccionar algunos elementos de un vector, llamaremos indexación a ese proceso.
- Para ello utilizamos paréntesis cuadrados

[]

- Supongamos que fabricamos un vector llamado **datos** con 12 elementos:

```
datos<-21:12
```

	Comando	Resultado
[1] 21 20 19 18 17 16 15 14 13 12	datos[3]	19
[1] 21 20 19 18 17 16 15 14 13 12	datos[2:9]	20 19 18 17 16 15 14 13

Indexando Vectores

```
> Carbon <- c(8, 54, 534, 1630, 6611)
```


```
> Carbon
```

```
[1] 8 54 534 1630 6611
```

```
> Carbon[5]
```

```
[1] 6611
```

**Obtener el 5to
elemento de
Carbon**



Números (índices) negativos implican
que esos elementos serán excluidos

```
Carbon[-2]
```

```
[1] 8 534 1630 6611
```

Indexando Vectores

```
> Carbon <- c(8, 54, 534, 1630, 6611)
```

```
> Carbon
```

```
[1] 8 54 534 1630 6611
```

```
> Carbon[c(2, 5)]
```

```
[1] 54 6611
```

**Extraer los
elementos**

2^{do} y 5^{to}

Carbon

Indexando Vectores

```
> Carbon
```

```
[1] 8 54 534 1630 6611
```

```
➤ Carbon[c(3, 4)]
```

```
[1] 534 6611
```

**3^{er} y 4^{to} elemento
del vector**

Indexando Vectores

```
> Carbon
```

```
[1] 8 54 534 1630 6611
```

```
> Carbon[2:4]
```

```
[1] 54 534 1630
```

**Del segundo
elemento al
cuarto**

Operadores lógicos

Los operadores lógicos nos servirán para indexar de acuerdo a criterios más complejos, como todos los números mayores que X. O de acuerdo a dos criterios a la vez o el uno o el otro.

== (igual—doble igual)

!= (no igual)

> (mayor que)

< (menor que)

>= (mayor o igual que)

<= (menor o igual que)

& (y)

| (o)

Indexando Vectores

```
> Carbon <- c(8, 54, 534, 1630, 6611)
```

```
> Carbon
```

```
[1] 8 54 534 1630 6611
```

```
> Carbon[Carbon > 1000]
```

**Extraer los
elementos de
Carbon que
son > 1000**

```
[1] 1630 6611
```

En este caso se pregunta por 2 criterios simultáneamente

```
> Carbon
```

```
[1] 8 54 534 1630 6611
```

```
➤ bool <- (Carbon>1000) & (Carbon<5000)
```

➤ Se buscan datos que sean mayor a 1000 y al mismo tiempo menor a 5000.

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

```
> Carbon[bool]
```

```
[1] 1630
```

Pero como funciona realmente la selección con operadores logicos?

```
> Carbon[Carbon > 1000]
```

```
> Carbon > 1000
```

Mayor que

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

Los resultados de los operadores lógicos son TRUE o FALSE (verdadero o falso). Al usar luego este vector en la indexación, se verán solo aquellos elementos que son TRUE

Indexando vectores

```
> Carbon[c(F,F,F,T,T)]
```

```
[1] 1630 6611
```

**Los elementos
pueden ser
extraídos usando
TRUE & False—
debe usar c()**

Diferencia entre [] y ()

```
> Carbon [ (Carbon>1000) | (Carbon<5000) ]  
[1]      8    54   534  1630  6611
```

[] – paréntesis cuadrado se usa para indexar objetos—rescatando valores específicos

() – paréntesis redondos son usados por funciones u operadores