Laboratorio Nacional de Políticas Públicas

CIDE
CENTRO DE INVESTIGACIÓN
Y DOCENCIA ECONÓMICAS A.C.

ESTADÍSTICA

LABORATORIO II
Objetos y Funciones

MAESTRÍA EN PERIODISMO DE POLÍTICAS PÚBLICAS.

JORGE JUVENAL CAMPOS FERREIRA.

Objetos.

1. Qué es un objeto?

En R, un objeto es un lugar en la memoria de la computadora que almacena un valor, y al cuál se le asigna un nombre. Por ejemplo:

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

Puede leerse como: “Creamos un objeto, llamado x, conteniendo los valores 1, 2 y 3”

Objetos.

2. Tipos primarios.

En R, como ya se comentó en clase existen varios tipos de cuatro tipos de vectores atómicos:

- Tipo `logical`, que almacena información del tipo `TRUE` o `FALSE`
- Tipo `integer` y tipo `numeric`, que almacenan información numérica.
- Tipo `character`, que almacena información de texto.

Objetos.

E igualmente existen otras clases de objetos, como las siguientes:

- Objeto `dataframe` o `tibble`, para almacenar bases de datos en forma de tabla.
- Objeto `xts`, para almacenar Series de tiempo.
- Objeto `sf`, para almacenar información geográfica,

etc.

Objetos.

Todos estos objetos tienen elementos comunes, como por ejemplo:

1. Todos deben tener un *nombre* que los identifique.
2. Una *clase* de objeto que les brinde atributos y características.
3. *Funciones* compatibles con ese tipo de objeto.

Objetos.

Declarando un objeto.

Para declarar un objeto, seguimos la estructura básica siguiente:

```
nombre_objeto <- contenido_del_objeto
```

Nombres de Objetos.

Nota sobre los nombres

Prohibiciones

1. Estan prohibidos los espacios.
2. Esta prohibido llamar a un objeto `TRUE` o `FALSE` o `if` o `else` o `for` (palabras reservadas).
3. Esta prohibido usar sólo números en el nombre.
4. Esta prohibido empezar un nombre con guion bajo `_` o con un número (aunque si lo pueden llevar después).
5. Pero si son tercos y no quieren respetar las prohibiciones, pueden envolver el nombre en `` ` `` (backticks).

Ejemplo:

```
`_12345Soy una variable rebelde y con un nombre extremadamente largo e innecesario`
<- c(1,2,3,4,5)
```

Nombres de Objetos.

Sugerencias

- Utiliza nombres cortos y útiles, que te den información de calidad sobre el contenido de la variable.

```
edad <- c(55L, 22L, 33L, 45L)
```

Nombres de Objetos.

- De preferencia usa minúsculas. Si tu variable tiene un nombre mas o menos complejo, utiliza la forma de llamar a las variables como `camelCase`. Igualmente, puedes usar guiones bajos o puntos para separar nombres, pero no es tan recomendable.

```
iPhone <- c(TRUE, TRUE, FALSE, FALSE)
edadViejitos <- c(100, 99, 77, 200)
nombresAlumnos <- c("Natalia", "Luis David", "Oscar Eder", "Luis David", "Pablo",
"Julio")
nombres_laboratoristas <- c("Lila", "Pablo", "Juvenal")
nombres.profesores <- c("Sebastian", "Salvador")
```

Nombres de Objetos.

- Trata de usar simbolos sin acentos, o eñes, en la medida de lo posible. Igualmente, trata de usar las letras del alfabeto inglés.

```
# Ejemplo, se pueden escribir nombres de variables en kanji, pero no es muy practi
co.
青春 <- "Juve en Japonés"
print(青春)
```

```
## [1] "Juve en Japonés"
```

Funciones

Qué es una función?

Una función es un grupo de instrucciones que toma un "input" o datos de entrada, usa estos datos para computar otros valores y retorna un resultado/producto.

Fuente: https://www.r-bloggers.com/lang/spanish/2512

Funciones

Partes de una función.

```
nombre_funcion(argumento1, argumento2, argumento3... argumentoN)
```

Generalmente, estas devuelven un resultado.

Funciones

a. Función c()

La función `c()` es básica en R. La función de la función consiste en combinar todos sus argumentos para formar un vector. Ejemplo.

```
# Nombre de la
# funcion
combinación <-  c(1,2,3,TRUE, "Hola", factor("a", "b", "c"), c("VectorDentroDe Vec
tor"), rep(1:10, 10), sum(1, 2))
```

Y si no me acuerdo de los argumentos de una función?

En este caso, utilizamos la ayuda o documentación de R.

Ejemplo, para acceder a la ayuda/documentación de la función `c()`, escribimos `?c`.

Y así para cada función.

Funciones – Documentación.

c {base} R Documentation

Combine Values into a Vector or List

Description

This is a generic function which combines its arguments.

The default method combines its arguments to form a vector. All arguments are coerced to a common type which is the type of the returned value, and all attributes except names are removed.

Usage

```
## S3 Generic function
c(...)

## Default S3 method:
c(..., recursive = FALSE, use.names = TRUE)
```

Funciones exclusivas por objetos

Los numeros tienen funciones exclusivas, como las siguientes.

```
numeros <- 1:100
sum(numeros)    # Sumamos un conjunto de numeros
```

```
## [1] 5050
```

```
sd(numeros)    # Obtenemos la desviacion estandar de un vector numerico
```

```
## [1] 29.01149
```

```
mean(numeros)    # Obtenemos la media de un vector numerico
```

```
## [1] 50.5
```

Funciones exclusivas por objetos

```
sqrt(numeros)   # Obtenemos la raiz cuadrada de cada uno de los numeros de un vector.
```

```
##   [1]  1.000000  1.414214  1.732051  2.000000  2.236068  2.449490  2.645751
##   [8]  2.828427  3.000000  3.162278  3.316625  3.464102  3.605551  3.741657
##  [15]  3.872983  4.000000  4.123106  4.242641  4.358899  4.472136  4.582576
##  [22]  4.690416  4.795832  4.898979  5.000000  5.099020  5.196152  5.291503
##  [29]  5.385165  5.477226  5.567764  5.656854  5.744563  5.830952  5.916080
##  [36]  6.000000  6.082763  6.164414  6.244998  6.324555  6.403124  6.480741
##  [43]  6.557439  6.633250  6.708204  6.782330  6.855655  6.928203  7.000000
##  [50]  7.071068  7.141428  7.211103  7.280110  7.348469  7.416198  7.483315
##  [57]  7.549834  7.615773  7.681146  7.745967  7.810250  7.874008  7.937254
##  [64]  8.000000  8.062258  8.124038  8.185353  8.246211  8.306624  8.366600
##  [71]  8.426150  8.485281  8.544004  8.602325  8.660254  8.717798  8.774964
##  [78]  8.831761  8.888194  8.944272  9.000000  9.055385  9.110434  9.165151
##  [85]  9.219544  9.273618  9.327379  9.380832  9.433981  9.486833  9.539392
##  [92]  9.591663  9.643651  9.695360  9.746794  9.797959  9.848858  9.899495
##  [99]  9.949874 10.000000
```

Funciones exclusivas por objetos

Y asi para cada tipo de objeto. Por ejemplo, los *factores* tienen la función `levels()`, para poder acceder a sus categorías asociadas, los *character* tienen la función `paste()` para poder pegar cadenas de texto en una sola. Las fechas tienen funciones para realizar operaciones de fechas, y así, para cada uno de los tipos de objetos.

Repaso.

b. Funciones para crear vectores, matrices y tibbles. (Vistas en clase, repasadas en la tarea).

Repaso.

Ejercicio 1.

Explique la diferencia entre las siguientes instrucciones.

```
# Instruccion 1
c(1,2,3)
```

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

```
# Instruccion 2
a <- c(1,2,3)
```

Repaso.

Ejercicio 2.

Explique como imprimir los valores del siguiente vector

```
printMe <- c("Imprimeme!!", "Imprimeme!!", "Imprimeme!!")
```

Igualmente, explique que significa el `[1]` que aparece del lado izquierdo de la impresión de la consola.

Repaso.

Ejercicio 3.

- Explique dos formas de obtener un vector que tenga los siguientes valores.

1 2 3 4 5 6 7 8 9 11 12 13 14 15 16 17 18 19 21 22 23 24 25 26 27 28 29

Repaso.

Ejercicio 4.

- Explique como construir un *vector* que repita 50 veces la frase: “Mi nombre es:”
- Construya *dos vectores con cincuenta nombres propios* (repetidos o no, pero al menos mas de 2), y genere una matriz con el vector construido en el ejercicio anterior.
- A dicha matriz, *añádale un numero de lista* unico.
- Genere un `dataframe` con dicha información (use la funcion `data.frame` o transforme la matriz con la funcion `as.data.frame()`)
- Genere una `tibble` con dicha información.

Repaso.

Respuesta.

```r
# 1. * Explique como construir un vector que repita 50 veces la frase: "Mi nombre es: "
frase <- rep("Mi nombre es: ", times = 50)
frase
```

```
##  [1] "Mi nombre es: " "Mi nombre es: " "Mi nombre es: " "Mi nombre es: "
##  [5] "Mi nombre es: " "Mi nombre es: " "Mi nombre es: " "Mi nombre es: "
##  [9] "Mi nombre es: " "Mi nombre es: " "Mi nombre es: " "Mi nombre es: "
## [13] "Mi nombre es: " "Mi nombre es: " "Mi nombre es: " "Mi nombre es: "
## [17] "Mi nombre es: " "Mi nombre es: " "Mi nombre es: " "Mi nombre es: "
## [21] "Mi nombre es: " "Mi nombre es: " "Mi nombre es: " "Mi nombre es: "
## [25] "Mi nombre es: " "Mi nombre es: " "Mi nombre es: " "Mi nombre es: "
## [29] "Mi nombre es: " "Mi nombre es: " "Mi nombre es: " "Mi nombre es: "
## [33] "Mi nombre es: " "Mi nombre es: " "Mi nombre es: " "Mi nombre es: "
## [37] "Mi nombre es: " "Mi nombre es: " "Mi nombre es: " "Mi nombre es: "
## [41] "Mi nombre es: " "Mi nombre es: " "Mi nombre es: " "Mi nombre es: "
## [45] "Mi nombre es: " "Mi nombre es: " "Mi nombre es: " "Mi nombre es: "
## [49] "Mi nombre es: " "Mi nombre es: "
```

Repaso.

```r
# 2. * Construya dos vectores con cincuenta nombres propios (repetidos o no, pero al men
os mas de 2), y genere una matriz con el vector construido en el ejercicio anterior.
nombres <- rep(c("Melchor", "Gaspar", "Baltazar", "Olga", "Eugenia"), times = 10)
nombres <- sample(nombres) # Funcion sample()
nombres
```

```
##  [1] "Melchor"  "Olga"     "Gaspar"   "Olga"     "Gaspar"   "Gaspar"
##  [7] "Baltazar" "Eugenia"  "Melchor"  "Baltazar" "Gaspar"   "Baltazar"
## [13] "Gaspar"   "Gaspar"   "Baltazar" "Olga"     "Melchor"  "Olga"
## [19] "Melchor"  "Melchor"  "Melchor"  "Olga"     "Eugenia"  "Eugenia"
## [25] "Melchor"  "Eugenia"  "Eugenia"  "Gaspar"   "Baltazar" "Eugenia"
## [31] "Baltazar" "Baltazar" "Olga"     "Olga"     "Melchor"  "Melchor"
## [37] "Eugenia"  "Baltazar" "Baltazar" "Eugenia"  "Baltazar" "Olga"
## [43] "Olga"     "Gaspar"   "Olga"     "Eugenia"  "Gaspar"   "Gaspar"
## [49] "Eugenia"  "Melchor"
```

Repaso.

```
matriz <- cbind(frase, nombres) # La funcion genera automaticamente una matriz
matriz
```

```
##       frase             nombres
##  [1,] "Mi nombre es: " "Melchor"
##  [2,] "Mi nombre es: " "Olga"
##  [3,] "Mi nombre es: " "Gaspar"
##  [4,] "Mi nombre es: " "Olga"
##  [5,] "Mi nombre es: " "Gaspar"
##  [6,] "Mi nombre es: " "Gaspar"
##  [7,] "Mi nombre es: " "Baltazar"
##  [8,] "Mi nombre es: " "Eugenia"
##  [9,] "Mi nombre es: " "Melchor"
## [10,] "Mi nombre es: " "Baltazar"
## [11,] "Mi nombre es: " "Gaspar"
## [12,] "Mi nombre es: " "Baltazar"
## [13,] "Mi nombre es: " "Gaspar"
## [14,] "Mi nombre es: " "Gaspar"
## [15,] "Mi nombre es: " "Baltazar"
## [16,] "Mi nombre es: " "Olga"
## [17,] "Mi nombre es: " "Melchor"
## [18,] "Mi nombre es: " "Olga"
```

Repaso.

```r
# 3. * A dicha matriz, añádale un numero de lista unico.
matriz <- cbind(matriz, 1:50)
matriz
```

```
##       frase             nombres
##  [1,] "Mi nombre es: " "Melchor"  "1"
##  [2,] "Mi nombre es: " "Olga"     "2"
##  [3,] "Mi nombre es: " "Gaspar"   "3"
##  [4,] "Mi nombre es: " "Olga"     "4"
##  [5,] "Mi nombre es: " "Gaspar"   "5"
##  [6,] "Mi nombre es: " "Gaspar"   "6"
##  [7,] "Mi nombre es: " "Baltazar" "7"
##  [8,] "Mi nombre es: " "Eugenia"  "8"
##  [9,] "Mi nombre es: " "Melchor"  "9"
## [10,] "Mi nombre es: " "Baltazar" "10"
## [11,] "Mi nombre es: " "Gaspar"   "11"
## [12,] "Mi nombre es: " "Baltazar" "12"
## [13,] "Mi nombre es: " "Gaspar"   "13"
## [14,] "Mi nombre es: " "Gaspar"   "14"
## [15,] "Mi nombre es: " "Baltazar" "15"
## [16,] "Mi nombre es: " "Olga"     "16"
## [17,] "Mi nombre es: " "Melchor"  "17"
## [18,] "Mi nombre es: " "Olga"     "18"
## [19,] "Mi nombre es: " "Melchor"  "19"
## [20,] "Mi nombre es: " "Melchor"  "20"
## [21,] "Mi nombre es: " "Melchor"  "21"
## [22,] "Mi nombre es: " "Olga"     "22"
```

Repaso.

```r
# 4. * Genere un dataframe con dicha información (use la funcion data.frame o transforme
 la matriz con la funcion as.data.frame*())
baseDeDatos <- as.data.frame(matriz)
class(baseDeDatos)
```

```
## [1] "data.frame"
```

```r
baseDeDatos
```

```
##              frase  nombres V3
## 1  Mi nombre es:   Melchor  1
## 2  Mi nombre es:      Olga  2
## 3  Mi nombre es:    Gaspar  3
## 4  Mi nombre es:      Olga  4
## 5  Mi nombre es:    Gaspar  5
## 6  Mi nombre es:    Gaspar  6
## 7  Mi nombre es:  Baltazar  7
## 8  Mi nombre es:   Eugenia  8
## 9  Mi nombre es:   Melchor  9
## 10 Mi nombre es:  Baltazar 10
## 11 Mi nombre es:    Gaspar 11
## 12 Mi nombre es:  Baltazar 12
```

Repaso.

```
# 5. * Genere una tibble con dicha información.
library(tibble)
tibbla <- as_tibble(baseDeDatos)
names(tibbla) <- c("Frase", "Nombre", "No_Lista")
tibbla
```

```
## # A tibble: 50 x 3
##    Frase             Nombre   No_Lista
##    <fct>             <fct>    <fct>
##  1 "Mi nombre es: " Melchor  1
##  2 "Mi nombre es: " Olga     2
##  3 "Mi nombre es: " Gaspar   3
##  4 "Mi nombre es: " Olga     4
##  5 "Mi nombre es: " Gaspar   5
##  6 "Mi nombre es: " Gaspar   6
##  7 "Mi nombre es: " Baltazar 7
##  8 "Mi nombre es: " Eugenia  8
##  9 "Mi nombre es: " Melchor  9
## 10 "Mi nombre es: " Baltazar 10
## # … with 40 more rows
```