# Functions

*Juan Ramón Gómez Berzosa*

*23/10/2018*

## Functions

**1. Crea una función impares que dado un vector devuelva la cantidad de elementos impares que contiene.**

```
impares = function(vector){

  cantidadImpar = length( vector[ (vector %% 2) != 0] )
  cantidadImpar
}
v = c(1:20)
impares(v)
```

```
## [1] 10
```

**2. Crea una función cambio que dada una matriz de numeros enteros reemplaze todos los NA por el valor 0.**

```
cambio = function(matriz){
  matriz[is.na(matriz)] = 0
  matriz
}
matriz = matrix(c(1:4,NA,6:10), nrow = 5, ncol = 4)
cambio(matriz )
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    6    1    6
## [2,]    2    7    2    7
## [3,]    3    8    3    8
## [4,]    4    9    4    9
## [5,]    0   10    0   10
```

**3. Crea una función unir que dados dos vectores devuelva un nuevo vector con los elementos de ambos vectores sin repetidos.**

```
#método1
unir1 = function(vector1, vector2){
  union(vector1,vector2)
}

#método2
unir2 = function(vector1, vector2){
  unique(c(vector1,vector2))
}
```

```r
v1 = c(1:8)
v2 = c(5:15)

unir1(v1,v2)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

```r
unir2(v1,v2)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

**4. Crea una función vyc que dado un string devuelva una lista de dos componentes que contenga las vocales y las consonantes.**

```r
vyc = function(string){
  palabra = unlist(strsplit(string, split=""))
  lista = list(vocales = palabra[grep("[aeiouAEIOU]",palabra)],
          consonantes = palabra[-grep("[aeiouAEIOU ]",palabra)])
  lista
}
s = "HOLA mundo"
vyc(s)
```

```
## $vocales
## [1] "O" "A" "u" "o"
##
## $consonantes
## [1] "H" "L" "m" "n" "d"
```

**5. Crea una función partir que dado un vector v y dos valores x e y (siendo y opcional), retorne un vector con los valores que aparecen luego del primer x y hasta el primer y. De no indicarse el valor de y se devolveran todos los valores que aparecen luego del primer x hasta el final del vector.**

```r
## Método el cuál si no existe x devolverá NA en vez desde la primera posición del vector
partir= function(v,x,y=NA){
  #Eliminamos inicialmente los valores NA del vector para que no de problemas
  na.omit(v)
  tama = length(v)
  #Creamos un vector de posiciones con la posición inicial y final + 1
  posiciones = c(1,tama +1)
  #Buscamos x en el vector y lo almacenamos, si no lo encontramos almacenamos tama + 1
  posiciones[1] = match(x,v,nomatch = tama + 1)

  #Creamos un vector parcial desde x hasta tama + 1 para que no encuentre después
  #un valor de y antes que el valor de x
  aux = v[posiciones[1]:posiciones[2]]
  #Si hemos encontrado x, almacenamos el 1 ya que nuestro vector final irá desde la
  #primera posición de nuestro vector parcial hasta y o hasta el final
  posiciones[!(posiciones==tama+1)] = 1
  #Omitimos los valores NA que hayamos introducido al ir hasta tama + 1
  aux = as.integer(na.omit(aux))
```

```
  tama = length(aux)
  #Buscamos la posición y, en caso de no encontrarla devolvemos la última posición
  #del vector parcial para imprimir hasta allí
  posiciones[2] = match(y,aux,nomatch = tama)

  #Imprimimos desde posicion[1] hasta posicion[2]
  sol = aux[posiciones[1]:posiciones[2]]

  #Suprimimos los valores NA y devolvemos el vector
  sol = as.integer(na.omit(sol))
  sol

}
v = seq(1,21,by=3)
partir(v,7)
```

```
## [1]  7 10 13 16 19
```

```
partir(v,7,7)
```

```
## [1] 7
```

```
partir(v,7,10)
```

```
## [1]  7 10
```

```
partir(v,7,13)
```

```
## [1]  7 10 13
```

```
partir(v,4,15)
```

```
## [1]  4  7 10 13 16 19
```

```
partir(v,10,4)
```

```
## [1] 10 13 16 19
```

```
#En este caso tendrá que devolver vacío, en nuestro caso integer(0) ya que no existe 12 en el vector
partir(v,12,13)
```

```
## integer(0)
```