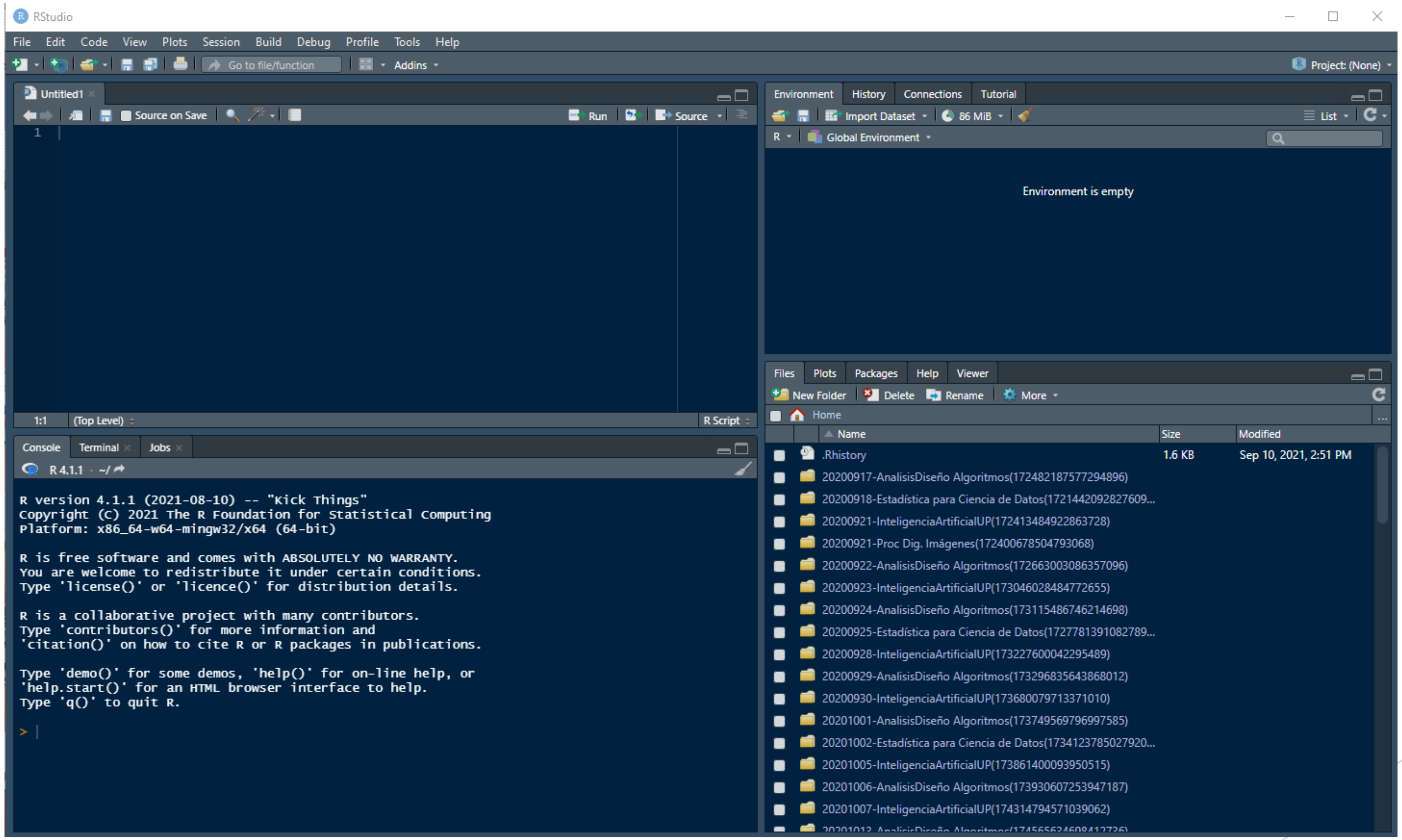
Enviroment

Environment/
variables

Script/
files

History/
list of commnads



Environment is empty

```
R version 4.1.1 (2021-08-10) -- "Kick Things"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Name	Size	Modified
.Rhistory	1.6 KB	Sep 10, 2021, 2:51 PM
20200917-AnalisisDiseño Algoritmos(172482187577294896)		
20200918-Estadística para Ciencia de Datos(1721442092827609...		
20200921-InteligenciaArtificialUP(172413484922863728)		
20200921-Proc Dig. Imágenes(172400678504793068)		
20200922-AnalisisDiseño Algoritmos(172663003086357096)		
20200923-InteligenciaArtificialUP(173046028484772655)		
20200924-AnalisisDiseño Algoritmos(173115486746214698)		
20200925-Estadística para Ciencia de Datos(1727781391082789...		
20200928-InteligenciaArtificialUP(173227600042295489)		
20200929-AnalisisDiseño Algoritmos(173296835643868012)		
20200930-InteligenciaArtificialUP(173680079713371010)		
20201001-AnalisisDiseño Algoritmos(173749569796997585)		
20201002-Estadística para Ciencia de Datos(1734123785027920...		
20201005-InteligenciaArtificialUP(173861400093950515)		
20201006-AnalisisDiseño Algoritmos(173930607253947187)		
20201007-InteligenciaArtificialUP(174314794571039062)		

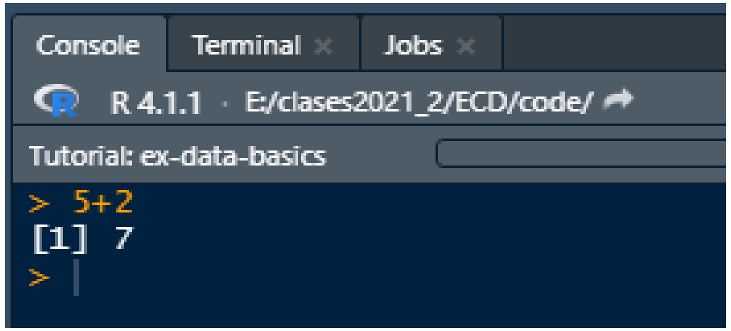
files

Console/
typing commands

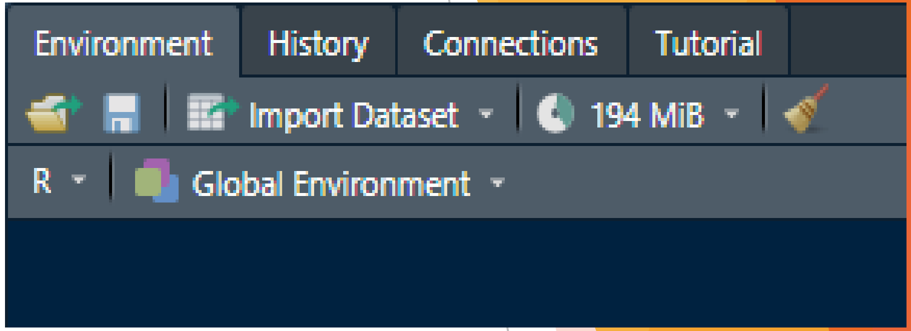
setwd("E:/clases2021_2/ECD/code")

console

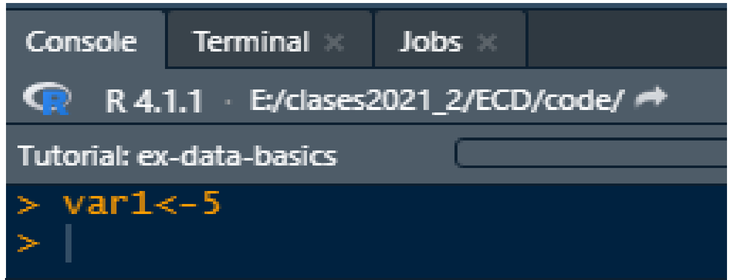
operations



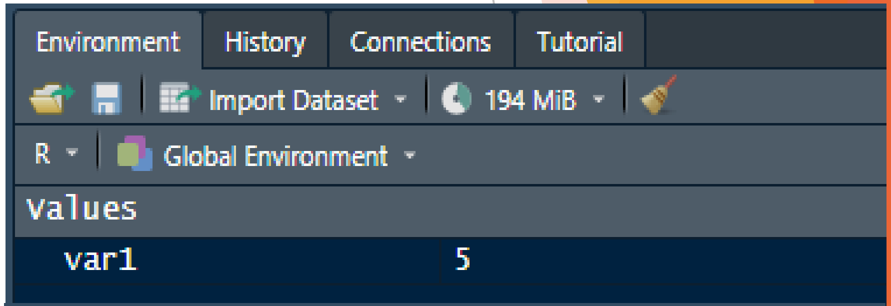
```
> 5+2
[1] 7
>
```



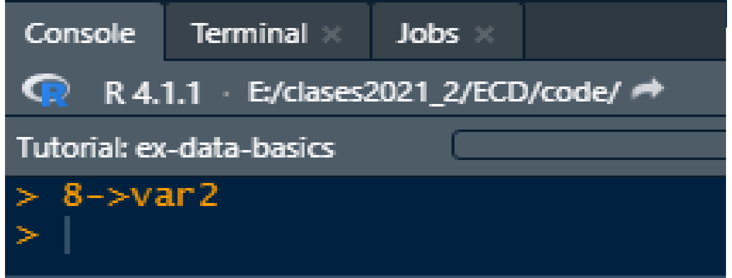
Data/objects



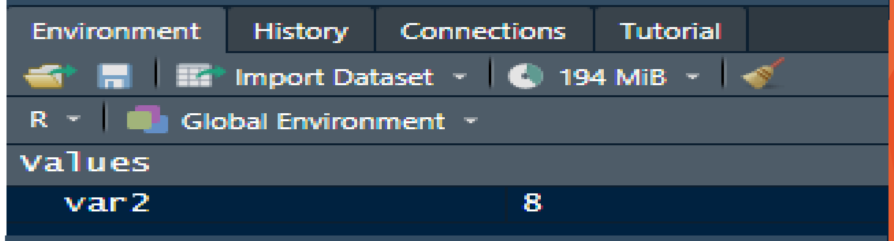
```
> var1<-5
>
```



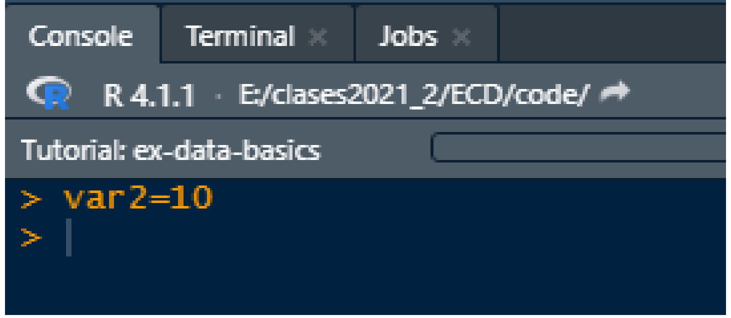
```
Values
  var1                5
```



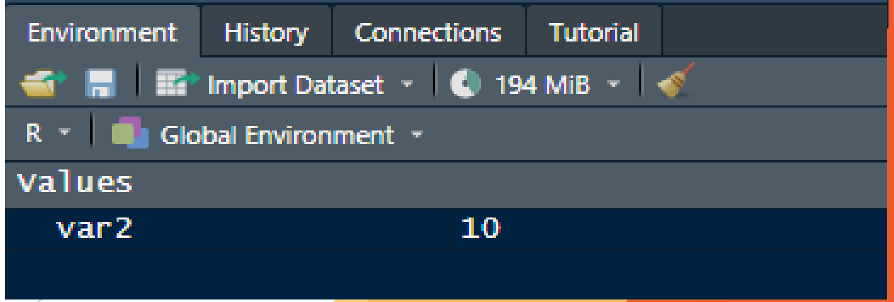
```
> 8->var2
>
```



```
Values
  var2                8
```



```
> var2=10
>
```



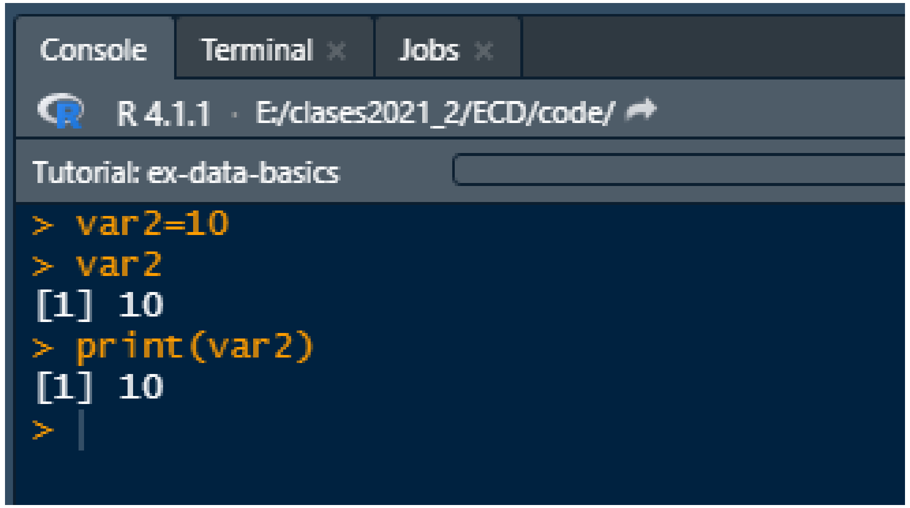
```
Values
  var2                10
```

->

<-

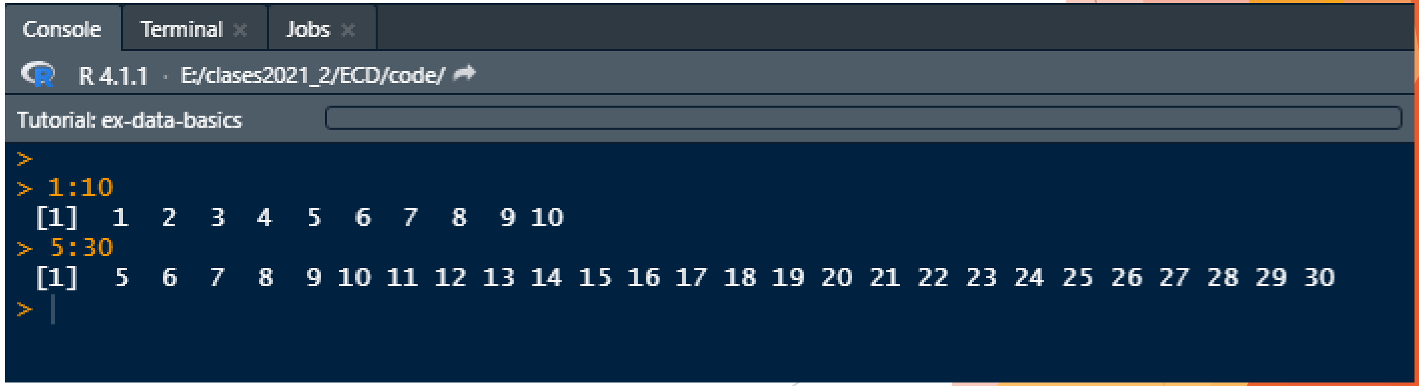
=

Print objects



```
> var2=10
> var2
[1] 10
> print(var2)
[1] 10
> 
```

Sequence of numbers



```
> 
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
> 5:30
 [1]  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
> 
```

Data in R

Basic type

- numeric
- integer
- logical
- character
- complex

Other objects

- vectors
- factors
- list
- matrix
- Data frames

numeric	1 , 3.5 , 2345.23, 12
character	“a” “hola” “casado” “nissan”
integer	1L 23L
logical	TRUE FALSE T F
complex	4+ 3i

Others objects

vectors

matrix

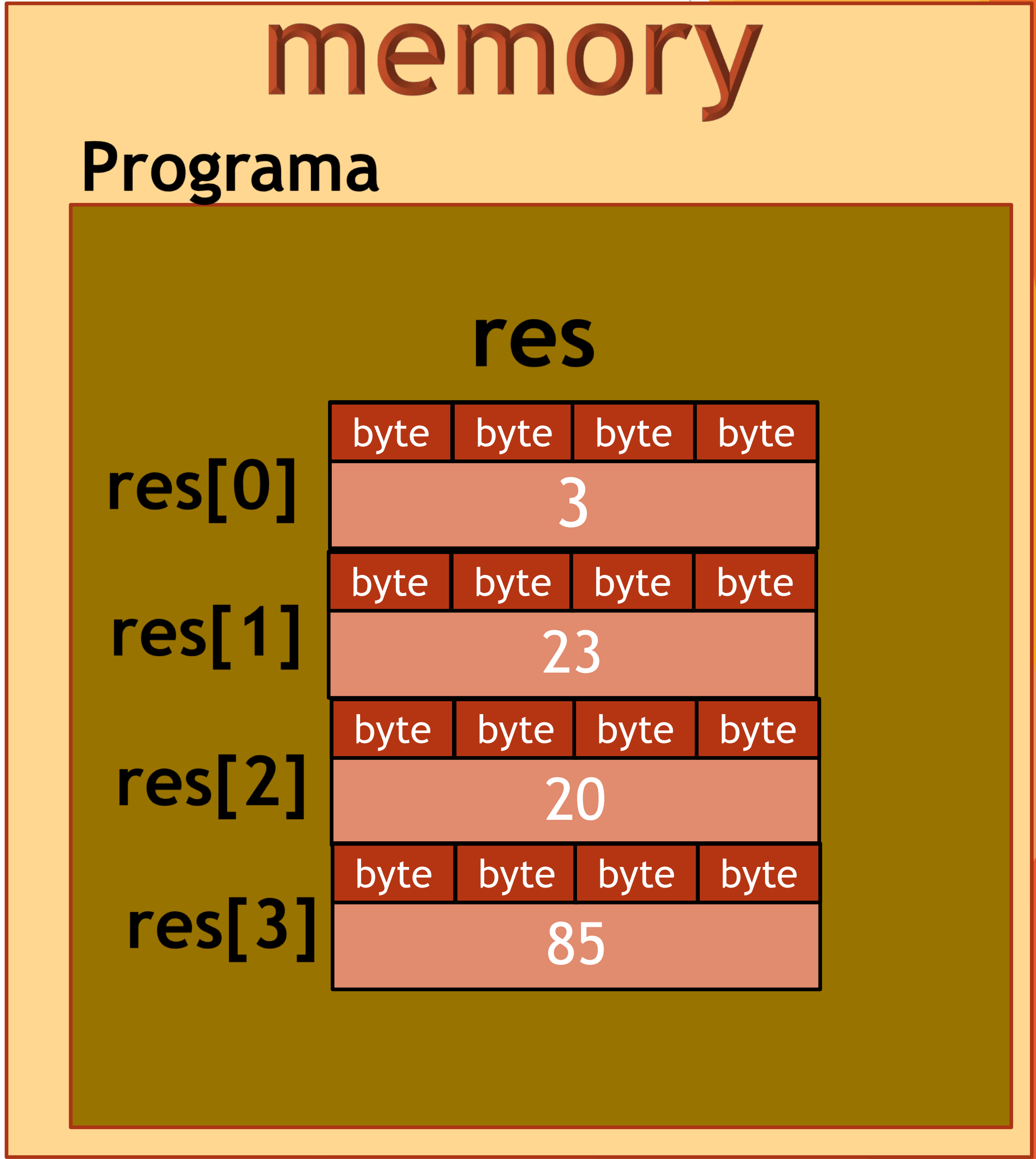
list

factors

Data frames

vectors

array



memory

Programa

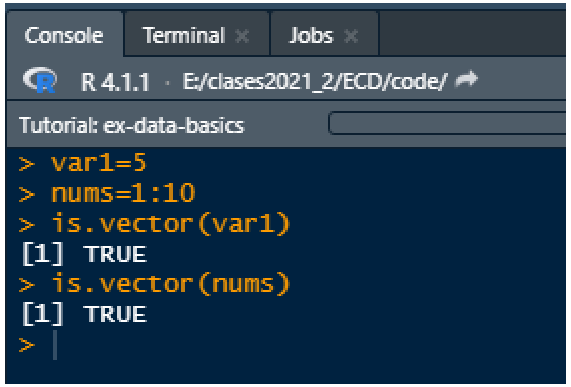
res

	byte	byte	byte	byte
res[0]	3			
	byte	byte	byte	byte
res[1]	23			
	byte	byte	byte	byte
res[2]	20			
	byte	byte	byte	byte
res[3]	85			

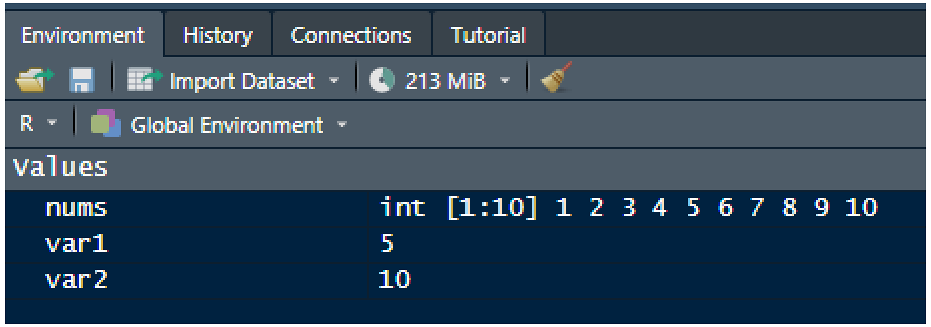
vectors

atomic vector-

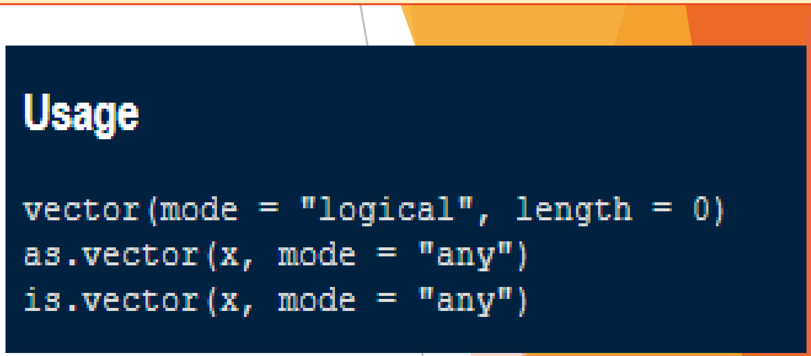
Sequence of elements of the same data type



```
> var1=5
> nums=1:10
> is.vector(var1)
[1] TRUE
> is.vector(nums)
[1] TRUE
>
```



```
Values
  nums                  int [1:10] 1 2 3 4 5 6 7 8 9 10
  var1                  5
  var2                  10
```



Usage

```
vector(mode = "logical", length = 0)
as.vector(x, mode = "any")
is.vector(x, mode = "any")
```

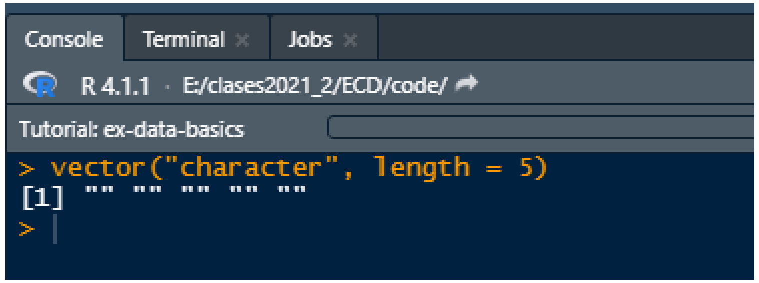
Specifying the data type

vector("character", length = 5)

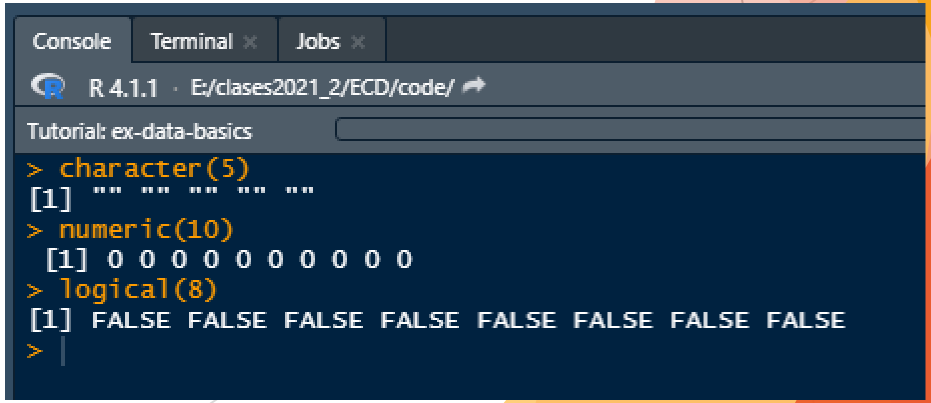
character(5)

logical(8)

numeric(10)



```
> vector("character", length = 5)
[1] "" "" "" "" ""
>
```



```
> character(5)
[1] "" "" "" "" ""
> numeric(10)
 [1] 0 0 0 0 0 0 0 0 0 0
> logical(8)
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
>
```

vectors

Specifying the content

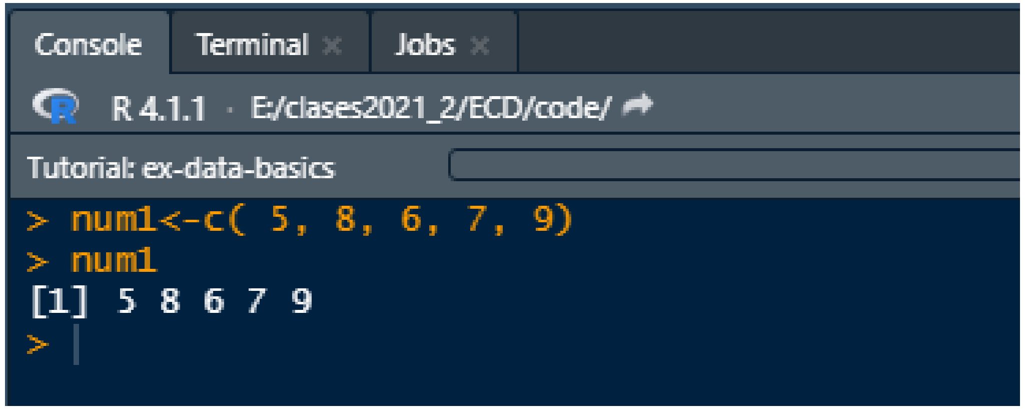
c combine

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

```
nombres <- c(“Juan", “Pedro", “luis")
```

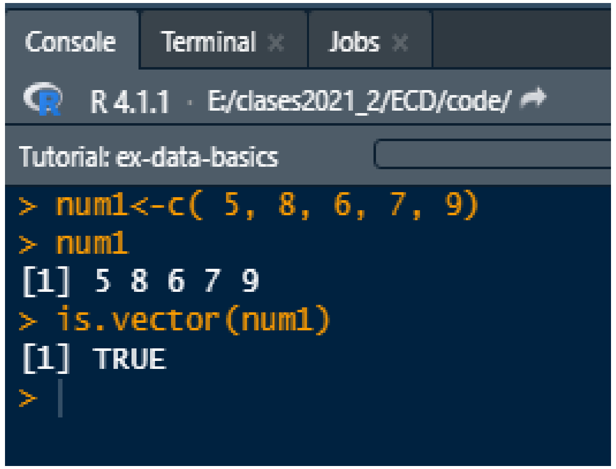
```
var1 <- c(1L, 2L, 3L)
```

Combine

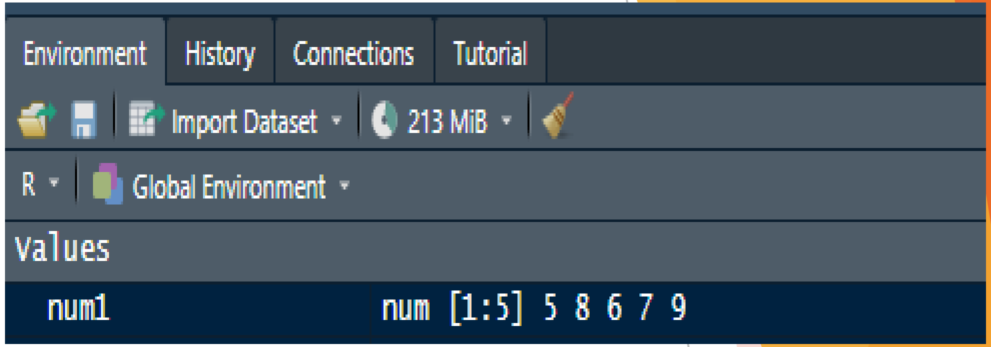


```
> num1<-c( 5, 8, 6, 7, 9)
> num1
[1] 5 8 6 7 9
> 
```

Combine values in a vector

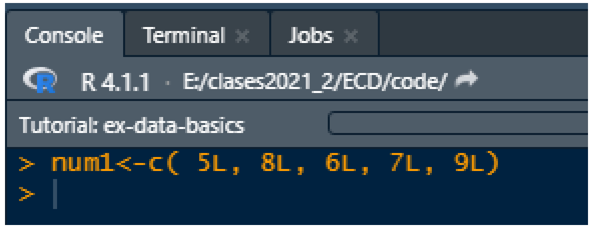


```
> num1<-c( 5, 8, 6, 7, 9)
> num1
[1] 5 8 6 7 9
> is.vector(num1)
[1] TRUE
> 
```

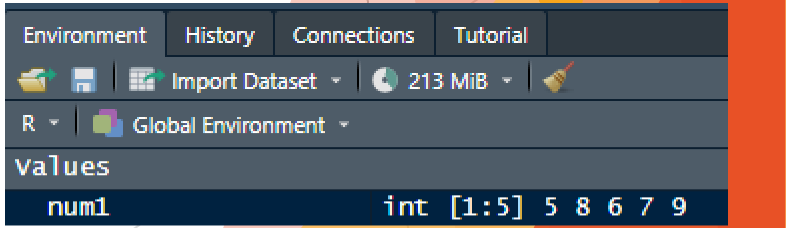


```
Values
  num1          num [1:5] 5 8 6 7 9
```

Default data type is Double

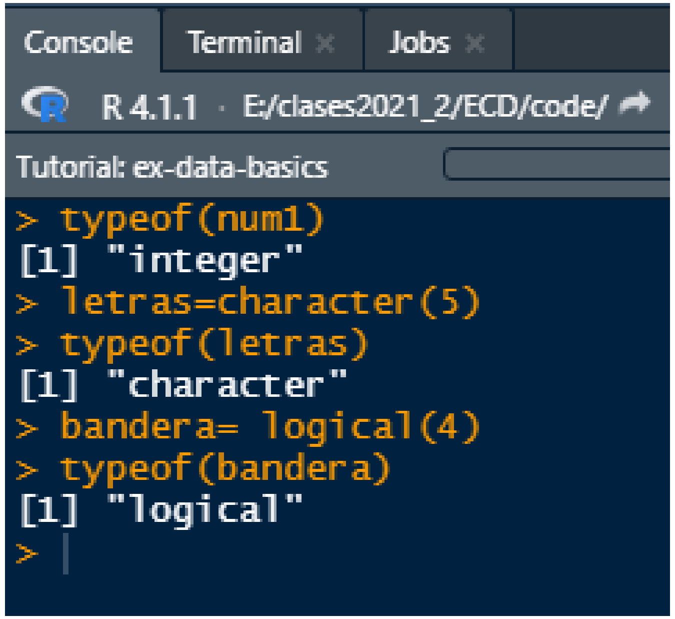


```
> num1<-c( 5L, 8L, 6L, 7L, 9L)
> 
```

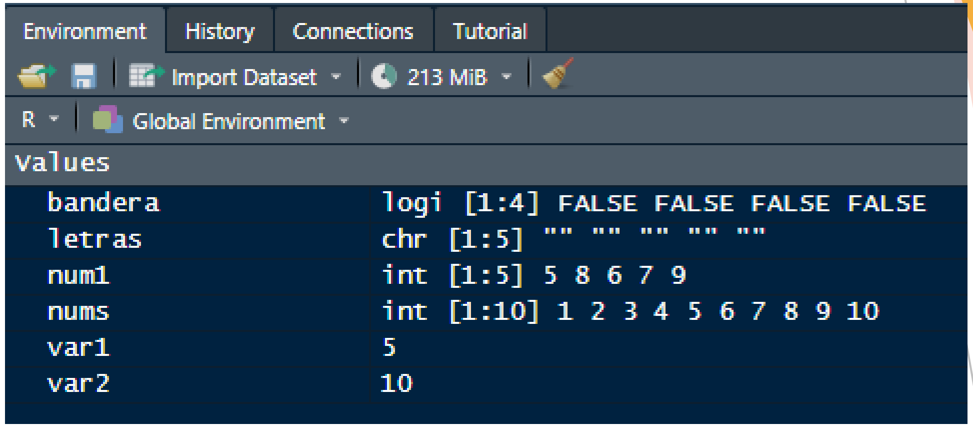


```
Values
  num1          int [1:5] 5 8 6 7 9
```

typeof()



```
> typeof(num1)
[1] "integer"
> letras=character(5)
> typeof(letras)
[1] "character"
> bandera= logical(4)
> typeof(bandera)
[1] "logical"
>
```

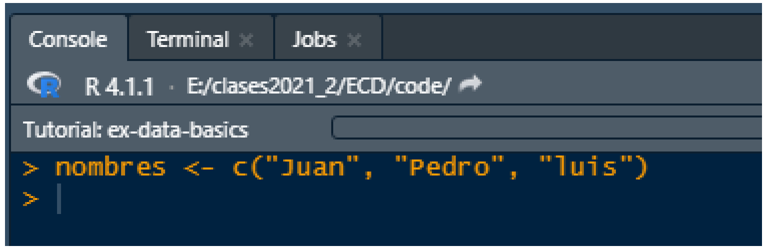


```
Values
  bandera            logi [1:4] FALSE FALSE FALSE FALSE
  letras             chr [1:5] "" "" "" "" ""
  num1               int [1:5] 5 8 6 7 9
  nums               int [1:10] 1 2 3 4 5 6 7 8 9 10
  var1               5
  var2               10
```

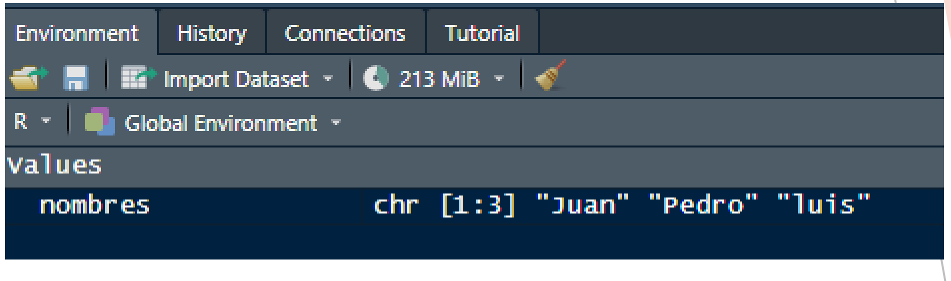
Display the data type of the object/variable

Adding element in the begin and end

nombres <- c(“Juan", “Pedro", “luis")



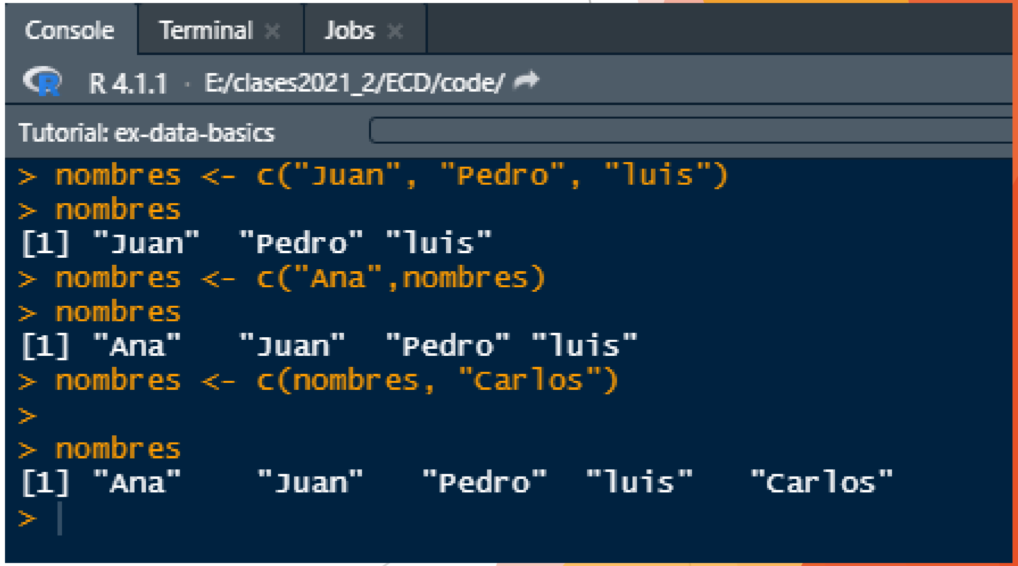
```
> nombres <- c("Juan", "Pedro", "luis")
> 
```



```
Values
  nombres            chr [1:3] "Juan" "Pedro" "luis"
```

nombres <- c(“Ana",nombres)

nombres <- c(nombres, “Carlos")

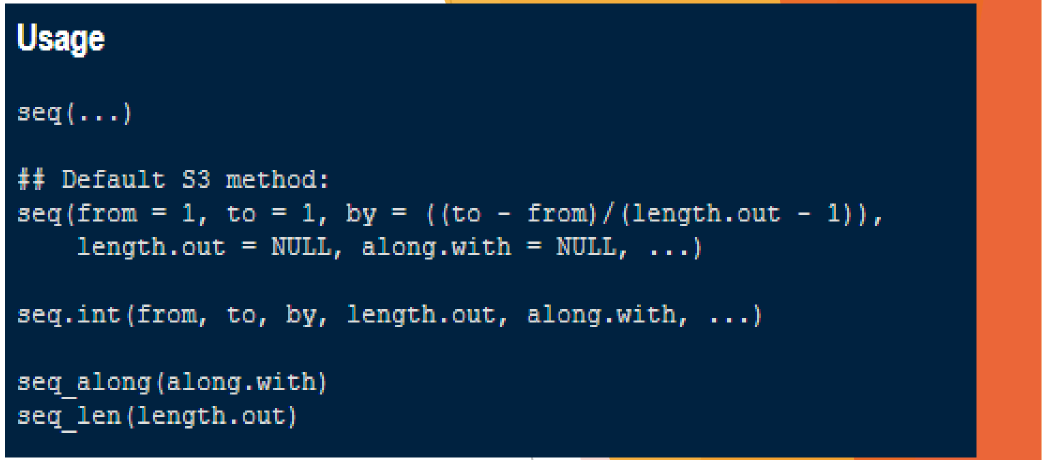


```
> nombres <- c("Juan", "Pedro", "luis")
> nombres
[1] "Juan"  "Pedro" "luis"
> nombres <- c("Ana",nombres)
> nombres
[1] "Ana"   "Juan"  "Pedro" "luis"
> nombres <- c(nombres, "Carlos")
> 
> nombres
[1] "Ana"    "Juan"   "Pedro"  "luis"   "Carlos"
> 
```

exercise

[1, 1, 0, 0, 1,1, 0,0,1,1,0,0, 1,1,0,0, 1,1,0,0] series

misnum <- 1:10



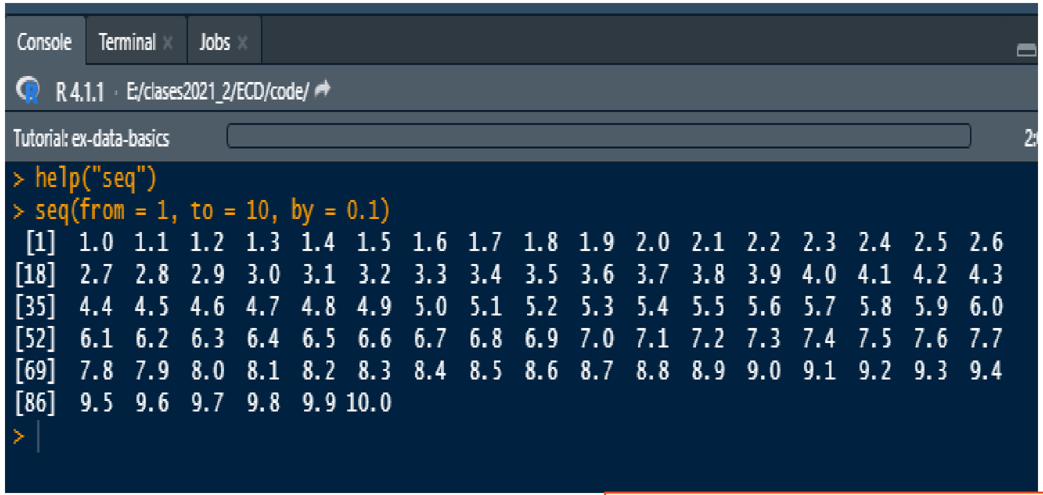
Usage

```
seq(...)

## Default S3 method:
seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),
    length.out = NULL, along.with = NULL, ...)

seq.int(from, to, by, length.out, along.with, ...)

seq_along(along.with)
seq_len(length.out)
```

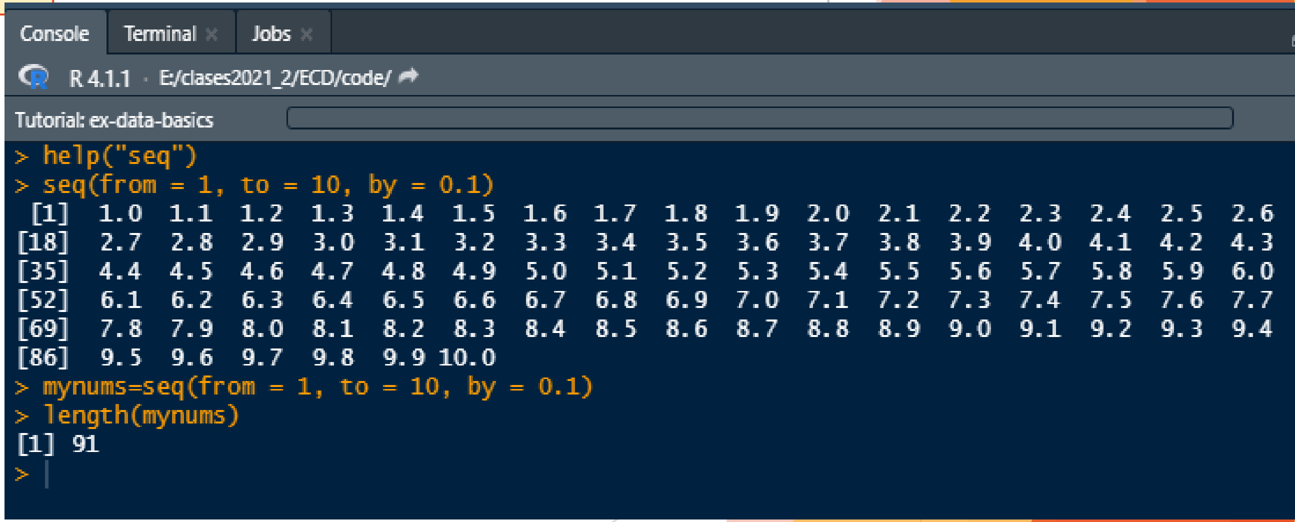
sequences

misnum <- seq(10)

seq(from = 1, to = 10, by = 0.1)



```
> help("seq")
> seq(from = 1, to = 10, by = 0.1)
 [1]  1.0  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4  2.5  2.6
[18]  2.7  2.8  2.9  3.0  3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9  4.0  4.1  4.2  4.3
[35]  4.4  4.5  4.6  4.7  4.8  4.9  5.0  5.1  5.2  5.3  5.4  5.5  5.6  5.7  5.8  5.9  6.0
[52]  6.1  6.2  6.3  6.4  6.5  6.6  6.7  6.8  6.9  7.0  7.1  7.2  7.3  7.4  7.5  7.6  7.7
[69]  7.8  7.9  8.0  8.1  8.2  8.3  8.4  8.5  8.6  8.7  8.8  8.9  9.0  9.1  9.2  9.3  9.4
[86]  9.5  9.6  9.7  9.8  9.9 10.0
> 
```

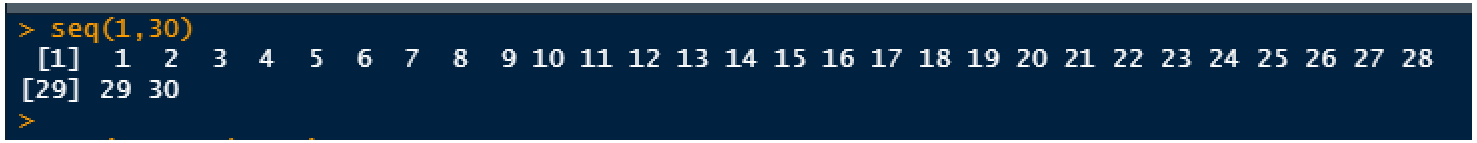


```
> help("seq")
> seq(from = 1, to = 10, by = 0.1)
 [1]  1.0  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4  2.5  2.6
[18]  2.7  2.8  2.9  3.0  3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9  4.0  4.1  4.2  4.3
[35]  4.4  4.5  4.6  4.7  4.8  4.9  5.0  5.1  5.2  5.3  5.4  5.5  5.6  5.7  5.8  5.9  6.0
[52]  6.1  6.2  6.3  6.4  6.5  6.6  6.7  6.8  6.9  7.0  7.1  7.2  7.3  7.4  7.5  7.6  7.7
[69]  7.8  7.9  8.0  8.1  8.2  8.3  8.4  8.5  8.6  8.7  8.8  8.9  9.0  9.1  9.2  9.3  9.4
[86]  9.5  9.6  9.7  9.8  9.9 10.0
> mynums=seq(from = 1, to = 10, by = 0.1)
> length(mynums)
[1] 91
> 
```

length

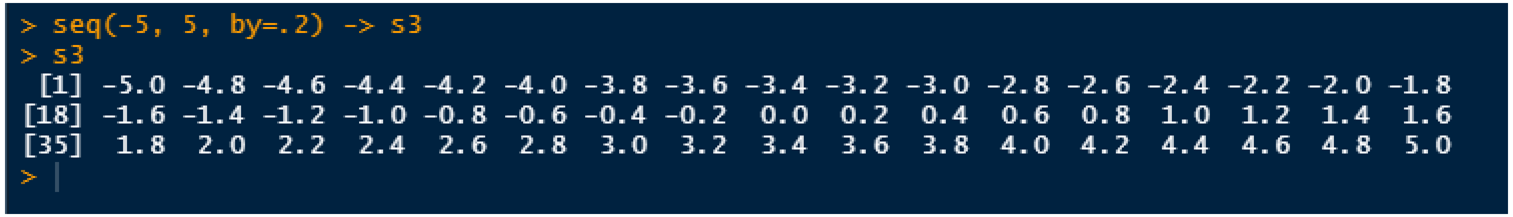
longuitud

```
seq(1,30)
```



```
> seq(1,30)
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
[29] 29 30
>
```

```
seq(-5, 5, by=.2) -> s3
```

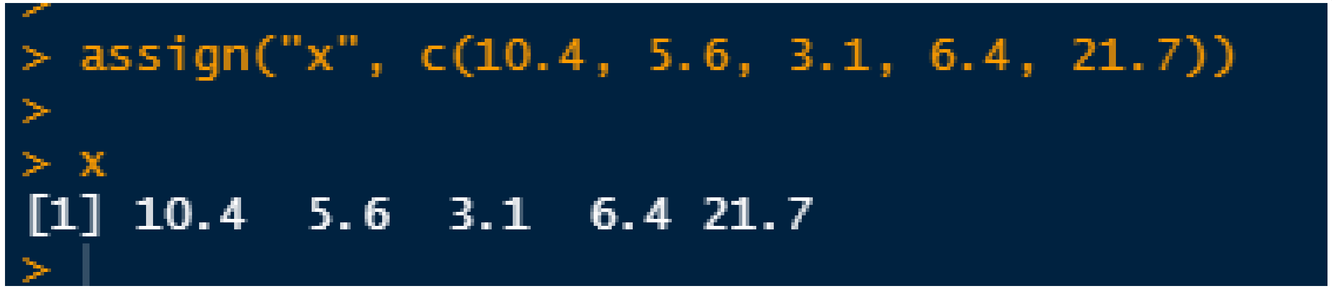


```
> seq(-5, 5, by=.2) -> s3
> s3
 [1] -5.0 -4.8 -4.6 -4.4 -4.2 -4.0 -3.8 -3.6 -3.4 -3.2 -3.0 -2.8 -2.6 -2.4 -2.2 -2.0 -1.8
[18] -1.6 -1.4 -1.2 -1.0 -0.8 -0.6 -0.4 -0.2  0.0  0.2  0.4  0.6  0.8  1.0  1.2  1.4  1.6
[35]  1.8  2.0  2.2  2.4  2.6  2.8  3.0  3.2  3.4  3.6  3.8  4.0  4.2  4.4  4.6  4.8  5.0
> |
```

```
s4 <- seq(length=51, from=-5, by=.2)
```

Secuencia con 200 elementos multiplos de 3

```
assign("x", c(10.4, 5.6, 3.1, 6.4, 21.7))
```

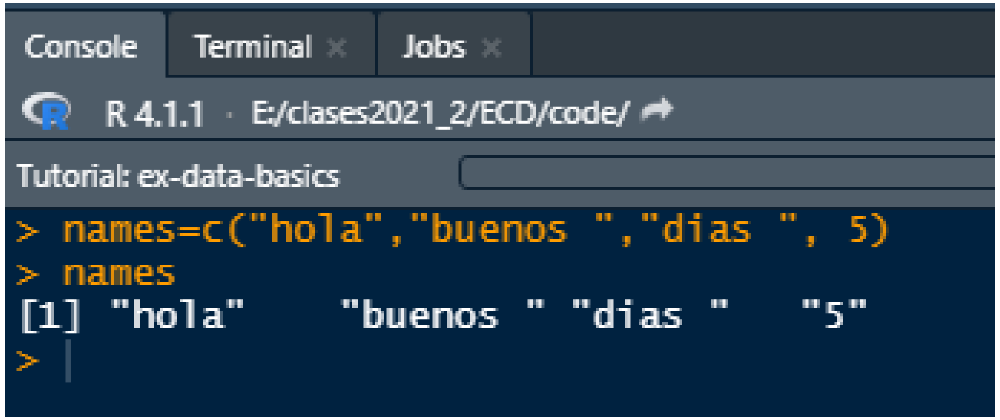


```
> assign("x", c(10.4, 5.6, 3.1, 6.4, 21.7))
>
> x
[1] 10.4  5.6  3.1  6.4 21.7
> 
```

```
c(10.4, 5.6, 3.1, 6.4, 21.7) -> x
```

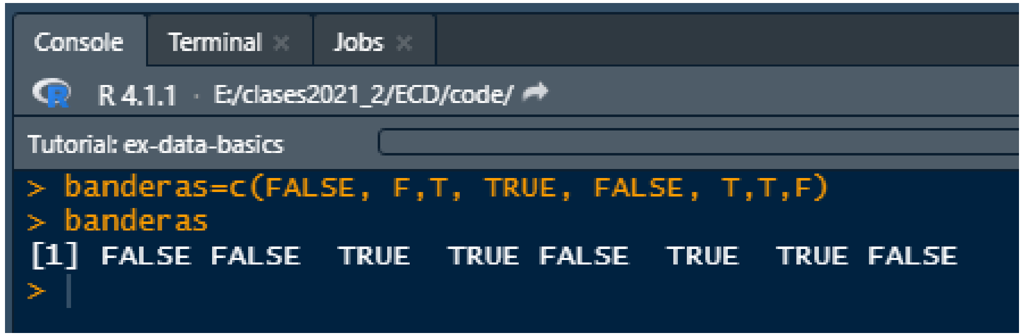
```
y <- c(x, 0, x)
```

coersion



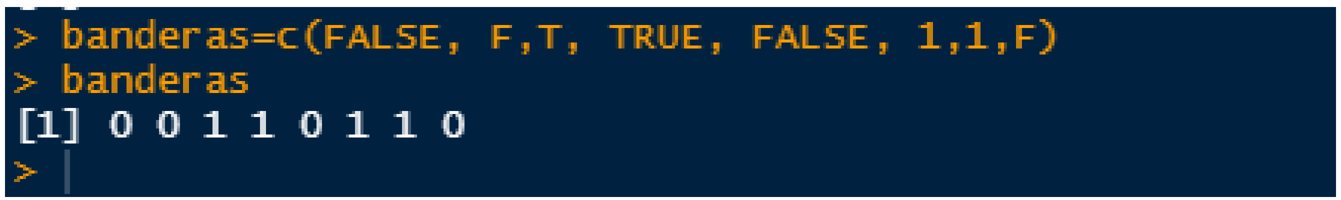
```
> names=c("hola","buenos ","dias ", 5)
> names
[1] "hola"     "buenos " "dias "    "5"
> 
```

If one element is character, the vector is data type character



```
> banderas=c(FALSE, F,T, TRUE, FALSE, T,T,F)
> banderas
[1] FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
> 
```

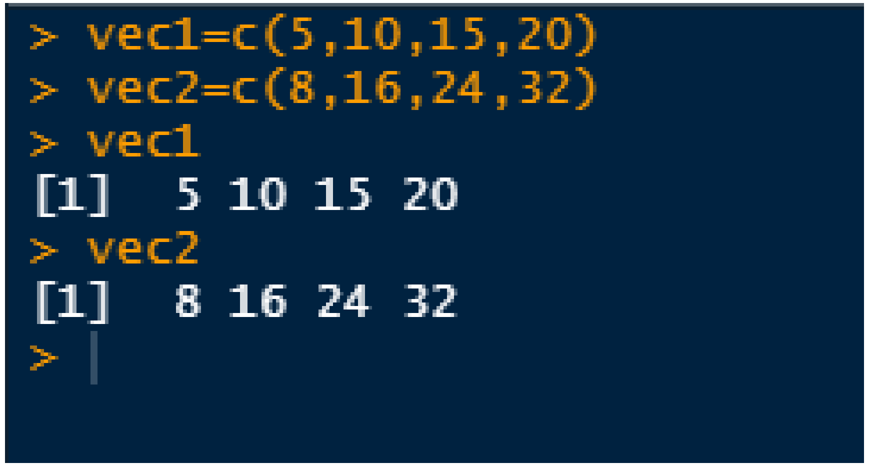
Be careful with upper-case letters



```
> banderas=c(FALSE, F,T, TRUE, FALSE, 1,1,F)
> banderas
[1] 0 0 1 1 0 1 1 0
> 
```

If one element is number, the vector is data type numeric

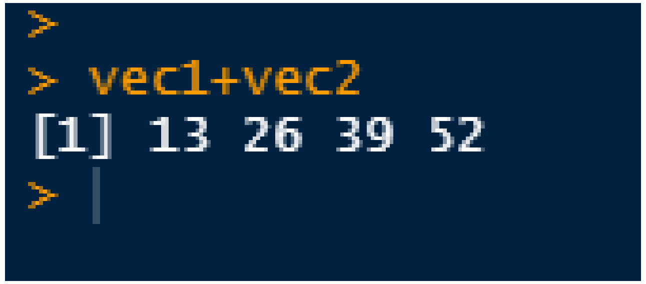
Vector operations element-wise



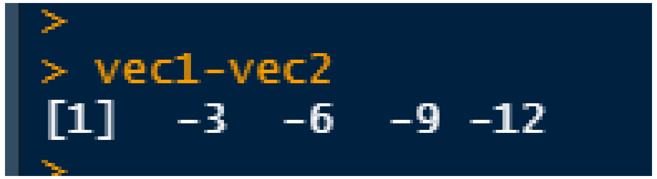
```
> vec1=c(5,10,15,20)
> vec2=c(8,16,24,32)
> vec1
[1]  5 10 15 20
> vec2
[1]  8 16 24  32
> 
```

$$a \circ b = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}_{(n \times 1)} \circ \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}_{(n \times 1)} = \begin{bmatrix} a_1b_1 \\ a_2b_2 \\ a_3b_3 \\ a_4b_4 \\ a_5b_5 \end{bmatrix}_{(n \times 1)}$$

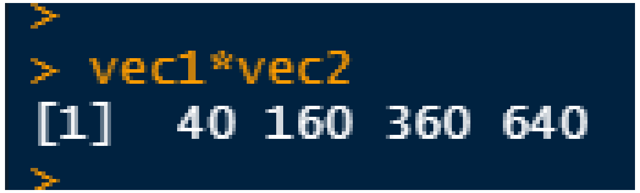
Element wise Product



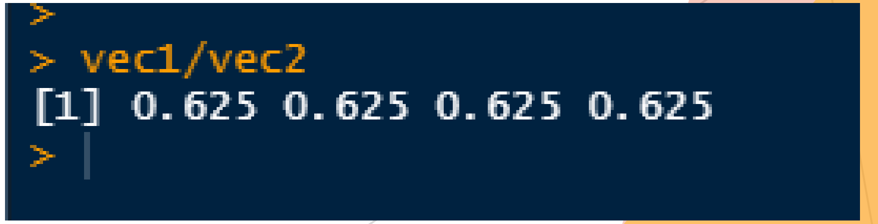
```
>
> vec1+vec2
[1] 13 26 39 52
> 
```



```
>
> vec1-vec2
[1]  -3  -6  -9 -12
>
```



```
>
> vec1*vec2
[1]  40 160 360 640
>
```



```
>
> vec1/vec2
[1] 0.625 0.625 0.625 0.625
> 
```

Useful functions

mean

median

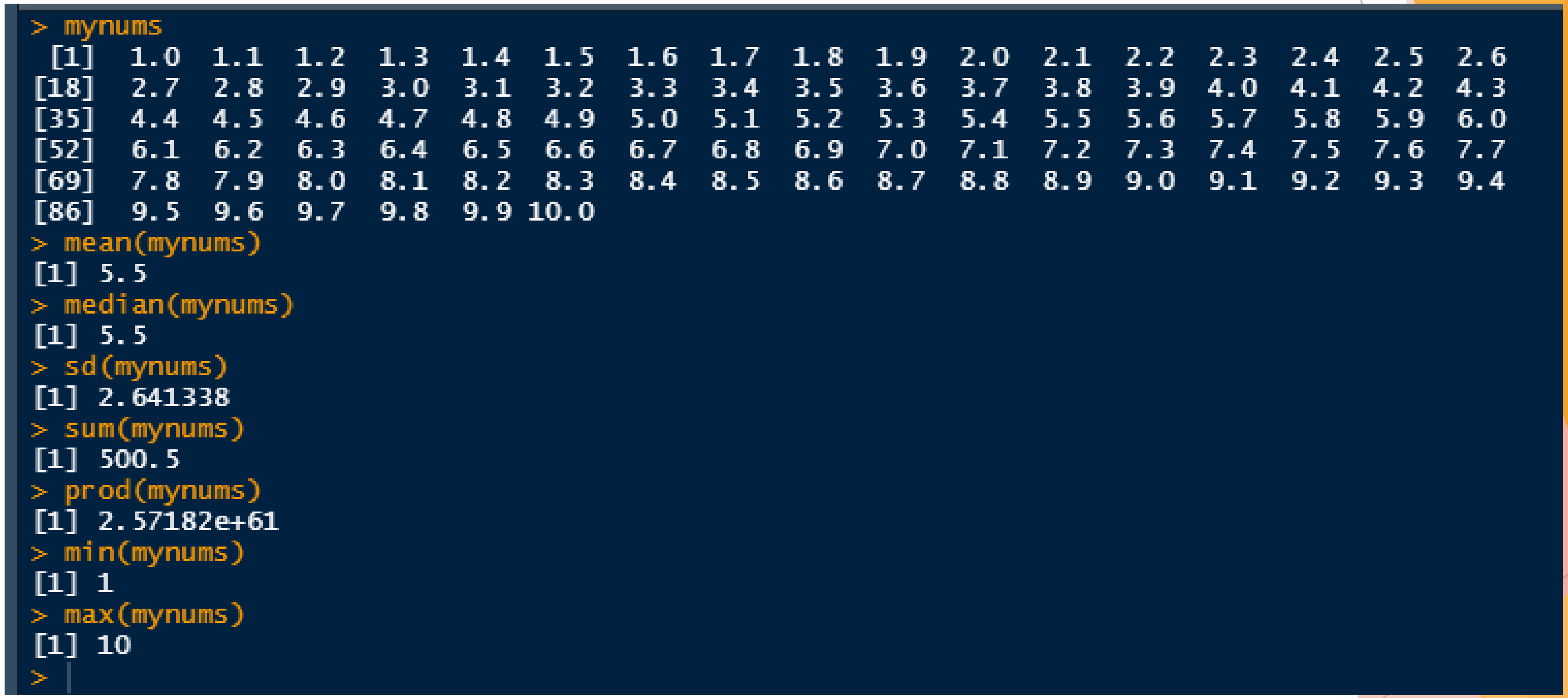
sd

sum

prod

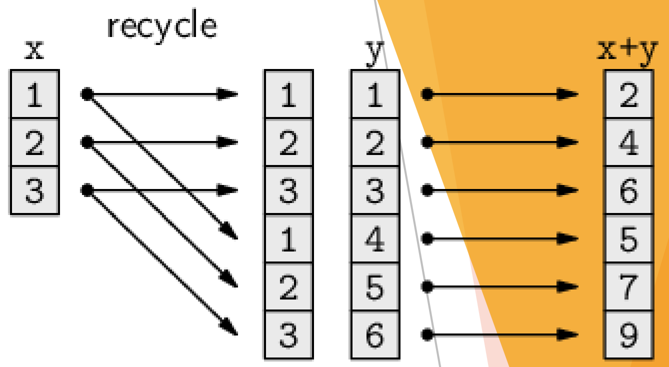
min

max



```
> mynums
 [1]  1.0  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4  2.5  2.6
[18]  2.7  2.8  2.9  3.0  3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9  4.0  4.1  4.2  4.3
[35]  4.4  4.5  4.6  4.7  4.8  4.9  5.0  5.1  5.2  5.3  5.4  5.5  5.6  5.7  5.8  5.9  6.0
[52]  6.1  6.2  6.3  6.4  6.5  6.6  6.7  6.8  6.9  7.0  7.1  7.2  7.3  7.4  7.5  7.6  7.7
[69]  7.8  7.9  8.0  8.1  8.2  8.3  8.4  8.5  8.6  8.7  8.8  8.9  9.0  9.1  9.2  9.3  9.4
[86]  9.5  9.6  9.7  9.8  9.9 10.0
> mean(mynums)
[1] 5.5
> median(mynums)
[1] 5.5
> sd(mynums)
[1] 2.641338
> sum(mynums)
[1] 500.5
> prod(mynums)
[1] 2.57182e+61
> min(mynums)
[1] 1
> max(mynums)
[1] 10
> 
```

Recycling



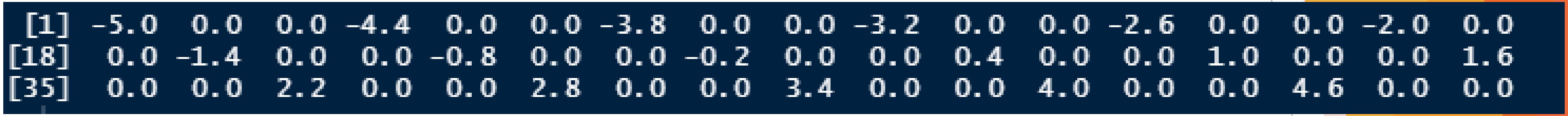
x	y
1	8
2	9
3	10
	11
	12
	13
	14
	15
	16

x+y

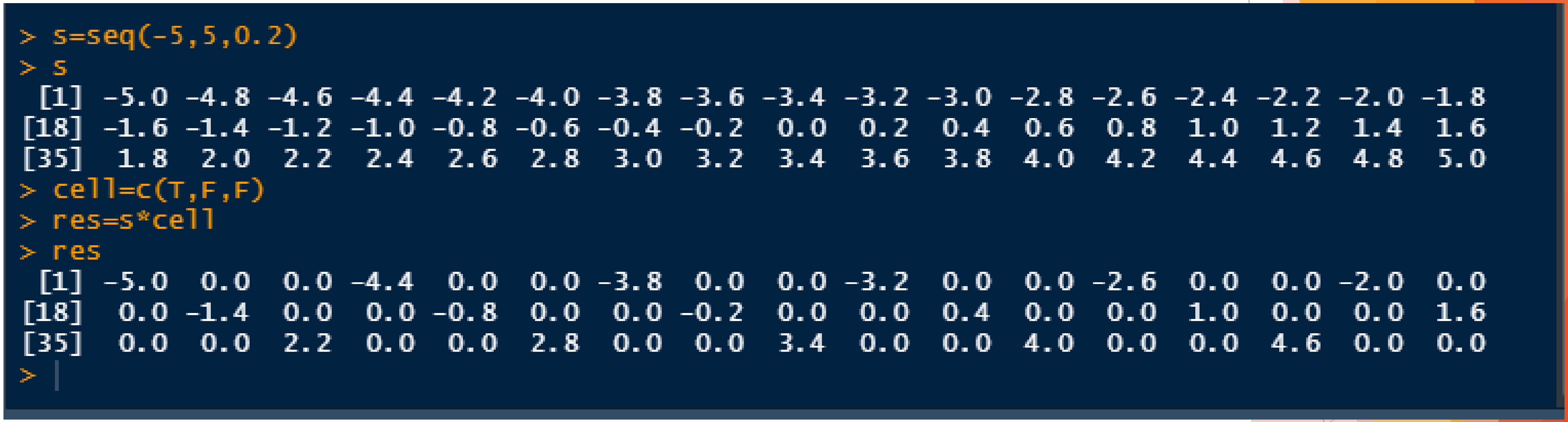
x	y
1	8
2	9
3	10
1	11
2	12
3	13
1	14
2	15
3	16

x+y

exercise



```
 [1] -5.0  0.0  0.0 -4.4  0.0  0.0 -3.8  0.0  0.0 -3.2  0.0  0.0 -2.6  0.0  0.0 -2.0  0.0
[18]  0.0 -1.4  0.0  0.0 -0.8  0.0  0.0 -0.2  0.0  0.0  0.4  0.0  0.0  1.0  0.0  0.0  1.6
[35]  0.0  0.0  2.2  0.0  0.0  2.8  0.0  0.0  3.4  0.0  0.0  4.0  0.0  0.0  4.6  0.0  0.0
```



```
> s=seq(-5,5,0.2)
> s
 [1] -5.0 -4.8 -4.6 -4.4 -4.2 -4.0 -3.8 -3.6 -3.4 -3.2 -3.0 -2.8 -2.6 -2.4 -2.2 -2.0 -1.8
[18] -1.6 -1.4 -1.2 -1.0 -0.8 -0.6 -0.4 -0.2  0.0  0.2  0.4  0.6  0.8  1.0  1.2  1.4  1.6
[35]  1.8  2.0  2.2  2.4  2.6  2.8  3.0  3.2  3.4  3.6  3.8  4.0  4.2  4.4  4.6  4.8  5.0
> cell=c(T,F,F)
> res=s*cell
> res
 [1] -5.0  0.0  0.0 -4.4  0.0  0.0 -3.8  0.0  0.0 -3.2  0.0  0.0 -2.6  0.0  0.0 -2.0  0.0
[18]  0.0 -1.4  0.0  0.0 -0.8  0.0  0.0 -0.2  0.0  0.0  0.4  0.0  0.0  1.0  0.0  0.0  1.6
[35]  0.0  0.0  2.2  0.0  0.0  2.8  0.0  0.0  3.4  0.0  0.0  4.0  0.0  0.0  4.6  0.0  0.0
> 
```

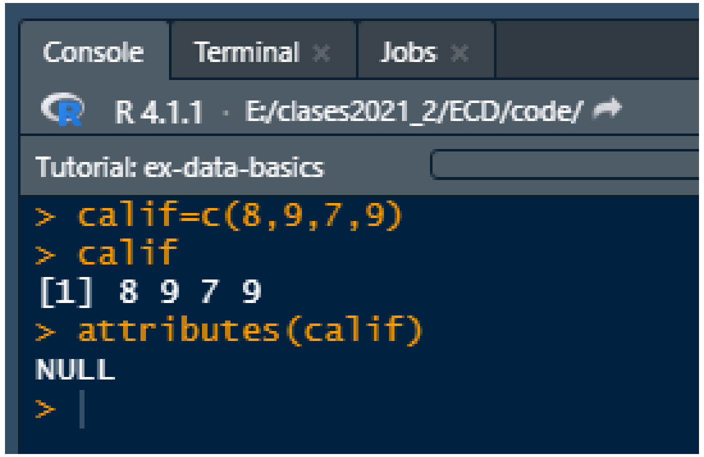
Attributes

names

dimensions

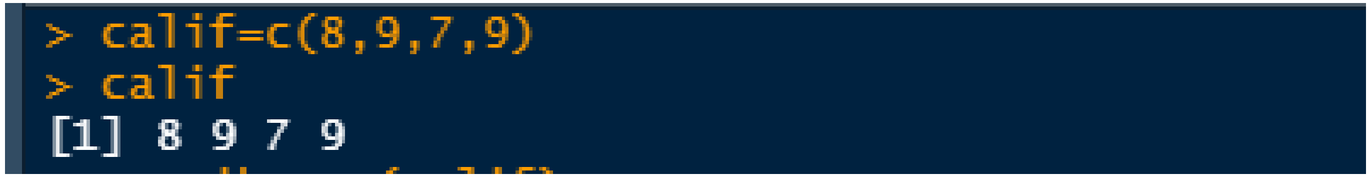
classes

attributes(object)

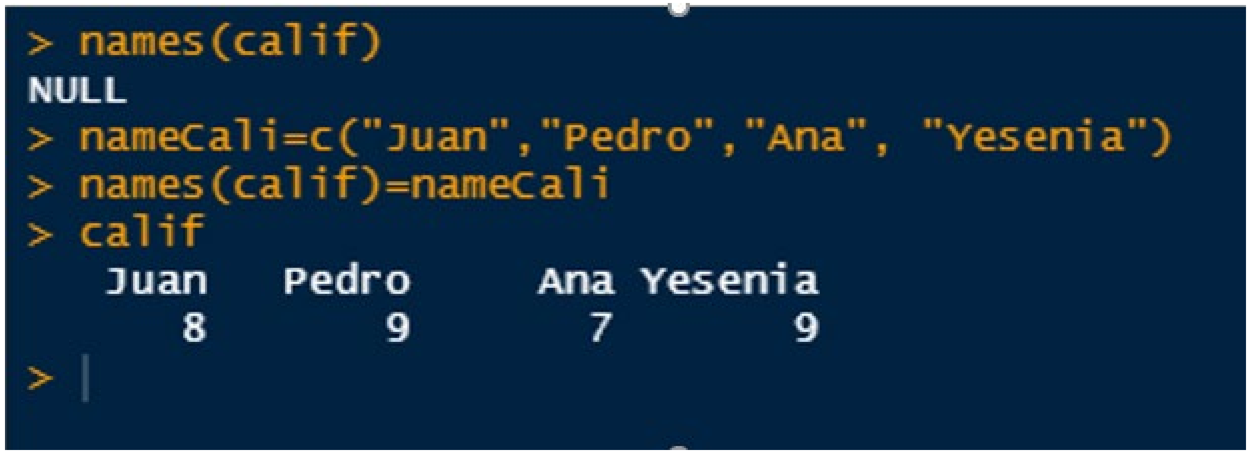


```
> calif=c(8,9,7,9)
> calif
[1] 8 9 7 9
> attributes(calif)
NULL
> 
```

Naming objects



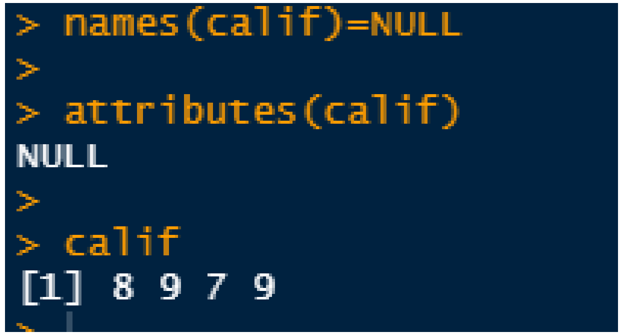
```
> calif=c(8,9,7,9)
> calif
[1] 8 9 7 9
```



```
> names(calif)
NULL
> nameCali=c("Juan","Pedro","Ana", "Yesenia")
> names(calif)=nameCali
> calif
   Juan   Pedro     Ana Yesenia 
      8       9       7       9 
> |
```

Create vector with names

Assign names to the first vector

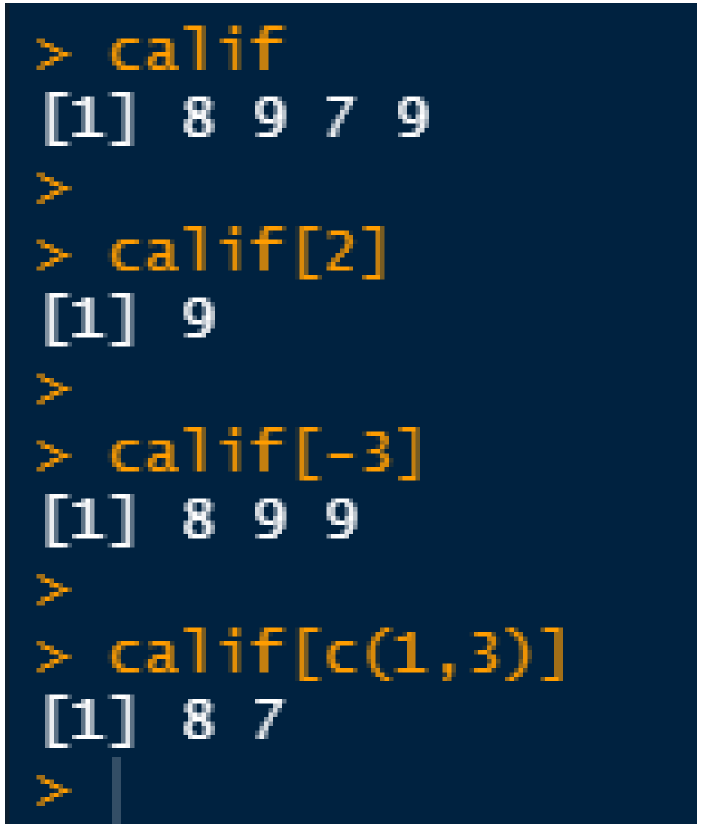


```
> names(calif)=NULL
> 
> attributes(calif)
NULL
> 
> calif
[1] 8 9 7 9
```

Eliminate names

Display elements of a vector

Display vector elements, using index



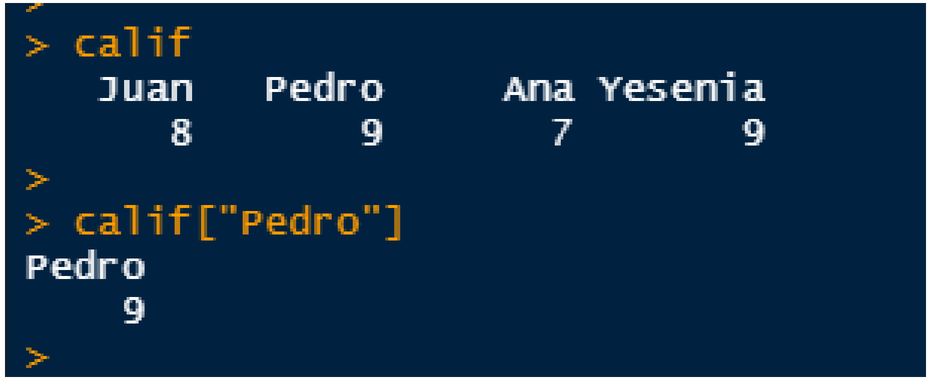
```
> calif
[1] 8 9 7 9
>
> calif[2]
[1] 9
>
> calif[-3]
[1] 8 9 9
>
> calif[c(1,3)]
[1] 8 7
> 
```

vectorName [index]

vectorName [-index]

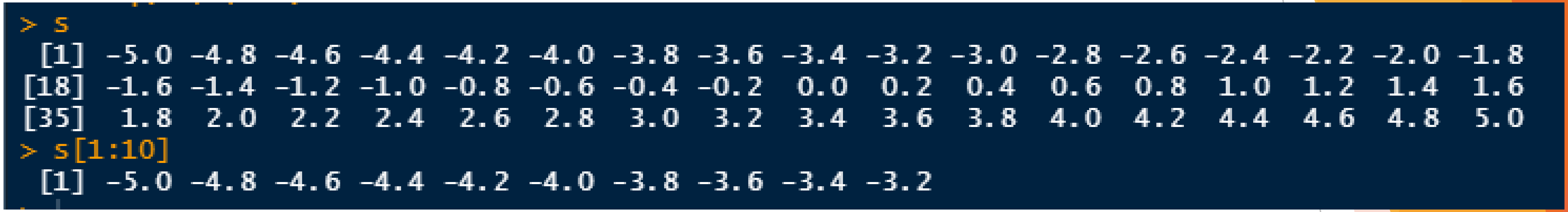
vectorName [c(list index)]

Display elements of a vector, using names



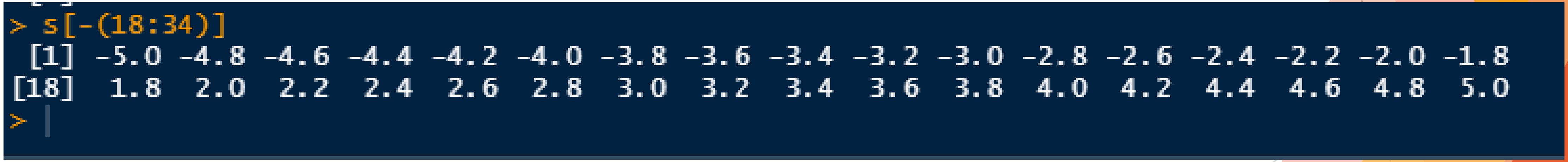
```
>
> calif
   Juan    Pedro      Ana Yesenia
      8        9        7       9
>
> calif["Pedro"]
Pedro
    9
>
```

Select several elements of a vector



```
> s
 [1] -5.0 -4.8 -4.6 -4.4 -4.2 -4.0 -3.8 -3.6 -3.4 -3.2 -3.0 -2.8 -2.6 -2.4 -2.2 -2.0 -1.8
[18] -1.6 -1.4 -1.2 -1.0 -0.8 -0.6 -0.4 -0.2  0.0  0.2  0.4  0.6  0.8  1.0  1.2  1.4  1.6
[35]  1.8  2.0  2.2  2.4  2.6  2.8  3.0  3.2  3.4  3.6  3.8  4.0  4.2  4.4  4.6  4.8  5.0
> s[1:10]
 [1] -5.0 -4.8 -4.6 -4.4 -4.2 -4.0 -3.8 -3.6 -3.4 -3.2
```

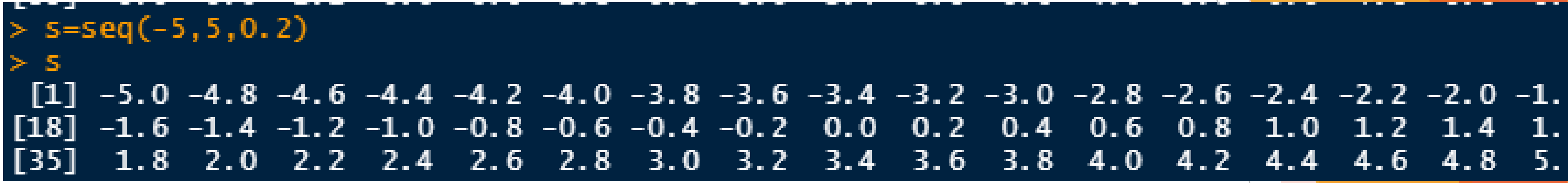
vectorName [indIni: indEnd]



```
> s[-(18:34)]
 [1] -5.0 -4.8 -4.6 -4.4 -4.2 -4.0 -3.8 -3.6 -3.4 -3.2 -3.0 -2.8 -2.6 -2.4 -2.2 -2.0 -1.8
[18]  1.8  2.0  2.2  2.4  2.6  2.8  3.0  3.2  3.4  3.6  3.8  4.0  4.2  4.4  4.6  4.8  5.0
> 
```

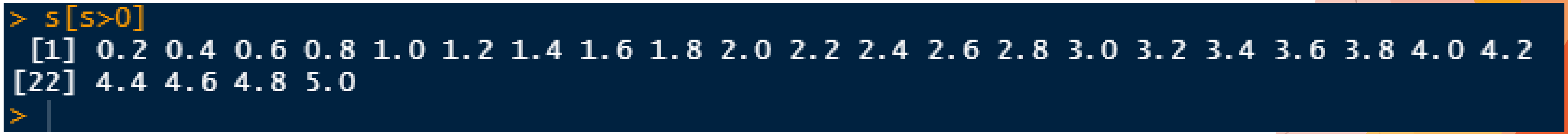
vectorName [-(indIni: indEnd)]

Select elements of a vector, using a condition



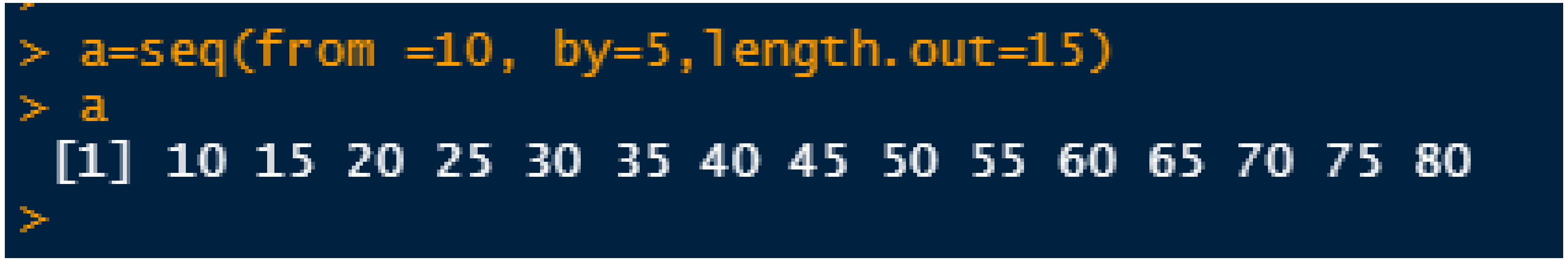
```
> s=seq(-5,5,0.2)
> s
 [1] -5.0 -4.8 -4.6 -4.4 -4.2 -4.0 -3.8 -3.6 -3.4 -3.2 -3.0 -2.8 -2.6 -2.4 -2.2 -2.0 -1.
[18] -1.6 -1.4 -1.2 -1.0 -0.8 -0.6 -0.4 -0.2  0.0  0.2  0.4  0.6  0.8  1.0  1.2  1.4  1.
[35]  1.8  2.0  2.2  2.4  2.6  2.8  3.0  3.2  3.4  3.6  3.8  4.0  4.2  4.4  4.6  4.8  5.
```

vectorName [condition]



```
> s[s>0]
 [1] 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0 3.2 3.4 3.6 3.8 4.0 4.2
[22] 4.4 4.6 4.8 5.0
> 
```

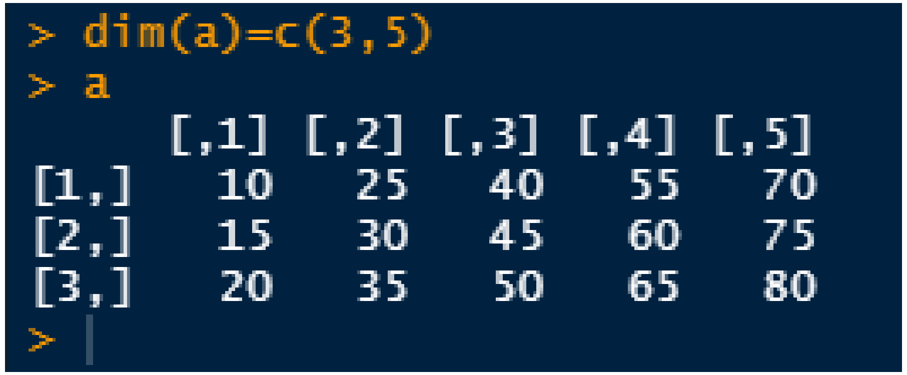
Change the dimension of a vector



```
> a=seq(from =10, by=5,length.out=15)
> a
 [1] 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80
>
```

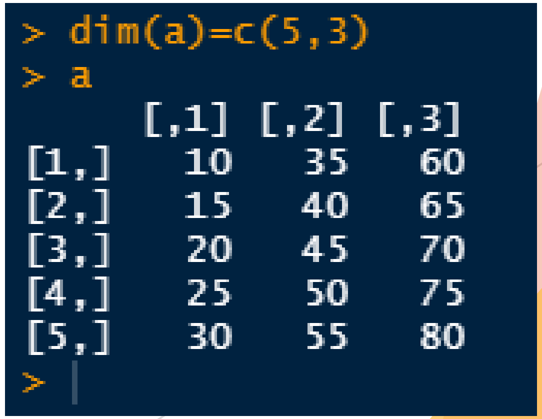
dim(vectorName) = c(d1,d2)

(reng, col)



```
> dim(a)=c(3,5)
> a
     [,1] [,2] [,3] [,4] [,5]
[1,]   10   25   40   55   70
[2,]   15   30   45   60   75
[3,]   20   35   50   65   80
>
```

Fill columns first



```
> dim(a)=c(5,3)
> a
     [,1] [,2] [,3]
[1,]   10   35   60
[2,]   15   40   65
[3,]   20   45   70
[4,]   25   50   75
[5,]   30   55   80
>
```

Other objects

vectors

matrix

list

factors

Data frames

Analizando vectores

class()	Despliega la clase del objeto
typeof()	Despliega el tipo de dato
length()	La longitud del objeto
str()	La estructura del objeto

```
f1 <- factor(c("a", "a", NA, NA, "b", NA, "a", "c", "a", "c", "b"))
str(f1)

 Factor w/ 3 levels "a","b","c": 1 1 NA NA 2 NA 1 3 1 3 ...

levels(f1) <- c(levels(f1),"No Answer")

f1[is.na(f1)] <- "No Answer“

str(f1)

 Factor w/ 4 levels "a","b","c","No Answer": 1 1 4 4 2 4 1 3 1 3 ...
```

```
f <- factor(c(3.4, 1.2, 5))
as.numeric(f)
```

```
levels(f)[f]
```

```
f2<- levels(f)[f]
f3 <- as.numeric(f2)
```

```
f <- factor(c(3.4, 1.2, 5))
f[4]<-3.4
f[5]<-2.4
```

```
mdat[ ,3, drop=FALSE]
```

Analizando vectores

class()	Despliega la clase del objeto
typeof()	Despliega el tipo de dato
length()	La longitud del objeto
str()	La estructura del objeto

Basic functions

class()	Tipo de objeto
typeof()	Tipo de dato del objeto
length	longuitud
as	Aplica una coerción a un objeto para convertirse a una determinada clase as(x,”numeric”) as.numeric(x)