

Tipos básicos de dados:

Integer: Conjunto dos números inteiros.

Numeric: Conjunto dos números reais.

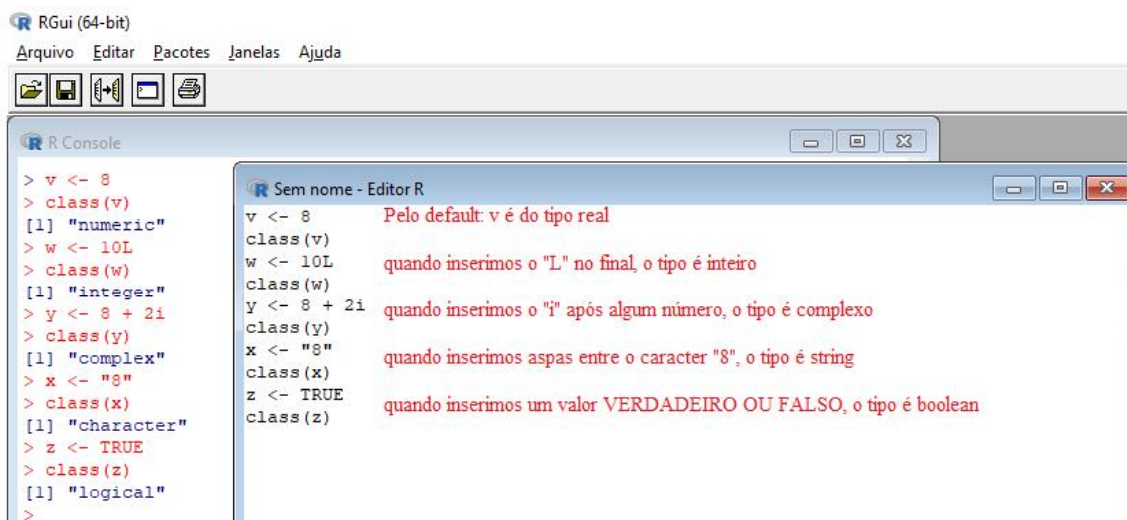
Complex: Conjunto dos números complexos.

Character: cadeia de caracteres (string)

Logical: valor booleano (TRUE or FALSE).

No default (padrão) do *software* R, os valores são do tipo numeric.

A função "class(objeto)" verifica o tipo de dado que corresponde ao objeto.



```
> v <- 8
> class(v)
[1] "numeric"
> w <- 10L
> class(w)
[1] "integer"
> y <- 8 + 2i
> class(y)
[1] "complex"
> x <- "8"
> class(x)
[1] "character"
> z <- TRUE
> class(z)
[1] "logical"
>
```

R Sem nome - Editor R

Pelo default: v é do tipo real

quando inserimos o "L" no final, o tipo é inteiro

quando inserimos o "i" após algum número, o tipo é complexo

quando inserimos aspas entre o caracter "8", o tipo é string

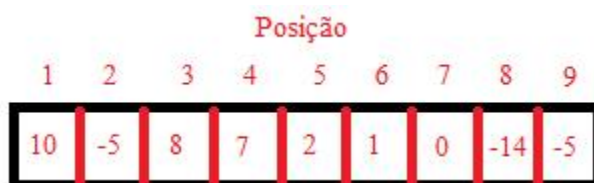
quando inserimos um valor VERDADEIRO OU FALSO, o tipo é boolean

Portanto, podemos escolher em qual classe (tipos de dados) será o nosso objeto, ou seja: o número 8, por default é numeric, porém se acrescentarmos a letra L, ou seja: 8L, agora é do tipo integer e se colocarmos entre aspas o número 8 e o 8L, temos "8" e "8L", logo ambos são do tipo character.

Até o momento, nós apenas conseguimos armazenar um único valor para cada objeto. Veremos agora como armazenar mais de um valor em um objeto. Para isto vamos utilizar o conceito de vetores.

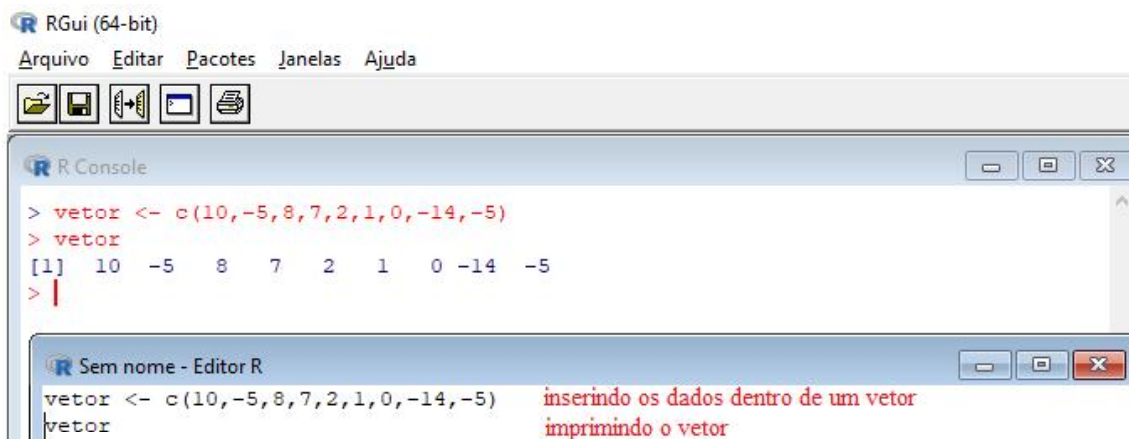
Vetores:

O que é um vetor? Vetor é uma estrutura de dado, em que você consegue armazenar um conjunto ordenado de dados de **MESMO TIPO** e se refere a cada dado pela sua **posição**.



Assim, um objeto passa a ter várias posições dentro dele.

Para utilizar um vetor em R, basta digitar a letra **c** antes dos dados.



The screenshot shows the RGui interface. The R Console window contains the following code and output:

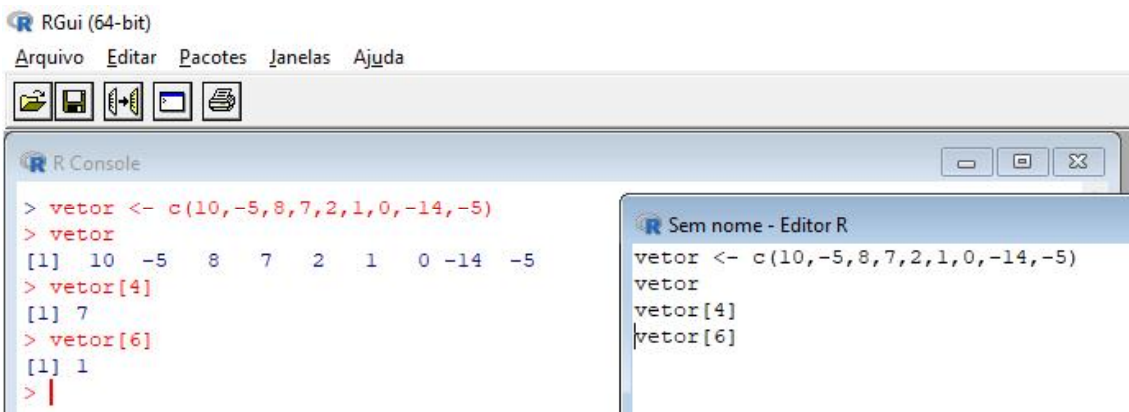
```
> vetor <- c(10,-5,8,7,2,1,0,-14,-5)
> vetor
[1] 10 -5 8 7 2 1 0 -14 -5
> |
```

The Editor R window shows the code being typed:

```
vetor <- c(10,-5,8,7,2,1,0,-14,-5)
vetor
```

Red annotations in the editor window indicate: "inserindo os dados dentro de um vetor" (inserting data into a vector) and "imprimindo o vetor" (printing the vector).

Podemos buscar um valor na posição de um vetor.



The screenshot shows the RGui interface. The R Console window contains the following code and output:

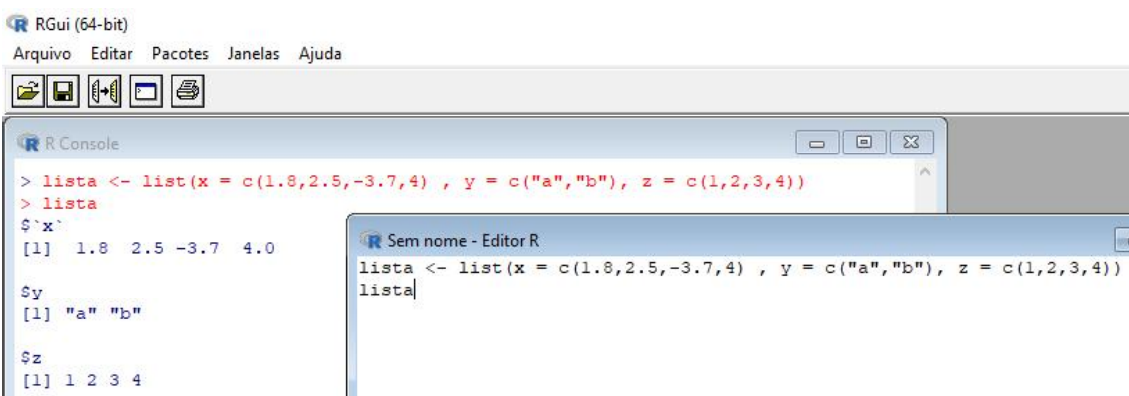
```
> vetor <- c(10,-5,8,7,2,1,0,-14,-5)
> vetor
[1] 10 -5 8 7 2 1 0 -14 -5
> vetor[4]
[1] 7
> vetor[6]
[1] 1
> |
```

The Editor R window shows the code being typed:

```
vetor <- c(10,-5,8,7,2,1,0,-14,-5)
vetor
vetor[4]
vetor[6]
```

A vantagem de trabalhar com vetores é que podemos armazenar muitas informações em apenas um objeto, porém essas informações DEVEM ser do MESMO TIPO de dados. Para armazenar VÁRIOS TIPOS de dados não podemos mais utilizar o conceito de vetor e sim o conceito de listas.

Para armazenar valores em uma lista:



The screenshot shows the RGui interface. The R Console window contains the following code and output:

```
> lista <- list(x = c(1.8,2.5,-3.7,4) , y = c("a","b"), z = c(1,2,3,4))
> lista
$`x`
[1] 1.8 2.5 -3.7 4.0

$y
[1] "a" "b"

$z
[1] 1 2 3 4
```

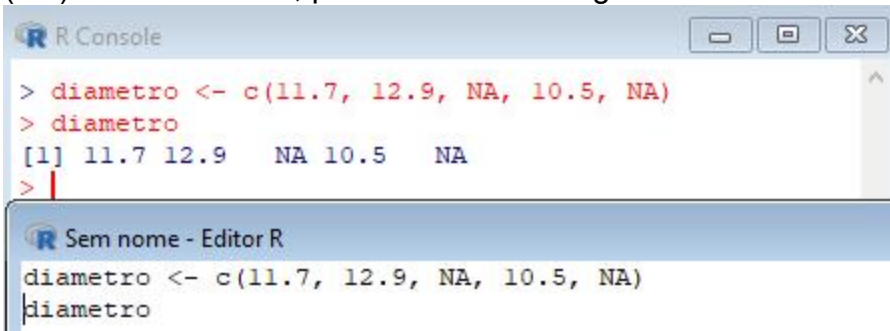
The Editor R window shows the code being typed:

```
lista <- list(x = c(1.8,2.5,-3.7,4) , y = c("a","b"), z = c(1,2,3,4))
lista|
```

Em resumo: A principal diferença é que em uma "lista", podemos armazenar dados de diferentes tipos e em vetores devem ser do mesmo tipo!

Obs: Para inserir um valor Not Available (NA) em um vetor, basta digitar **NA**

Exemplo: Ao medir os diâmetros das árvores, algumas delas não possuem o diâmetro mínimo estabelecido. Logo para armazenar um valor Not Available (NA) dentro do vetor, podemos fazer o seguinte:

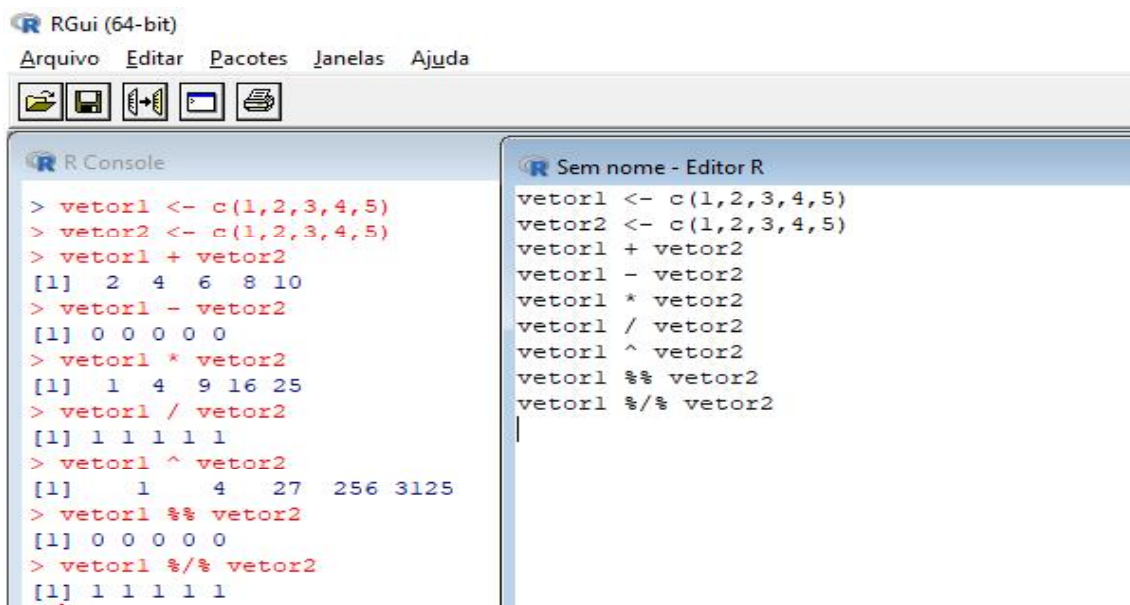


```
R Console
> diametro <- c(11.7, 12.9, NA, 10.5, NA)
> diametro
[1] 11.7 12.9 NA 10.5 NA
>

Sem nome - Editor R
diametro <- c(11.7, 12.9, NA, 10.5, NA)
diametro
```

Operações com Vetores:

As mesmas operações mencionadas anteriormente, podem ser utilizadas quando estamos trabalhando com vetores.



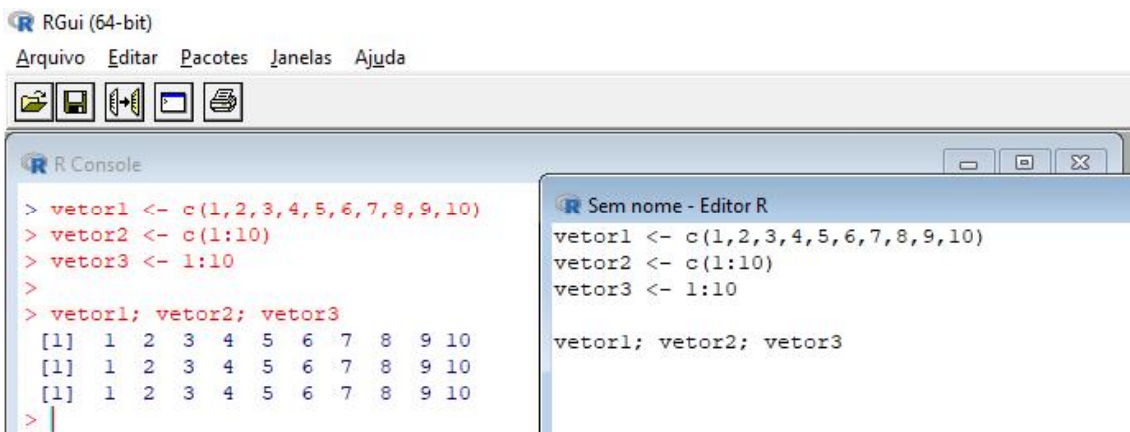
```
RGui (64-bit)
Arquivo Editar Pacotes Janelas Ajuda

R Console
> vetor1 <- c(1,2,3,4,5)
> vetor2 <- c(1,2,3,4,5)
> vetor1 + vetor2
[1] 2 4 6 8 10
> vetor1 - vetor2
[1] 0 0 0 0 0
> vetor1 * vetor2
[1] 1 4 9 16 25
> vetor1 / vetor2
[1] 1 1 1 1 1
> vetor1 ^ vetor2
[1] 1 4 27 256 3125
> vetor1 %% vetor2
[1] 0 0 0 0 0
> vetor1 %/% vetor2
[1] 1 1 1 1 1

Sem nome - Editor R
vetor1 <- c(1,2,3,4,5)
vetor2 <- c(1,2,3,4,5)
vetor1 + vetor2
vetor1 - vetor2
vetor1 * vetor2
vetor1 / vetor2
vetor1 ^ vetor2
vetor1 %% vetor2
vetor1 %/% vetor2
```

Obs: Podemos também utilizar funções já implementadas no R para fazer as operações dentro dos vetores. Isto será abordado na parte de funções.

Temos outras opções para definir um vetor. (dados consecutivos)



```
RGui (64-bit)
Arquivo Editar Pacotes Janelas Ajuda

R Console
> vetor1 <- c(1,2,3,4,5,6,7,8,9,10)
> vetor2 <- c(1:10)
> vetor3 <- 1:10
>
> vetor1; vetor2; vetor3
[1] 1 2 3 4 5 6 7 8 9 10
[1] 1 2 3 4 5 6 7 8 9 10
[1] 1 2 3 4 5 6 7 8 9 10
>

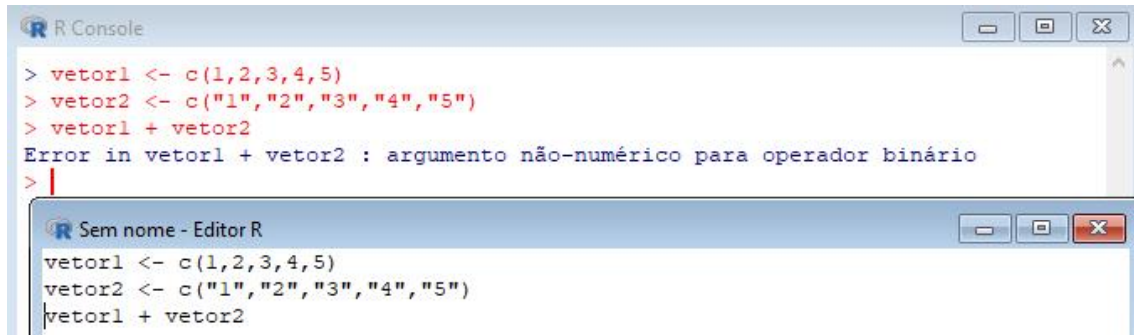
Sem nome - Editor R
vetor1 <- c(1,2,3,4,5,6,7,8,9,10)
vetor2 <- c(1:10)
vetor3 <- 1:10

vetor1; vetor2; vetor3
```

E se eu tivesse um vetor com tipos de dados diferentes? Por exemplo:

O vetor1 é do tipo “numeric” e o vetor2 é do tipo “character”.

Não é possível utilizar as operações; neste caso devemos transformar o vetor2 que é do tipo “character” para um tipo “numeric” ou “integer”.



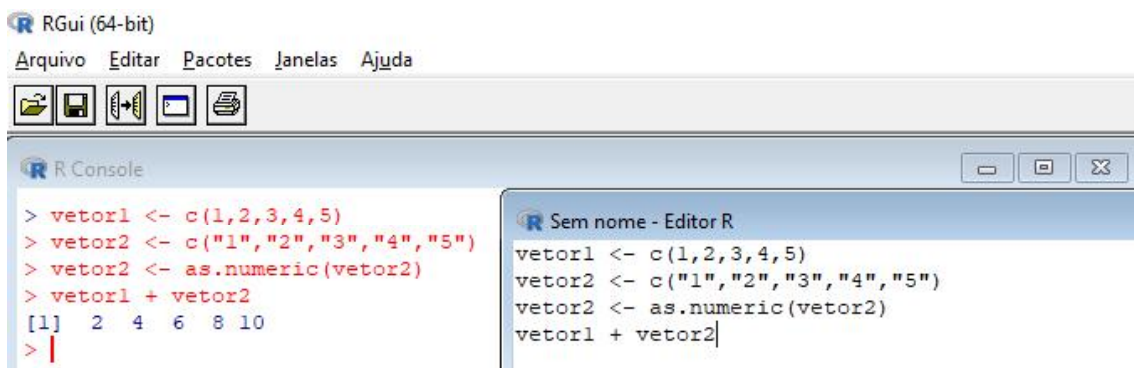
```
R Console
> vetor1 <- c(1,2,3,4,5)
> vetor2 <- c("1","2","3","4","5")
> vetor1 + vetor2
Error in vetor1 + vetor2 : argumento não-numérico para operador binário
> |

Sem nome - Editor R
vetor1 <- c(1,2,3,4,5)
vetor2 <- c("1","2","3","4","5")
vetor1 + vetor2
```

Para fazer a conversão de um vetor, podemos utilizar a seguinte sintaxe.

objeto <- as.integer(objeto)

objeto <- as.numeric(objeto)



```
RGui (64-bit)
Arquivo Editar Pacotes Janelas Ajuda

R Console
> vetor1 <- c(1,2,3,4,5)
> vetor2 <- c("1","2","3","4","5")
> vetor2 <- as.numeric(vetor2)
> vetor1 + vetor2
[1] 2 4 6 8 10
> |

Sem nome - Editor R
vetor1 <- c(1,2,3,4,5)
vetor2 <- c("1","2","3","4","5")
vetor2 <- as.numeric(vetor2)
vetor1 + vetor2
```

Com a transformação do vetor2 em “numeric”, foi possível fazer a operação desejada.

Exercícios:

Este é o momento de fixação, portanto **PRATIQUE!**

- 1) Como que o software R identifica o tipo da variável:
 - a) Integer (inteira).
 - b) Numeric (real).
 - c) Character (String = cadeia de caracteres).
 - d) Complex (complexa).
 - e) Logical (booleana).
- 2) Como você verifica o tipo de dado?
- 3) Qual a diferença em utilizar um objeto “normal”, e utilizar um objeto do tipo vetor?
- 4) Como diferenciar se um objeto é do tipo normal e do tipo vetor?

- 5) Armazene em dois objetos do tipo vetor, os seguintes valores: vetor1 = (1,3,7,-5,10) e vetor2 = (10,5,-2,4,15) e efetue as seguintes operações:
- Some os dois vetores.
 - Subtraia os dois vetores.
 - Multiplique os dois vetores.
 - Divida o primeiro vetor pelo segundo.
 - O resto do vetor1 pelo vetor2.
 - Eleve o vetor2 pelo vetor1.
 - Eleve o vetor1 ao quadrado.
- 6) Como você faria uma busca de um valor dentro de um vetor? Por exemplo, considere um vetor com 10 posições e você deseja saber qual o valor da posição 5? `vetor <- c(10,-5,0,-4,12,1,1,3,3,10)`.
- 7) Qual a principal diferença entre trabalhar com **vetores** e **listas**?
- 8) Como converter um vetor do tipo numeric para o tipo integer?
- 9) Como converter um vetor do tipo integer para character?
- 10) Como converter um vetor do tipo character para integer? Sempre é possível?
- 11) Por que é necessário saber o tipo de dado?

Respostas:

- 1) a) Todo o valor que estiver com um "L" ao final é do tipo Inteiro. (5L)
 - b) O padrão do R, é declarar as variáveis como Numeric (real). (5)
 - c) Quando está entre aspas duplas. ("5L" ou "5")
 - d) Quando há a parte imaginária (letra i). (5 + 2i)
 - e) Quando há um valor verdade ou falso. (TRUE or FALSE)
- 2) Utilizando a função "class(objeto)".
- 3) Um objeto "normal" armazena apenas um valor nele, já o objeto do tipo vetor armazena um valor por posição, portanto um objeto do tipo vetor pode armazenar diversos valores, sendo cada um em uma posição. O *software* R, inicia com a posição 1.
- 4) Quando utilizamos a letra "c" antes da leitura, o R entende que estamos utilizando vetores. (obs: 5:24, também é uma abreviatura para um vetor, porém este vetor terá 20 posições, com os números crescentes, iniciando em 5 e terminando em 24).

```
R Console
> vetor1 <- c(1,3,7,-5,10)
> vetor2 <- c(10,5,-2,4,15)
> vetor1 + vetor2
[1] 11 8 5 -1 25
> vetor1 - vetor2
[1] -9 -2 9 -9 -5
> vetor1 * vetor2
[1] 10 15 -14 -20 150
> vetor1 / vetor2
[1] 0.1000000 0.6000000 -3.5000000 -1.2500000 0.6666667
> vetor1 %% vetor2
[1] 1 3 -1 3 10
> vetor2 ^ vetor1
[1] 1.000000e+01 1.250000e+02 -1.280000e+02 9.765625e-04 5.766504e+11
> vetor1^2
[1] 1 9 49 25 100
<|

Sem nome - Editor R
vetor1 <- c(1,3,7,-5,10)
vetor2 <- c(10,5,-2,4,15)
vetor1 + vetor2
vetor1 - vetor2
vetor1 * vetor2
vetor1 / vetor2
vetor1 %% vetor2
vetor2 ^ vetor1
vetor1^2
```

- 6) vetor[5]
- 7) Em vetores, podemos armazenar apenas os dados com o mesmo tipo e em listas podemos armazenar em cada posição diferentes tipos de dados, inclusive podemos armazenar um vetor dentro de cada posição da lista.
- 8) `vet <- as.integer(vet)`
- 9) `vet <- as.character(vet)`
- 10) `vet <- as.integer(vet)`, só é possível a conversão se os caracteres são números.
- 11) É necessário, quando desejamos efetuar as operações entre os vetores.

Não sendo possível, efetuar operações em vetores do tipo character com numeric, ou character com integer. Obs: O R, trabalha com operações entre tipos integer e numeric, resultando no tipo numeric.