R para iniciantes
Aula II
Funções matemáticas

Carlos Henrique Tonhatti

Dúvidas da última Aula?

Sumário

Sumário

Sumário

Sumário

Sumário

Operadores

Assim como outras linguagens o R possui operadores aritméticos comparativos, lógicos entre outros.

Operadores

Assim como outras linguagens o R possui operadores aritméticos comparativos, lógicos entre outros.
Para mais detalhes veja `?Syntax`

Operadores

Operadores aritméticos

+	adição
-	subtração
*	multiplicação
^	potência
/	divisão
%%	módulo
%/%	divisão de inteiros

Exemplos aritméticos

```
> 1+2
 [1] 3
> 3^3
 [1] 9
> 5/2
 [1] 2.5
```

Exemplos aritméticos

```
> 1+2
 [1] 3
> 3^3
 [1] 9
> 5/2
 [1] 2.5
> 5%%2
 [1] 1
```

Exemplos aritméticos

```
> 1+2
 [1] 3
> 3^3
 [1] 9
> 5/2
 [1] 2.5
> 5%%2
 [1] 1
> 5%/%2
 [1] 2
```

Operadores

Operadores de comparação

<	menor que	>=	maior ou igual
>	maior que	==	igual
<=	menor ou igual	!=	diferente

Comparam dois objetos e retornam verdadeiro ou falso (`TRUE`, `FALSE`).

Exemplos de comparação

```
> 1<5
 [1] TRUE
```

Exemplos de comparação

```
> 1<5
 [1] TRUE
> 1>5
 [1] FALSE
```

Exemplos de comparação

```
> 1<5
 [1] TRUE
> 1>5
 [1] FALSE
> 1==5
 [1] FALSE
```

Exemplos de comparação

```
> 1<5
 [1] TRUE
> 1>5
 [1] FALSE
> 1==5
 [1] FALSE
> 1!=5
 [1] TRUE
```

Exemplos de comparação

```
> 1<5
 [1] TRUE
> 1>5
 [1] FALSE
> 1==5
 [1] FALSE
> 1!=5
 [1] TRUE
> 5==5
 [1]TRUE
```

Exemplos de comparação

Operadores de comparação podem ser usados sobre caracteres:

```
> "casa"=="casa"
 [1] TRUE
```

Exemplos de comparação

Operadores de comparação podem ser usados sobre caracteres:

```
> "casa"=="casa"
 [1] TRUE
```

É usado a ordem alfabética para determinar quem é maior ou menor.

```
> "a"<"z"
  [1] TRUE
```

Exemplos de comparação

Operadores de comparação podem ser usados sobre caracteres:

```
> "casa"=="casa"
 [1] TRUE
```

É usado a ordem alfabética para determinar quem é maior ou menor.

```
> "a"<"z"
  [1] TRUE
> "a"<"A"
 [1] TRUE
```

Exemplos de comparação

Operadores de comparação podem ser usados sobre caracteres:

```
> "casa"=="casa"
 [1] TRUE
```

É usado a ordem alfabética para determinar quem é maior ou menor.

```
> "a"<"z"
  [1] TRUE
> "a"<"A"
 [1] TRUE
> "aguia"<"zebrafish"
 [1] TRUE
```

Precedência entre operadores

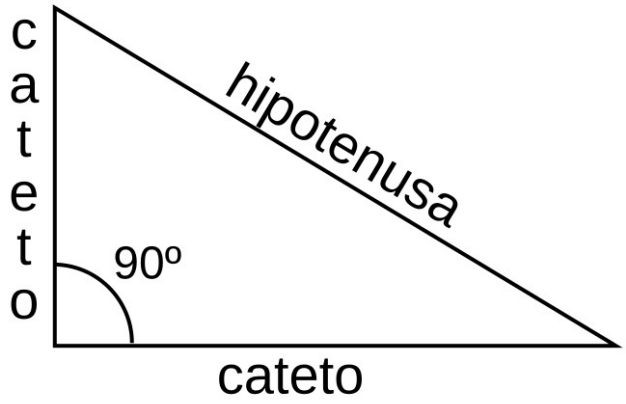
Dentro de uma expressão os operadores de igual precedência são avaliados da esquerda para a direita exceto quando indicados por parênteses.

Ordem de precedência

```
...^  %% %/% * / + - > < >= ...
```

Para mais detalhes e outros operadores veja `?Syntax`

Exemplo “hipotenusa”



$$hipotenusa^2 = catetoA^2 + catetoB^2$$

$$hipotenusa = \sqrt{catetoA^2 + catetoB^2}$$

Exemplo “Hipotenusa”

```
> catetoA<- 4
> catetoB<- 3
```

Exemplo “Hipotenusa”

```
> catetoA<- 4
> catetoB<- 3
> quad.hipotenusa<- catetoA^2 + catetoB^2
> hipotenusa<- sqrt(quad.hipotenusa)
```

Exemplo “Hipotenusa”

```
> catetoA<- 4
> catetoB<- 3
> quad.hipotenusa<- catetoA^2 + catetoB^2
> hipotenusa<- sqrt(quad.hipotenusa)

OU

> hipotenusa<- sqrt(catetoA^2 + catetoB^2)
> hipotenusa
 [1] 5
```

Arredondamentos

Funções de arredondamento

```
# Arredonda o número até o número de casas decimais indicado
(default 0 casas)
round(x, digits=0)

# Arredonda o número "para cima".
ceiling(x)

# Arrendonda o número "para baixo".
floor(x)
```

Exemplos

```
> round(2.718282)
 [1] 3

> round(2.718282,digits=3)
 [1] 2.718

> ceiling(2.718282)
 [1] 3

> floor(2.718282)
 [1] 2
```

Sumário

Criação de vetores

Funções que criam vetores

```
# Combina os argumentos em um vetor ou lista.
c(...)
```

Criação de vetores

Funções que criam vetores

```
# Combina os argumentos em um vetor ou lista.
c(...)

# Gera uma sequência regular
seq(from=1,to=1, by= ...)
from:to
```

Criação de vetores

Funções que criam vetores

```
# Combina os argumentos em um vetor ou lista.
c(...)

# Gera uma sequência regular
seq(from=1,to=1, by= ...)
from:to

# Replica os elementos de um vetor
rep(x, times)
```

Exemplos

```
> Animal<- c(“Cachorro”, “Gato”, “Mosquito”)
> Peso<- c(1500,1000,1)
```

Exemplos

```
> Animal<- c(“Cachorro”, “Gato”, “Mosquito”)
> Peso<- c(1500,1000,1)

> temperatura<- seq(from= 35,to=37,by=.5)
> temperatura
 [1] 35.0 35.5 36.0 36.5 37.0
```

Exemplos

```
> Animal<- c(“Cachorro”, “Gato”, “Mosquito”)
> Peso<- c(1500,1000,1)

> temperatura<- seq(from= 35,to=37,by=.5)
> temperatura
 [1] 35.0 35.5 36.0 36.5 37.0

> notas<- 0:10
> notas
 [1] 0 1 2 3 4 5 6 7 8 9 10
```

Exemplos

```
> Animal<- c(“Cachorro”, “Gato”, “Mosquito”)
> Peso<- c(1500,1000,1)

> temperatura<- seq(from= 35,to=37,by=.5)
> temperatura
 [1] 35.0 35.5 36.0 36.5 37.0

> notas<- 0:10
> notas
 [1] 0 1 2 3 4 5 6 7 8 9 10
```

Exemplos

```
> conceito<- rep(“A”,5)
> conceito
 [1] “A” “A” “A” “A” “A”

> conceitos<- rep(c(“A”,“B”),times=5)
> conceitos
  [1] “A” “B” “A” “B” “A” “B” “A” “B” “A” “B”

> conceitos<- rep(c(“A”, “B”),each=5)
 [1] “A” “A” “A” “A” “A” “B” “B” “B” “B” “B”
```

Operações com um vetor

Operações aplicadas a um vetor normalmente operam em cada do seus elementos.

```
> notas
 [1] 0 1 2 3 4 5 6 7 8 9 10
> notas*2
 [1] 0 2 4 6 8 10 12 14 16 18 20
```

Operações entre vetores

Operações aplicadas entre vetores operam pareando os elementos de ambos os vetores.

```
> a<- c(1,2,3,4,5,6)
> b<- c(6,5,4,3,2,1)
> a+b
 [1] 7 7 7 7 7 7
```

Operações entre vetores — tamanhos diferentes

Regra da ciclagem

Os elementos do vetor mais curto são repetidos sequencialmente até que a operação seja aplicada a todos os elementos do vetor mais longo.

```
> a<- c(1,2,3,4,5,6)
> b<- c(0,1)
> a*b
 [1] 0 2 0 4 0 6
```

Operações entre vetores — tamanhos diferentes

Qaundo o vetor menor não for multiplo do vetor maior a operações e realizada mas aparece um aviso.

```
> a<- c(1,2,3,4,5,6)
> b<- c(0,1,2,3)
> a*b
 [1] 0 2 6 12 0 6
Mensagens de aviso perdidas:
In a * b :
  comprimento do objeto maior não é múltiplo do comprimento
do objeto menor
```

Hipotenusas

Usando operações entre vetores é possível realizar a mesma operação para vários dados ao mesmo tempo.

```
> catetosA<- c(2,3,4,7,8)
> catetosB<- c(2,5,3,6,5)
> hipotenusas<- sqrt(catetosA^2 + catetosB^2)
> hipotenusas
 [1] 2.828427 5.830952 5.000000 9.219544 9.433981
```

Funções que operam sobre todo o vetor

Algumas funções operam sobre todos os elementos do vetor.

```
> a< c(1,2,3,4,5,6,7,8,9)

# soma
> sum(a)
 [1] 45

# média
> mean(a)
 [1] 5

# diferença entre elementos
> diff(a)
 [1] 1 1 1 1 1 1 1 1
```

Funções que operam sobre todo o vetor

Algumas funções operam sobre todos os elementos do vetor.

```
> a< c(1,2,3,4,5,6,7,8,9)

# soma
> sum(a)
 [1] 45

# média
> mean(a)
 [1] 5

# diferença entre elementos
> diff(a)
 [1] 1 1 1 1 1 1 1 1
```

Funções que operam sobre todo o vetor

```
# soma cumulativa
> cumsum(a)
 [1] 1 3 6 10 15 21 28 36 45
```

Funções que operam sobre todo o vetor

```
# soma cumulativa
> cumsum(a)
 [1] 1 3 6 10 15 21 28 36 45

# produto cumulativo
> cumprod(a)
 [1] 1 2 6 24 120 720 5040 40320 362880
```

Funções que operam sobre todo o vetor

```
# soma cumulativa
> cumsum(a)
 [1] 1 3 6 10 15 21 28 36 45

# produto cumulativo
> cumprod(a)
 [1] 1 2 6 24 120 720 5040 40320 362880
```

Outras funções que operam sobre todo o vetor:

`var,sd,min,max,range ...`

Sumário

Operadores

Operações com matriz

+	soma elemento por elemento
*	multiplicação elemento por elemento
%*%	multiplicação entre matrizes

Funções com matriz

Funções com matrizes

Sendo **A** uma matriz

Funções com matriz

Funções com matrizes

Sendo **A** uma matriz

# Matriz transposta
t(A)

Funções com matriz

Funções com matrizes

Sendo **A** uma matriz

```
# Matriz transposta
t(A)

# Diagonal
diag(A)
```

Funções com matriz

Funções com matrizes

Sendo **A** uma matriz

```
# Matriz transposta
t(A)

# Diagonal
diag(A)

# Autovalores e autovetores
eigen(A)
```

Exemplos

```
> A
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

> t(A)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9

> diag(A)
[1] 1 5 9
```

```
> eigen(A)
$values
 [1] 1.611684e+01 -1.116844e+00 -5.700691e-16

$vectors
           [,1]       [,2]       [,3]
[1,] -0.4645473 -0.8829060  0.4082483
[2,] -0.5707955 -0.2395204 -0.8164966
[3,] -0.6770438  0.4038651  0.4082483
```

Dados faltantes

No R dados faltantes são representados por `NA` (*"Not Available"*).

```
> x<- c(2,4,5,NA)
> mean(x)
```

Dados faltantes

No R dados faltantes são representados por `NA` (“ *Not Available*”).

```
> x<- c(2,4,5,NA)
> mean(x)
 [1] NA
```

Dados faltantes

No R dados faltantes são representados por `NA` (*"Not Available"*).

```
> x<- c(2,4,5,NA)
> mean(x)
 [1] NA
> mean(x, na.rm=T)
 [1] 3.666667
```

Valores infinitos

No R valores infinitos são representados por `Inf`.

```
>1/0
 [1] Inf
>1/Inf
 [1] 0
```

Valores indefinidos

No R valores que não podem ser definidos são representados por `NaN` ("Not a Number").

```
> 0/0
 [1] NaN
> sqrt(-1)
 [1] NaN
```

Sumário

Operações com outras classes

Para aplicar operações e funções sobre objetos de outras classes (fatores, *data frames*, listas, *arrays*, etc) é necessário o uso de uma função auxiliar.

Operações com outras classes

Para aplicar operações e funções sobre objetos de outras classes (fatores, *data frames*, listas, *arrays*, etc) é necessário o uso de uma função auxiliar.

Existem várias funções e métodos para isso.

Algumas funções auxiliares

Funções auxiliares

# Aplica uma função sobre *array* ou matriz.
apply(X, MARGIN, FUN, . . .)

Algumas funções auxiliares

Funções auxiliares

# Aplica uma função sobre *array* ou matriz.
apply(X, MARGIN, FUN, . . .)

# Aplica uma função sobre uma lista ou vetor.
lapply(X, FUN, . . .)

Algumas funções auxiliares

Funções auxiliares

# Aplica uma função sobre *array* ou matriz.
apply(X, MARGIN, FUN, . . .)

# Aplica uma função sobre uma lista ou vetor.
lapply(X, FUN, . . .)

# Aplica uma função sobre um vetor agrupado por fatores.
tapply(X, INDEX, FUN,. . .)

Algumas funções auxiliares

Funções auxiliares

# Aplica uma função sobre *array* ou matriz.
apply(X, MARGIN, FUN, . . .)

# Aplica uma função sobre uma lista ou vetor.
lapply(X, FUN, . . .)

# Aplica uma função sobre um vetor agrupado por fatores.
tapply(X, INDEX, FUN,. . .)

X= objeto
MARGIN = dimensão na qual será aplicada a função (1 para linha e 2 para coluna)
INDEX= fator usado para agrupar
FUN = função principal

Exemplos

```
> notas<-matrix(c(5,6,4,9,9,7,4,3,2),ncol=3)
> notas
     [,1] [,2] [,3]
[1,]    5    9    4
[2,]    6    9    3
[3,]    4    7    2
```

Exemplos

```
> notas<-matrix(c(5,6,4,9,9,7,4,3,2),ncol=3)
> notas
     [,1] [,2] [,3]
[1,]    5    9    4
[2,]    6    9    3
[3,]    4    7    2

> apply(X=notas, MARGIN=1, FUN=mean)
[1] 6.000000 6.000000 4.333333
```

Exemplos

```
> notas<-matrix(c(5,6,4,9,9,7,4,3,2),ncol=3)
> notas
     [,1] [,2] [,3]
[1,]    5    9    4
[2,]    6    9    3
[3,]    4    7    2

> apply(X=notas, MARGIN=1, FUN=mean)
 [1] 6.000000 6.000000 4.333333
```

No caso de médias por linha ou por coluna é mais fácil usar `rowMeans()` ou `colMeans()`.

Exemplos

```
> notas<-list(aluno1=c(5,6,8),aluno2=c(4,5))
> notas
$aluno1
 [1] 5 6 8
$aluno2
 [1] 4 5

> lapply(notas,mean)
$aluno1
 [1] 6.333333
$aluno2
 [1] 4.5
```

Exemplos

```
> idade<- c(25,26,55,37,21,42)
> sexo<- c(“F”, “F”, “M”, “F”, “M”, “M”)
> tapply(X=idade, INDEX=sexo, FUN=mean)
        F         M
 29.33333  39.33333
```

Para a próxima aula

- Ler o cap. 3 da apostila
- Fazer o tutorial 2 “Funções matemáticas”
- Fazer os exercícios “Funções matemáticas”