

Treinamento em Programação no Ambiente R

GENt

May 16, 2019

Dia 1

Este relatório foi feito utilizando R Markdown. Ele pode ser exportado em `.html` ou `.pdf`, basta alterar o `output`: no cabeçalho entre `html_document` e `pdf_document`.

Primeiro, um “oi” para o mundo.

```
cat("Hello world!")
```

```
## Hello world!
```

A função `cat` significa “concatenar” e ela vai imprimir o que eu escrevi no console.

Estabelecendo diretório de trabalho

É a pasta no meu computador que o R “conversa”, ou seja, que vai buscar os arquivos de entrada e solta os arquivos de saída. É uma boa prática salvar os scripts, os dados, os gráficos (tudo referente à análise) num mesmo diretório.

```
# Depende do seu computador  
# setwd("~/Documents/CursoR")
```

```
getwd() # Se eu não souber onde estou
```

```
## [1] "/home/cristiane/github/GENt-esalq.github.io/cursor"
```

Operações básicas

O R é uma grande calculadora.

```
1+1.3 #Decimal definido com "."
```

```
## [1] 2.3
```

```
2*3
```

```
## [1] 6
```

```
2^3
```

```
## [1] 8
```

```
4/2
```

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

```
sqrt(4) #raiz quadrada
```

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

```
log(100, base = 10) #logarítimo na base 10
```

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

```
log(100) #logarítimo com base neperiana
```

```
## [1] 4.60517
```

```
# Resolvendo problema
```

```
((13+2+1.5)/3) + log(96, base = 4)
```

```
## [1] 8.792481
```

Lembrando que o que vem antes do parênteses é uma função, e, sendo uma função, existe um manual para ela dentro do R, acesse com:

```
# Pedindo ajuda sobre função do R  
?log
```

Operação com vetores

```
# Diferentes formas de criar um vetor  
c(1,3,2,5,2)
```

```
## [1] 1 3 2 5 2
```

```
1:10
```

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

```
seq(from=0, to=100, by=5)
```

```
## [1] 0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80
```

```
## [18] 85 90 95 100
```

```
# ou
```

```
seq(0,100,5) # Se você já souber a ordem dos argumentos da função
```

```
## [1] 0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80
```

```
## [18] 85 90 95 100
```

```
seq(from=4, to=30, by=3)
```

```
## [1] 4 7 10 13 16 19 22 25 28
```

```
rep(3:5, 2)
```

```
## [1] 3 4 5 3 4 5
```

```
# Operações
```

```
c(1,4,3,2)*2 # Multiplica todos os elementos por 2
```

```
## [1] 2 8 6 4
```

```
c(4,2,1,5)+c(5,2,6,1) # Soma 4+5, 2+2, 1+6 e assim por diante
```

```
## [1] 9 4 7 6
```

```
c(4,2,1,5)*c(5,2,6,1) # Multiplica 4*5, 2*2, 1*6 e assim por diante
```

```
## [1] 20 4 6 5
```

Criando objetos

```
x = c(30.1,30.4,40,30.2,30.6,40.1)
# ou
x <- c(30.1,30.4,40,30.2,30.6,40.1)

y = c(0.26,0.3,0.36,0.24,0.27,0.35)
```

Operações com os objetos

```
x*2
## [1] 60.2 60.8 80.0 60.4 61.2 80.2
x + y
## [1] 30.36 30.70 40.36 30.44 30.87 40.45
x*y
## [1] 7.826 9.120 14.400 7.248 8.262 14.035
z <- (x+y)/2
z
## [1] 15.180 15.350 20.180 15.220 15.435 20.225
# Aplicando algumas funções
sum(z) # soma dos valores de z
## [1] 101.59
mean(z) # média
## [1] 16.93167
var(z) # variância
## [1] 6.427507
```

Obtendo valores internos dos objetos por indexação

```
z[3] # elemento na terceira posição do vetor
## [1] 20.18
z[2:4]
## [1] 15.35 20.18 15.22
```

Para saber algumas características do objeto

```
str(z)
## num [1:6] 15.2 15.3 20.2 15.2 15.4 ...
```

Vetor de caracteres

```
clone <- c("GRA02", "URO01", "URO03", "GRA02", "GRA01", "URO01")
```

Vetor de fatores (ou variáveis categóricas)

```
clone_fator <- as.factor(clone)
str(clone_fator)
```

```
## Factor w/ 4 levels "GRA01","GRA02",...: 2 3 4 2 1 3
```

```
levels(clone_fator)
```

```
## [1] "GRA01" "GRA02" "URO01" "URO03"
```

```
length(clone_fator)
```

```
## [1] 6
```

Vetor lógico

```
logico <- x > 40
logico  # Os elementos são maiores que 40?
```

```
## [1] FALSE FALSE FALSE FALSE FALSE TRUE
```

```
# Indica a posição dos TRUE
```

```
which(logico) # Obtendo as posições dos elementos TRUE
```

```
## [1] 6
```

```
x[which(logico)] # Obtendo os números maiores que 40 do vetor x pela posição
```

```
## [1] 40.1
```

Para ficar esperto/a

```
(a <- 1:10)
```

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

```
b <- seq(from = 0.1, to = 1, 0.1)
```

```
(b <- b*10)
```

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

```
a==b # Existe um problema computacional de armazenamento
```

```
## [1] TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE TRUE TRUE
```

```
a==round(b) # Evitar que isso aconteça arredondando o resultado
```

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
?round      # Fiquei com dúvida nessa função  
errado <- c(TRUE, "vish", 1) # Não podemos misturar classes num mesmo vetor  
errado  
## [1] "TRUE" "vish" "1"
```