Introdução à Computação (MAC113 - FEA – 2020)

Ana C V de Melo

Atribuição, Tipos Compostos: Vetores e Listas

Nosso 1o. comando (instrução) R

A linguagem e suas instruções

- um *Script* (programa) é formado por uma sequência de instruções...
- toda linguagem de programação possui um repertório de instruções (visto anteriormente)
- não podemos escrever qq coisa, as instruções que escrevemos nos *Scripts* devem ser aquelas já definidas para a linguagem
- precisamos respeitar a forma como elas são escritas...
- vamos ver cada um dos comandos ao longo do curso...

Atribuição - comando que já estamos utilizando...

- Guarda valores em variáveis
- Variáveis: qq nome formado por letras, dígitos, "." e "_" (precisa começar com uma letra).

```
x <- 15
y = 22
z <- x + y
print(z)
```

```
## [1] 37
```

Problema: calcular média aritmética

- Dados 3 números inteiros calcular a média aritmética entre eles.
- Vamos resolver o problema em um script
  - Passo 1: planejar o que faremos no programa

```
# Programa
# Dados 3 números inteiros calcular a média aritmética

# lê os números

# calcula a media aritmética

# imprime o valor da média
```

Problema: calcular média aritmética

- Passo 2: fazer o programa

```
# Programa
# Dados 3 números inteiros calcular a média aritmética

# lê os números
a <- as.integer(readline("Digite o primeiro número: "))
b <- as.integer(readline("Digite o segundo número: "))
c <- as.integer(readline("Digite o terceiro número: "))

# calcula a media aritmética

# imprime o valor da média
```

Exemplo de Execução

```
Digite o primeiro número: 1
Digite o segundo número: 2
Digite o terceiro número: 3
```

Programa inteiro

```
# Programa
# Dados 3 números inteiros calcular a média aritmética

# lê os números
a <- as.integer(readline("Digite o primeiro número: "))
b <- as.integer(readline("Digite o segundo número: "))
c <- as.integer(readline("Digite o terceiro número: "))

# calcula a media aritmética

media <- (a + b + c) / 3

# imprime o valor da média

cat("Média dos Números:", media)
```

Exemplo de Execução...

```
Digite o primeiro número: 34
Digite o segundo número: 23
Digite o terceiro número: 56
Média dos Números: 37.66667
```

Planilhas: entrada/saída - revisão

Considere a planilha: "alunosnotas3.csv"

```
Nome,P1,P2,Sub
Alexandre ,3,5.5,8.5
Allan ,5,6,
Andre ,8,7.5,
Bernardo ,9,5.5,
Bruno,1.5,7.5,
Carlos ,5,4.5,6.5
Carolina ,10,10,
Claudio , ,3,
Daniel ,3,5,6
Denis, 3.8,3.5,
```

- as provas não realizadas ficam em branco, ex: Claudio, ,3, só fez a P2

Entrada: arquivo texto

- Leitura de dados .csv

```
# define o caminho do diretório de execução do programa
diretorio <- getwd()
setwd(diretorio)
print(diretorio)   #imprime o o caminho completo do diretório
```

```
## [1] "/Users/ana/Home/home-ana/Activities/teaching/courses/mac113-
```

Sobre o read.csv()

-fig comando

- veja no *Help*

Ler planilhas (.csv)

- definindo configurações no read.csv - *NA* (Not Available)

```
dados2 <- read.csv("alunosnotas3.csv",
                   header = TRUE, sep = ",", fill = TRUE)
print(dados2)     # preenche com NA o que está em branco
```

```
##          Nome   P1   P2 Sub
## 1  Alexandre   3.0  5.5 8.5
## 2      Allan   5.0  6.0  NA
## 3      Andre   8.0  7.5  NA
## 4   Bernardo   9.0  5.5  NA
## 5       Bruno  1.5  7.5  NA
## 6     Carlos   5.0  4.5 6.5
## 7   Carolina  10.0 10.0  NA
## 8    Claudio    NA  3.0  NA
## 9      Daniel  3.0  5.0 6.0
## 10      Denis  3.8  3.5  NA
```

Sobre o write.csv()

- fig do comando

Gravar planilhas (.csv)

```
write.csv(dados2, "nova.plan.csv")
temp <- read.csv("nova.plan.csv")
print(temp)
```

```
##     X       Nome   P1   P2 Sub
## 1   1 Alexandre   3.0  5.5 8.5
## 2   2     Allan   5.0  6.0  NA
## 3   3     Andre   8.0  7.5  NA
## 4   4  Bernardo   9.0  5.5  NA
## 5   5      Bruno  1.5  7.5  NA
## 6   6    Carlos   5.0  4.5 6.5
## 7   7  Carolina  10.0 10.0  NA
## 8   8   Claudio    NA  3.0  NA
## 9   9    Daniel  3.0  5.0 6.0
## 10 10     Denis  3.8  3.5  NA
```

Gravar planilhas sem identificação de linhas na gravação

```
write.csv(dados2, "nova.plan2.csv", row.names=FALSE)
temp <- read.csv("nova.plan2.csv")
print(temp)
```

```
##          Nome   P1   P2 Sub
## 1  Alexandre   3.0  5.5 8.5
## 2      Allan   5.0  6.0  NA
## 3      Andre   8.0  7.5  NA
## 4   Bernardo   9.0  5.5  NA
## 5       Bruno  1.5  7.5  NA
## 6     Carlos   5.0  4.5 6.5
## 7   Carolina  10.0 10.0  NA
## 8    Claudio    NA  3.0  NA
## 9      Daniel  3.0  5.0 6.0
## 10      Denis  3.8  3.5  NA
```

Como aparece na planilha

```
"","Nome","P1","P2","Sub"
"1","Alexandre ",3,5.5,8.5
"2","Allan ",5,6,NA
"3","Andre ",8,7.5,NA
"4","Bernardo ",9,5.5,NA
"5","Bruno",1.5,7.5,NA
"6","Carlos ",5,4.5,6.5
"7","Carolina ",10,10,NA
"8","Claudio ",NA,3,NA
"9","Daniel",3,5,6
"10","Denis",3.8,3.5,NA
```

Como aparece na planilha

```
"Nome","P1","P2","Sub"
"Alexandre ",3,5.5,8.5
"Allan ",5,6,NA
"Andre ",8,7.5,NA
"Bernardo ",9,5.5,NA
"Bruno",1.5,7.5,NA
"Carlos ",5,4.5,6.5
"Carolina ",10,10,NA
"Claudio ",NA,3,NA
"Daniel",3,5,6
"Denis",3.8,3.5,NA
```

- observe que não temos mais a numeração das linhas...
- outras formas de entrada e saída - ao longo do curso

Para que servem?

- já trabalhamos com valores inteiros, reais, ... eles possuem um único valor (atômico)
- vimos que podemos atribuir a uma variável um valor atômico
- mas na matemática usamos vetores, matrizes... eles possuem vários valores
- as planilhas que usamos também possuem vários valores...
- vamos aprender um pouco sobre como trabalhar com valores compostos (vários valores) ...
- aula de hoje: Vetores e Listas

Tipos Compostos

Vetores

Armazenamento - criar

```
v1 <- c(1,8,5)
print(v1)
```

```
## [1] 1 8 5
```

```
typeof(v1)
```

```
## [1] "double"
```

```
v2 <- c("casa", "das", "rosas")
print(v2)
```

```
## [1] "casa"  "das"   "rosas"
```

```
typeof(v2)
```

```
## [1] "character"
```

Criar vetores com operadores

```
v3 <- 1:15
print(v3)
```

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

```
v4 <- 4.8:12.6
print(v4)
```

```
## [1]  4.8  5.8  6.8  7.8  8.8  9.8 10.8 11.8
```

Criar vetores com operadores

```
v5 <- seq(1, 5, by= 0.5)  # by - incremento
print(v5)
```

```
## [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```
v6 <- seq(1, 5, length.out = 6) # número de elementos
print(v6)
```

```
## [1] 1.0 1.8 2.6 3.4 4.2 5.0
```

Acesso a 1 elemento do vetor

```
print(v4)
```

```
## [1]  4.8  5.8  6.8  7.8  8.8  9.8 10.8 11.8
```

```
v4[2]
```

```
## [1] 5.8
```

```
v4[17]
```

```
## [1] NA
```

Acesso a alguns elementos do vetor - seleção por índice

```
print(v4)
```

```
## [1]  4.8  5.8  6.8  7.8  8.8  9.8 10.8 11.8
```

```
v4[c(3,5,7)]
```

```
## [1]  6.8  8.8 10.8
```

```
v4[3:7]
```

```
## [1]  6.8  7.8  8.8  9.8 10.8
```

Acesso a alguns elementos do vetor - seleção por valor (subset)

```
print(v4)
```

```
## [1]  4.8  5.8  6.8  7.8  8.8  9.8 10.8 11.8
```

```
temp1 <- subset(v4, v4 <= 3)  #só os valores < ou = a 3
print(temp1)
```

```
## numeric(0)
```

```
temp2 <- subset(v4, v4 >  7)  #só os valores > 7
print(temp2)
```

```
## [1]  7.8  8.8  9.8 10.8 11.8
```

Adicionar 1 elemento ao vetor

```
print(v2)
```

```
## [1] "casa"  "das"   "rosas"
```

```
v2[4] <- "Av Paulista"
print(v2)
```

```
## [1] "casa"        "das"         "rosas"       "Av Paulista"
```

Modificar 1 elemento do vetor

```
print(v2)
```

```
## [1] "casa"        "das"         "rosas"       "Av Paulista"
```

```
v2[3] <- "Bruxas"
print(v2)
```

```
## [1] "casa"        "das"         "Bruxas"      "Av Paulista"
```

Remover todos os elementos do vetor

```
print(v2)
```

```
## [1] "casa"        "das"         "Bruxas"      "Av Paulista"
```

```
v2 <- NULL
print(v2)
```

```
## NULL
```

Algumas operações sobre vetores

```
print(v5)
```

```
## [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```
length(v5)   # num. de elementos do vetor
```

```
## [1] 9
```

```
min(v5)      # menor valor
```

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

Algumas operações sobre vetores

```
print(v5)
```

```
## [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```
max(v5)      # maior valor
```

```
## [1] 5
```

```
mean(v5)     # média dos valores
```

```
## [1] 3
```

- existem muitas outras operações...

Operações aritméticas

```
print(v1)
```

```
## [1] 1 8 5
```

```
print(v3)
```

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

```
v1+v3
```

```
##  [1]  2 10  8  5 13 11  8 16 14 11 19 17 14 22 20
```

```
v1-v3
```

```
##  [1]   0   6   2  -3   3  -1  -6   0  -4  -9  -3  -7 -12  -6 -10
```

Operações aritméticas

```
print(v1)
```

```
## [1] 1 8 5
```

```
print(v3)
```

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

```
v1*v3
```

```
##  [1]   1  16  15   4  40  30   7  64  45  10  88  60  13 1
```

```
v1/v3
```

```
##  [1] 1.00000000 4.00000000 1.66666667 0.25000000 1.6000000
##  [7] 0.14285714 1.00000000 0.55555556 0.10000000 0.7272727
## [13] 0.07692308 0.57142857 0.33333333
```

Dá para misturar tipos de dados?

```
v7 <- c(1, TRUE, "casa")
print(v7)
```

```
## [1] "1"    "TRUE" "casa"
```

```
typeof(v7)
```

```
## [1] "character"
```

- NÃO - todos são convertidos para o tipo mais amplo
- Precisamos de outro tipo de estrutura: Listas

Listas

Criar listas - a partir de valores

```
l <- list(1,8, 5:7)  # criação de uma lista
print(l)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 8
##
## [[3]]
## [1] 5 6 7
```

Criar listas - a partir de elementos existentes

```
v1 <- c(1,2,6,7:10)
v2 <- c(4,8,2)
l1 <- list(v1,v2)  # criação de uma lista
print(l1)          # a partir de vetores
```

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

Criar listas com tipos diferentes

- Listas - podemos misturar valores

```
l1 <- list("qualquer coisa",c(1:3))
print(l1)
```

```
## [[1]]
## [1] "qualquer coisa"
##
## [[2]]
## [1] 1 2 3
```

```
print(typeof(l1))
```

```
## [1] "list"
```

Listas - elementos com nomes

- nomes para cada um dos elementos na definição

```
l2 <- list("cel1"= v1, "cel2"= v2)
print(l2)
```

```
## $cel1
## [1]  1  2  6  7  8  9 10
##
## $cel2
## [1] 4 8 2
```

Listas - elementos com nomes

- nomes para cada um dos elementos após a definição (*names*)

```
l1 <- list("qualquer coisa",c(1:3))
names(l1) <- c("prim_elem","seg_elem")
print(l1)
```

```
## $prim_elem
## [1] "qualquer coisa"
##
## $seg_elem
## [1] 1 2 3
```

Acesso aos elementos da lista

```
print(l1[1])        # acesso ao primeiro elemento [1]
```

```
## $prim_elem
## [1] "qualquer coisa"
```

```
print(l1[[1]])      # acesso ao valor do primeiro elemento [
```

```
## [1] "qualquer coisa"
```

```
print(l1$prim_elem) #acesso ao valor pelo nome do elemento
```

```
## [1] "qualquer coisa"
```

Acesso a elementos da lista - subset

```
l2 <- list("qualquer coisa",c(1:3), c(20,12,5,39))
print(subset(l2, l2 != "qualquer coisa"))
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] 20 12  5 39
```

```
names(l2) <-c("a", "b", "c")
print(subset(l2$c, l2$c > 12))
```

```
## [1] 20 39
```

Adicionar um elementos à lista

```
l1$terc_elem <- list("z","b") #elemento é uma lista
print(l1)
```

```
## $prim_elem
## [1] "qualquer coisa"
##
## $seg_elem
## [1] 1 2 3
##
## $terc_elem
## $terc_elem[[1]]
## [1] "z"
##
## $terc_elem[[2]]
## [1] "b"
```

Modificar um elemento da lista

```
l1$terc_elem <- "ano" #elemento é uma lista
print(l1)
```

```
## $prim_elem
## [1] "qualquer coisa"
##
## $seg_elem
## [1] 1 2 3
##
## $terc_elem
## [1] "ano"
```

Remover elementos da lista

```
l1$terc_elem <- NULL
print(l1)
```

```
## $prim_elem
## [1] "qualquer coisa"
##
## $seg_elem
## [1] 1 2 3
```

Algumas operações sobre listas

```
l <- list("elem1"=c(1,2), "elem2"="z")
length(l)        # tamanho da lista
```

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

```
v10 <- unlist(l) # converte lista em vetor
print(v10)
```

```
## elem11 elem12  elem2
##    "1"    "2"    "z"
```

Algumas operações sobre listas

```
ltemp <- unname(l) # remove os nomes dos
print(ltemp)       # elementos da lista
```

```
## [[1]]
## [1] 1 2
##
## [[2]]
## [1] "z"
```