

# Universidade Federal do Ceará (UFC/Sobral)

## Aula 10 - Métodos Computacionais Aplicados

Prof. Weligton Gomes

16/10/2023

```
library(ISwR)

data(energy)
data(thuesen)
```

### Funções que facilitam a manipulação de dados no R

```
thue1 <- subset(thuesen, blood.glucose < 7)
thue1
```

```
##      blood.glucose short.velocity
## 6              5.3             1.49
## 11             6.7             1.25
## 12             5.2             1.19
## 15             6.7             1.52
## 17             4.2             1.12
## 22             4.9             1.03
```

```
thue2 <- transform(thuesen, log.gluc = log(blood.glucose))
thue2
```

```
##      blood.glucose short.velocity log.gluc
## 1              15.3             1.76 2.727853
## 2              10.8             1.34 2.379546
## 3               8.1             1.27 2.091864
## 4              19.5             1.47 2.970414
## 5               7.2             1.27 1.974081
## 6               5.3             1.49 1.667707
## 7               9.3             1.31 2.230014
## 8              11.1             1.09 2.406945
## 9               7.5             1.18 2.014903
## 10             12.2             1.22 2.501436
## 11              6.7             1.25 1.902108
## 12              5.2             1.19 1.648659
## 13             19.0             1.95 2.944439
## 14             15.1             1.28 2.714695
## 15              6.7             1.52 1.902108
## 16              8.6              NA 2.151762
## 17              4.2             1.12 1.435085
## 18             10.3             1.37 2.332144
## 19             12.5             1.19 2.525729
```

```
## 20      16.1      1.05 2.778819
## 21      13.3      1.32 2.587764
## 22       4.9      1.03 1.589235
## 23       8.8      1.12 2.174752
## 24       9.5      1.70 2.251292
```

```
thue3 <- within(thuesen,{
  log.gluc <- log(blood.glucose)
  m <- mean(log.gluc)
  centered.log.gluc <- log.gluc - m
  rm(m)
})
thue3
```

```
##      blood.glucose short.velocity centered.log.gluc log.gluc
## 1      15.3      1.76      0.481879807 2.727853
## 2      10.8      1.34      0.133573113 2.379546
## 3       8.1      1.27     -0.154108960 2.091864
## 4      19.5      1.47      0.724441444 2.970414
## 5       7.2      1.27     -0.271891996 1.974081
## 6       5.3      1.49     -0.578266201 1.667707
## 7       9.3      1.31     -0.015958621 2.230014
## 8      11.1      1.09      0.160972087 2.406945
## 9       7.5      1.18     -0.231070001 2.014903
## 10      12.2      1.22      0.255462930 2.501436
## 11       6.7      1.25     -0.343865495 1.902108
## 12       5.2      1.19     -0.597314396 1.648659
## 13      19.0      1.95      0.698465958 2.944439
## 14      15.1      1.28      0.468721722 2.714695
## 15       6.7      1.52     -0.343865495 1.902108
## 16       8.6      NA      -0.094210818 2.151762
## 17       4.2      1.12     -0.810888496 1.435085
## 18      10.3      1.37      0.086170874 2.332144
## 19      12.5      1.19      0.279755623 2.525729
## 20      16.1      1.05      0.532846250 2.778819
## 21      13.3      1.32      0.341791014 2.587764
## 22       4.9      1.03     -0.656737817 1.589235
## 23       8.8      1.12     -0.071221300 2.174752
## 24       9.5      1.70      0.005318777 2.251292
```

```
summary(thue3)
```

```
##      blood.glucose      short.velocity      centered.log.gluc      log.gluc
## Min.      : 4.200      Min.      :1.030      Min.      : -0.81089      Min.      :1.435
## 1st Qu.: 7.075      1st Qu.:1.185      1st Qu.: -0.28989      1st Qu.:1.956
## Median : 9.400      Median :1.270      Median : -0.00532      Median :2.241
## Mean   :10.300      Mean   :1.326      Mean    : 0.00000      Mean    :2.246
## 3rd Qu.:12.700      3rd Qu.:1.420      3rd Qu.: 0.29526      3rd Qu.:2.541
## Max.    :19.500      Max.    :1.950      Max.     : 0.72444      Max.     :2.970
##                                     NA's      :1
```

o R é uma linguagem que permite criar novas funções. Na verdade, muitas das funções em R são atualmente funções de funções. A estrutura de uma função é dada abaixo:

```
myfunction<-function(arg1, arg2,...){ statements ou afirmações return(object) }
```

O R é uma verdadeira linguagem de programação que permite a execução condicional e também construções

de loop. Por exemplo:

```
for(i in 1:5){  
  print(i)  
}
```

```
## [1] 1  
## [1] 2  
## [1] 3  
## [1] 4  
## [1] 5
```

```
# criando vetor de exemplo  
x <- 10:20
```

```
# divide cada elemento por 10  
for(i in seq_along(x))  
  x[i] <- x[i]/10
```

```
# resultado  
x
```

```
## [1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
```

```
y <- 12345  
x <- y/2  
while (abs(x*x-y) > 1e-10) x <- (x + y/x)/2  
x
```

```
## [1] 111.1081
```

```
x^2
```

```
## [1] 12345
```

Observe a construção da expressão while (condição), que diz que a expressão deve ser avaliada enquanto a condição for TRUE.

O teste ocorre na parte superior do loop

Uma variação do mesmo algoritmo com o teste na parte inferior do loop pode ser escrita com uma construção repetida:

```
y<-12345  
x <- y/2  
repeat{  
  x <- (x + y/x)/2  
  if (abs(x*x-y) < 1e-10) break  
}  
x
```

```
## [1] 111.1081
```

```
x^2
```

```
## [1] 12345
```

## Progressão Aritmética (PA) e Progressão Geométrica (PG)

Criando funções para calcular os n-ésimos termos e a soma de uma PA:

```
nesimotermo<-function(a1, n, r){
  a1+(n-1)*r
}
```

```
nesimotermo(107,101,6)
```

```
## [1] 707
```

```
termonpa<-function(a1, an, r){
  ((an-a1)/r)+1
}
```

```
termonpa(2,100,2)
```

```
## [1] 50
```

```
somapa<-function(a1, n, r){
  an=a1+(n-1)*r
  ((a1+an)*n)/2
}
```

```
somapa(2,5,2)
```

```
## [1] 30
```

## Progressão Geométrica (PG)

Criando funções para calcular os n-ésimos termos e a soma de uma PG finita e infinita:

### PG finita

```
nesimotermopg<-function(a1,q,n){
  a1*q^(n-1)
}
```

```
nesimotermopg(3,2,4)
```

```
## [1] 24
```

```
somapg<-function(a1, q, n){
  (a1*(q^n -1))/(q-1)
}
```

```
somapg(1,10,5)
```

```
## [1] 11111
```

### PG infinita

```
sompginf<-function(a1, q){
  a1/(1-q)
}
```

```
sompginf(1, 0.5)
```

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

## Exercícios de P.A e P.G

Questão 01: Qual é o centésimo primeiro termo de uma P.A. cujo primeiro termo é 107 e a razão é 6?

```
nesimotermo<-function(a1, n, r){  
  a1+(n-1)*r  
}  
  
nesimotermo(107,101,6)
```

```
## [1] 707
```

Questão 02: Sabendo que o primeiro termo é 10, o último é 109 e a razão é 3, basta usar a fórmula do termo geral para encontrar a posição do termo 109.

```
termonpa<-function(a1, an, r){  
  ((an-a1)/r)+1  
}  
termonpa(10,109,3)
```

```
## [1] 34
```

Questão 03: Determine o décimo quinto termo da progressão geométrica a seguir: (1, 3, 9, 27, . . .).

```
nesimotermopg<-function(a1,q,n){  
  a1*q^(n-1)  
}  
  
nesimotermopg(1,3,15)
```

```
## [1] 4782969
```

## Transformar a temperatura em grau Fahrenheit para grau Celsius.

Lembre-se que em grau Fahrenheit a escala de temperatura varia de  $32^{\circ}F$  ( $0^{\circ}C$ ) até  $212^{\circ}F$  ( $100^{\circ}C$ ).

```
fahrenheit_to_celsius <- function(temp_F) {  
  temp_C <- (temp_F - 32) * 5 / 9  
  return(temp_C)  
}  
  
fahrenheit_to_celsius(55)
```

```
## [1] 12.77778
```

```
fahrenheit_to_celsius(212)
```

```
## [1] 100
```

## Entrada de Dados no R.

Observação: No Windows o endereço do arquivo a \ deve ser substituída por /.

### Inserindo dados com o Data Entry

#### Arquivo no formato .txt (Bloco de Notas)

```
x401k <- read.delim("~/Base de Dados_MCA/401k.txt")
```

## Arquivo no formato .csv (Separador de Vírgula)

```
nerlove <- read.csv2("~/Base de Dados_MCA/nerlove.csv")
```

## Arquivo no formato .xlsx (Excel)

```
library(xlsx)
```

```
rental1 <- read.xlsx("~/Base de Dados_MCA/rental1.xlsx", 1)
```

## Arquivo no formato .dta (Stata)

```
library(haven)
```

```
# mydata <- read.dta("c:/mydata.dta")
```

```
campus <- read_dta("~/Base de Dados_MCA/campus.dta")
```

## Manipulação de Bases de Dados

```
library(readxl)
```

```
rental1 <- read_excel("~/Base de Dados_MCA/rental1.xlsx")
```

```
rental2 <- read_excel("~/Base de Dados_MCA/rental2.xlsx")
```

```
rental3 <- read_excel("~/Base de Dados_MCA/rental3.xlsx")
```

```
rental4 <- read_excel("~/Base de Dados_MCA/rental4.xlsx")
```

## A Função Merge

```
# mydata<-merge(mydata1, mydata2, by=c("id_1","id_2"))
```

```
rental12<-merge(rental1, rental2, by=c("city","year"))
```

## A Função Append

```
rental34<-rbind(rental3, rental4)
```