# Aula 04 - MCA_2021.1

Prof. Weligton Gomes

03/04/2023

### Instalação e Ativação de Pacotes

```
#install.packages("ISwR")
library(ISwR)
```

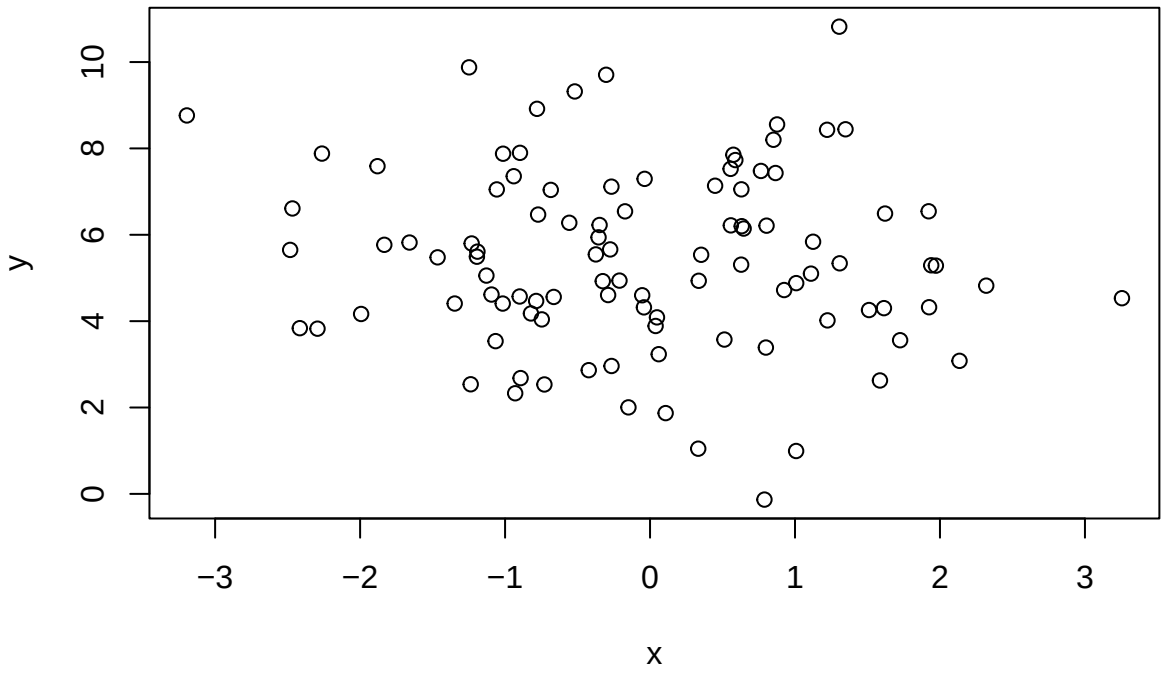
## Extrair números aleatórios de uma distribuição normal - rnorm()

Observação: r de random ou aleatório e norm - de distribuição normal

```
x<-rnorm(100)
y<-rnorm(100,mean = 5, sd = 2)
print(x)
```

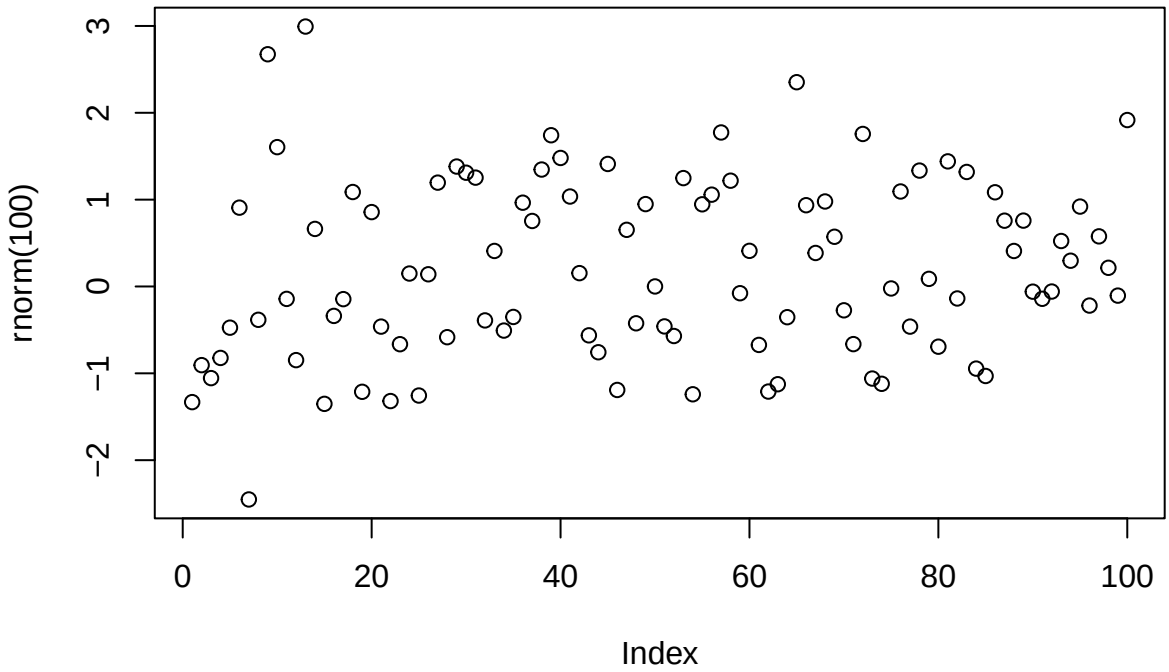
```
##   [1] -0.78559412  0.79915815 -1.24836921 -0.94025732 -0.14900718  0.80374784
##   [7]  0.58843477 -3.19510246 -0.89294969  0.86549991  1.11011579  0.55583800
##  [13]  0.35260833  3.25544173 -1.46533441 -1.34740092 -1.83299926 -2.29382299
##  [19] -0.21079755  0.78921925  1.30520143  1.30822289  0.63243609 -1.88014471
##  [25]  0.51302340  1.00775335 -2.46670141  0.33596604 -0.04209504  0.64566779
##  [31] -0.34808107  0.05999311 -0.68468088 -0.35521637  1.22393652 -1.23698465
##  [37]  1.61345946 -0.72863804  1.92199442 -0.82158603  1.62095436 -0.27490938
##  [43] -0.89712806 -1.99298775  1.72505876 -0.42272569  0.03846032  0.04783546
##  [49]  0.57474378  0.87635478  1.34840700 -0.05399493  1.22131692 -0.89909555
##  [55]  0.85134432  1.97177545 -0.37382583  0.55742370 -0.03728665  1.51066518
##  [61] -1.12878136 -0.51927489  0.92503507  1.93957814  2.13480666  1.00827381
##  [67] -0.17258186 -0.55696019 -0.77960385 -2.48244736 -1.01379333 -1.65913331
##  [73] -1.09357274 -1.05709538  1.12460605  0.76513821 -0.26641634  0.10768489
##  [79] -1.01612725 -1.23035996  2.31905170 -1.19413106 -2.26313526  0.62847086
##  [85] -0.32620418  1.92484263  1.58595613 -1.19054000  0.63009297 -0.30273775
##  [91] -1.06561922 -0.74672351 -0.77205765 -0.28922840 -0.93024582  0.44879981
##  [97]  0.33248616 -0.66437242 -0.26562627 -2.41581293
```

```
plot(x,y)
```




#Visualização de dados pelo comando View

```
View(rnorm(100))
plot(rnorm(100))
```




Os dois caracteres <- devem ser lidos como um único símbolo: uma seta apontando para a variável à qual o valor é atribuído. Isso é conhecido como operador de atribuição.

```
x <- 2
x + x
```

```
## [1] 4
```

#Operação com Vetores:

Um ponto forte do R é que ele pode manipular vetores de dados inteiros como objetos únicos. Um vetor de dados é simplesmente uma matriz de números e vetor pode ser construído assim:

```
weight <- c(60, 72, 57, 90, 95, 72)
height <- c(1.75, 1.80, 1.65, 1.90, 1.74, 1.91)
bmi <- weight/height^2
bmi
```

```
## [1] 19.59184 22.22222 20.93664 24.93075 31.37799 19.73630
```

```
mean(bmi)
```

```
## [1] 23.13262
```

```
summary(bmi)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   19.59   20.04   21.58   23.13   24.25   31.38
```

Considere, por exemplo, o cálculo da média e do desvio padrão da variável weight de forma manual

Média = Somatório de weight/número de observações

```
sum(weight)
```

```
## [1] 446
```

```
sum(weight)/length(weight)
```

```
## [1] 74.33333
```

Desvio padrão = A raíz quadrada da variância. Onde a variáncia é o somatório dos desvios médio quadrado / n-1

```
xbar <- sum(weight)/length(weight)
weight - xbar
```

```
## [1] -14.333333  -2.333333 -17.333333  15.666667  20.666667  -2.333333
```

```
(weight - xbar)^2
```

```
## [1] 205.444444   5.444444 300.444444 245.444444 427.111111   5.444444
```

```
sum((weight - xbar)^2)
```

```
## [1] 1189.333
```

```
sqrt(sum((weight - xbar)^2)/(length(weight) - 1))
```

```
## [1] 15.42293
```

De forma mais fácil e ágil, tem-se a função mean():

```
mean(weight)
```

```
## [1] 74.33333
```

```
sd(weight)
```

```
## [1] 15.42293
```

Como um exemplo um pouco mais complicado do que R pode fazer, considere o seguinte: A regra prática é que o IMC (BMI) para um indivíduo com peso normal deve estar entre 20 e 25, e queremos saber se nossos

dados se desviam sistematicamente deste. Você pode usar um teste t de uma amostra para avaliar se o IMC das seis pessoas pode ser considerado como tendo uma média de 22,5, dado que eles vêm de uma distribuição normal. Para isso, você pode usar a função t.test.

(Você pode não conhecer a teoria do teste t ainda. O exemplo é incluído aqui principalmente para dar alguma indicação de como é a saída estatística "real").

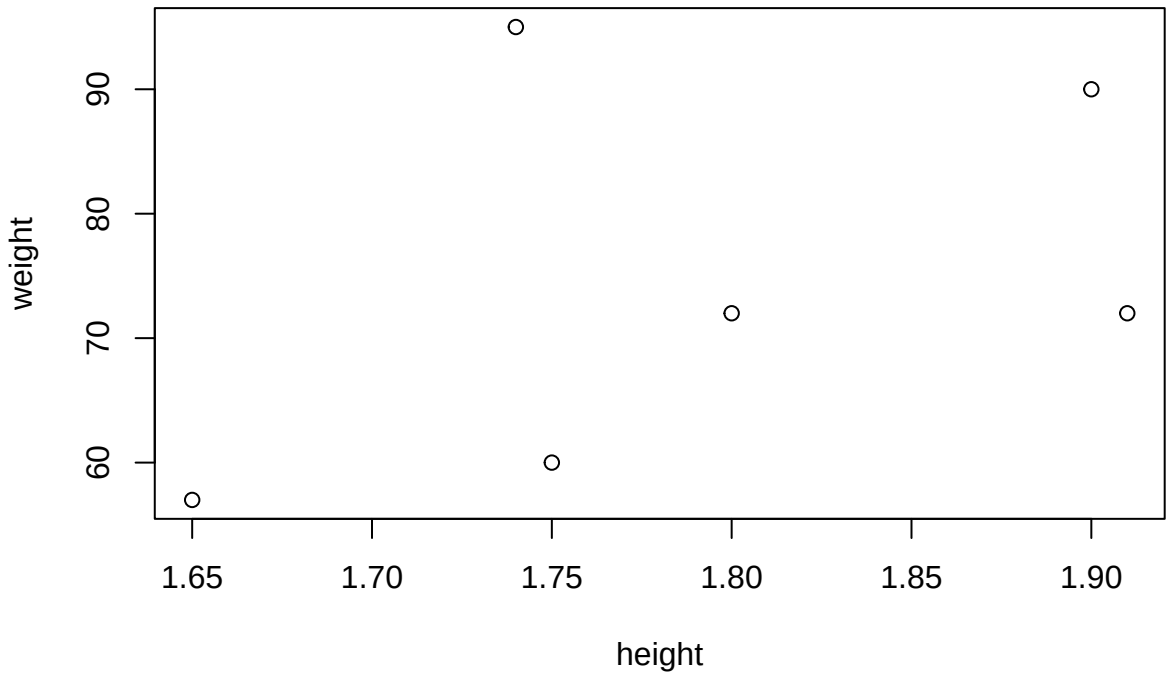
```
t.test(bmi, mu=22.5)
```

```
##
##  One Sample t-test
##
## data:  bmi
## t = 0.34488, df = 5, p-value = 0.7442
## alternative hypothesis: true mean is not equal to 22.5
## 95 percent confidence interval:
##  18.41734 27.84791
## sample estimates:
## mean of x
##  23.13262
```

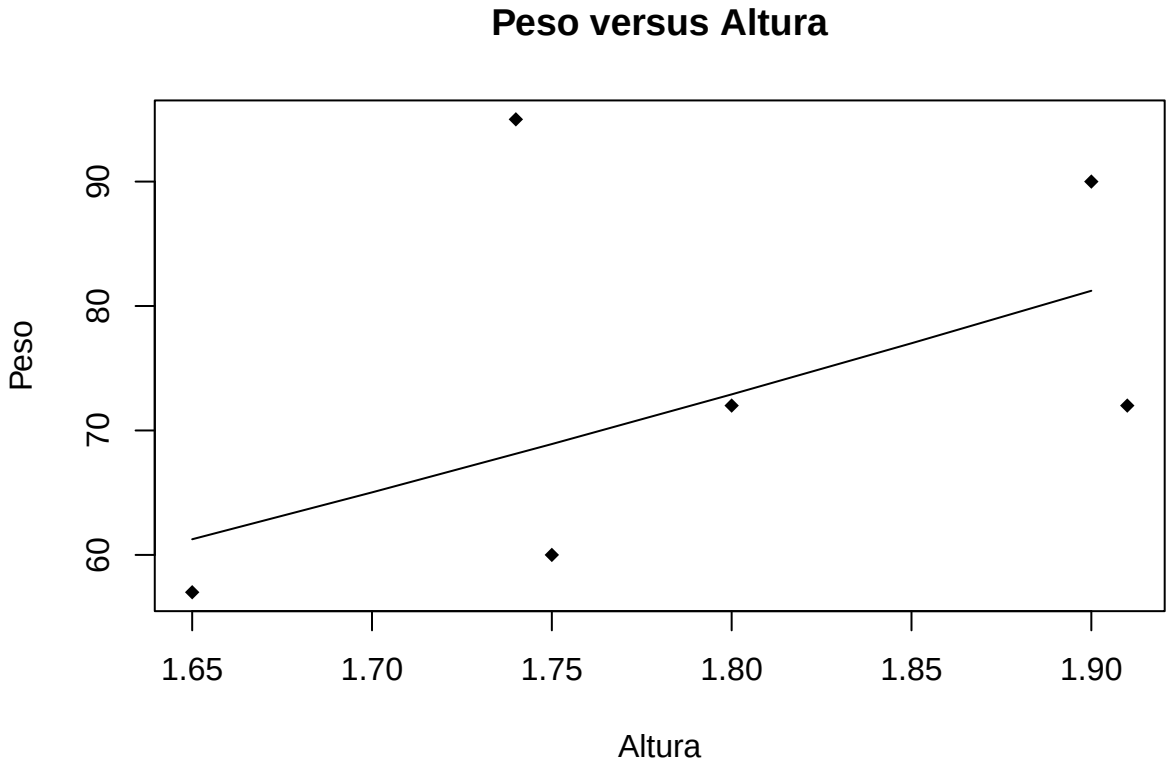
```
hh <- c(1.65, 1.70, 1.75, 1.80, 1.85, 1.90)
```

#Gráficos no R

```
plot(height,weight)
```




```
plot(height, weight, pch=18, main = "Peso versus Altura", xlab = "Altura", ylab = "Peso")
#pch = plotting Character (0:18)
lines(hh, 22.5 * hh^2)
```




```
args(plot.default)
```

```
## function (x, y = NULL, type = "p", xlim = NULL, ylim = NULL,
##     log = "", main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
##     ann = par("ann"), axes = TRUE, frame.plot = axes, panel.first = NULL,
##     panel.last = NULL, asp = NA, xgap.axis = NA, ygap.axis = NA,
##     ...)
## NULL
```

#Funções e Argumentos

Muitas coisas em R são feitas usando chamadas de função, comandos que se parecem com uma aplicação de uma função matemática de uma ou várias variáveis; por exemplo, log (x) ou plot (altura, peso).

```
args(plot.default)
```

```
## function (x, y = NULL, type = "p", xlim = NULL, ylim = NULL,
##     log = "", main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
##     ann = par("ann"), axes = TRUE, frame.plot = axes, panel.first = NULL,
##     panel.last = NULL, asp = NA, xgap.axis = NA, ygap.axis = NA,
##     ...)
## NULL
```

#Vetores:

Existem mais dois tipos, vetores de caracteres e vetores lógicos. Um vetor de caracteres é um vetor de strings de texto, cujos elementos são especificados e impressos entre aspas:

```
c("Huey","Dewey","Louie")
```

```
## [1] "Huey"  "Dewey" "Louie"
```

Não importa se você usa símbolos de aspas simples ou duplas, desde que a aspa esquerda seja igual à direita:

```
a<-c('Huey','Dewey','Louie')
c(T,T,F,T)
```

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

```
bmi > 25
```

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

Se você imprimir um vetor de caracteres, geralmente ele sairá com aspas adicionadas a cada elemento. Existe uma maneira de evitar isso, ou seja, usar a função cat Por exemplo,

```
cat(c("Huey","Dewey","Louie"))
```

```
## Huey Dewey Louie
```

#Valores Missing no R: NA

#Funções que criam vetores

Aqui, apresentamos três funções, c, seq e rep, que são usadas para criar vetores em várias situações. O primeiro deles, c, já foi introduzido. É a abreviação de "concatenar", juntando itens de ponta a ponta, que é exatamente o que a função faz:

```
c(42,57,12,39,1,3,4)
```

```
## [1] 42 57 12 39  1  3  4
```

```
x <- c(1, 2, 3)
y <- c(10, 20)
c(x, y, 5)
```

```
## [1]  1  2  3 10 20  5
```

A segunda função, seq ("sequência"), é usada para séries de números equidistantes.

```
seq(4,9)
```

```
## [1] 4 5 6 7 8 9
```

```
4:9
```

```
## [1] 4 5 6 7 8 9
```

Se você quiser uma sequência em saltos de 2.

```
seq(4,10,2)
```

```
## [1]  4  6  8 10
```

A terceira função, rep ("replicar"), é usada para gerar valores repetidos. Ele é usado em duas variantes, dependendo se o segundo argumento é um vetor ou um único número:

```
oops <- c(7,9,13)
rep(oops,3)
```

```
## [1]  7  9 13  7  9 13  7  9 13
```

```
rep(oops,1:3)
```

```
## [1]  7  9  9 13 13 13
```

```
rep(1:2,c(10,15))
```

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

```
rep(1:2,each=10) #Igual ao comando abaixo.
```

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

```
rep(1:2,c(10,10))
```

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

Em R, a noção de matriz é estendida a elementos de qualquer tipo, então você poderia ter, por exemplo, uma matriz de caracteres (string). Matrizes e arranjos são representadas como vetores com dimensões:

#Exemplo de matrizes nas notas de aula

```
x <- 1:12
dim(x) <- c(3,4)

y<-matrix(c(2,4,5,8,3,2,1,12,9),nrow=3,byrow=TRUE)

x <- matrix(1:12,nrow=3,byrow=T)
rownames(x) <- LETTERS[1:3]
colnames(x) <- LETTERS[4:7]
x
```

```
##   D  E  F  G
## A 1  2  3  4
## B 5  6  7  8
## C 9 10 11 12
```

```
colnames(x)<-c("Cajá","Melão", "Abacaxi", "Maçã")
x
```

```
##   Cajá Melão Abacaxi Maçã
## A    1     2       3    4
## B    5     6       7    8
## C    9    10      11   12
```

#Matriz Transposta

```
t(x)
```

```
##         A B  C
## Cajá    1 5  9
## Melão   2 6 10
## Abacaxi 3 7 11
## Maçã    4 8 12
```

#Inversa de Matriz

```
solve(y)
```

```
##             [,1]        [,2]        [,3]
## [1,]  0.01570681  0.12565445 -0.03664921
## [2,] -0.36649215  0.06806283  0.18848168
## [3,]  0.48691099 -0.10471204 -0.13612565
```

Você pode "colar" vetores ou juntar coluna ou linha, usando as funções cbind (coluna) e rbind (linha).

```
cbind(A=1:4,B=5:8,C=9:12)
```

```
##      A B  C
## [1,] 1 5  9
```

```
## [2,] 2 6 10
## [3,] 3 7 11
## [4,] 4 8 12
```

```
rbind(A=1:4,B=5:8,C=9:12)
```

```
##   [,1] [,2] [,3] [,4]
## A    1    2    3    4
## B    5    6    7    8
## C    9   10   11   12
```