Universidade Federal do Ceará (UFC/Sobral)

Aula 06 - Métodos Computacionais Aplicados

Prof. Weligton Gomes

18/09/2023

Instalação e Ativação de Pacotes

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

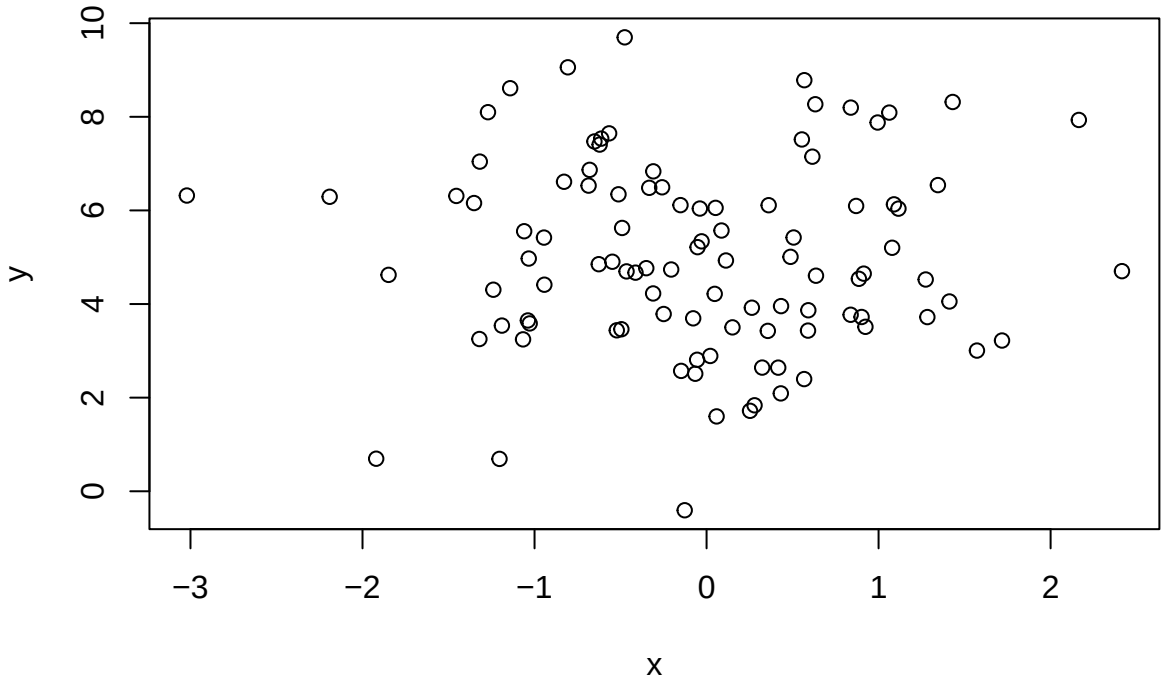
Extrair números aleatórios de uma distribuição normal - rnorm()

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

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

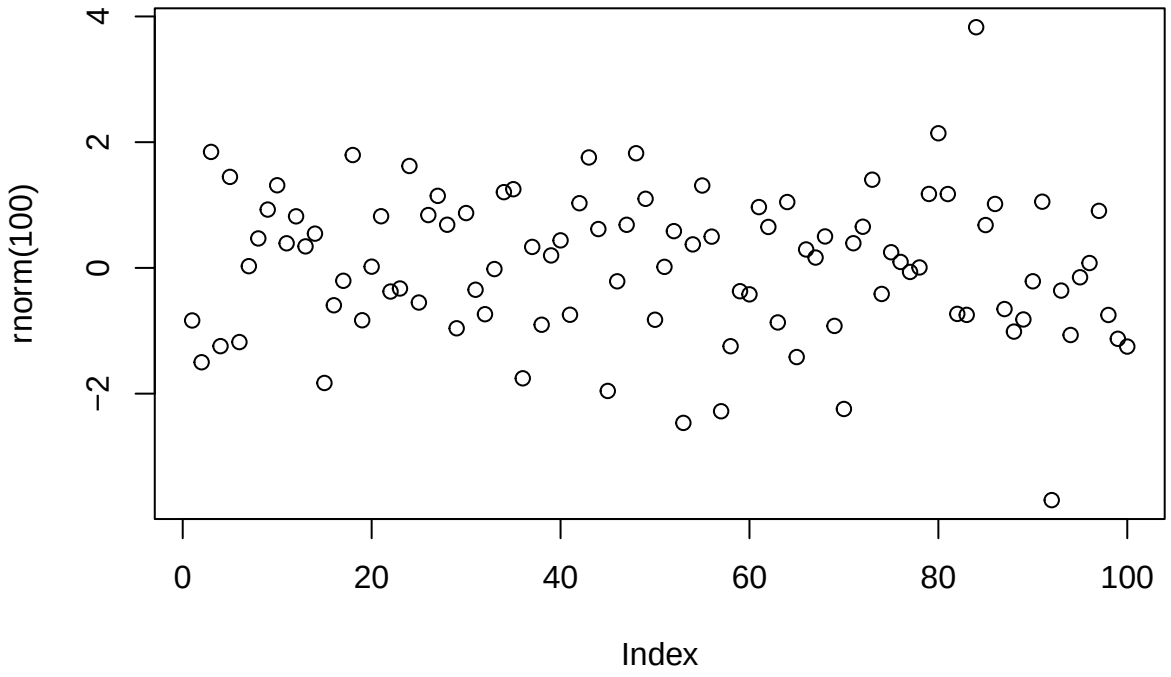
```
##   [1] -0.65192579 -0.25839784 -0.03900028 -0.30971450  1.43065398  0.48799731
##   [7]  0.63198890  0.05833844  0.27897749  1.28350103  0.83767485  0.02144243
##  [13] -0.31068578 -0.61226686 -1.27059663  0.61504982  0.90125004  0.91353118
##  [19]  0.86933764  0.26334882 -1.03874787 -3.02118301 -0.47591963 -0.05226447
##  [25] -1.14160699  0.56825996 -0.35049301 -0.67961263  0.05241692 -0.52189237
##  [31] -0.62142379 -0.14775806  0.59148718 -0.54760502  0.58974207 -1.23952029
##  [37]  0.92344687 -0.80621646 -0.56638873 -0.94291488  1.57172601  0.99454466
##  [43] -1.92055103  1.11542343 -1.03370055  1.27384014 -0.94515575 -0.12714567
##  [49]  0.83846146 -1.06034065 -0.24939801  2.41483995 -1.18979625  0.88556325
##  [55] -0.51200818  1.08942919 -1.02812460  0.43297632 -1.31822029  0.35610269
##  [61]  0.41660856  1.71752505 -0.06579881 -0.68582510 -0.49521716 -0.82849193
##  [67] -0.20562818  0.43157260  1.41185710 -1.06679951 -1.84851345 -0.02899313
##  [73] -0.33318977  0.04728532  1.06222539  0.15063665 -1.32050203 -0.05488179
##  [79]  1.07875931  0.25275128  0.36101848 -1.20369331 -0.41314037 -0.49084048
##  [85]  0.56729345  0.55411532 -1.35167621 -2.19107328  0.08677114 -0.62658075
##  [91]  0.32349324  1.34485857 -1.45468100 -0.46523092  0.50601296 -0.07781165
##  [97] -0.15159028  0.11252134  2.16410780  0.63603938
```

```
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(height)
```

```
## [1] 1.791667
```

```
sd(height)
```

```
## [1] 0.1002829
```

```
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)
```

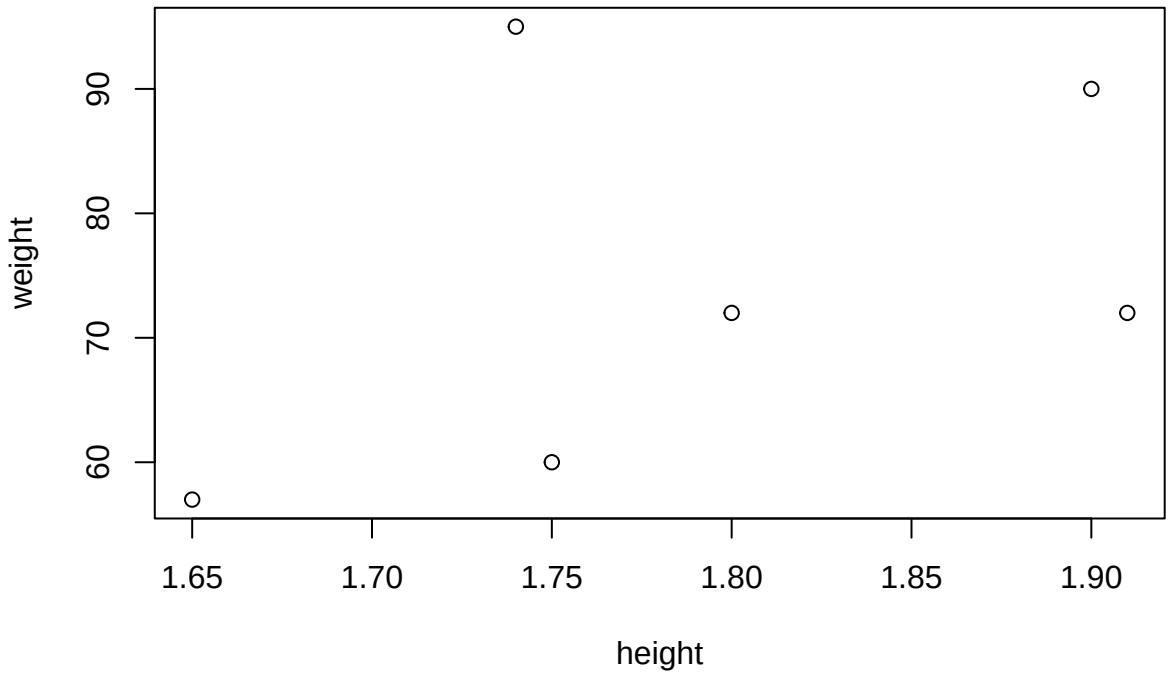
```
##
##  One Sample t-test
##
## data:  bmi
## t = 0.34488, df = 5, p-value = 0.7442
## alternative hypothesis: true mean is not equal to 22.5
## 95 percent confidence interval:
##  18.41734 27.84791
## sample estimates:
## mean of x
##  23.13262
```

O argumento $\mu = 22.5$ atribui um valor ao argumento formal mu, que representa a letra grega μ convencionalmente usada para a média teórica. Se isso não for fornecido, o teste t usaria o valor padrão $mu = 0$.

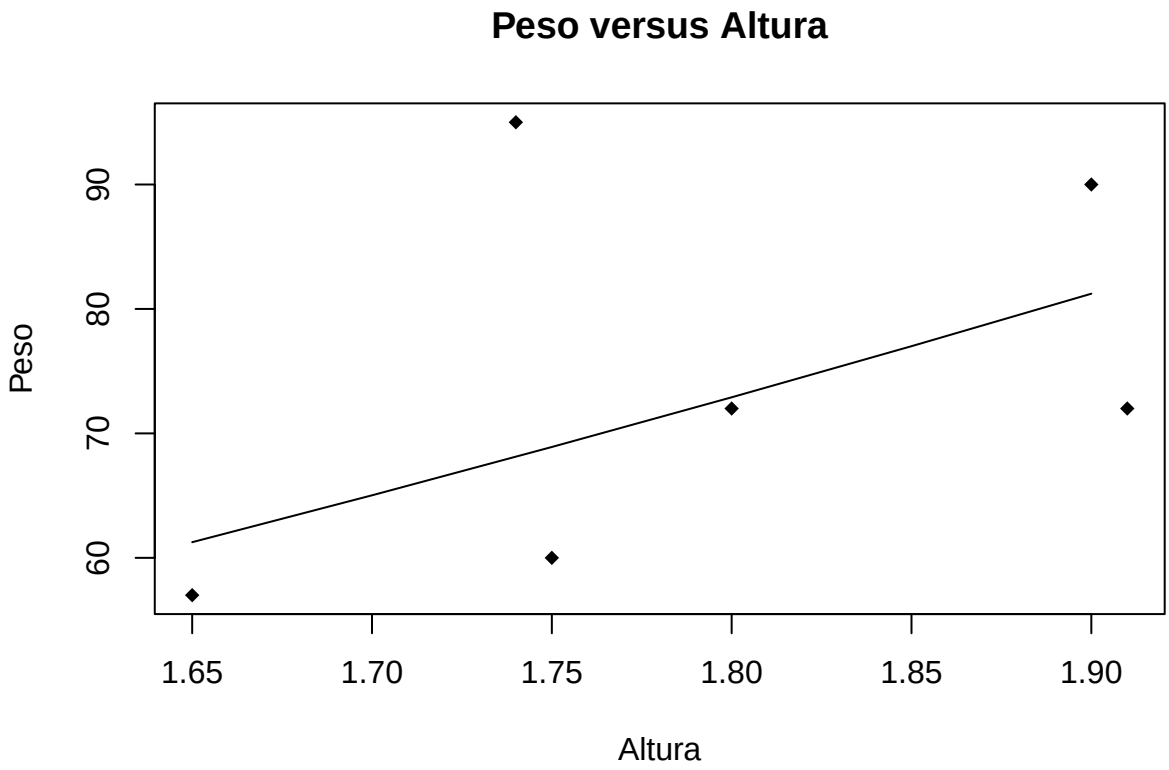
Para um teste como este, obtemos uma impressão mais extensa do que nos exemplos anteriores.Você pode se concentrar no valor de p, que é usado para testar a hipótese de que a média é $22,5$. O valor de p não é pequeno, indicando que não é improvável obter dados como os observados se a média fosse de fato $22,5$. (Falando de forma simplificada; na verdade, p é a probabilidade de obter um valor t maior que $0,3449$ ou menor que $-0,3449$). No entanto, você também pode observar o intervalo de confiança de 95% para a verdadeira média. Este intervalo é bastante amplo, indicando que realmente temos muito pouca informação sobre a verdadeira média.

Gráficos no R

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




```
plot(height, weight, pch=18, main = "Peso versus Altura", xlab = "Altura", ylab = "Peso")
#pch = plotting Character (0:18)
hh <- c(1.65, 1.70, 1.75, 1.80, 1.85, 1.90)
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
```

Observação: Valores Missing no R são representados por 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
```

```r
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.

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

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

```r
4:9
```

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

Se você quiser uma sequência em saltos de 2.

```r
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:

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

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

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

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

```r
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
```

```r
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
```

```r
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

```r
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
```

```
library(ISwR)
data(energy)
```

Fator

É comum em dados estatísticos ter variáveis categóricas, indicando alguma subdivisão dos dados, como classe social, etc. Essas são inseridas por meio de um código numérico. Essas variáveis devem ser especificadas como fatores em R.

```
pain <- c(0,3,2,2,1)
fpain <- factor(pain,levels=0:3)
levels(fpain) <- c("none","mild","medium","severe")

fpain
```

```
## [1] none   severe medium medium mild
## Levels: none mild medium severe
```

```
as.numeric(fpain)
```

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

```
levels(fpain)
```

```
## [1] "none"   "mild"   "medium" "severe"
```

Listas

Às vezes, é útil combinar uma coleção de objetos em um objeto composto maior. Isso pode ser feito usando listas.

function list

intake = ingestão

```
intake.pre <- c(5260,5470,5640,6180,6390,6515,6805,7515,7515,8230,8770)
intake.post <- c(3910,4220,3885,5160,5645,4680,5265,5975,6790,6900,7335)
length(intake.pre)
```

```
## [1] 11
```

```
length(intake.post)
```

```
## [1] 11
```

Para combinar esses vetores individuais em uma lista, você pode dizer

```
mylist <- list(before=intake.pre,after=intake.post)
mylist
```

```
## $before
##  [1] 5260 5470 5640 6180 6390 6515 6805 7515 7515 8230 8770
##
## $after
##  [1] 3910 4220 3885 5160 5645 4680 5265 5975 6790 6900 7335
```

```
mylist$after
```

```
##  [1] 3910 4220 3885 5160 5645 4680 5265 5975 6790 6900 7335
```

Dataframe

Um quadro de dados corresponde ao que outros pacotes estatísticos chamam de "matriz de dados" ou "conjunto de dados".

É uma lista de vetores e / ou fatores do mesmo comprimento que estão relacionados "transversalmente" de forma que os dados na mesma posição venham da mesma unidade experimental (sujeito, animal, etc.). Além disso, ele possui um conjunto exclusivo de nomes de linhas.

```
d <- data.frame(intake.pre,intake.post)
d$intake.pre
```

```
##  [1] 5260 5470 5640 6180 6390 6515 6805 7515 7515 8230 8770
```

Indexação: Caso de um ou mais índices

Se você precisa de um elemento específico em um vetor, por exemplo, o indivíduo de número 5, você pode fazer:

```
intake.pre[5] #Caso de um elemento
```

```
## [1] 6390
```

```
intake.pre[5]<-6395
intake.pre[c(3,5,7)] #Caso de mais de um elemento do vetor
```

```
## [1] 5640 6395 6805
```

ou de outra forma;

```
v <- c(3,5,7)
intake.pre[v]
```

```
## [1] 5640 6395 6805
```

```
intake.pre[1:5] #No caso de uma sequência de elementos
```

```
## [1] 5260 5470 5640 6180 6395
```

```
intake.pre[-c(3,5,7)]
```

```
## [1] 5260 5470 6180 6515 7515 7515 8230 8770
```

```
#Apresenta todas as informações, exceto aquelas
#com a indexação ou número 3, 5 e 7.

intake.pre1<-intake.pre[-c(3,5,7)]
#Criação de um vetor excluindo alguns elementos do vetor original.
```