

Mergulhando no R & K³ Parte 1

Automatizar, nada mais do que um processo de várias etapas, organizado em sequências de ações ou dados processados em 'lote', agrupando as partes que necessitam serem repetidas. Para realizar essa ação, as linguagens utilizam os loops, ciclos de repetição ou interação.

De uma maneira geral, existem dois tipos dessas construções especiais ou loops. Alguns loops são executados por um determinado número de vezes, controlado por um contador ou índice, sendo incrementado a cada ciclo de iteração. Podemos dizer neste caso, que fazem parte da família de loops, o **for**, por exemplo.

Mas alguns loops são baseados no início e na verificação de uma condição lógica. A condição é testada no início ou no final da construção do loop. Essas variações pertencem à **while** ou **repeat**

Vamos trazer isto para a base da linguagem R, os comandos de fluxo de controle, nas construções de loop, são:

FOR, WHILE, REPEAT, BREAK e NEXT.

Expressão genérica:

```
for (var in seq) {  
  expr  
}
```

Script em R:

```
cities <- c('Bahia','São Paulo','Americana')  
cities  
for(i in cities){  
  print(i)  
}
```

Resultado:

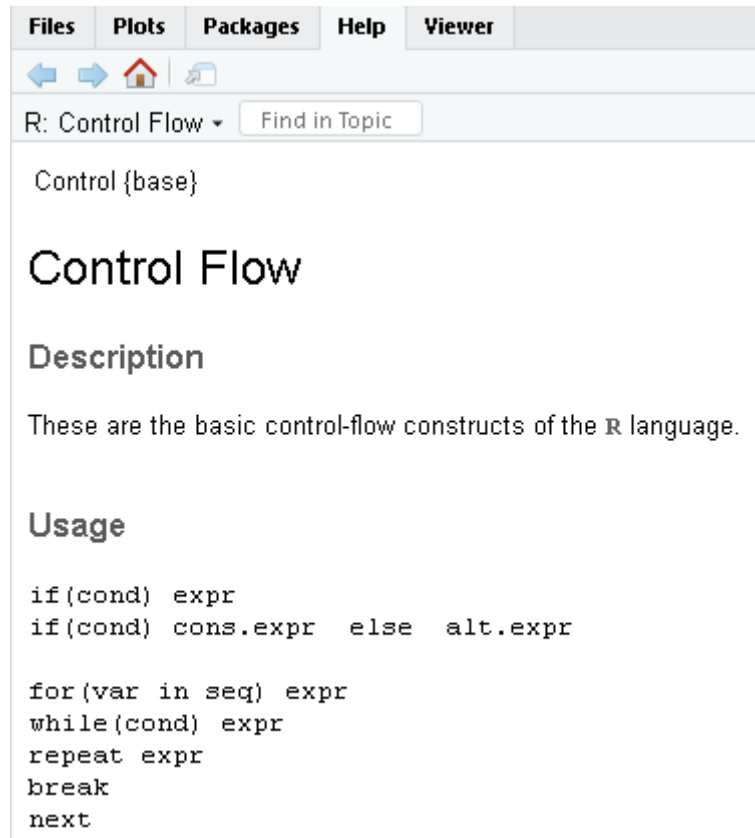
```
[1] "Bahia"  
[1] "São Paulo"  
[1] "Americana"
```

Então existem comandos de controle que servem de tomada de decisão de uma certa maneira, automatizam a decisão.

Mergulhando no R & K³ Parte 1

Vamos olhar o comando a seguir:

? Control



Exemplo de if simples:

Vamos criar uma matriz de 30 linha por 30 colunas, vazia. Veja o script a seguir.

```
mymat <- matrix(nrow=30, ncol=30)
mymat
for(i in 1:dim(mymat)[1]) {
  for(j in 1:dim(mymat)[2]) {
    mymat[i,j] = i*j
  }
}
mymat[1:20, 1:20]
```

Execute linha a linha com run

```
> mymat <- matrix(nrow=30, ncol=30)
> mymat
```

Mergulhando no R & K³ Parte 1

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]
[1,]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[2,]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[3,]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[4,]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[5,]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[6,]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[7,]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[8,]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[9,]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[10,]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[11,]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[12,]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[13,]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[14,]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
[15,]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
...												

Depois executa a função for, multiplicará cada linh para cada uma das colunas conforme o índice i (linha) e j(coluna) da matríz de 30 x 30

```
> mymat[1:10, 1:10]
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]
[1,]	1	2	3	4	5	6	7	8	9	10
[2,]	2	4	6	8	10	12	14	16	18	20
[3,]	3	6	9	12	15	18	21	24	27	30
[4,]	4	8	12	16	20	24	28	32	36	40
[5,]	5	10	15	20	25	30	35	40	45	50
[6,]	6	12	18	24	30	36	42	48	54	60
[7,]	7	14	21	28	35	42	49	56	63	70
[8,]	8	16	24	32	40	48	56	64	72	80
[9,]	9	18	27	36	45	54	63	72	81	90
[10,]	10	20	30	40	50	60	70	80	90	100

Se você alterar a exibição de mymat para 20 linhas e 20 coulnas, a exibição ficará assim:

```
> mymat[1:20, 1:20]
```

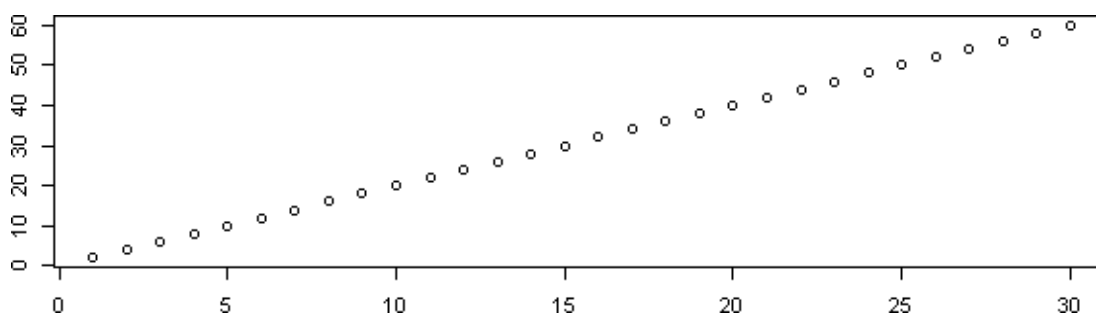
	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]
[1,]	1	2	3	4	5	6	7	8	9	10	11	12
[2,]	2	4	6	8	10	12	14	16	18	20	22	24
[3,]	3	6	9	12	15	18	21	24	27	30	33	36
[4,]	4	8	12	16	20	24	28	32	36	40	44	48
[5,]	5	10	15	20	25	30	35	40	45	50	55	60
[6,]	6	12	18	24	30	36	42	48	54	60	66	72
[7,]	7	14	21	28	35	42	49	56	63	70	77	84
[8,]	8	16	24	32	40	48	56	64	72	80	88	96
[9,]	9	18	27	36	45	54	63	72	81	90	99	108

Mergulhando no R & K³ Parte 1

```
[10,] 10 20 30 40 50 60 70 80 90 100 110 120
[11,] 11 22 33 44 55 66 77 88 99 110 121 132
[12,] 12 24 36 48 60 72 84 96 108 120 132 144
[13,] 13 26 39 52 65 78 91 104 117 130 143 156
[14,] 14 28 42 56 70 84 98 112 126 140 154 168
[15,] 15 30 45 60 75 90 105 120 135 150 165 180
[16,] 16 32 48 64 80 96 112 128 144 160 176 192
[17,] 17 34 51 68 85 102 119 136 153 170 187 204
[18,] 18 36 54 72 90 108 126 144 162 180 198 216
[19,] 19 38 57 76 95 114 133 152 171 190 209 228
[20,] 20 40 60 80 100 120 140 160 180 200 220 240
      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20]
[1,] 13 14 15 16 17 18 19 20
[2,] 26 28 30 32 34 36 38 40
[3,] 39 42 45 48 51 54 57 60
[4,] 52 56 60 64 68 72 76 80
[5,] 65 70 75 80 85 90 95 100
[6,] 78 84 90 96 102 108 114 120
[7,] 91 98 105 112 119 126 133 140
[8,] 104 112 120 128 136 144 152 160
[9,] 117 126 135 144 153 162 171 180
[10,] 130 140 150 160 170 180 190 200
[11,] 143 154 165 176 187 198 209 220
[12,] 156 168 180 192 204 216 228 240
[13,] 169 182 195 208 221 234 247 260
[14,] 182 196 210 224 238 252 266 280
[15,] 195 210 225 240 255 270 285 300
[16,] 208 224 240 256 272 288 304 320
[17,] 221 238 255 272 289 306 323 340
[18,] 234 252 270 288 306 324 342 360
[19,] 247 266 285 304 323 342 361 380
[20,] 260 280 300 320 340 360 380 400
```

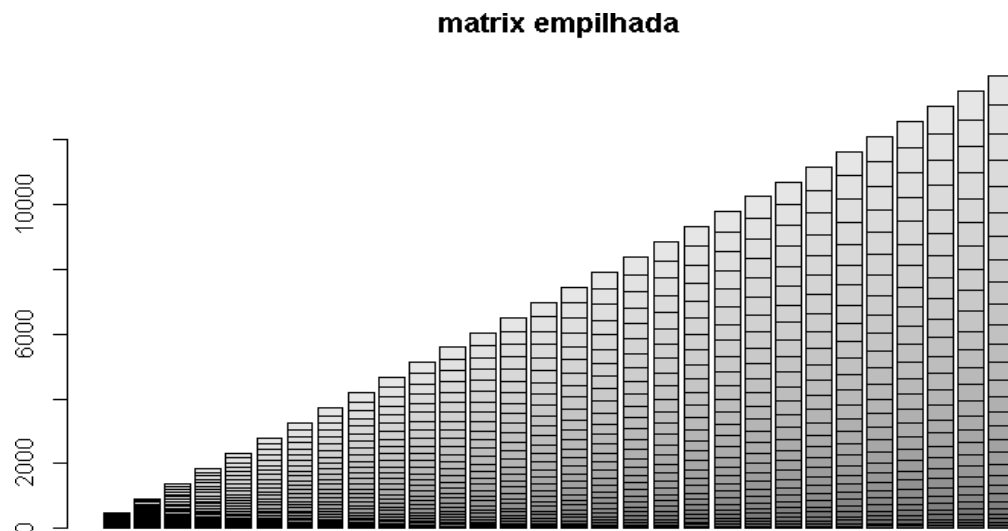
Vamos exibir os dados em forma de representação gráfica, apenas vamos “plotar” cada dado em um plano 2D.

```
plot(mymat)
```



Mergulhando no R & K³ Parte 1

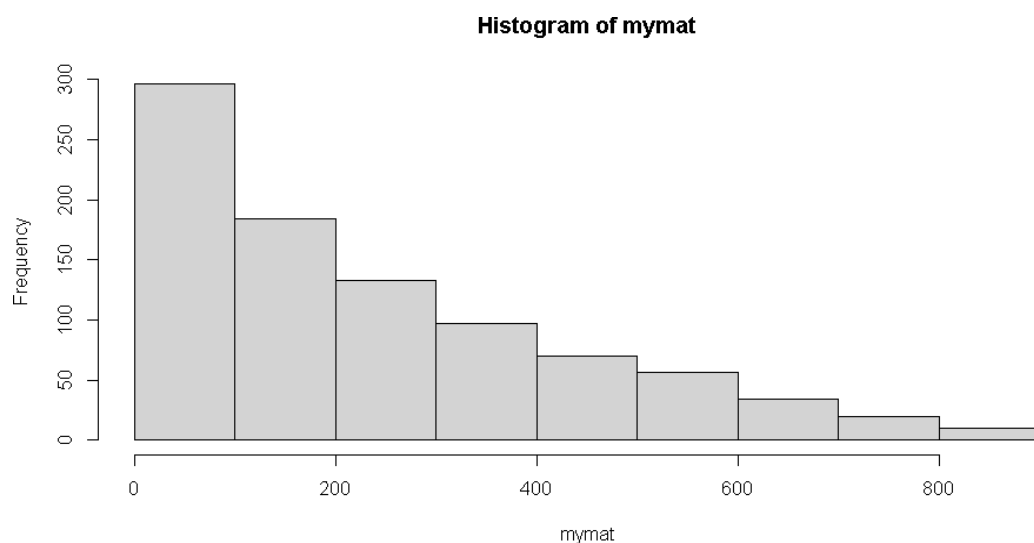
```
barplot(mymat, main="matrix empilhada", horiz=FALSE)
```



Para você ter ideia do empilhamento, faça:

```
pilhamaior<-sum(mymat[30,])  
> pilhamaior  
[1] 13950  
> pilhamaior<-sum(mymat[10,])  
> pilhamaior  
[1] 4650
```

```
hist(mymat)
```



Mergulhando no R & K³ Parte 1

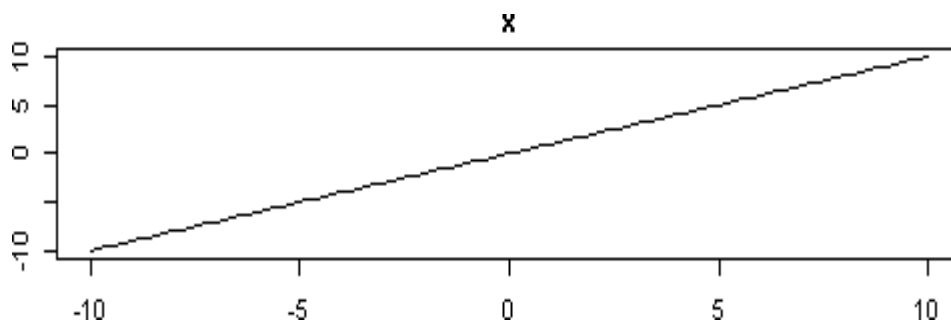
Ao analisar este gráfico, qual modelo matemático ele pertence, ou melhor, que modelo matemático explicaria este conjunto de dados?

Resposta: Dada uma função de 1º grau, em que $f(x) = x + b$

Onde b , é o parâmetro constante da curva, neste caso linear.

Faça o teste em R:

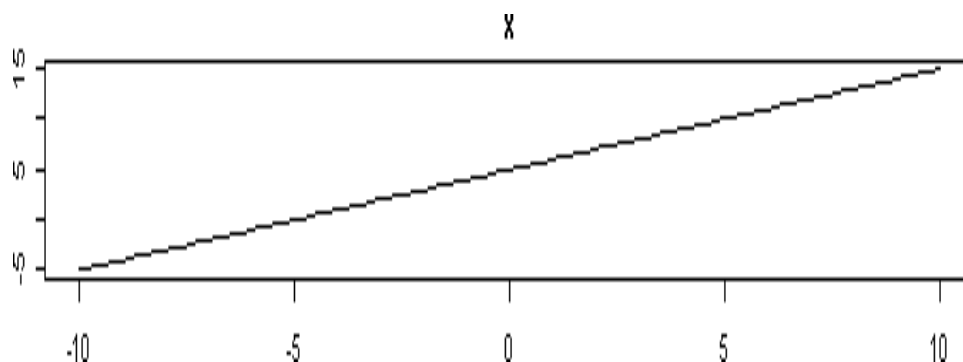
```
curve(x+0, from = -10, to = 10, main = "x")
```



Temos como resultado o gráfico de uma curva em linha reta, que varia de -10 a 10, cuja inclinação é igual a zero.

Vamos alterar a função $f(x) = x + b$ para $b = 5$

```
curve(x+5, from = -10, to = 10, main = "x")
```



Perceba o deslocamento da função pela adição de uma constante

Vamos estudar o controle while

```
while (test_expression)
{
  declaração
}
```

Mergulhando no R & K³ Parte 1

```
i <- 1  
while (i < 6) {  
  print (i)  
  i = i+1  
}
```

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

Agora vamos analisar um pequeno trecho de código para controle de repetição, que é usado para iteragir sobre o bloco de código várias vezes.

```
repeat {  
  declaração  
}
```

```
x <- 1  
repeat {  
  print (x)  
  x = x+1  
  if (x == 6){  
    break  
  }  
}
```

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

Vamos ver a declaração break,

Uma instrução break é usada dentro de um loop (repeat , for , while) para interromper as iterações e fluir o controle para fora do loop.

```
if (test_expression) {  
  break  
}
```

Mergulhando no R & K³ Parte 1

```
x <- 1:5  
for (val in x) {  
  if (val == 3) {  
    break  
  }  
  print(val)  
}
```

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

Vamos ver a declaração next,

Uma instrução next é útil quando queremos pular a iteração atual de um loop sem encerrá-la. Ao encontrar next, o analisador R pula avaliações adicionais e inicia a próxima iteração do loop.

```
if (test_expression) {  
  next  
}
```

```
x <- 1:5  
for (val in x) {  
  if (val == 3) {  
    next  
  }  
  print(val)  
}
```

```
[1] 1  
[1] 2  
[1] 4  
[1] 5
```

SÓ PRA LEMBRAR:

TUDO SÃO DADOS, ESTRUTURADOS, NÃO ESTRUTURADOS

OS DADOS SÃO REPRESENTADOS PELOS MODELOS MATEMÁTICOS

NÃO EXISTE MATEMÁTICA SEM DADOS

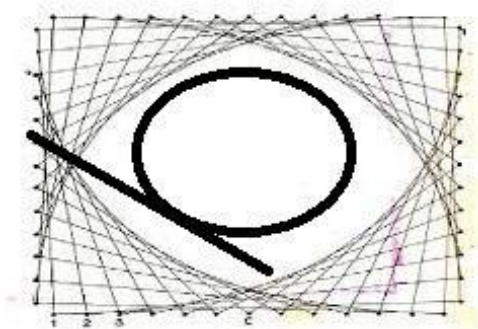
**PRIMEIRO VEM OS DADOS DEPOIS USAMOS ALGUM MODELO MATEMÁTICO PARA
EXPLICAR ESSES DADOS**

Mergulhando no R & K³ Parte 1

Então vamos lá, já que tudo são dados. Os conjuntos de dados são curvas.

As curvas mais simples são as retas. Cujas grau de curvatura $k=0$.

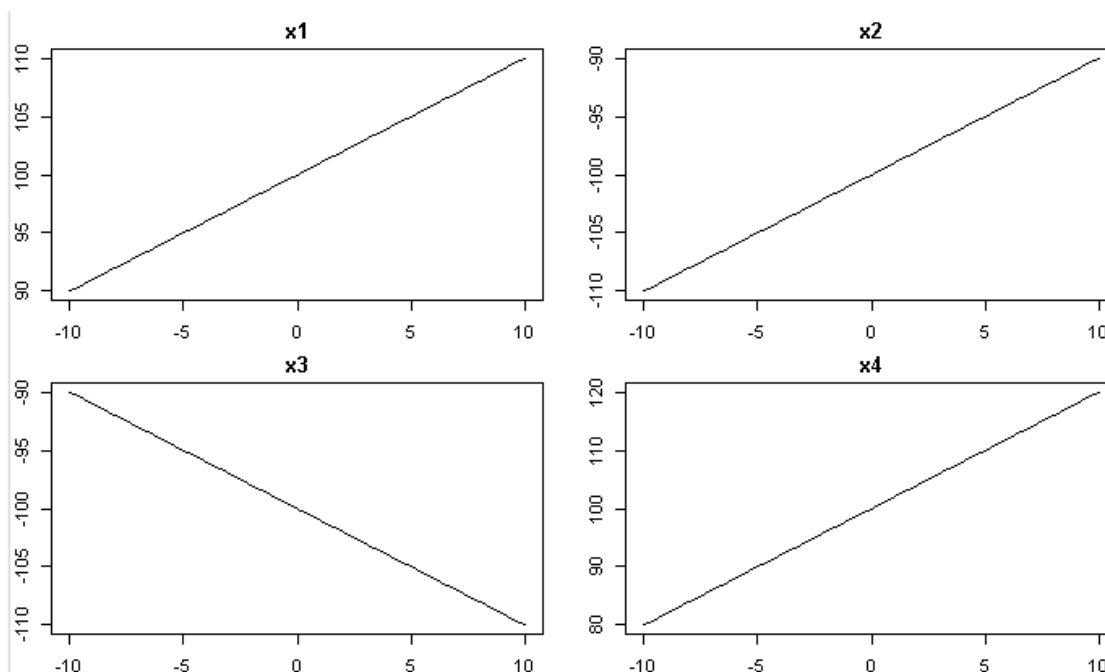
Matematicamente falando, curva é um conjunto unidimensional de pontos, alinhados ou não, que só tem comprimento e não espessura ou largura. As curvas, ou linhas, podem ser suaves ou quebradas, retas ou encurvadas, planas ou reversas.



Adaptado de Reis,L (2012) - Retas que forma curvas

Agora faça,

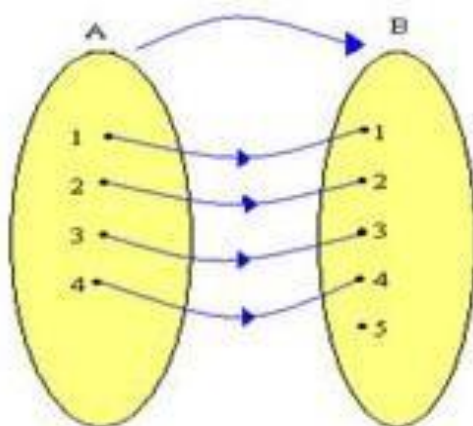
```
par(mfrow=c(4,2), mar=c(2,2,2,2))
curve(x+100, from = -10, to = 10, main = "x1")
curve(x-100, from = -10, to = 10, main = "x2")
curve(-x-100, from = -10, to = 10, main = "x3")
curve(2*x+100, from = -10, to = 10, main = "x4")
```



Mergulhando no R & K³ Parte 1

Funções

Uma relação estabelecida entre dois conjuntos A e B, onde exista uma associação entre cada elemento de A com um único de B através de uma lei de formação é considerada uma função.



Tudo é função, aquilo que rege uma formação de dados é um modelo ou função.

Paradinha do conhecimento, um exemplo de função auxiliar

A linguagem R e demais linguagens são ricas em funções. Modelos pré estabelecidos que auxiliam a relação entre os dados brutos e os modelos que os explicam. Além disso várias funções auxiliares ajudam e integração de diferentes modelos.

par(mfrow) → Um vetor de comprimento 2, onde o primeiro argumento especifica o número de linhas e o segundo o número de colunas dos gráficos.

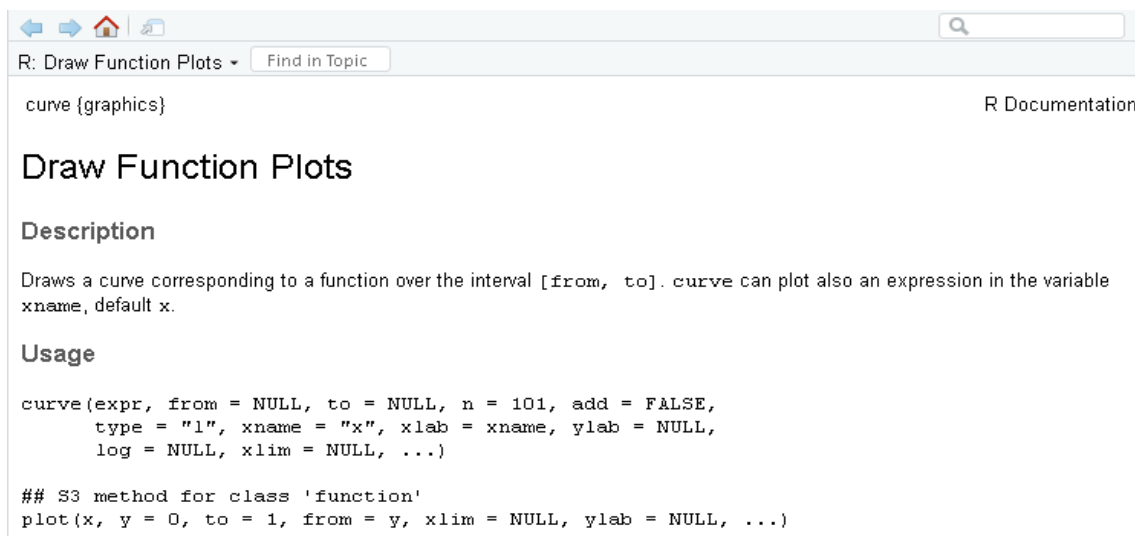
Esta função cria uma janela de plotagem com vários painéis. A função par (mfrow) é útil para criar um gráfico simples com vários painéis, enquanto o layout deve ser usado para gráficos de painel personalizados de tamanhos variados.

par(mfrow=c(5,2)) → 5 colunas e duas linhas

mar → margens entre os gráficos plotados

Mergulhando no R & K³ Parte 1

Em linguagem R, existe a função curve



The screenshot shows the R Documentation page for the 'curve' function. The page title is 'Draw Function Plots'. The description states: 'Draws a curve corresponding to a function over the interval [from, to]. curve can plot also an expression in the variable xname, default x.' The usage section shows the following code:

```
curve(expr, from = NULL, to = NULL, n = 101, add = FALSE,
      type = "l", xname = "x", xlab = xname, ylab = NULL,
      log = NULL, xlim = NULL, ...)

## S3 method for class 'function'
plot(x, y = 0, to = 1, from = y, xlim = NULL, ylab = NULL, ...)
```

O que é uma função?

O conceito de função surge de maneira natural e espontânea toda vez que consideramos duas grandezas que estejam relacionadas entre si, de maneira que cada valor de uma delas corresponde a um único valor da outra. Assim, relações funcionais ocorrem em todos os ramos do conhecimento: o salário varia com o número de horas trabalhadas, a receita varia com o número de artigos vendidos, a velocidade e a aceleração de um móvel variam com o tempo. Em Ciências Biológicas, também é bastante conveniente expressar este tipo de correspondência entre quantidades variáveis, como se pode observar nos exemplos seguintes: o tamanho de uma população varia com o tempo, a taxa de crescimento de um tecido canceroso relaciona-se com o efeito do tratamento radioativo e a amplitude dos impulsos elétricos gerados no músculo cardíaco também é função do tempo (Santiago & Paiva, 2015, on line) .

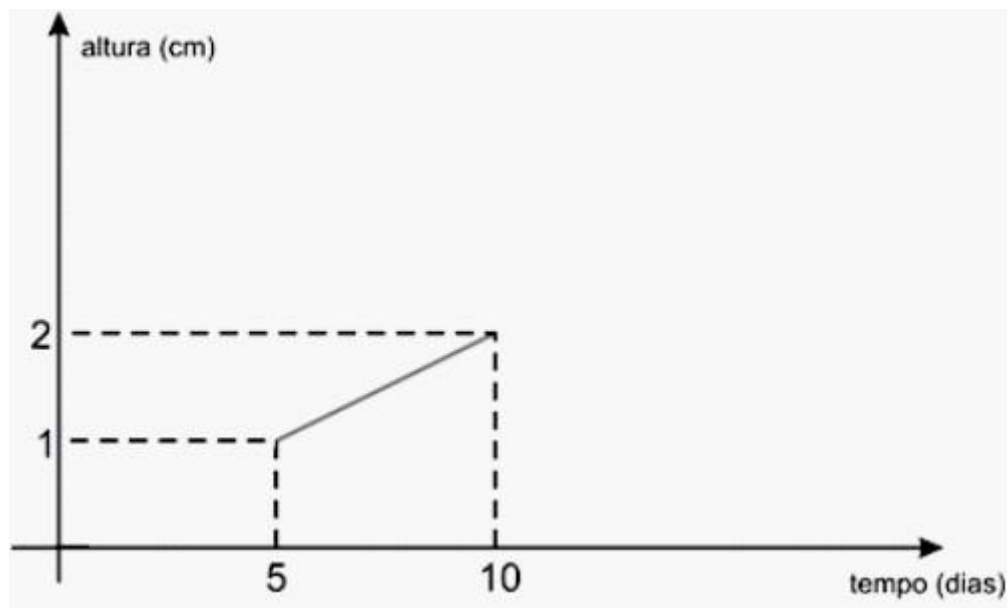
Exemplo 1 - Um paciente foi submetido a um eletrocardiograma em uma clínica, e o resultado é mostrado no quadro 1, onde t representa o tempo e v a voltagem (tensão).

Tempo (t)	0	1	2	3	5	7	8
Voltagem (v)	0	4	0	4	0	2	4

Como podemos observar, para cada valor da variável t, existe um único valor da variável v, isto é, para um mesmo tempo t, não podem ser registradas duas voltagens, o que motiva a seguinte definição.

Exemplo2 - Um botânico mede o crescimento de uma planta, em centímetros, todos os dias. Ligando os pontos colocados por ele num gráfico, resulta a figura 1.4 abaixo. Se for mantida sempre esta relação entre tempo e altura, que altura a planta terá no 30º dia?

Mergulhando no R & K³ Parte 1



Para acharmos o valor pedido, devemos determinar a equação que expressa a altura a , como função tempo t .

De acordo com o gráfico, a reta passa pelos pontos (5,1) e (10,2), logo o coeficiente angular da reta é dado por

$$m = \frac{2-1}{10-5} = \frac{1}{5}$$

e, então, a equação da reta é dada pela expressão $a - a_0 = m(t - t_0)$, obtida a partir da inclinação m quando (t_0, a_0) é um ponto dado e (t, a) é um ponto qualquer da reta. Assim, tomando o ponto (5,1), vem que

$$a - 1 = \frac{1}{5}(t - 5) \Rightarrow a(t) = \frac{t}{5}$$

é a equação da reta que permite calcular qualquer altura sendo dado o tempo. Logo, $t = 30$ dias para dias,

$$a = \frac{30}{5} = 6 \text{ cm}$$

Mergulhando no R & K³ Parte 1

Os programas especializados já disponibilizam essas equações, sem arcarmos com o custo de desenvolvê-las. Vamos conhecer os modelos matemáticos gerais.

Exemplos de funções matemáticas:

Função de 1º grau. Com $a = 0$

Função módulo

Função segundo grau ou quadrática

Função cúbica

Função quarta

Função quártupla

Função raiz quadrada

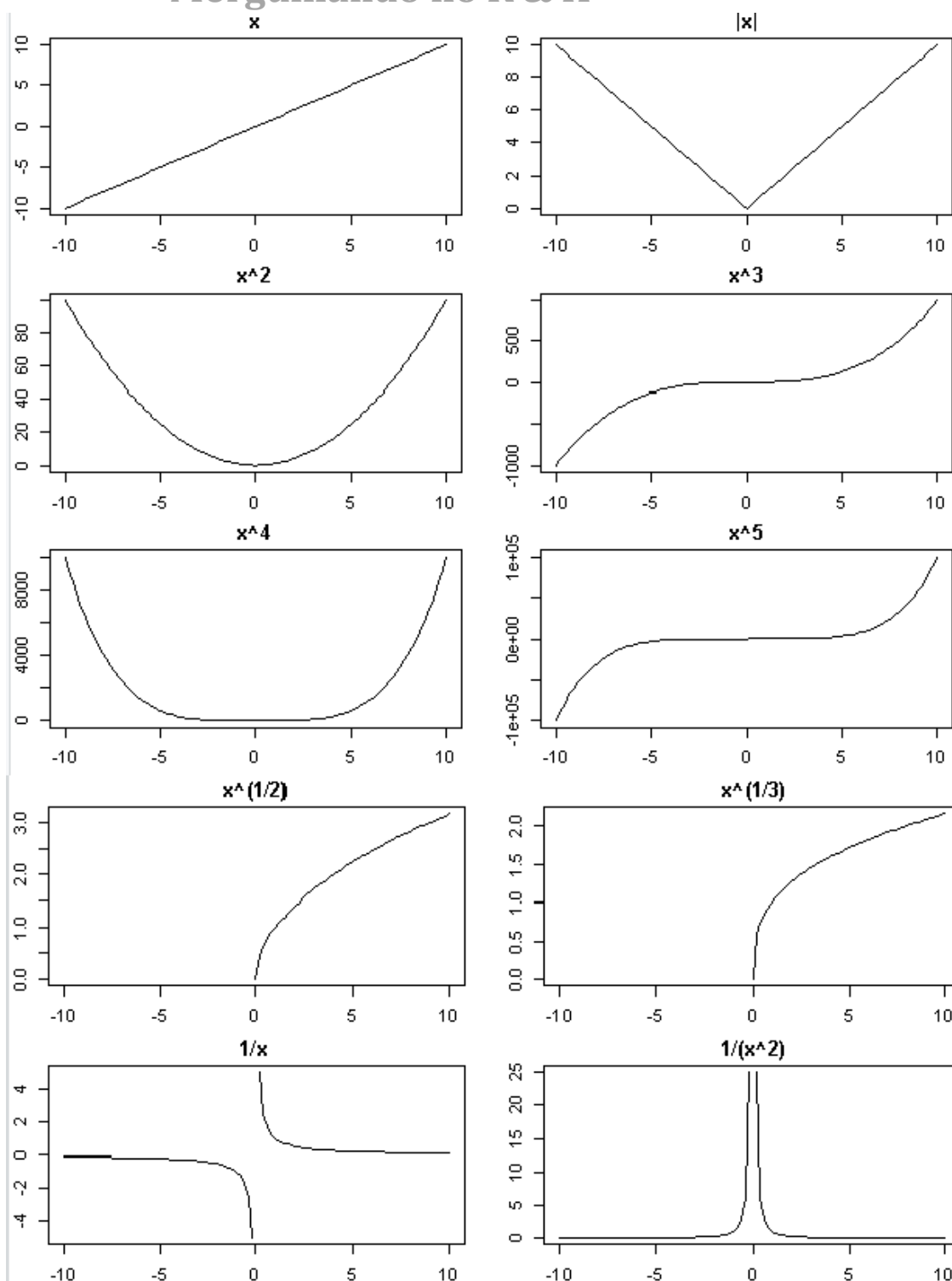
Função raiz cúbica

Função racional ou função recíproco

Função racional cujo limite tende a 1

```
par(mfrow=c(5,2), mar=c(2,2,2,2))
curve(x+0, from = -10, to = 10, main = "x")
curve(abs(x), from = -10, to = 10, main = "|x|")
curve(x^2, from = -10, to = 10, main = "x^2")
curve(x^3, from = -10, to = 10, main = "x^3")
curve(x^4, from = -10, to = 10, main = "x^4")
curve(x^5, from = -10, to = 10, main = "x^5")
curve(x^(1/2), from = -10, to = 10, main = "x^(1/2)")
curve(x^(1/3), from = -10, to = 10, main = "x^(1/3)")
curve(1/x, from = -10, to = 10, main = "1/x")
curve(1/(x^2), from = -10, to = 10, main = "1/(x^2)")
```

Mergulhando no R & K³ Parte 1



Vamos fazer a mesmo código com a parte positiva dos dados.

Mergulhando no R & K³ Parte 1

```
par(mfrow=c(5,2), mar=c(2,2,2,2))  
curve(x+0, from = 0, to = 10, main = "x")  
curve(abs(x), from = 0, to = 10, main = "|x|")  
curve(x^2, from = 0, to = 10, main = "x^2")  
curve(x^3, from = 0, to = 10, main = "x^3")  
curve(x^4, from = 0, to = 10, main = "x^4")  
curve(x^5, from = 0, to = 10, main = "x^5")  
curve(x^(1/2), from = 0, to = 10, main = "x^(1/2)")  
curve(x^(1/3), from = 0, to = 10, main = "x^(1/3)")  
curve(1/x, from = 0, to = 10, main = "1/x")  
curve(1/(x^2), from = 0, to = 10, main = "1/(x^2)")
```

