BIS0005 - Bases Computacionais da Ciência

Aula 02 - Representação Gráfica de Funções

Saul Leite
Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

Q2 2018

Motivação

- Em diferentes áreas da Ciência busca-se modelar fenômenos por meio de funções matemáticas a fim de reproduzir os comportamentos observados na natureza.
  - comportamento de gases;
  - escoamento de fluídos;
  - propagação de ondas;
  - etc...

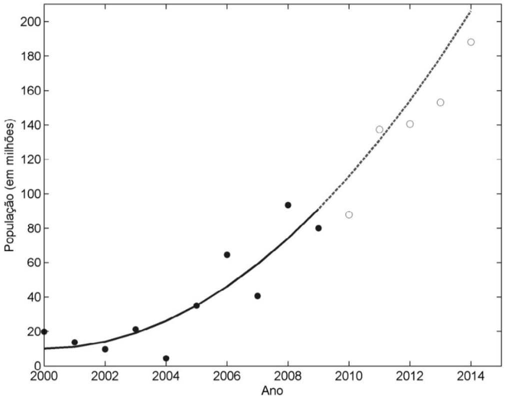
- Dado um modelo matemático, muitas vezes, temos a necessidade de visualizar o comportamento do mesmo.

- Gráficos de funções auxiliam o entendimento dos fenômenos.

Motivação: Crescimento populacional

Modelo para o crescimento de uma faixa sócio-econômica da população.

Dados disponíveis (pontos cheios) são usados para montar a função matemática que descreve os dados. Curva tracejada é a **extrapolação** da função em datas futuras.

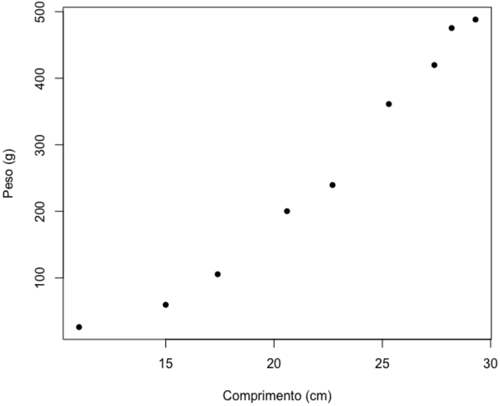


Motivação: Exemplo Tilápia do Nilo

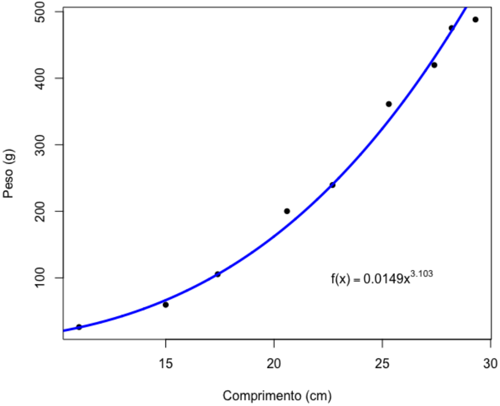
idade	comprimento medio (cm)	peso medio (g)
0	11.0	26.0
1	15.0	59.5
2	17.4	105.4
3	20.6	200.2
4	22.7	239.5
5	25.3	361.2
6	27.4	419.8
7	28.2	475.4
8	29.3	488.2

(Retirado de Bassanezi, 2009)

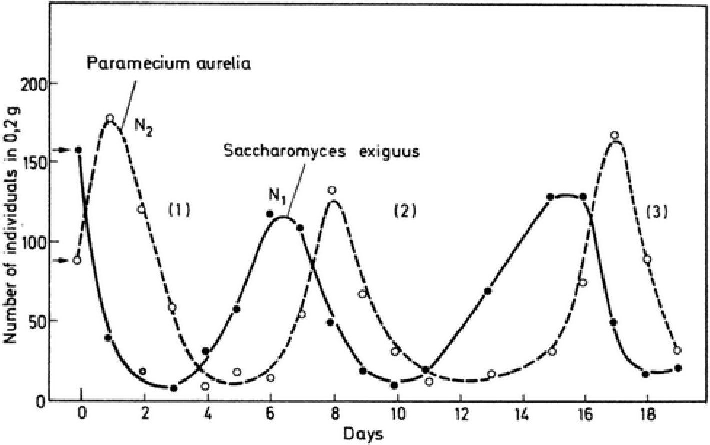
Motivação: Exemplo Tilápia do Nilo



Motivação: Exemplo Tilápia do Nilo

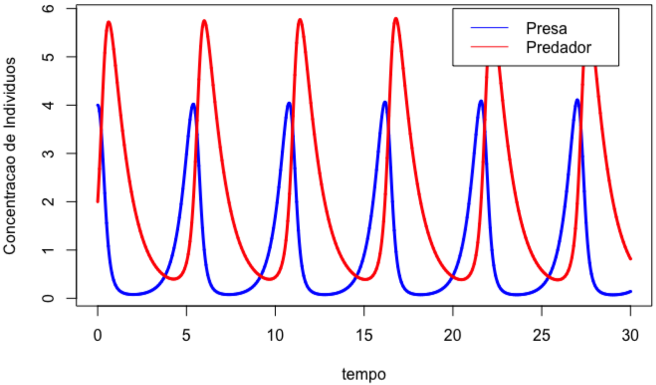


Motivação: Dinâmica populacional



Flutuação no tamanho da população de *Paramecium aurelia* que se alimenta de *Saccharomyces exiguus*. (Bassanezi, 2009)

Motivação Dinâmica populacional



Lotka-Volterra ~1920, um dos primeiros modelos matemáticos para essa interação presa-predador.

Construção do modelo: exemplo população de bactérias

Considere os dados da tabela que mostram o crescimento de uma população (em milhares) de bactérias.

Qual a equação que descreve esse crescimento populacional de bactérias?

geracao	populacao
0	140.000
1	182.000
2	236.600
3	307.580
4	399.854
5	519.810
6	675.753

Construção do modelo: exemplo população de bactérias

Populações, em geral, crescem muito rapidamente, pois a cada geração são mais indivíduos para se reproduzir.

Dividindo a população de cada geração pela da geração anterior, obtém-se:

$$\frac{p(1)}{p(0)} = \frac{182}{140} = 1.3 \qquad \frac{p(3)}{p(2)} = \frac{307.580}{236.6} = 1.3$$

$$\frac{p(2)}{p(1)} = \frac{236.6}{182} = 1.3 \qquad \frac{p(4)}{p(3)} = \frac{399.854}{307.580} = 1.3,$$

em que $p(x)$ representa a população na geração x.

Construção do modelo: exemplo população de bactérias

Desta forma, temos que

$$\frac{p(x)}{p(x-1)} = 1.3 \quad \Rightarrow \quad p(x) = p(x-1)1.3, \quad \text{para } x = 1, 2, 3, 4, \ldots.$$

Portanto,

$$\begin{aligned}
p(1) &= p(0)1.3 \\
p(2) &= p(1)\ 1.3 = (p(0)\ 1.3)\ 1.3 = p(0)\ (1.3)^2 \\
p(3) &= p(2)\ 1.3 = (p(0)\ (1.3)^2)\ 1.3 = p(0)\ (1.3)^3
\end{aligned}$$

Construção do modelo: exemplo população de bactérias

Desta forma, temos que

$$\frac{p(x)}{p(x-1)} = 1.3 \quad \Rightarrow \quad p(x) = p(x-1)1.3, \quad \text{para } x = 1, 2, 3, 4, \ldots.$$

Portanto,

$$\begin{aligned}
p(1) &= p(0)1.3 \\
p(2) &= p(1)\ 1.3 = (p(0)\ 1.3)\ 1.3 = p(0)\ (1.3)^2 \\
p(3) &= p(2)\ 1.3 = (p(0)\ (1.3)^2)\ 1.3 = p(0)\ (1.3)^3
\end{aligned}$$

Podemos concluir que $p(x) = p(0)\ (1.3)^x$.

Construção do modelo: exemplo população de bactérias

Desta forma, temos que

$$\frac{p(x)}{p(x-1)} = 1.3 \quad \Rightarrow \quad p(x) = p(x-1)1.3, \quad \text{para } x = 1, 2, 3, 4, \ldots.$$

Portanto,

$$\begin{aligned}
p(1) &= p(0)1.3 \\
p(2) &= p(1)\,1.3 = (p(0)\,1.3)\,1.3 = p(0)\,(1.3)^2 \\
p(3) &= p(2)\,1.3 = (p(0)\,(1.3)^2)\,1.3 = p(0)\,(1.3)^3
\end{aligned}$$

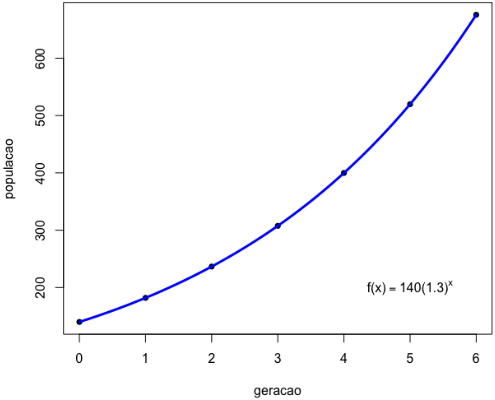
Podemos concluir que $p(x) = p(0)\,(1.3)^x$. Logo temos:

$$p(x) = 140\,(1.3)^x \text{ para } x = 1, 2, 3, \ldots$$

Construção do modelo: exemplo população de bactérias

$$p(x) = 140\ (1.3)^x$$

Esta é uma função exponencial com base 1.3.
A base representa um fator de crescimento pelo qual a população muda a cada geração.
Neste caso, considerando r a taxa percentual, diz-se que a taxa de crescimento é $r = 30\% = 0.3$.



Motivação

O estudo de funções decorre da necessidade de:

- Analisar fenômenos, visualizando o comportamento de um sistema.
- Interpretar interdependências, entendendo como uma variável comporta-se com relação à outra.
- Encontrar soluções de problemas.
- Descrever regularidades.
- Generalizar.

PARTE PRÁTICA

Ferramentas de Programação e Visualização

Existem diversas ferramentas para utilizadas em cálculos matemáticos avançados.

- Matlab
- Maple
- Mathematica
- Octave
- Scilab (livro)
- Python
- R

Geralmente contam com bibliotecas de funções matemáticas prontas e recursos avançados.

R

R é um conjunto integrado de recursos de software para manipulação de dados, cálculo e exibição gráfica.

Um dos pontos fortes do R é a facilidade de trabalhar com conjuntos de dados e gerar gráficos de qualidade.

R está disponível como Software Livre (na forma de código fonte). Ele roda em uma ampla variedade de plataformas (Linux, Windows, MacOS).

Link para download: http://vps.fmvz.usp.br/CRAN/

RStudio

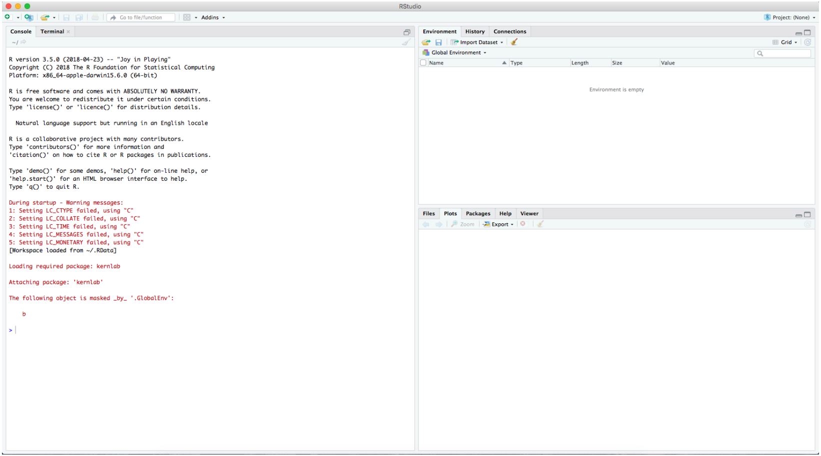
O RStudio é um ambiente de desenvolvimento integrado (IDE) para R. Ele inclui:

- um console para executar comandos;
- editor de código com realce de sintaxe que suporta execução direta de código;
- ferramentas para plotagem
- histórico
- depuração
- gerenciamento de espaço de trabalho.

O RStudio está disponível em edições open source. E roda em diversas plataformas (Linux, Windows, MacOS).

Link para download:
https://www.rstudio.com/products/rstudio/download/

RStudio


RStudio
Go to file/function
Addins
Project: (None)
Console
Terminal
~/
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
During startup - Warning messages:
1: Setting LC_CTYPE failed, using "C"
2: Setting LC_COLLATE failed, using "C"
3: Setting LC_TIME failed, using "C"
4: Setting LC_MESSAGES failed, using "C"
5: Setting LC_MONETARY failed, using "C"
[Workspace loaded from ~/.RData]
Loading required package: kernlab
Attaching package: 'kernlab'
The following object is masked _by_ '.GlobalEnv':
b
>
Environment
History
Connections
Import Dataset
Grid
Global Environment
Name
Type
Length
Size
Value
Environment is empty
Files
Plots
Packages
Help
Viewer
Zoom
Export

A interação do usuário com o R pode ocorrer de duas formas distintas:

- Na primeira forma, os comandos são digitados diretamente no prompt do R: ao ser pressionada a tecla Enter, os comandos digitados são interpretados e imediatamente executados.

Neste caso, o R funciona como uma sofisticada calculadora

- Na segunda forma, um conjunto de comandos é digitado em um arquivo texto: Este arquivo, em seguida, é levado para o ambiente R e executado.

Neste modo, o R funciona como um ambiente de programação.

R: Exemplo

Exemplo de código em R, executado na linha de comando:

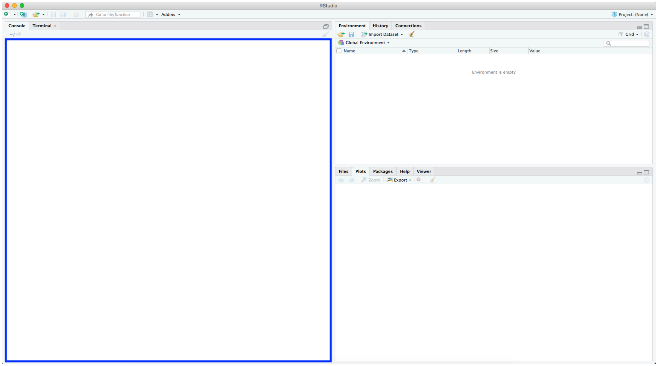
```r
x <- 2
y <- x + 5
y
```

```
## [1] 7
```

Neste exemplo, x, e y são variáveis.

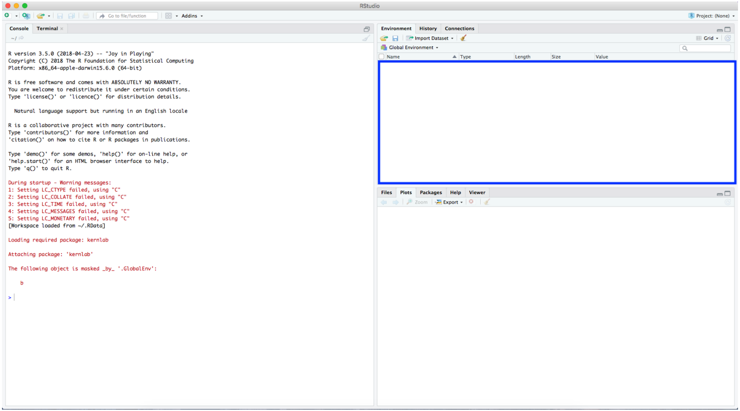
Executando pelo RStudio

Entre com os comandos na área marcada no RStudio.



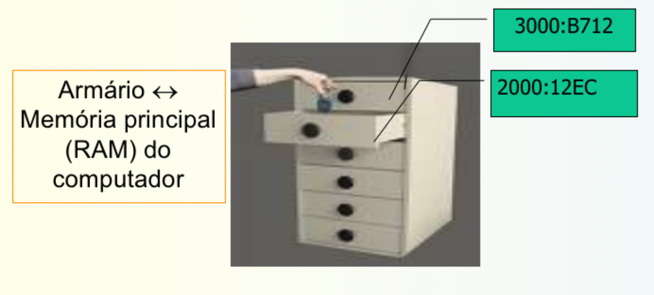
Executando pelo RStudio

Note que as variáveis criadas aparecem do lado superior direito, como ilustrado abaixo.



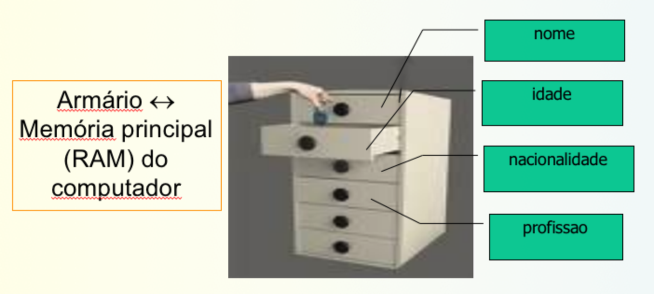
Variáveis

Em programas computacionais precisamos armazenar informações para utilizarmos durante a execução do programa.



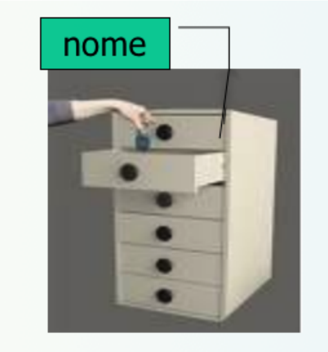
Variáveis

As linguagens de programação permitem que os usuários atribuam nomes para as posições de memória da máquina.



Variáveis

Uma variável é um endereço da memória de acesso randômico (RAM), representada por um nome (rótulo), criado pelo usuário, cujo conteúdo pode se alterar no decorrer do programa.



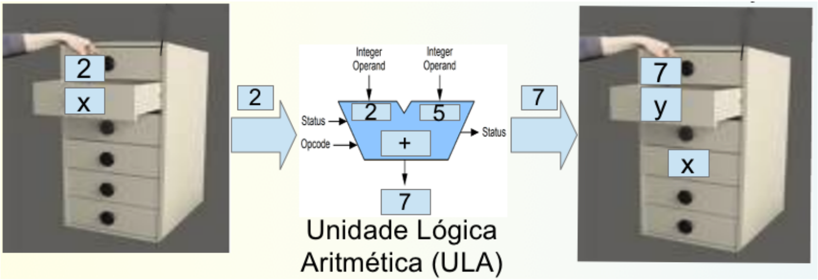
Uma variável é composta por dois elementos:

- **Identificador**: nome dado pelo programador à variável
- **Conteúdo**: valor atual da variável

Processamento pelo computador

O que acontece internamente no computador quando aplicamos o seguinte comando?

```
x <- 2
y <- x + 5
```



Processamento pelo computador

Lembrando que a Unidade de Controle do processador (CPU) está sempre dando as ordens para os demais componentes

- "Memória RAM, quero o valor de x".
- "ULA, some o valor 2 com o valor 5".
- "Memória RAM, associe um novo endereço a variável denominada y e escreva nesse endereço o valor 7".

R

O R possui várias operações e funções matemáticas que podem ser facilmente utilizadas, como por exemplo:

Operação Função	Descrição
+, -	soma, subtração
/, *	divisão, multiplicação
^ ou **	potência
sin(x)	seno em radianos
cos(x)	cosseno em radianos
tan(x)	tangente em radianos
log(x)	logaritmo base natural
log10(x)	logaritmo base 10
sqrt(x)	raíz quadrada $\sqrt{x}$
exp(x)	exponencial e^x

R: Criando funções

Podemos criar nossas próprias funções no R. Abaixo, definimos a função $f(x) = 140\ (1.3)^x$:

```
f <- function(x) 140 * (1.3)^x
```

Nesse exemplo, a variável f representa a nossa função. Podemos calcular a função para diferentes valores de x da seguinte forma:

```
f(1)
```

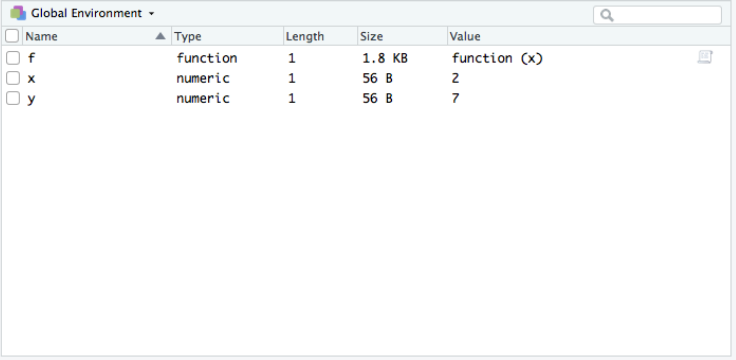
```
## [1] 182
```

```
f(2)
```

```
## [1] 236.6
```

RStudio

Note que as funções também aparecem no lado superior direito do RStudio:



Global Environment

Name	Type	Length	Size	Value
f	function	1	1.8 KB	function (x)
x	numeric	1	56 B	2
y	numeric	1	56 B	7

R: Criando funções

Outro exemplo de função em R: vamos definir $g(x) = \sqrt{x} \cdot cos(x)$

```
g <- function(x) sqrt(x) * cos(x)
```

Podemos calcular a função $g(x)$ para diferentes valores de x:

```
g(pi)
```

```
## [1] -1.772454
```

```
g(2*pi)
```

```
## [1] 2.506628
```

em que pi é uma variável pré-definida em R com o valor de $\pi = 3.141593$.

R: Gerando um gráfico

Sempre que desejamos produzir um gráfico de uma função, precisamos definir em quais pontos gostaríamos de visualizar a função, ou seja, para quais valores de x.

Existem duas formas para se definir estes valores:

- Definindo diretamente os pontos x nos quais queremos plotar a função.
- Definindo um intervalo de valores de x no qual queremos plotar a função

R: Gerando um gráfico

Forma 1: definindo diretamente os pontos x. Podemos definir os pontos de interesse usando o comando **c()**, onde listamos todos os valores que desejamos para a função.

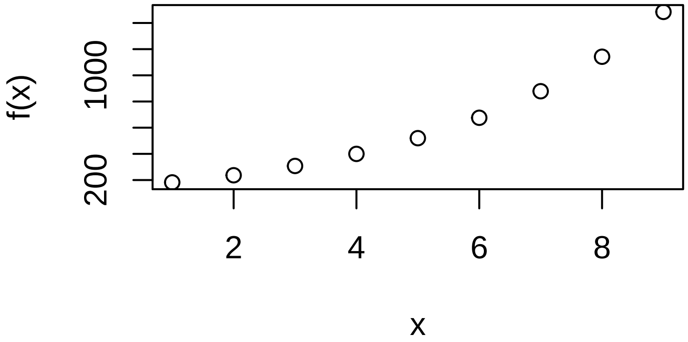
```
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
```

Neste caso, x é um vetor (falaremos mais sobre eles mais para frente).

R: Gerando um gráfico

Para gerar o gráfico, basta chamar a função **plot**, da seguinte forma:

```
f <- function(x) 140 * (1.3)^x
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
plot(x,f(x))
```



R: Gerando um gráfico

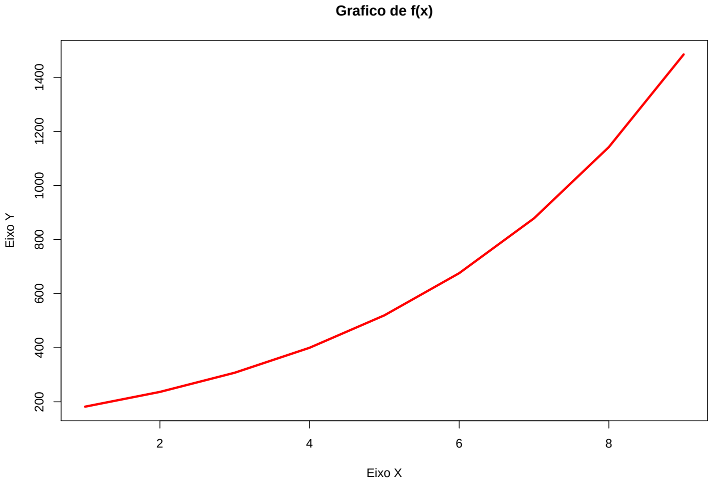
Podemos modificar o gráfico de diversas maneiras. Por exemplo,

- podemos ligar os pontos discretos com uma curva usando o parâmetro **type = “l”** no comando plot.
- podemos também dar um nome aos eixos x e y usando **xlab** e **ylab**.
- podemos dar um título para o gráfico usando **main**.
- podemos trocar a cor do gráfico com o argumento **col**.
- podemos aumentar a grossura da linha com o argumento **lwd**.

```
f <- function(x) 140 * (1.3)^x
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
plot(x,f(x),type="l",xlab="Eixo X", ylab="Eixo Y",
     main = "Grafico de f(x)", col="red", lwd=3)
```

R: Gerando um gráfico

Gráfico resultante:



R: Gerando um gráfico

Outros parâmetros da função plot

parâmetro	Descrição
lty	especifica o tipo de linha (1=solida, 2=tracejada, 3=pontilhada)
ylim	intervalo de valores para y usado no gráfico
xlim	intervalo de valores para x usado no gráfico
type	especifica o tipo de gráfico "p" = pontos, "l" = linhas, "b" = ambos, "h" = histograma.
sub	define um subtítulo para o gráfico.

R: Gerando um gráfico

Forma 2: definindo um intervalo de valores de x no qual queremos plotar a função. Podemos definir os pontos de interesse em um intervalo usando o comando **seq(from,to,by)**, com os seguintes parâmetros:

- *from*: inicio do intervalo
- *to:* fim do intervalo
- *by*: incremento da sequencia

```
x <- seq(1,9,1)
```

O comando acima é equivalente ao que fizemos anteriormente.

R: Gerando um gráfico

Outro exemplo: um gráfico da função $g(x) = \sqrt{x} \cdot cos(x)$ no intervalo $[0, 2\pi]$:

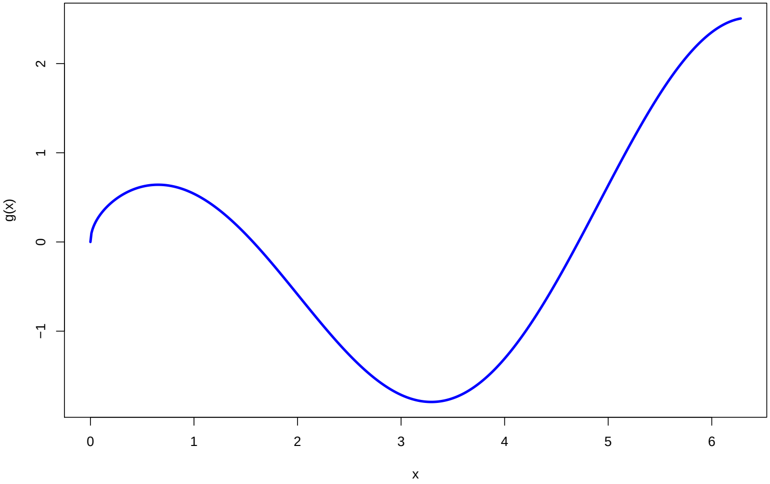
```
g <- function(x) sqrt(x) * cos(x)
x <- seq(0,2*pi,0.01)
plot(x,g(x),type="l",col="blue",lwd=3)
```

Note que o x acima é um vetor, com valores:

$$x = (0, 0.01, 0.02, 0.03, 0.04, 0.05, \ldots, 2\pi)$$

R: Gerando um gráfico

Gráfico resultante:



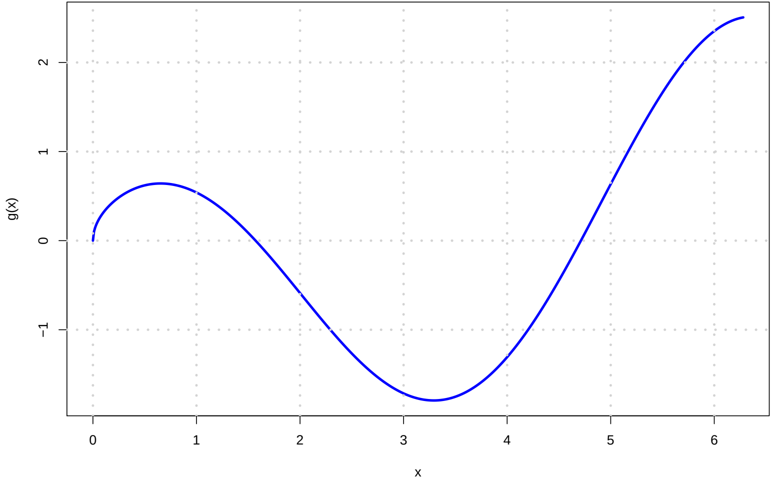
R: Gerando um gráfico

Podemos gerar um grid usando o comando **grid** do R. Considere o exemplo abaixo:

```
g <- function(x) sqrt(x) * cos(x)
x <- seq(0,2*pi,0.01)
plot(x,g(x),type="l",col="blue",lwd=3)
grid(lwd=3)
```

R: Gerando um gráfico

Gráfico resultante:



R: Combinando gráficos de diversas funções.

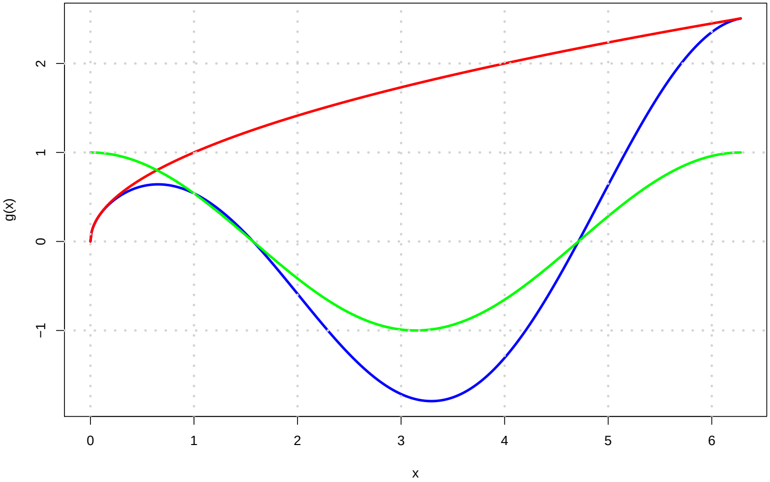
É possível fazer o gráfico de diferentes funções na mesma figura. Para isso, basta fazer chamadas ao comando **lines** após a execusão do comando **plot**, como ilustrado abaixo:

```
g <- function(x) sqrt(x) * cos(x)
g1<- function(x) sqrt(x)
g2<- function(x) cos(x)

x <- seq(0,2*pi,0.01)
plot(x,g(x),type="l",col="blue",lwd=3)
lines(x,g1(x),col="red",lwd=3)
lines(x,g2(x),col="green",lwd=3)
grid(lwd=3)
```

R: Combinando gráficos de diversas funções.

Gráfico resultante:



R: Adicionando pontos no gráfico

Podemos também adicionar pontos nos gráficos usando o comando *points*:

```
points(pos_x, pos_y, pch)
```

que necessita dos seguintes parâmetros:

- *pos_x*: coordenada x do ponto;
- *pox_y*: coordenada y do ponto;
- *pch*: tipo do ponto para representação gráfica, ilustrado na figura abaixo.

ATIVIDADES EM AULA

Exercício 1

A empresa COLKS é uma indústria automobilística em um pais, onde a moeda oficial é o dubila. O lucro mensal da COLKS é função do número de carros produzidos no mês.

Ela tem um custo fixo de 50 dubilas e um custo variável em função do número de carros produzidos no mês (N_c) dado por $48(N_c)^{0.9}$. Vamos dizer que ela venda cada carro por 50 dubilas.

Assim, o seu lucro L mensal é dado por

$$L = 50N_c - 48(N_c)^{0.9} - 50$$

Exercício 1

1. Determine o lucro L da COLKS ao produzir $N_c = 1$, $N_c = 4$ e $N_c = 10$ carros. Interprete os resultados que você obteve.
2. Agora faça um gráfico de L em função de N_c para $0 \leq N_c \leq 20$. A partir de quantos carros mensalmente vendidos a COLKS começa a ter lucro?
3. Analisando o gráfico, quantos carros a COLKS tem que produzir no mês para ter um lucro de cerca de 100 dubilas?

Exercício 2

A acidez $A(x)$ de uma solução de hidróxido de magnésio em ácido clorídrico, sob certas condições experimentais, é dada pela equação

$$A(x) = x^3 + 3x^2 - 54,$$

na qual x é a concentração de íons hidrônio. Pede-se:

1. Use o R para gerar o gráfico de $A(x)$ em função de x para $0 \leq x \leq 8$;
2. A partir da analise do gráfico, determine a concentração x do íon de hidrônio que resulta em solução saturada (i.e., com acidez nula). Acrescente uma instrução que gere um ponto vermelho no gráfico correspondente à saturação da solução.

MATERIAL EXTRA

Fazendo gráficos tridimensionais

Vejamos agora como podemos fazer gráficos tridimensionais. Como exemplo, vamos fazer o gráfico da função:

$$f(x,y) = x^2 + y^2$$

Para fazer esse tipo de gráfico, iremos usar a biblioteca *plot3D* do R. Você pode instalar essa biblioteca no seu computador, chamado o seguinte comando na linha da comando do R:

```r
install.packages("plot3D")
```

Bibliotecas são usadas para estender as funcionalidades do R.

Fazendo gráficos tridimensionais

Vamos definir nossa função:

```
f <- function(x,y) x^2 + y^2
```

que agora depende de dois parâmetros. Com a biblioteca *plot3D* instalada, podemos gerar o gráfio.

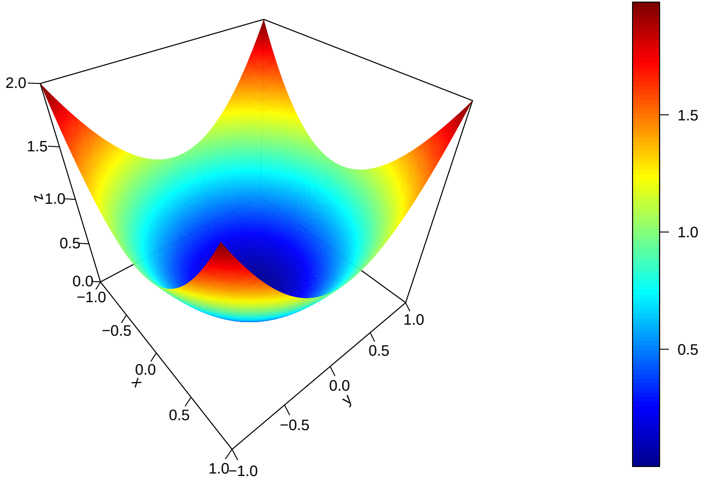
Geramos o gráfico com a seguinte sequencia de comandos:

```
library(plot3D)       #carrega a biblioteca plot3D
x <- seq(-1,1,0.01)  #define pontos de x para o gráfico
y <- seq(-1,1,0.01)  #define pontos de y para o gráfico
z <- outer(x, y, f)  # calcula a função nos pontos x e y

#gera o gráfico
persp3D(x, y, z, phi = 40, theta = 50,
        ticktype="detailed")
```

Fazendo gráficos tridimensionais

Gráfico resultante:



Fazendo gráficos tridimensionais (iterativos)

Para fazer gráficos iterativos, utilize a biblioteca *plot3Drgl* e siga os mesmos passos dos slides anteriores, chamando a seguinte função para gerar o gráfico:

```
persp3Drgl(x, y, z, phi = 40, theta = 50,
         ticktype="detailed")
```

Atividades para casa

Atividades para fazer até a próxima aula:

- Fazer a Lista de Exercícios 1 no Tidia (lista01.pdf).
- Ler o **Capítulo 3** "Noções de Estatística, Correlação e Regressão" do livro "Bases Computacionais da Ciência."

Referências

- Aulas dos Profs. David Correa Martins Jr, Wagner Tanaka Botelho e Jesús P. Mena-Chalco.
- Livro Bases Computacionais da Ciência.