# Como programar em R

## Lógica de programação e programação para o R

Rafael Etto, Carolina Gavão, Rodrigo Nascimento, Renann Rodrigues,
Daniel Gonçalves, Diego Santi e Douglas Tomacheswki

UEPG

Laboratório de Biologia Molecular Microbiana
LABMOM

# Lógica de programação

# Introdução Lógica de Programação e Algoritmo

- **Lógica de programação** é o modo como se escreve um programa de computador, por meio de *algoritmo*.

- **Algoritmo** é uma sequência de passos para se executar uma função. Um exemplo de algoritmo, fora da computação, é uma receita de bolo.

# Introdução a Linguagem R

- Uma linguagem desenvolvida para manipulação, análise e visualização gráfica de dados;
- O nome R provém em parte das iniciais dos criadores (Ross e Robert);
- A linguagem R é largamente usada entre estatísticos e analistas de dados para desenvolver software de estatística e análise de dados. Pesquisas e levantamentos com profissionais da área mostram que a popularidade do R aumentou substancialmente nos últimos anos.

# Linguagem de Programa R

- Projeto de sucesso e popular no meio acadêmico e empresarial;
- Projeto ativo, com lançamento de duas novas versões por ano;
- Linguagem de programação interpretada e procedural para estatística e mineração de dados;
- Ambiente de visualização;
- Milhares de funções distribuídas de forma livre.
- Informações  são processadas em memória.

# Ambientes de programação

- RGUI;
- Eclipse
- Vim;
- Gedit;
- Notepad++;
- Deducer;
- JGE;
- RKWard;
- R Studio;
- Microsoft Open R

- Windows
- Unix
- Linux
- Macintosh
- Chrome OS

# Linguagem R

- Compilado e interpretado;
- Orientado a objetos;
- imperativo e dinâmico.

# Instalação das dependências do R no Linux

$ apt-get install build-essential g++ gfortran libbz2-dev libcurl4-openssl-dev liblzma-dev_5.2.4-1 libpango1.0-dev libreadline-dev xorg-dev

# Instalação do R a partir do código fonte

```
$ tar -xzvf R-{versao_do_R}.tar.gz
$ cd R-{versao_do_R}
$ ./configure --prefix=/opt/R/{versao_do_R} --enable-memory-profiling --enable-R-shlib --with-blas --with-lapack
$ make
$ sudo make install
```

# Gerenciamento de múltiplas versões do R em distribuições baseadas no Debian

$ sudo update-alternatives --install /usr/bin/R R /opt/R/{versao_do_R}/bin/R 1
$ sudo update-alternatives --install /usr/bin/R R /opt/R/{versao_do_R}/bin/R 2
$ sudo update-alternatives --install /usr/bin/R R /opt/R/{versao_do_R}/bin/R 3
$ sudo update-alternatives --config R

```
rodrigo@dream:~$ update-alternatives --config R
Existem 2 escolhas para a alternativa R (disponibiliza /usr/bin/R).

  Selecção    Caminho              Prioridade Estado
------------------------------------------------------------
  0            /opt/R/3.6.0/bin/R   100       modo automático
* 1            /opt/R/3.5.1/bin/R   10        modo manual
  2            /opt/R/3.6.0/bin/R   100       modo manual

Pressione <enter> para manter a escolha actual[*], ou digite o número da selecção:
```

# Gerenciamento de múltiplas versões do R em distribuições baseadas no Debian

$ sudo update-alternatives --install /usr/bin/Rscript Rscript /opt/R/{versao_do_R}/bin/Rscript 1
$ sudo update-alternatives --install /usr/bin/Rscript Rscript /opt/R/{versao_do_R}/bin/Rscript 2
$ sudo update-alternatives --install /usr/bin/Rscript Rscript /opt/R/{versao_do_R}/bin/Rscript 3
$ sudo update-alternatives --config Rscript

```
rodrigo@dream:~/Documentos/bio/Python/Dia_01$ sudo update-alternatives --config Rscript
Existem 3 escolhas para a alternativa Rscript (disponibiliza /usr/bin/Rscript).

  Selecção   Caminho                    Prioridade Estado
------------------------------------------------------------
  0            /opt/R/3.4.2/bin/Rscript   3         modo automático
  1            /opt/R/3.4.2/bin/Rscript   3         modo manual
* 2            /opt/R/3.5.1/bin/Rscript   1         modo manual
  3            /opt/R/3.6.0/bin/Rscript   2         modo manual

Pressione <enter> para manter a escolha actual[*], ou digite o número da selecção:
```

# Especificações gerais no R

- Caixa branca;
  - Pacotes ou funções de código livre;
- Caixa preta;
  - Pacote ou funções compiladas que não se tenha acesso.
- Sensível a letras maiúsculas e minúsculas;
- Conjunto de códigos podem ser executadas por linha de comando;
- Funções podem requerer parâmetros ou não;

# Packages (Pacotes)

- Potencializa o uso da ferramenta;
- Milhares de pacotes disponíveis;
- Hospedagem no CRAN (Comprehensive R Archive Network);
- Existem espelhos da nuvem CRAN no Brasil
  - UFPR;
  - USP;
  - Fundação Oswaldo Cruz;
  - Universidade Estadual de Santa Cruz

# Instalação de pacotes no R

Instalação a partir da nuvem:
install.packages( “”, dependencies = TRUE)

Instalação a partir de um arquivo local:
install.packages(file.choose(), repos=NULL, type=”source”)

Instalação a partir de um link:
install.packages(“url”, repos = NULL, type=”source”)

# Operadores

Formas como os operadores básicos matemáticos e operadores lógicos são interpretados no R.

| Operador | Descrição |
|---|---|
| [, [[ | Subscrição e subconjunto |
| ^ | Exponencial |
| +, - | Operador de Soma e Subtração |
| *, / | Operador multiplicação e divisão |
| <, >, <=, >=, ==, != | Comparação |
| ! | Negação |
| &, && | E |
| \|, \|\| | Ou |
| =, -> | Atribuição da esquerda para direita |
| =, <- | Atribuição da direita para esquerda |
| ? | Ajuda |

# Variáveis no R

Inteiro: {0; 1; 2; 3; 4; 5; 6; 7; …; 8}

float: {0,1; 0,2; 0,3; …; 0,0009}

booleano: {TRUE, FALSE}

String: {Linguagem; Manaus; Barcelos}

Caracter: {a, e, i, o, u}

# Palavras Reservadas

Palavras reservadas nunca devem ser utilizadas como nome de variáveis, por exemplo.:

if, else, repeat, while, for, in, next, break, TRUE, FALSE, NULL, Inf, NaN, NA.

## Operações matemáticas básicas

- O R por ser uma ferramenta desenvolvida com propósito de manipulação de dados, visualização gráfica e desenvolvimento de modelos estatísticos, também pode ser utilizado para calcular expressões algébricas básicas.

Soma: 2 + 2
Subtração: 3 - 1
Potência: 2 ^ 60
Multiplicação: 9 * 8
Divisão: 2 / 2
Função Raiz quadrada: sqrt(4)

# Operadores Lógicos

| Função/Operador | Sintaxe |
|---|---|
| Pi | pi |
| Potência | ^ |
| Valor absoluto | abs |
| Logaritmo | log |
| Fatorial | factorial |
| Raiz quadrada | sqrt |
| Seno | sin |
| Coseno | cos |
| Tangente | tan |

# Uso inicial do R

## Ordem de precedência

- 1º Parêntesis;
- 2º Expoentes;
- 3º Multiplicações e Divisões; (da esquerda para a direita)
- 4º Somas e Subtrações. (da esquerda para a direita)

$$((4 + 2)/12) \text{ ^ } 3$$
$$\neq$$
$$4 + 2/12 \text{ ^ } 3$$

# Uso inicial do R

## Acessando diretório do projeto

Comando para gerenciar os arquivos salvos do projeto

getwd() - Comando exibe a pasta atual do projeto;

setwd() - Comando que muda o diretório de seu projeto;

Exemplo:
Linux - setwd("/home/usuario/Documentos/cursoProgramarR")
Windows - setwd("C:\\Users\\Usuario\\Documents\\cursoProgramarR")

Obs.: No Windows o ideal é usar duas barras para mudar de dirétorio

# Exercício B

- Calcule o valor numérico da expressão

$$[(18 + 3 * 2) \div 8 + 5 * 3] \div 6$$

- Calcule o valor numérico da expressão

$$\{[(8 * 4 + 3) \div 7 + (3 + 15 \div 5) * 3] * 2 - (19 - 7) \div 6\} * 2 + 12$$

- $\frac{(1+\sqrt{225})}{2}$

# Exercício B

- Calcule o valor numérico da expressão

$$[(18 + 3 * 2) \div 8 + 5 * 3] \div 6 = 3$$

- Calcule o valor numérico da expressão

$$\{[(8 * 4 + 3) \div 7 + (3 + 15 \div 5) * 3] * 2 - (19 - 7) \div 6\} * 2 + 12 = 100$$

- $(1 + \sqrt{255}))/2 = 8$

# Comentário no R

Na linguagem R o comentário não é executado.

# Este símbolo serve para comentar uma linha

# Manipulação de strings

email <- paste(“nascimento.rodrigo”, “hotmail.com.br”, sep=”@”)

email <- paste(“rsn.rodrigosn”, “gmail.com”, sep=”@”)

# Salvar

```
> save.image()
> save.image( file=”minha-sessao-introdutoria.RData” )
```

```
>#  Carrega um arquivo de workspace no mesmo diretório
> load(file="minha-sessao-introdutoria.RData")
```

# Exercício

1. Crie um diretório para seus exercícios da disciplina.
2. Chame o R, clicando no ícone da área de trabalho ou na barra de tarefas.
3. Verifique o seu diretório de trabalho.
4. Mude o diretório de trabalho para o diretório que você criou.
5. Verifique o conteúdo da área de trabalho.
6. Carregue o arquivo letras.RData.
7. Verifique novamente sua área de trabalho.
8. Saia do R, tomando o cuidado de salvar sua área de trabalho.
9. Repita os passos 2 a 5.

# Tipo de dados do objeto

- Caractere
- Numérico
- Inteiro
- Fator
- Data

Comando R para verificar o tipo de dados

- class()

# Estrutura de dados

| Classe | Descrição |
|---|---|
| Vetor | Conjunto unidimensional de valores de mesmo tipo de dados. |
| Matrizes | Conjunto multidimensional com linhas e colunas de mesmo tipo de dados. |
| Listas | Um conjunto de diversos agrupamento de diferentes objetos, como por exemplo vetor, matrizes, data frames entre outros e inclusive de diferentes tipos de dados e tamanhos. |
| Data Frames | Semelhante a um banco de dados. |
| Séries temporais | Estrutura de dados com informações diárias. Ex.: Temperatura do clima. |

# Vetor

- Criar um vetor: Usa-se o comando vetor que armazena em uma variável vetorA um vetor de tamanho 5:
  - vetorA <- vector(length = 5)
- Acessar um vetor:
  - vetorA[1]
  - vetorA[2]
  - vetorA[3]
  - vetorA[4]
  - vetorA[5]
- Visualização de vetor no R
  - fix()
  - edit()
  - view()

# Matrizes

- Criar uma matriz:
  - matrizA <- matrix(data=NA, nrow = 3, ncol = 3)
- Acessar uma matriz:
  - Para acessar a matriz levamos em consideração a linha e coluna que queremos acessar, matrizA[linha, coluna];
  - matrizA[1,1] = imprimindo o conteúdo da linha 1 e coluna 1
  - matrizA[1,2] = imprimindo o conteúdo da linha 1 e coluna 2
  - matrizA[1,3] = imprimindo o conteúdo da linha 1 e coluna 3
  - matrizA[2,1] = imprimindo o conteúdo da linha 1 e coluna 1
  - matrizA[2,2] = imprimindo o conteúdo da linha 1 e coluna 2
  - matrizA[2,3] = imprimindo o conteúdo da linha 1 e coluna 3
- Formas de visualização de Matriz no R
  - fix()
  - edit()
  - view()

# Listas

- Criar de uma lista:
  - listaA <- list()
- Visualizar a lista:
  - listaA
- Inserir uma matriz e um vetor do exemplo anterior na lista:
  - listaA <- list(vetorA, matrizA)
- Formas de visualização de uma lista
  - view()
  - fix()
  - edit()

# Data Frames

Utilizamos o data frame para armazenar tabelas de dados. É uma lista de vetores de igual tamanho.

Por exemplo, criaremos 5 vetores de tamanhos iguais:

```
a = c(12, 24, 33, 35)
b = c("aa", "bb", "cc", "dd")
c = c(TRUE, FALSE, TRUE, FALSE)
d = c(0.152, 0.1453, 0.1548, 0.6585)
e = c(“Árvore”, “Chapéu”, “Feijão”, “Açucar”)
```

# Data Frames

A partir dos vetores criados no slide anterior criaremos a tabela com todas as informações;

dataFame <- data.frame(a, b, c, d, e) #A variável dataFrame se torna um data.frame

Visualização de um dataFrame

- fix()
- edit()
- view()

# Serie temporais

- Uma série de tempo é uma coleção de variáveis aleatórias ordenadas ao longo do tempo.
  - Exemplo:
    - Coleta de temperatura a cada hora, diária, mensal ou anual;
    - Produção agrícola em determinadas safras;
    - Tempo de reprodução bacteriana;

Recomendação de máterial sobre series temporais:
https://aprender.ead.unb.br/pluginfile.php/235342/mod_resource/content/1/CursoCompleto.pdf

# For e While

Executa a repetição de um bloco de instruções enquanto uma condição é verdadeira.

```
for (variable in vector) {
 ...
}

while (condition) {
 ...
}
```

# Exemplo de For e While para percorrer um vetor

```
for(i in 1:length(vetorA)){
        print(vetorA[i])
}
```

```
j <- 1
while(j <= length(vetorA)){
        print(vetorA[i])
        j = j + 1
}
```

# Condicionais

- Na linguagem R existem três tipos de condicionais, que ao serem atendidos o bloco de códigos entre as chaves “{...}” são executadas:
  - O comando “if()”

```
if (condição) {
  # comandos que será executado caso condição = TRUE
}
```

  - if() else

```
if (condição){
  # comandos que será executado caso condição = TRUE
} else {
  # comandos que será executado caso condição = FALSE
}
```

  - ifelse()

```
ifelse(condição, TRUE, FALSE)
```

# Condicionais Exemplos

- O comando “if()”

```
x <- 12
if (x < 18) {
        print(“X é menor que 18”)
}
```

- if() else

```
if (x < 18){
        print(“X é menor que 18”)
} else {
        print(“X é maior que 18”)
}
```

- ifelse()

```
ifelse(x < 18, print(“X é menor que 18”),print(“X é maior que 18”))
```

# Exercício C

- Criar um vetor com  elementos do tipo numeric;
- Criar um vetor com elementos do tipo caracter;
- Criar um For e percorrer todo o vetor;
- Criar um While e percorrer todo o vetor.

# Funções

- Grande vantagem da linguagem é a possibilidade de automatizar determinadas análises desta forma utilizando as funções internas do R;
- Contudo, ganha ainda mais flexibilidade e agilidade criando suas próprias funções.

Sintaxe de uma função:

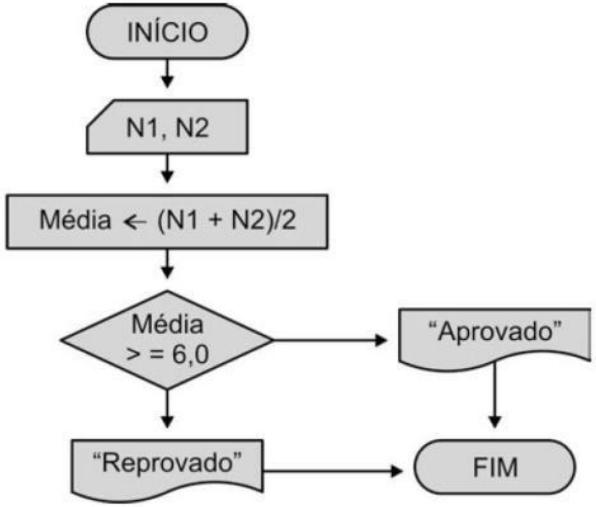
```
nome = function (argumento1 , … , argumento n) {
    Comandos da função
}
```

# Criando nossas primeiras Funções

```
quadrado <- function(x){
    return(x^2) # comando para retorna o valor
}

baskara <- function(a, b, c){
    valorA = sqrt( (b^2) - 4 * a * c )
    valorB = ((-b) + valorA) / (2 * a)
    valorC = ((-b) - valorA) / (2 * a)
    return( list(valorB, valorC) )
}
```

# Exercício D



# Gráficos de visualizações - Plot

Gráficos simples

Conjunto de gráficos normais:

```
x <- 1:100 # cria os valores x para os gráficos
y <- x^2 # cria os valores y para os gráficos (como x^2)

plot(x, y) # gráfico simples com os pontos
plot(x, y, type = "l") # gráfico simples com linha
plot(x, y, type = "b") # gráfico simples com pontos e linhas
```

# Gráficos de visualizações - Plot

Gráficos simples limite entre os eixos

```
# Gráfico com x entre 0 e 10, e y entre 0 e 20
plot(x, y, xlim = c(0, 10), ylim = c(0, 20))

#Nome nos eixos e no título
plot(x, y, xlab=”valores X (cm)”, ylab=”valores y”, main=”Título”, col=”red”)
```

# Gráficos de visualizações - Plot

Gráficos simples limite entre os eixos

```
# Gráfico com x entre 0 e 10, e y entre 0 e 20
plot(x, y, xlim = c(0, 10), ylim = c(0, 20))

#Nome nos eixos e no título
plot(x, y, xlab=”valores X (cm)”, ylab=”valores y”, main=”Título”, col=”red”)
```

# Gráficos de visualizações - Histogramas

Histogramas

```
x <- rnorm(n = 2000)
hist(x, main = “Título”, xlab=”dados”, ylab=”frequências absolutas”)
```

Alteração das cores das bordas e o fundo do gráfico

```
hist(x, main = “Título”, xlab=”dados”, ylab=”frequências absolutas”, col="darkblue", border="black");
```

# Gráficos de visualizações - Histogramas

Se preferir apresentar a densidade deve acrescentar o argumento prob como sendo verdadeiro;
hist(x, main = “Título”, xlab=”dados”, ylab=”frequências absolutas”, col="darkblue", border="black", prob=TRUE);

Para acessar a média basta acrescentar uma linha de código seguinte à linha em que o histograma é criado:
#Mediana
hist(x, main = “Título”, xlab=”dados”, ylab=”frequências absolutas”, col="darkblue", border="black", prob=TRUE);
abline(v=median(x), col="green",lwd=2)

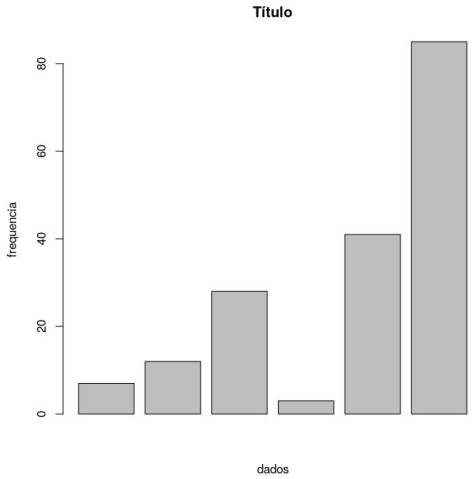
# Gráficos de visualizações - Histogramas

```
#Media
hist(x, main = “Título”, xlab=”dados”, ylab=”frequências absolutas”, col="darkblue",
border="black",                                                    prob=TRUE);
abline(v=mean(x), col = "red", lwd = 2);

Acrescentando                quadro                com                legenda
legend(x="topright",              #posicao              da              legenda
      c("Mediana","Média"),              #nomes              da              legenda
      col=c("blue","red"),                                                  #cores
      lty=c(1,2),                #estilo                da                linha
      lwd=c(2,2)) #grossura das linhas
```

# Gráficos de visualizações - Bar Charts

```
x <- c(7, 12, 28, 3, 41)
barplot(x, main="Título", xlab="dados", ylab="frequência")
```



# Gráficos de visualizações - Bar Char e Stacked Bar Char

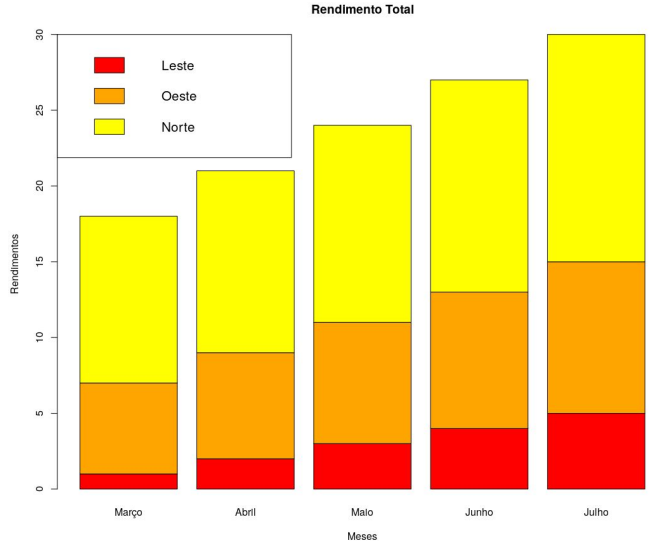
```
#Criação de entrada dos vetores
Cor <- c("Red", "Orange", "Yellow")
Mes = c("Março", "Abril", "Maio", "Junho", "Julho")
Regiao = c("Leste", "Oeste", "Norte")

#Criar matrizes de valores
Valores = matrix(data=1:15, nrow=3, ncol=5, byrow = TRUE)

#Criar o barchar
barplot(Valores, main = "Rendimento Total", names.arg = Mes, xlab = "Meses",
ylab = "Rendimentos", col = Cor)
```

# Gráficos de visualizações - Bar Char e Stacked Bar Char

```
#Legenda para o chart
legend("topleft", Regiao, cex = 1.3, fill = Cor)
```

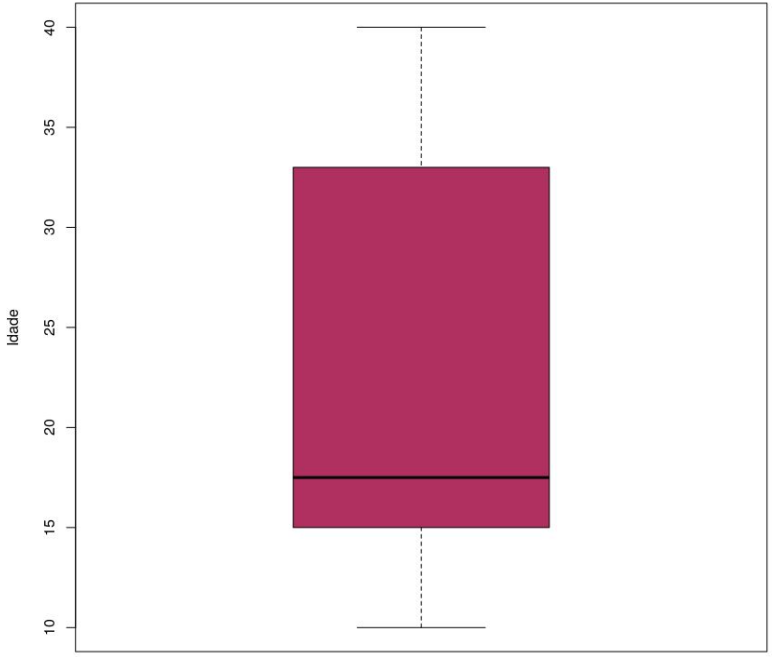


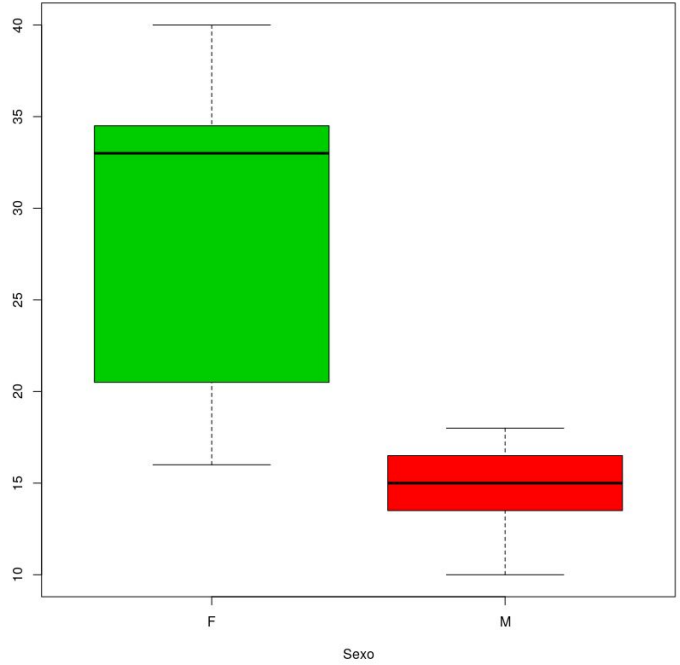

Prof: Rodrigo Nascimento

# Gráficos de visualizações - Boxplot

```
Idade = c(15, 16, 13, 18, 10, 17, 14, 16, 20, 21, 35, 34, 33, 40)
Peso = c(53, 44, 44, 40, 50, 55, 48, 49, 65, 75, 70, 56, 46, 47)
Sexo = rep(c("M", "F"), c(7,7))
data = data.frame(Idade, Peso, Sexo)
data
boxplot(data$Idade, col = "maroon", ylab="Idade")
boxplot(data$Idade~data$Sexo, col=27:2, lwd=1, xlab="Sexo")
```

# Gráficos de visualizações - Boxplot



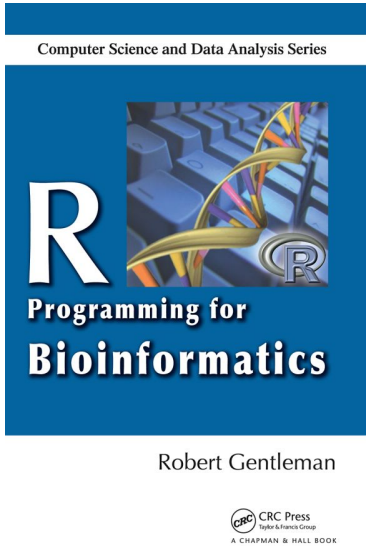



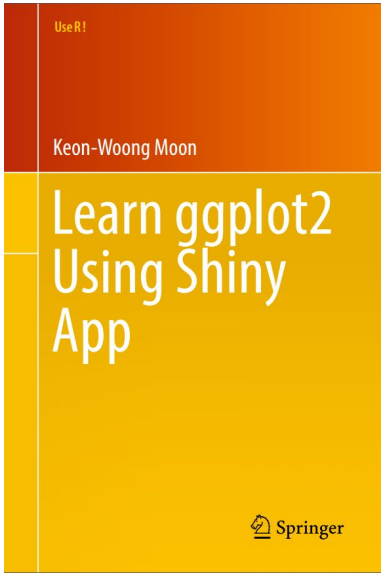
# Sites importantes

- https://www.kaggle.com/
- https://archive.ics.uci.edu/
- https://toolbox.google.com/datasetsearch/
- https://cran.r-project.org/web/packages/
- https://rdrr.io/
- https://www.bioconductor.org/

# Referências bibliográficas


- GENTLEMAN, Robert. **R programming for bioinformatics**. Chapman and Hall/CRC, 2008.
- NAVARRO, Omar Trejo. **R Programming By Example: Practical, hands-on projects to help you get started with R**. Packt Publishing Ltd, 2017.
- MOON, Keon-Woong. **Learn ggplot2 using shiny App**. Springer, 2017.

# Obrigado!

Contato:

E-mail: nascimento.rodrigo@hotmail.com.br