

capa

intro



Relatório aula de Laboratório #1

Configuração da linguagem R e do ambiente de trabalho

2º semestre, 2023/2024 - 28 de fevereiro



**Autores:**

**18364 Nuno Gomes**

**46638 Ricardo Ramos**

**47663 Rafael Carvalho**

Índice

[Introdução 2](#_Toc160052152)

[Instruções na linguagem R 3](#_Toc160052153)

[i. Escalares e Operadores. 3](#_Toc160052154)

[ii. Manipulação de vetores. 3](#_Toc160052155)

[iii. Manipulação de matrizes. 3](#_Toc160052156)

[iv. Manipulação de data frames. 3](#_Toc160052157)

[v. Manipulação de fatores. 4](#_Toc160052158)

[vi. Algumas funções úteis. 4](#_Toc160052159)

[vii. Pacotes. 4](#_Toc160052160)

[viii. Estatísticas descritivas básicas e mais. 4](#_Toc160052161)

[Configuração do Spark. 6](#_Toc160052162)

[Conclusao Referencias Anexos 7](#_Toc160052163)

# Introdução

A análise de grandes conjuntos de dados para extrair informações valiosas e padrões significativos é o foco da disciplina de Big Data Mining. Com o aumento exponencial do volume de dados na era digital, tornou-se essencial contar com ferramentas e tecnologias especializadas para lidar eficientemente com esses vastos conjuntos de informações.

O Apache Spark destaca-se como uma ferramenta poderosa no contexto do Big Data, proporcionando um ambiente de computação em cluster rápido e versátil para o processamento de dados. A linguagem de programação R é amplamente reconhecida no âmbito da estatística e análise de dados, sendo uma escolha popular devido à sua sintaxe amigável e à extensa coleção de pacotes, especialmente para quem trabalha com dados em grande escala.

Neste trabalho introdutório à unidade curricular, pretende-se testar, ensaiar e familiarizar quer com o ambiente Spark quer com a linguagem de programação R, constituindo estes o alicerce para os demais trabalhos e desenvolvimento das aprendizagens da unidade curricular.

Em síntese, o relatório subdivide-se em dois temas: Instruções na linguagem R, onde é abordado a prática de implementação de instruções R básicas, essenciais, e configuração do Spark, onde, sucintamente são descritos os resultados da aplicação do código.

## Instruções na linguagem R

### i. Escalares e Operadores.

1. var1 <− 3 # Atribui o valor 3 à variável var1

2. show( var1 ) # Mostra o valor da variável var1

3. var2 <− var1 ∗ var1 # Atribui à variável var2 o quadrado do valor de var1

4. var3 <− var1 ∗∗ 2 # Atribui à variável var3 o valor de var1 elevado ao quadrado

5. var4 <− var1 ˆ 2 # (^ operador de potência) Atribui à variável var4 o valor de var1 elevado ao quadrado

6. var1 < var1 # Retorna o resultado booleano da comparação do valor de var1 com o valor de var1

7. var3 != var4 # Verifica se var3 é diferente de var4 e retorna TRUE ou FALSE

8. var2 == var4 # Verifica se var2 é igual a var4 e retorna TRUE ou FALSE

9. var2 <− var2 − var2 # Subtrai o valor atual de var2 do próprio var2, resultando em 0 e guarda em var2

10. var5 <− var3 / var2 # Divide var3 pelo valor de var2 e atribui o resultado à variável var5

11. var5 + 1 # Adiciona 1 ao valor de var5 e retorna o resultado

### ii. Manipulação de vetores.

|  |  |
| --- | --- |
| 1. a <- c(1, 2, 5.3, 6, -2, 4, 3.14159265359)  2. a  3. b <- c("1", "2", "3")  4. "2" %in% b  5. "5" %in% b  6. c <- c(TRUE, TRUE, FALSE, TRUE)  7. a <- c  8. a[0]  9. a[1]  10. a[-1]  a[-3]  11. a[8]  12. a[c(1, 3)]  13. a[c(3, 1)] | # Cria um vetor chamado 'a' com os valores especificados  # Mostra o vetor 'a'  # Cria um vetor de caracteres chamado 'b'  # Verifica se "2" está presente no vetor 'b'  # Verifica se "5" está presente no vetor 'b'  # Cria um vetor lógico chamado 'c'  # Atribui o vetor 'c' a 'a'  # Tenta aceder ao elemento na posição 0 de 'a' (o índice deve começar em 1)  # Acede ao primeiro elemento de 'a'  # Apresenta todos os valores exceto o primeiro elemento de 'a'  # Apresenta todos os valores exceto o terceiro elemento de 'a'  # Acede ao oitavo elemento de 'a' (se existir)  # Seleciona os elementos nas posições 1 e 3 de 'a'  # Seleciona os elementos nas posições 3 e 1 de 'a' |

14. a[a>2]

1) a>2 # 1º gera um vetor booleano da operação logica a>2 aplicado a cada elemento de a;

[FALSE, FALSE, TRUE, TRUE,FALSE, TRUE TRUE]

2) a[a>2] # 2º este vetor logico indexa o vetor a retornando só os elementos cuja posição é “true”

### iii. Manipulação de matrizes.

1. m <- matrix(1:6, nrow=3, ncol=2) # Cria uma matriz 3x2 com valores de 1 a 6

2. show(m) # Mostra a matriz 'm'

3. n <- matrix(2:7, nrow=2, ncol=3) # Cria uma matriz 2x3 com valores de 2 a 7

4. n # Mostra a matriz 'n'

5. m[, 2] # Seleciona a segunda coluna da matriz 'm'

6. n[1, ] # Seleciona a primeira linha da matriz 'n'

7. m[2:3, 1:2] # Seleciona as linhas 2 e 3 e as colunas 1 e 2 da matriz 'm'

8. n %\*% m # Multiplica as matrizes 'n' e 'm'

9. m %\*% n # Multiplica as matrizes 'm' e 'n'

10. n %\*% n # Multiplica a matriz 'n' por ela mesma

11. nˆ2 # Eleva cada elemento da matriz 'n' ao quadrado

12. sqrt(n) # Calcula a raiz quadrada de cada elemento da matriz 'n'

### iv. Manipulação de data frames.

1. d <- c(1, 2, 3, 4) # Cria um vetor 'd' com valores de 1 a 4

2. e <- c("Bob", "Alice", NA, "Joe") # Cria um vetor ‘e’ de strings

3. f <- c(TRUE, TRUE, FALSE, TRUE) # Cria um vetor lógico 'f'

4. my.data <- data.frame(d, e, f, stringsAsFactors=FALSE) # Cria um data frame chamado 'my.data'

5. show(my.data) # Mostra o conteúdo do data frame 'my.data'

6. names(my.data) <- c("ID", "Name", "Passed") # Renomeia as colunas do data frame 'my.data'

7. View(my.data) # Visualização da matriz do data frame 'my.data'

8. my.data$Name # Seleciona a coluna 'Name' do data frame 'my.data'

9. my.data <- cbind(my.data, Failed=!f) # Adiciona uma nova coluna 'Failed' ao data frame 'my.data'

10. View(my.data) # Mostra o data frame 'my.data' após a alteração

11. my.data <- rbind(my.data, c(5, "Carol", FALSE, TRUE)) # Adiciona uma nova linha ao data frame 'my.data'

12. View(my.data) # Mostra o data frame 'my.data' após a alteração

13. ?data.frame # Abre a documentação para a função data.frame

### v. Manipulação de fatores.

1. colour <- c(rep("red", 20), rep("blue", 30)) # Cria um vetor chamado 'colour' com repetição de "red" 20 vezes e "blue" 30 vezes

2. colour <- factor(colour) # Converte 'colour' num fator apresentado todos os valores diferentes presentes

3. summary(colour) # Apresenta um resumo estatístico do fator 'colour'

4. dimensions <- c("large", "medium", "small") # Cria um vetor de strings chamado 'dimensions'

5. show(dimensions) # Mostra o vetor 'dimensions'

6. dimensions <- ordered(dimensions) # Converte 'dimensions' num fator ordenado, a ordenação é feita alfabéticamente

7. show(dimensions) # Mostra o vetor 'dimensions' após a alteração

### vi. Algumas funções úteis.

1. length(colour) # Retorna o comprimento (número de elementos) do vetor 'colour'

2. length(a) # Retorna o comprimento do vetor 'a'

3. class(colour) # Retorna a classe do objeto 'colour'

4. class(dimensions) # Retorna a classe do objeto 'dimensions'

5. class(a) # Retorna a classe do objeto 'a'

6. class(my.data) # Retorna a classe do objeto 'my.data'

7. nrow(my.data) # Retorna o número de linhas do data frame 'my.data'

8. ncol(my.data) # Retorna o número de colunas do data frame 'my.data'

9. str(my.data) # Apresenta a estrutura do data frame 'my.data'

10. sessionInfo() # Mostra informações sobre a sessão do R

11. ls() # Lista os objetos no ambiente de trabalho

12. rm(a) # Remove o objeto 'a' do ambiente de trabalho

13. glimpse(my.data) # Apresenta uma visão resumida do data frame 'my.data'

### vii. Pacotes.

1. library() # Mostra os pacotes atualmente carregados

2. search() # Mostra a lista de pacotes disponíveis

3. library("MASS") # Carrega o pacote "MASS"

4. search() # Mostra a lista de pacotes após a carga do "MASS"

5. .libPaths() # Mostra os caminhos do diretório onde os pacotes são instalados

6. install.packages("e1071") # Instala o pacote "e1071"

7. install.packages("funModeling") # Instala o pacote "funModeling"

8. # (deliberadamente em branco)

9. x <- c("MASS", "dplyr", "e1071") # Cria um vetor 'x' com os nomes dos pacotes

10. lapply(x, require, character.only=TRUE) # Carrega os pacotes usando *lapply*

11. # (deliberadamente em branco)

12. require(funModeling) # Carrega o pacote "funModeling"

### viii. Estatísticas descritivas básicas e mais.

1. iris # Retorna o conjunto de dados 'iris'

2. summary(iris) # Apresenta um resumo estatístico do conjunto de dados 'iris'

3. fivenum(iris$Sepal.Length) # Calcula os cinco números resumidos para a variável Sepal.Length em 'iris'

4. # (deliberadamente em branco)

5. status(iris) # Chama a função 'status' com o conjunto de dados 'iris'

**(c)**

**i. Defina um ponto de interrupção na linha 6 e execute o código delimitado pela região EX.1, passo a passo.**

**ii. Explique o propósito dos três exemplos chamados EX.1, EX.2 e EX.3.**

**(d) Implemente e mostre um código que obtém as duas primeiras colunas de my.data, usando a indexação do operador de intervalo.**

my.data[, range(1,2)]

**(e) Implemente e mostre um código que obtém as duas primeiras colunas de my.data, usando a indexação vetorial.**

my.data[, 1:2]

**(f) Instale o pacote Hmisc e use a função describe na variável my.data.**

## Configuração do Spark.

(a) Instale o pacote sparklyr usando install.packages("sparklyr"). Pode demorar alguns minutos, tenha paciência.

(b) Instale o Spark usando library(sparklyr) spark\_install(version = ’3.4.2’, hadoop\_version = ’3’).

4. Introdução ao Spark.

Descreva os comandos usados (se não listados), os problemas (se houver) e os resultados dos seguintes pontos, linha

por linha:

(a) Verifique programaticamente se o pacote sparklyr está carregado. Em seguida, conecte-se ao Spark usando

ss <- spark\_connect('local', version = '3.4.2', hadoop\_version = '3', config = list()).

(b) Veja o conteúdo da variável ss.

(c) Use a função copy do pacote dplyr para copiar o conjunto de dados iris para o Spark.

1. library(dplyr)

2. df <− copy\_to(ss, iris)

3. show(df)

(d) Mostre alguns dados amostrais do conjunto de dados carregado.

1. head(select(df, Petal.Width, Species))

2. head(filter(df, Petal.Width > 0,3)

3. df %>% head

(e) Utilizando SQL

1. library (DBI)

2. df\_sql <− dbGetQuery ( ss , ”SELECT \* FROM iris WHERE Petal\_Width > 0 . 3 LIMIT 5” )

3. show ( df\_sql )

Falta o resto que não tive tempo

# Conclusão

*A simplicidade da linguagem R complementada pela capacidade de processamento distribuído do Spark, proporciona uma poderosa solução para explorar padrões complexos e extrair insights valiosos em ambientes de Big Data, contribuindo para uma análise robusta e informada.*

# Referencias

<https://didatica.tech/a-linguagem-r/>

https://pedropark99.github.io/Introducao\_R/no%C3%A7%C3%B5es-b%C3%A1sicas-do-r.html