MAC 113 — Introdução à Ciência da Computação

Aula 24

Nelson Lago

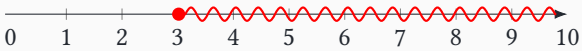
1°/2025

Previously on MAC113...

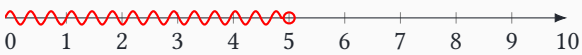
Booleanos e conjuntos

Às vezes vale a pena entender o operador “and” como “intersecção de conjuntos”
(e o operador “or” como “união de conjuntos”)

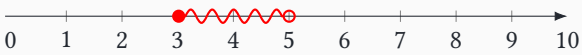
$3 \leq$ nota



nota $<$ 5



$3 \leq$ nota && nota $<$ 5



Álgebra booleana

Propriedades Comutativas	A and B = B and A	A or B = B or A
Propriedades Distributivas	A and (B or C) = (A and B) or (A and C)	A or (B and C) = (A or B) and (A or C)
Propriedades Associativas	(A or B) or C = A or (B or C)	(A and B) and C = A and (B and C)
Propriedades Idempotentes	A and A = A	A or A = A
Dupla Negação	not not A = A	
Elementos Absorventes	A or True = True	A and False = False
Elementos Neutros	A or False = A	A and True = A
Leis de De Morgan	not (A or B) = (not A) and (not B)	not (A and B) = (not A) or (not B)

Álgebra booleana

- **pizza**
- **sushi**
- **moqueca**
- **hambúrguer**

- **Sou guloso**
  - pizza **ou** hambúrguer
- **Sou alérgico a peixes**
  - **nem** sushi **nem** moqueca

Equivalentes! Qual usar?

O que facilita o entendimento

Álgebra booleana

- **Leis de De Morgan:**
  - not (A or B) = (not A) and (not B)
  - not (A and B) = (not A) or (not B)

- **nem sushi nem moqueca** — `(! sushi) && (! moqueca)`
- **não quero se for (sushi ou moqueca)** — `! (sushi || moqueca)`

- **Topa jantar e depois cinema?**
  - Não tenho grana para tudo isso!
    - » `! (jantar && cinema)`
  - Tem que abrir mão de (pelo menos) um deles
    - » `(! jantar) || (! cinema)`

Comandos básicos com vectors e lists

- `vec <- 1:10`
- `vec <- seq_len(4)`
- `vec <- rep(3.14, 4)`
- `vec <- c("a", "b", "c")`
- `lista <- list(a=1, b=2, c=3)`
- `length(blah)` (list ou vector)
- `tudo_junto <- c(blah, bleh)` (lists ou vectors)
- `vec[X]`
- `lista[[X]]` (numérico)
- `blah[X:Y]` (numérico – lists ou vectors)
- `lista[[X]]` (character)
- `lista$X` (character, atalho sem aspas)
- `for (item in blah)` (list ou vector)
- `names(lista)`
  - `for (nome in names(lista))`
- `unlist(lista)`
- `vec[X] <- valor` (atribui)
- `vec <- append(vec, valor)` (acrescenta)
- `lista$nome <- valor` (atribui ou acrescenta)

Exercício – Boletim escolar

Dada uma lista de alunos, faça um programa que imprime uma tabela com as notas de todos os alunos

```
library(glue)
alunos <- list(
  list(nome="Alan Turing", ID=1234, matemática=9.7,
       português=1.4, física=9.2, história=8.7),
  list(nome="Ada Lovelace", ID=4321, matemática=9.8,
       português=1.2, física=10, história=9.2)
)
disciplinas <- c("matemática", "português", "física", "história")
cat(format("Nome", width=15), format(disciplinas, width=10, justify="right"), "\n")
for (aluno in alunos) {
    notas <- aluno[disciplinas]
    cat(format(aluno$nome, width=15),
        format(unlist(notas), width=10, justify="right"))
    cat("\n")
}
```

```
Nome            matemática  português     física   história
Alan Turing            9.7        1.4        9.2        8.7
Ada Lovelace           9.8        1.2       10.0        9.2
```

Tabelas

Paciente	Colesterol (mg/dL)	Idade	Quarto	SUS
Fulano de Tal	164	49	132-A	S
Ciclano de Tal	227	83	231-B	N
Média/Total	195,6	66	–	1

- **Por que usamos tabelas?**
- **Provavelmente, várias razões** 😜
- **Uma delas é que se trata de uma visualização dupla**
- **Podemos olhar cada linha ou cada coluna**

Tabelas

- **Em tabelas desse tipo**
  - Cada linha corresponde a uma abstração (“pessoa”, “produto” etc.)
  - Cada coluna corresponde a uma coleção de valores de um mesmo tipo (“preço”, “colesterol”, “idade” etc.)
- **Podemos armazenar cada linha como uma list**
  - Mas processar as colunas (por exemplo, para calcular a média) é trabalhoso
- **Podemos armazenar cada coluna como um vector e usar os índices para identificar as linhas**
  - Mas já vimos antes que isso é um tanto trabalhoso também
- **Podemos usar uma list em que cada elemento é um vector correspondente a uma coluna**
  - Mas isso é apenas uma variação da ideia anterior

Precisamos de algo novo!

Algo que se assemelhe a uma tabela

Dataframes

(na verdade, tibbles)

Dataframes

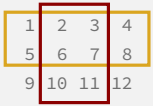
- **Dataframes são bastante similares a lists**
- **Cada item é um vector que representa uma coluna**

```
library(tibble)
#df <- data.frame(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- tibble(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
cat("Alunos:", df$nome, "\n")
cat("Média da turma:", mean(df$nota), "\n")
cat("Lista de chamada:\n")
cat(paste(df$nome, " (", df$ID, ") ______", sep=""), sep="\n")
```

```
Alunos: Zé Fê Má Lú
Média da turma: 8.225
Lista de chamada:
Zé (1) ______
Fê (2) ______
Má (3) ______
Lú (4) ______
```

Recortes com dataframes

- **A maneira mais razoável de nos referirmos a elementos em uma tabela é através de suas coordenadas**



1	2	3	4
5	6	7	8
9	10	11	12

- **tabela**$_{l,c}$
- **tabela**$_{1-2,2-3}$

```
2 3
6 7
```

Dataframes

```r
library(tibble)
#df <- data.frame(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- tibble(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
cat("Boletim de notas:\n")
for (i in seq_len(nrow(df))) {
  line <- df[i,]
  cat(unlist(line[c("nome", "nota")]), "\n")
}
```

```
Boletim de notas:
Zé 7
Fê 8.5
Má 8.3
Lú 9.1
```

Dataframes

- **Em geral, as operações vetoriais que fazemos em dataframes/tibbles processam as colunas**
  - Como cada coluna é um vector, sabemos que os elementos são todos do mesmo tipo
  - Em geral, coisas como `max()`, `mean()`, `sum()` etc., fazem sentido com as colunas
- **Para processar as linhas, podemos extrair uma linha específica por sua coordenada:**
  - `df[3,]`
- **ou, para processar todas, fazer um laço:**
  - `for (i in seq_len(nrow(df))) { df[i,] }`
- **ou usar `paste()`**
- **...**

Dataframes

```
library(tibble)
#df <- data.frame(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
df <- tibble(nome=c("Zé", "Fê", "Má", "Lú"), ID=c(1,2,3,4), nota=c(7, 8.5, 8.3, 9.1))
cat("Boletim de notas:\n")
for (i in seq_len(nrow(df))) {
  line <- df[i,]
  cat(unlist(line[c("nome", "nota")]), "\n")
}
```

```
Boletim de notas:
Zé 7
Fê 8.5
Má 8.3
Lú 9.1
```

Dataframes

Dataframes/tibbles: Como vivem? Onde moram? De que se alimentam?

- **Não tem muita graça digitar os dados de um dataframe/tibble como parte do programa**
- **O mais razoável é carregar de um arquivo**
- **Dataframes são tabelas → o arquivo deve representar uma tabela**
- **O formato mais comumente usado é o CSV (comma-separated values)**

Nome	matemática	português	física	história
Alan Turing	9.7	1.4	9.2	8.7
Ada Lovelace	9.8	1.2	10.0	9.2

Dataframes

```
Nome,matemática,português,física,história
Alan Turing,9.7,1.4,9.2,8.7
Ada Lovelace,9.8,1.2,10.0,9.2
```

```
Nome          ,matemática,português,física,história
Alan Turing   ,9.7       ,1.4      ,9.2   ,8.7
Ada Lovelace  ,9.8       ,1.2      ,10.0  ,9.2
```

- **CSV: "Comma" pode ser qualquer coisa (espaços, tabs...)**

```
Nome;matemática;português;física;história
Alan Turing;9,7;1,4;9,2;8,7
Ada Lovelace;9,8;1,2;10,0;9,2
```

Dataframes

- **E como ler um arquivo desses?**

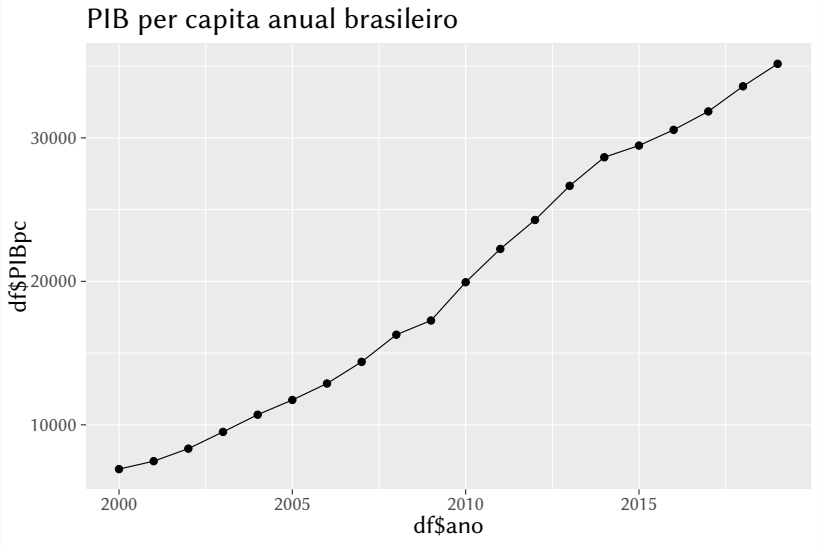
```r
endereço <- 'http://www.ime.usp.br/~lago/dadinhos/renda_vs_inflacao.csv'
#df <- read.table(endereço, sep=",") # dataframe
library(readr)
df <- read_delim(endereço, delim=",", col_types = "idd")
names(df)[2] <- "PIBpc"
print(head(df))
```

```
# A tibble: 6 × 3
    ano  PIBpc  IPCA
  <int>  <dbl> <dbl>
1  2000  6913.   7
2  2001  7467    6.8
3  2002  8341.   8.4
4  2003  9507.  14.7
5  2004 10706    6.6
6  2005 11734.   6.9
```

Dataframes

```
endereço <- 'http://www.ime.usp.br/~lago/dadinhos/renda_vs_inflacao.csv'
#df <- read.table(endereço, sep=",") # dataframe
library(readr)
df <- read_delim(endereço, delim=",", col_types = "idd")
names(df)[2] <- "PIBpc"
p <- ggplot(NULL, aes(x=df$ano, y=df$PIBpc)) +
         geom_line() +
         geom_point() +
         labs(title="PIB per capita anual brasileiro")
print(p)
```

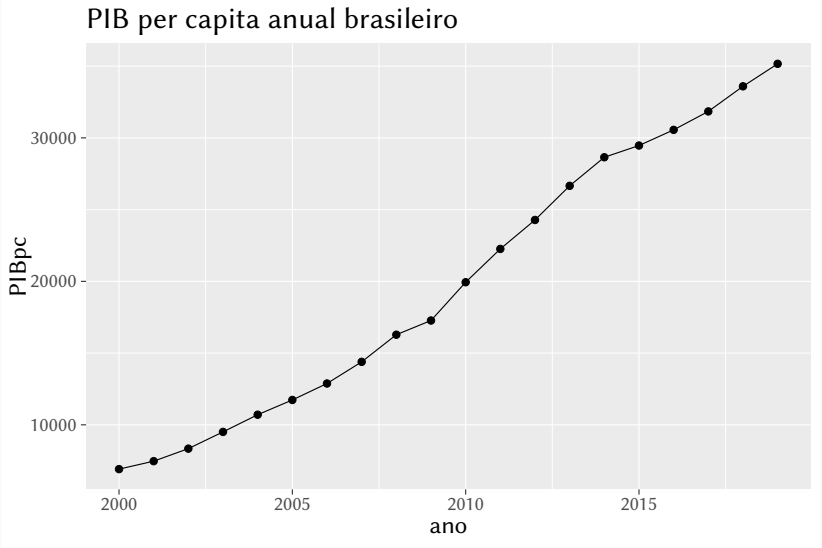
Dataframes


PIB per capita anual brasileiro
30000
20000
10000
df$PIBpc
2000
2005
2010
2015
df$ano

Dataframes

```
endereço <- 'http://www.ime.usp.br/~lago/dadinhos/renda_vs_inflacao.csv'
#df <- read.table(endereço, sep=",") # dataframe
library(readr)
df <- read_delim(endereço, delim=",", col_types = "idd")
names(df)[2] <- "PIBpc"
p <- ggplot(df, aes(x=ano, y=PIBpc)) +
         geom_line() +
         geom_point() +
         labs(title="PIB per capita anual brasileiro")
print(p)
```

Dataframes


PIB per capita anual brasileiro
PIBpc
30000
20000
10000
2000
2005
2010
2015
ano

Exercício — cálculo de notas

Escreva um programa que leia um arquivo CSV onde cada linha contém o nome e as notas de um estudante e, para cada estudante, calcule e imprima a média das notas. Ao final, o programa deve imprimir também a média da turma.

Exercício — cálculo de notas

Exemplo: para o arquivo http://www.ime.usp.br/~lago/dadinhos/notas.csv, similar a isto:

```
Nome,           P1, P2, T1, T2
José Augusto,  10, 15, 20, 30
Pedro Mé,      23, 16, 19, 22
Suzana Ré,      8, 22, 17, 14
Gisela Fox,    12, 28, 21, 45
João Mendonça, 14, 32, 25, 16
```

a saída deve ser:

```
José Augusto  : 18.75
Pedro Mé      : 20
Suzana Ré     : 15.25
Gisela Fox    : 26.5
João Mendonça : 21.75
Média da turma: 20.45
```

Exercício — cálculo de notas

```
library(readr)
main <- function() {
    end <- 'http://www.ime.usp.br/~lago/dadinhos/notas.csv'
    df <- read_delim(end, delim=",", col_types="cnnnn")
    for (i in seq_len(nrow(df))) { df[i,'média'] <- mean(unlist(df[i,2:5])) }
    cat(paste(format(df$Nome, width=14), ": ", df$média, sep=""), sep="\n")
    cat("Média da turma:", mean(df$média), "\n")
}
main()
```