MAC 113 — Introdução à Ciência da Computação

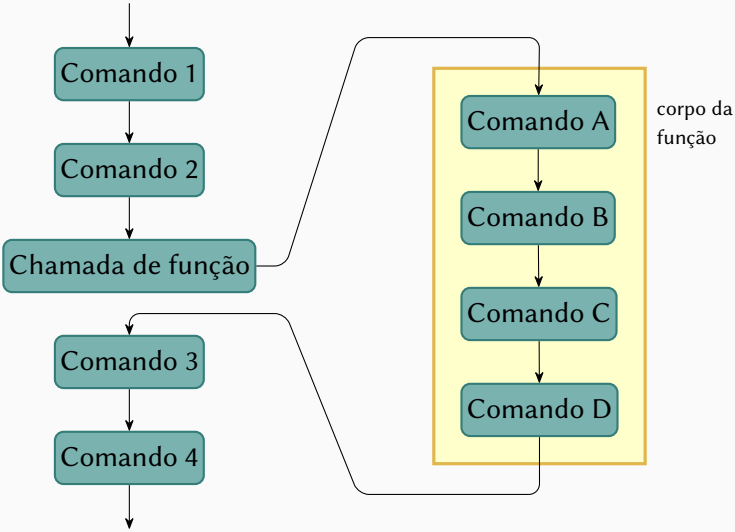
Aula 13

Nelson Lago

1°/2025

Previously on MAC113...

Funções são “desvios”



Coleções

Há três tipos principais de coleções:

- **Listas de coisas "iguais"**
  - Nomes de pacientes
  - Salários de funcionários
  - PIB anual
  - Objetos para comprar
- **Composição de dados formando uma abstração**

  Pessoa: nome, endereço, estado civil, CPF...
  Pessoa: nome, colesterol, cor do cabelo...

  País: capital, PIB per capita, área cultivável...
  Personagem: inteligência, força, destreza...
- **Tabelas (mistura dos dois)**

Coleções

A primeira coleção que vamos ver é o vector:

```
cores <- c("vermelho", "azul", "amarelo")
```

Um vector é um conjunto ordenado:

```
cat(cores[1], "\n")
cat(cores[3], "\n")
cores[2] <- "verde"
cat(cores[2], "\n")
```

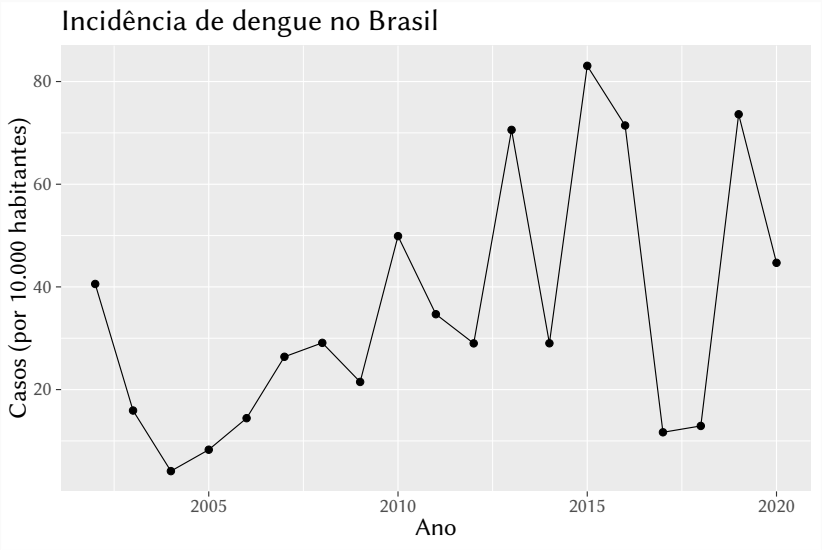
```
vermelho
amarelo
verde
```

- **Todos os elementos de um vector precisam ser do mesmo tipo**
  - Usamos vectors para representar coleções de coisas "iguais"

Tudo em todo lugar ao mesmo tempo

```
main <- function() {
  tmp <- read.csv("http://www.ime.usp.br/~lago/dadinhos/casos-dengue.csv")
  casos <- tmp[[1]]
  tmp <- read.csv("http://www.ime.usp.br/~lago/dadinhos/anos-dengue.csv")
  anos <- tmp[[1]]
  p <- ggplot(NULL, aes(x=anos, y=casos)) +
       geom_line() +
       geom_point() +
       labs(title="Incidência de dengue no Brasil", x="Ano",
            y="Casos (por 10.000 habitantes)")
  print(p)
}
main()
```

matplotlib


Incidência de dengue no Brasil
Casos (por 10.000 habitantes)
80
60
40
20
2005
2010
2015
2020
Ano

and now for something slightly different

podemos usar mais variáveis no laço além da variável de controle

Indicadores de passagem

Dada uma lista com a idade dos alunos de MAC113, imprima os elementos da lista, um em cada linha e diga se há algum menor de idade na turma

```
idades <- c(18, 18, 19, 18, 20, 18, 19, 17, 18, 19)
menores <- FALSE
i <- 1
while (i <= length(idades)) {
    if (idades[i] < 18) { menores <- TRUE }
    cat("Idade:", idades[i], "\n")
    i <- i + 1
}
if (menores) {
    cat("Há menores de idade na turma", "\n")
}
```

Indicadores de passagem

- **Chamamos uma variável similar a `menores` do exemplo anterior de indicador de passagem**
- **O uso de indicadores de passagem é um padrão de programação bastante comum**
  - Padrões de programação são técnicas (ou truques!) úteis e, portanto, comuns, porém não óbvios
- **Um indicador de passagem é usado para indicar se "alguma coisa" aconteceu durante o laço**
  - Quando usamos um indicador de passagem, não estamos preocupados com **qual** elemento do laço causou o evento
  - Na verdade, pode haver um ou mais elementos responsáveis por essa mudança; o valor do indicador de passagem é alterado no máximo uma vez

Indicadores de passagem

- **O indicador de passagem é diferente da variável de controle do laço**
  - A variável de controle **sempre** muda de valor **no mínimo** uma vez (na última repetição) para indicar o fim do laço
  - O indicador de passagem muda de valor **zero ou uma vezes** e a mudança pode acontecer em qualquer uma das iterações
    - » *(o comando que altera o valor do indicador de passagem pode ser executado mais de uma vez, mas o valor da variável só é efetivamente alterado na primeira)*

Exercício — contagem de dígitos

Dado um natural n ≥ 0 e um algarismo d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

E se n for zero?

```
n <- as.integer(readline("Digite o número n: "))
d <- as.integer(readline("Digite o algarismo d: "))
if (n == 0 && d == 0) { conta <- 1 }
val <- n
conta <- 0 # elemento neutro
while (val > 0) {
    este <- val %% 10
    if (este == d) {
        conta <- conta + 1
    }
    val <- val %/% 10
}
cat("O algarismo", d, "aparece", conta, "vezes em", n, "\n")
```

Exercício — Contando algarismos

Escreva uma função `contaAlgarismos(n, d)` que devolve o número de vezes que o algarismo `d` aparece no inteiro positivo `n` (zeros à esquerda obviamente não devem ser contados)

```
contaAlgarismos <- function(n, d) {
    if (n == 0 && d == 0) { conta <- 1 }
    val <- n
    conta <- 0 # elemento neutro
    while (val > 0) {
        este <- val %% 10
        if (este == d) {
            conta <- conta + 1
        }
        val <- val %/% 10
    }
    return(conta)
}
```

Exercício — Permutações

Um número *a* é uma *permutação* de um número *b* se é possível encontrar o número *b* reordenando os dígitos de *a*. Usando a função anterior, escreva um programa que lê dois números *a* e *b* e informa se *a* é uma permutação de *b*.

```
main <- function() {
    a <- as.integer(readline("Digite o valor de a: "))
    b <- as.integer(readline("Digite o valor de b: "))
    if (permutação(a, b)) {
        cat(glue("{a} é uma permutação de {b}"), "\n")
    } else {
        cat(glue("{a} não é uma permutação de {b}"), "\n")
    }
}
permutação <- function(a, b) {
    perm <- TRUE
    n <- 0
    while (n <= 9) {
        if (contaAlgarismos(a, n) != contaAlgarismos(b, n)) { perm <- FALSE }
        n <- n + 1
    }
    return(perm)
}
```

Indicadores de passagem

- **O indicador de passagem é diferente da variável de controle do laço**
  - A variável de controle **sempre** muda de valor **no mínimo** uma vez (na última repetição) para indicar o fim do laço
  - O indicador de passagem muda de valor **zero ou uma vezes** e a mudança pode acontecer em qualquer uma das iterações
    - » *(o comando que altera o valor do indicador de passagem pode ser executado mais de uma vez, mas o valor da variável só é efetivamente alterado na primeira)*
- **MAS...**
- **Às vezes podemos usar o indicador de passagem para terminar o laço antecipadamente**

Indicadores de passagem

```
idades <- c(18, 18, 19, 18, 20, 18, 19, 17, 18, 19)
menores <- FALSE
i <- 1
while (i <= length(idades) && ! menores) {
    if (idades[i] < 18) { menores <- TRUE }
    i <- i + 1
}
if (menores) {
    cat("Há menores de idade na turma", "\n")
}
```

Exercício — aniversários

Numa sala com n pessoas, estimar a probabilidade de que ao menos duas delas façam aniversário no mesmo dia usando o método de Monte Carlo

```
pessoas <- as.integer(readline('Número de pessoas: '))
experimentos <- as.integer(readline('Número de experimentos: '))
sucessos <- 0 # experimentos em que há aniversários repetidos
i <- 1
while (i <= experimentos) {
    dias_do_ano <- integer(365)
    achei <- FALSE
    j <- 1
    while (j <= pessoas) {
        dia <- sample(1:365, 1) # sorteia o aniversário da pessoa
        dias_do_ano[dia] <- dias_do_ano[dia] + 1 # incrementa o número de aniversariantes do dia
        if (dias_do_ano[dia] > 1) { # Pelo menos um aniversário repetido!
            achei <- TRUE
        }
        j <- j + 1
    }
    if (achei) {
        sucessos <- sucessos + 1
    }
    i <- i + 1
}
cat(glue('A probabilidade estimada é {100 * sucessos / experimentos}%'), "\n")
```

Para $n = 23$ a probabilidade é $\approx 51\%$

Funções

- O código está correto e não é longo
- MAS
- É um tanto complexo
- E se usarmos funções?

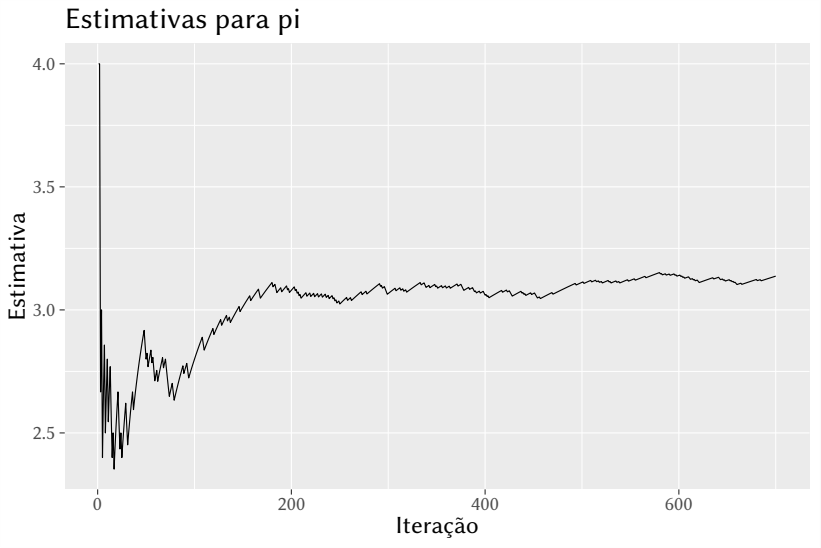
Exercício — aniversários

```r
checa_nivers_repetidos <- function(pessoas) {
    dias_do_ano <- integer(365)
    achei <- FALSE
    j <- 1
    while (j <= pessoas) {
        dia <- sample(1:365, 1) # sorteia o aniversário da pessoa
        dias_do_ano[dia] <- dias_do_ano[dia] + 1 # incrementa o número de aniversariantes do dia
        if (dias_do_ano[dia] > 1) { # Pelo menos um aniversário repetido!
            achei <- TRUE
        }
        j <- j + 1
    }
    return(achei)
}
```

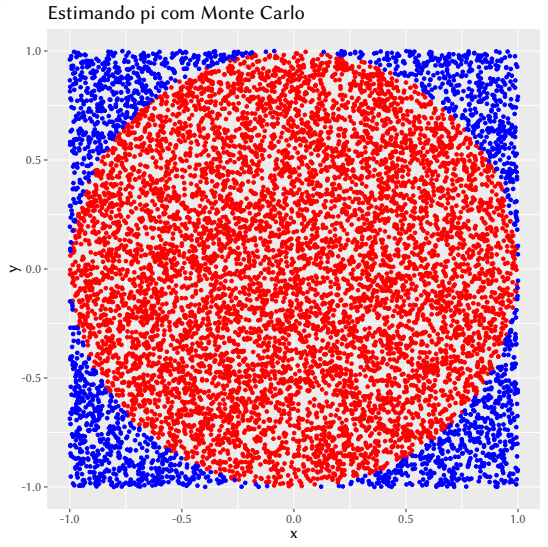
π e Monte Carlo, parte II

```
main <- function() {
    chutes <- as.integer(readline("Quantas amostras? "))
    dentro <- 0
    estimativas <- numeric(chutes)
    i <- 1
    while (i <= chutes) {
        x <- runif(1, -1, 1)
        y <- runif(1, -1, 1)
        if (x^2 + y^2 <= 1) { # dentro do círculo
            dentro <- dentro + 1
        }
        estimativas[i] <- 4 * dentro / i
        i <- i + 1
    }
    cat("Pi é aproximadamente", 4*dentro/chutes, "\n")
    p <- ggplot(NULL, aes(x=1:chutes, y=estimativas)) +
         geom_line() +
         labs(title="Estimativas para pi", x="Iteração", y="Estimativa")
   print(p)
}
main()
```

π e Monte Carlo, parte II



π e Monte Carlo, parte III



π e Monte Carlo, parte III

```
chutes <- as.integer(readline("Amostras? "))
x_círculo <- numeric(chutes)
y_círculo <- numeric(chutes)
x_quadrado <- numeric(chutes)
y_quadrado <- numeric(chutes)
dentro <- 0
fora <- 0
i <- 1
while (i <= chutes) {
  x <- runif(1, -1, 1)
  y <- runif(1, -1, 1)
  if (x^2 + y^2 <= 1) {
    dentro <- dentro + 1
    x_círculo[dentro] <- x
    y_círculo[dentro] <- y
  } else {
    fora <- fora + 1
    x_quadrado[fora] <- x
    y_quadrado[fora] <- y
  }
  i <- i + 1
}
```

```
x_círculo <- x_círculo[1:dentro]
y_círculo <- y_círculo[1:dentro]
x_quadrado <- x_quadrado[1:fora]
y_quadrado <- y_quadrado[1:fora]
p <- ggplot(NULL) +
  geom_point(aes(x=x_círculo, y=y_círculo),
             color="red", size=1) +
  geom_point(aes(x=x_quadrado, y=y_quadrado),
             color="blue", size=1) +
       labs(title="Estimativa de pi",
            x="x", y="y")
print(p)
```

Exercício — adivinhando números

Escreva um programa que tenta adivinhar um número de 1 a 10 escolhido pelo usuário (o usuário responde “<”, “>” ou “=” para indicar se o número correto é maior, menor ou igual ao número gerado pelo computador)

```
library(glue)
main <- function() {
    cat("Escolha um número de 1 a 10, vou tentar adivinhá-lo!", "\n")
    resposta <- "x" # qualquer coisa diferente de "="
    while (resposta != "=") {
        chute <- sample(1:10, 1)
        resposta <- readline(glue('Será {chute}? responda com "<", ">" ou "=" '))
    }
    cat("Aêh, sou vidente!", "\n")
}
main()
```

Exercício — adivinhando números

Melhore o programa anterior fazendo uma lista dos números já chutados, de maneira a não repetir tentativas.

```
cat("Escolha um número de 1 a 10, vou tentar adivinhá-lo!", "\n")
já_foram <- integer(10)
último <- 0
resposta <- "x" # qualquer coisa diferente de "="
while (resposta != "=") {
    repetido <- TRUE
    while (repetido) {
        repetido <- FALSE
        chute <- sample(1:10, 1)
        i <- 1
        while (i <= último) {
            if (chute == já_foram[i]) { repetido <- TRUE }
            i <- i + 1
        }
    }
    último <- último + 1
    já_foram[último] <- chute # O número certo também é incluído na lista!
    resposta <- readline(glue('Será {chute}? responda com "<", ">" ou "=" '))
}
cat("Aêh, sou vidente!", "\n")
```