MAC 113 — Introdução à Ciência da Computação

Aula 10

Nelson Lago

1°/2025

Previously on MAC 113...

Indentação

```
main <- function() {
…
while (condição) {
…
if (condição) {
…
} else {
…
}
…
}
…
}
```

Indentação

```
main <- function() {
…
while (condição) {
…
if (condição) {
…
} else {
…
}
…
}
…
}
```

- A **condição** do `if`/`while` fica entre parênteses

Indentação

```
main <- function() {
…
while (condição) {
…
if (condição) {
…
} else {
…
}
…
}
…
}
```

- A **condição** do `if`/`while` fica entre parênteses
- O **código** a ser executado ou não fica entre chaves

Indentação

```
main <- function() {
…
while (condição) {
…
if (condição) {
…
} else {
…
}
…
}
…
}
```

- A **condição** do `if`/`while` fica entre parênteses
- O **código** a ser executado ou não fica entre chaves
- Os **parâmetros** das funções ficam entre parênteses

Indentação

```
main <- function() {
…
while (condição) {
…
if (condição) {
…
} else {
…
}
…
}
…
}
```

- A **condição** do `if`/`while` fica entre parênteses
- O **código** a ser executado ou não fica entre chaves
- Os **parâmetros** das funções ficam entre parênteses
- O **código** das funções fica entre chaves

Indentação

```
main <- function() {
…
while (condição) {
…
if (condição) {
…
} else {
…
}
…
}
…
}
```

- A **condição** do `if`/`while` fica entre parênteses
- O **código** a ser executado ou não fica entre chaves
- Os **parâmetros** das funções ficam entre parênteses
- O **código** das funções fica entre chaves
- Em geral, o que vai entre parênteses é "curto", mas o que vai entre chaves pode ser bastante longo (várias linhas)

Indentação

```
main <- function() {
…
while (condição) {
…
if (condição) {
…
} else {
…
}
…
}
…
}
```

- A **condição** do `if`/`while` fica entre parênteses
- O **código** a ser executado ou não fica entre chaves
- Os **parâmetros** das funções ficam entre parênteses
- O **código** das funções fica entre chaves
- Em geral, o que vai entre parênteses é "curto", mas o que vai entre chaves pode ser bastante longo (várias linhas)
- É muito fácil para um humano se confundir com as chaves!

Indentação

```
main <- function() {
…
while (condição) {
…
if (condição) {
…
} else {
…
}
…
}
…
}
```

- A **condição** do `if`/`while` fica entre parênteses
- O **código** a ser executado ou não fica entre chaves
- Os **parâmetros** das funções ficam entre parênteses
- O **código** das funções fica entre chaves
- Em geral, o que vai entre parênteses é "curto", mas o que vai entre chaves pode ser bastante longo (várias linhas)
- É muito fácil para um humano se confundir com as chaves!

Indentação

```
main <- function() {
…
while (condição) {
…
if (condição) {
…
} else {
…
}
…
}
…
}
```

- A **condição** do `if`/`while` fica entre parênteses
- O **código** a ser executado ou não fica entre chaves
- Os **parâmetros** das funções ficam entre parênteses
- O **código** das funções fica entre chaves
- Em geral, o que vai entre parênteses é "curto", mas o que vai entre chaves pode ser bastante longo (várias linhas)
- É muito fácil para um humano se confundir com as chaves!
- A solução é a **indentação**: qualquer código entre chaves é deslocado para a direita

Indentação

```
main <- function() {
…
while (condição) {
…
if (condição) {
…
} else {
…
}
…
}
…
}
```

- A **condição** do `if`/`while` fica entre parênteses
- O **código** a ser executado ou não fica entre chaves
- Os **parâmetros** das funções ficam entre parênteses
- O **código** das funções fica entre chaves
- Em geral, o que vai entre parênteses é "curto", mas o que vai entre chaves pode ser bastante longo (várias linhas)
- É muito fácil para um humano se confundir com as chaves!
- A solução é a **indentação**: qualquer código entre chaves é deslocado para a direita
  - Qualquer tamanho, mas uma boa opção para **R** são 4 espaços

Indentação

```
main <- function() {
    …
    while (condição) {
        …
        if (condição) {
            …
        } else {
            …
        }
        …
    }
    …
}
```

- A **condição** do `if`/`while` fica entre parênteses
- O **código** a ser executado ou não fica entre chaves
- Os **parâmetros** das funções ficam entre parênteses
- O **código** das funções fica entre chaves
- Em geral, o que vai entre parênteses é "curto", mas o que vai entre chaves pode ser bastante longo (várias linhas)
- É muito fácil para um humano se confundir com as chaves!
- A solução é a **indentação**: qualquer código entre chaves é deslocado para a direita
  - Qualquer tamanho, mas uma boa opção para **R** são 4 espaços

Indentação

```
main <- function() {
    …
    while (condição) {
        …
        if (condição) {
            …
        } else {
            …
        }
        …
    }
    …
}
```

- A **condição** do `if`/`while` fica entre parênteses
- O **código** a ser executado ou não fica entre chaves
- Os **parâmetros** das funções ficam entre parênteses
- O **código** das funções fica entre chaves
- Em geral, o que vai entre parênteses é "curto", mas o que vai entre chaves pode ser bastante longo (várias linhas)
- É muito fácil para um humano se confundir com as chaves!
- A solução é a **indentação**: qualquer código entre chaves é deslocado para a direita
  - Qualquer tamanho, mas uma boa opção para **R** são 4 espaços

Indentação

```
main <- function() {
    …
    while (condição) {
        …
        if (condição) {
            …
        } else {
            …
        }
        …
    }
    …
}
```

- A **condição** do `if`/`while` fica entre parênteses
- O **código** a ser executado ou não fica entre chaves
- Os **parâmetros** das funções ficam entre parênteses
- O **código** das funções fica entre chaves
- Em geral, o que vai entre parênteses é "curto", mas o que vai entre chaves pode ser bastante longo (várias linhas)
- É muito fácil para um humano se confundir com as chaves!
- A solução é a **indentação**: qualquer código entre chaves é deslocado para a direita
  - Qualquer tamanho, mas uma boa opção para **R** são 4 espaços

Indentação

```
main <- function() {
    …
    while (condição) {
        …
        if (condição) {
            …
        } else {
            …
        }
        …
    }
    …
}
```

- A **condição** do `if`/`while` fica entre parênteses
- O **código** a ser executado ou não fica entre chaves
- Os **parâmetros** das funções ficam entre parênteses
- O **código** das funções fica entre chaves
- Em geral, o que vai entre parênteses é "curto", mas o que vai entre chaves pode ser bastante longo (várias linhas)
- É muito fácil para um humano se confundir com as chaves!
- A solução é a **indentação**: qualquer código entre chaves é deslocado para a direita
  - Qualquer tamanho, mas uma boa opção para **R** são 4 espaços

Cuidado com o amigo da onça!

- Talvez você tenha percebido que às vezes é possível omitir `return()` ...

Cuidado com o amigo da onça!

- Talvez você tenha percebido que às vezes é possível omitir `return()` ...
- Talvez você tenha percebido que às vezes é possível omitir as chaves com `if` / `else` / `while` ...

Cuidado com o amigo da onça!

- Talvez você tenha percebido que às vezes é possível omitir `return()` ...
- Talvez você tenha percebido que às vezes é possível omitir as chaves com `if` / `else` / `while` ...
- Talvez você tenha percebido que às vezes é possível omitir o `"\n"` no final do comando `cat()` ...

Cuidado com o amigo da onça!

- Talvez você tenha percebido que às vezes é possível omitir `return()` ...
- Talvez você tenha percebido que às vezes é possível omitir as chaves com `if` / `else` / `while` ...
- Talvez você tenha percebido que às vezes é possível omitir o `"\n"` no final do comando `cat()` ...

Cuidado!

Cuidado com o amigo da onça!

- Talvez você tenha percebido que às vezes é possível omitir `return()` ...
- Talvez você tenha percebido que às vezes é possível omitir as chaves com `if` / `else` / `while` ...
- Talvez você tenha percebido que às vezes é possível omitir o `"\n"` no final do comando `cat()` ...

Cuidado!

Não caia em tentação!

Cuidado com o amigo da onça!

- **Talvez você tenha percebido que às vezes é possível omitir `return()` ...**
- **Talvez você tenha percebido que às vezes é possível omitir as chaves com `if` / `else` / `while` ...**
- **Talvez você tenha percebido que às vezes é possível omitir o `"\n"` no final do comando `cat()` ...**

Cuidado!

Não caia em tentação!

Quando você menos esperar, isso vai causar problemas!

Funções – escopo

- Funções são **nomes** dados a trechos de código que representam uma **ideia** bem definida e auto-contida
- Uma das vantagens é que elas podem ser usadas em contextos diferentes para realizar a computação correspondente àquela ideia
- Portanto...
- Elas precisam ser capazes de funcionar de maneira independente do contexto

Cada um com seu cada um e ninguém se mete no cada um dos outros

Nomes e memória

- **Em R, só existe quem tem nome:**

```
saudação <- "Olá!"
saudação <- "Oi!"
```

- **O texto `"Olá!"` não existe mais em nenhum lugar da memória após a segunda linha ser executada!**

```
x <- 1
x <- x + 1
```

- **O número `1` não existe mais em nenhum lugar da memória após a segunda linha ser executada!**

Nomes e memória

- **Em R, só existe quem tem nome:**

```
x <- 1
y <- x
x <- x + 1
```

- **O número `1` *continua existindo* na memória após a terceira linha ser executada**
  - Mas o que acontece na segunda linha?!?!?
    - » *Temos duas "cópias" do número `1` na memória?*
- **DEPENDE**
  - Em R, sim!
  - Na terceira linha, voltamos a ter apenas uma cópia
    - » *(R na verdade evita copiar quando não é necessário, mas do ponto de vista do programador é "como se" sempre houvesse a cópia)*

O que são “variáveis”?

- **Nomes que usamos para acessar dados armazenados em algum lugar na memória do computador?**
- **Conceitos abstratos que representam uma informação relevante para a computação que estamos realizando?**

Ambos

- **Mas cada linguagem dá mais ênfase a este ou àquele ponto de vista**

Variáveis e memória

- **Em python**
  - `x = x + 1` não altera o número `1` armazenado na memória, mas sim o valor associado à variável `x` (o número `2` "nasce" e recebe o nome `x`, enquanto o número `1` é "jogado fora")
    - » *Pensando em variáveis como conceitos, isso faz sentido: o valor da variável é o conceito, e ele foi modificado; o número armazenado é* ***outro*** *número*
- **Em outras linguagens, como R e C, as coisas são diferentes:**
  - `x <- y` guarda duas "cópias" do mesmo número em lugares diferentes da memória
  - `x <- x + 1` procura o lugar na memória em que a variável `x` está armazenada e altera o valor que está ali
    - » *Pensando em variáveis como nomes para um dado na memória, isso faz sentido: o conteúdo daquele espaço de memória foi modificado*

Escopo

E por que eu preciso me preocupar com isso?

```
metade <- function(x) {
    x <- x / 2
    return (x)
}
x <- 10
cat(metade(x), "é a metade de", x, "\n")
```

- **Na chamada a `metade()`, o número `10`, que está armazenado em `val`, é copiado para o nome `x`**
  - O nome `x` só existe dentro da função `metade()`!
  - Quando a função termina, o nome `x` e seu valor deixam de existir
  - Quando o valor de `x` é modificado dentro da função, isso não altera nenhuma variável fora da função
    - » *E isso continua valendo mesmo que, "por coincidência", o nome "de fora" e o nome "de dentro" sejam iguais*

Funções – escopo

- Funções são **nomes** dados a trechos de código que representam uma **ideia** bem definida e auto-contida
- Uma das vantagens é que elas podem ser usadas em contextos diferentes para realizar a computação correspondente àquela ideia
- Portanto...
- Elas precisam ser capazes de funcionar de maneira independente do contexto

Cada um com seu cada um e ninguém se mete no cada um dos outros

Funções

- **Não confunda `cat()` e `return`!**
  - `cat()` e `readline()` ⟶ comunicação com o "mundo"
  - `return` e os parâmetros das funções ⟶ comunicação entre partes diferentes do programa

```
metade <- function(x) {
    x <- x / 2
    return (x)
}
val <- 10
cat(metade(val), "é a metade de", val, "\n")
```

```
metade <- function(x) {
    y <- x / 2
    cat(y, "é a metade de", x, "\n")
}
metade(10)
```

Exercício – calculadora

Escreva uma função que recebe dois números e devolve a soma dos dois

Exercício – calculadora

Escreva uma função que recebe dois números e devolve a soma dos dois

```
soma <- function(a, b) {


}
```

Exercício – calculadora

Escreva uma função que recebe dois números e devolve a soma dos dois

```
soma <- function(a, b) {
    return (a + b)
}
```

Exercício – calculadora

Escreva uma função que recebe dois números e devolve a diferença dos dois

Exercício – calculadora

Escreva uma função que recebe dois números e devolve a diferença dos dois

```
subtração <- function(a, b) {


}
```

Exercício – calculadora

Escreva uma função que recebe dois números e devolve a diferença dos dois

```
subtração <- function(a, b) {
    return (a - b)
}
```

Exercício – calculadora

Escreva uma função que recebe dois números e devolve o produto dos dois

Exercício – calculadora

Escreva uma função que recebe dois números e devolve o produto dos dois

```
multiplicação <- function(a, b) {


}
```

Exercício – calculadora

Escreva uma função que recebe dois números e devolve o produto dos dois

```
multiplicação <- function(a, b) {
    return (a * b)
}
```

Exercício – calculadora

Escreva uma função que recebe dois números e devolve o resultado da divisão do primeiro pelo segundo

Exercício – calculadora

Escreva uma função que recebe dois números e devolve o resultado da divisão do primeiro pelo segundo

```
divisão <- function(a, b) {




}
```

Exercício – calculadora

Escreva uma função que recebe dois números e devolve o resultado da divisão do primeiro pelo segundo

```
divisão <- function(a, b) {




    return (a / b)
}
```

Exercício – calculadora

Escreva uma função que recebe dois números e devolve o resultado da divisão do primeiro pelo segundo

```
divisão <- function(a, b) {
    if (b == 0) {



    }
    return (a / b)
}
```

Exercício – calculadora

Escreva uma função que recebe dois números e devolve o resultado da divisão do primeiro pelo segundo

```
divisão <- function(a, b) {
    if (b == 0) {
        cat("Tá bêbo?!", "\n")
        return()
    }
    return (a / b)
}
```

Exercício – calculadora

Escreva um programa que pergunta ao usuário qual operação executar, pede os dois operandos e mostra o resultado

Exercício – calculadora

Escreva um programa que pergunta ao usuário qual operação executar, pede os dois operandos e mostra o resultado

```
main <- function() {









}
main()
```

Exercício – calculadora

Escreva um programa que pergunta ao usuário qual operação executar, pede os dois operandos e mostra o resultado

```
main <- function() {
    op <- readline("Qual operação? (soma, sub, multi, div) ")









}
main()
```

Exercício – calculadora

Escreva um programa que pergunta ao usuário qual operação executar, pede os dois operandos e mostra o resultado

```
main <- function() {
    op <- readline("Qual operação? (soma, sub, multi, div) ")
    primeiro <- as.integer(readline("Insira o primeiro operando: "))
    segundo <- as.integer(readline("Insira o segundo operando: "))









}
main()
```

Exercício – calculadora

Escreva um programa que pergunta ao usuário qual operação executar, pede os dois operandos e mostra o resultado

```
main <- function() {
    op <- readline("Qual operação? (soma, sub, multi, div) ")
    primeiro <- as.integer(readline("Insira o primeiro operando: "))
    segundo <- as.integer(readline("Insira o segundo operando: "))









    cat("O resultado é", resultado, "\n")
}
main()
```

Exercício – calculadora

Escreva um programa que pergunta ao usuário qual operação executar, pede os dois operandos e mostra o resultado

```
main <- function() {
    op <- readline("Qual operação? (soma, sub, multi, div) ")
    primeiro <- as.integer(readline("Insira o primeiro operando: "))
    segundo <- as.integer(readline("Insira o segundo operando: "))
    if (op == "soma")









    cat("O resultado é", resultado, "\n")
}
main()
```

Exercício – calculadora

Escreva um programa que pergunta ao usuário qual operação executar, pede os dois operandos e mostra o resultado

```
main <- function() {
    op <- readline("Qual operação? (soma, sub, multi, div) ")
    primeiro <- as.integer(readline("Insira o primeiro operando: "))
    segundo <- as.integer(readline("Insira o segundo operando: "))
    if (op == "soma")
        { resultado <- soma(primeiro, segundo) }









    cat("O resultado é", resultado, "\n")
}
main()
```

Exercício – calculadora

Escreva um programa que pergunta ao usuário qual operação executar, pede os dois operandos e mostra o resultado

```
main <- function() {
    op <- readline("Qual operação? (soma, sub, multi, div) ")
    primeiro <- as.integer(readline("Insira o primeiro operando: "))
    segundo <- as.integer(readline("Insira o segundo operando: "))
    if (op == "soma")
        { resultado <- soma(primeiro, segundo) }
    if (op == "sub")
        { resultado <- subtração(primeiro, segundo) }




    cat("O resultado é", resultado, "\n")
}
main()
```

Exercício – calculadora

Escreva um programa que pergunta ao usuário qual operação executar, pede os dois operandos e mostra o resultado

```
main <- function() {
    op <- readline("Qual operação? (soma, sub, multi, div) ")
    primeiro <- as.integer(readline("Insira o primeiro operando: "))
    segundo <- as.integer(readline("Insira o segundo operando: "))
    if (op == "soma")
        { resultado <- soma(primeiro, segundo) }
    if (op == "sub")
        { resultado <- subtração(primeiro, segundo) }
    if (op == "multi")
        { resultado <- multiplicação(primeiro, segundo) }


    cat("O resultado é", resultado, "\n")
}
main()
```

Exercício – calculadora

Escreva um programa que pergunta ao usuário qual operação executar, pede os dois operandos e mostra o resultado

```
main <- function() {
    op <- readline("Qual operação? (soma, sub, multi, div) ")
    primeiro <- as.integer(readline("Insira o primeiro operando: "))
    segundo <- as.integer(readline("Insira o segundo operando: "))
    if (op == "soma")
        { resultado <- soma(primeiro, segundo) }
    if (op == "sub")
        { resultado <- subtração(primeiro, segundo) }
    if (op == "multi")
        { resultado <- multiplicação(primeiro, segundo) }
    if (op == "div")
        { resultado <- divisão(primeiro, segundo) }
    cat("O resultado é", resultado, "\n")
}
main()
```

Exercício – calculadora

Modifique o programa anterior para repetir até o usuário escolher “fim”.

Exercício – calculadora

Modifique o programa anterior para repetir até o usuário escolher “fim”.

```
op <- readline("Qual operação? (soma, sub, multi, div) ")
```

Exercício – calculadora

Modifique o programa anterior para repetir até o usuário escolher “fim”.

```
op <- readline("Qual operação? (soma, sub, multi, div, fim) ")
```

Exercício – calculadora

Modifique o programa anterior para repetir até o usuário escolher “fim”.

```
op <- readline("Qual operação? (soma, sub, multi, div, fim) ")

    primeiro <- as.integer(readline("Insira o primeiro operando: "))
    segundo <- as.integer(readline("Insira o segundo operando: "))
    if (op == "soma")
        { resultado <- soma(primeiro, segundo) }
    if (op == "sub")
        { resultado <- subtração(primeiro, segundo) }
    if (op == "multi")
        { resultado <- multiplicação(primeiro, segundo) }
    if (op == "div")
        { resultado <- divisão(primeiro, segundo) }
    cat("O resultado é", resultado, "\n")
```

Exercício – calculadora

Modifique o programa anterior para repetir até o usuário escolher “fim”.

```
op <- readline("Qual operação? (soma, sub, multi, div, fim) ")
while (op != "fim") {
    primeiro <- as.integer(readline("Insira o primeiro operando: "))
    segundo <- as.integer(readline("Insira o segundo operando: "))
    if (op == "soma")
        { resultado <- soma(primeiro, segundo) }
    if (op == "sub")
        { resultado <- subtração(primeiro, segundo) }
    if (op == "multi")
        { resultado <- multiplicação(primeiro, segundo) }
    if (op == "div")
        { resultado <- divisão(primeiro, segundo) }
    cat("O resultado é", resultado, "\n")

}
cat("Fim do programa", "\n")
```

Exercício – calculadora

Modifique o programa anterior para repetir até o usuário escolher “fim”.

```
op <- readline("Qual operação? (soma, sub, multi, div, fim) ")
while (op != "fim") {
    primeiro <- as.integer(readline("Insira o primeiro operando: "))
    segundo <- as.integer(readline("Insira o segundo operando: "))
    if (op == "soma")
        { resultado <- soma(primeiro, segundo) }
    if (op == "sub")
        { resultado <- subtração(primeiro, segundo) }
    if (op == "multi")
        { resultado <- multiplicação(primeiro, segundo) }
    if (op == "div")
        { resultado <- divisão(primeiro, segundo) }
    cat("O resultado é", resultado, "\n")
    op <- readline("Qual operação? (soma, sub, multi, div, fim) ")
}
cat("Fim do programa", "\n")
```

Exercício — sistema de *login*

Imagine um sistema de *login* com três usuários:

- **Alan Turing** — UID **`turing`**, senha **`tmachine`**
- **Ada Lovelace** — UID **`llace`**, senha **`anengine`**
- **Grace Hopper** — UID **`hopper`**, senha **`business`**

Crie um sistema que lê o UID (*login*) e a senha do usuário e, se os dados estiverem corretos, escreve “Bem-vindo!”; caso contrário, o sistema escreve “Login ou senha incorreto”.

Exercício — sistema de *login*

```
uid <- readline("username: ")
senha <- readline("senha: ")
if (uid == "turing" && senha == "tmachine"
    || uid == "llace" && senha == "anengine"
    || uid == "hopper" && senha == "business") {
  cat("Bem-vindo!", "\n")
} else {
  cat("Login ou senha incorreto", "\n")
}
```

Exercício — sistema de *login*

Imagine um sistema de *login* com três usuários:

- **Alan Turing** — UID **`turing`**, senha **`tmachine`**
- **Ada Lovelace** — UID **`llace`**, senha **`anengine`**
- **Grace Hopper** — UID **`hopper`**, senha **`business`**

Crie um sistema que lê o UID (*login*) e a senha do usuário e, se os dados estiverem corretos, escreve “Bem-vindo, [nome]!”; caso contrário, o sistema escreve “Login ou senha incorreto”.

Exercício — sistema de *login*

```
library(glue)
```

Exercício — sistema de *login*

```
library(glue)
uid <- readline("username: ")
senha <- readline("senha: ")
```

Exercício — sistema de *login*

```r
library(glue)
uid <- readline("username: ")
senha <- readline("senha: ")









if (nome != "") {
    cat(glue("Bem-vindo, {nome}!"), "\n")
} else {
    cat("Login ou senha incorreto", "\n")
}
```

Exercício — sistema de *login*

```
library(glue)
uid <- readline("username: ")
senha <- readline("senha: ")
nome <- ""









if (nome != "") {
    cat(glue("Bem-vindo, {nome}!"), "\n")
} else {
    cat("Login ou senha incorreto", "\n")
}
```

Exercício — sistema de *login*

```
library(glue)
uid <- readline("username: ")
senha <- readline("senha: ")
nome <- ""
if (uid == "turing" && senha == "tmachine")
    { nome <- "Alan Turing" }




if (nome != "") {
    cat(glue("Bem-vindo, {nome}!"), "\n")
} else {
    cat("Login ou senha incorreto", "\n")
}
```

Exercício — sistema de *login*

```
library(glue)
uid <- readline("username: ")
senha <- readline("senha: ")
nome <- ""
if (uid == "turing" && senha == "tmachine")
    { nome <- "Alan Turing" }
if (uid == "llace" && senha == "anengine")
    { nome <- "Ada Lovelace" }
if (uid == "hopper" && senha == "business")
    { nome <- "Grace Hopper" }
if (nome != "") {
    cat(glue("Bem-vindo, {nome}!"), "\n")
} else {
    cat("Login ou senha incorreto", "\n")
}
```

Exercício — sistema de *login*

```
main <- function() {
    uid <- readline("username: ")
    senha <- readline("senha: ")
    nome <- checa_login(uid, senha)
    if (nome == "") {
        cat("Login ou senha incorreto", "\n")
    } else {
        cat(glue("Bem-vindo, {nome}!"), "\n")
    }
}
```

Exercício — sistema de *login*

```
main <- function() {
    uid <- readline("username: ")
    senha <- readline("senha: ")
    nome <- checa_login(uid, senha)
    if (nome == "") {
        cat("Login ou senha incorreto", "\n")
    } else {
        cat(glue("Bem-vindo, {nome}!"), "\n")
    }
}
checa_login <- function(uid, senha) {
    nome <- ""
    if (uid == "turing" && senha == "tmachine")
        { nome <- "Alan Turing" }
    if (uid == "llace" && senha == "anengine")
        { nome <- "Ada Lovelace" }
    if (uid == "hopper" && senha == "business")
        { nome <- "Grace Hopper" }
    return (nome)
}
```

Exercício — sistema de *login*

```
main <- function() {
    uid <- readline("username: ")
    senha <- readline("senha: ")
    nome <- checa_login(uid, senha)
    if (nome == "") {
        cat("Login ou senha incorreto", "\n")
    } else {
        cat(glue("Bem-vindo, {nome}!"), "\n")
    }
}
checa_login <- function(uid, senha) {
    name <- ""
    if (uid == "turing" && senha == "tmachine")
        { name <- "Alan Turing" }
    if (uid == "llace" && senha == "anengine")
        { name <- "Ada Lovelace" }
    if (uid == "hopper" && senha == "business")
        { name <- "Grace Hopper" }
    return (name)
}
```

Exercício — sistema de *login*

```
main <- function() {
    uid <- readline("username: ")
    senha <- readline("senha: ")
    nome <- checa_login(uid, senha)
    if (nome == "") {
        cat("Login ou senha incorreto", "\n")
    } else {
        cat(glue("Bem-vindo, {nome}!"), "\n")
    }
}
checa_login <- function(uid, senha) {

    if (uid == "turing" && senha == "tmachine")
        { return("Alan Turing") }
    if (uid == "llace" && senha == "anengine")
        { return("Ada Lovelace") }
    if (uid == "hopper" && senha == "business")
        { return("Grace Hopper") }
    return ("")
}
```

Pré-exercício

Dada uma coordenada x, y, diga se o ponto correspondente está dentro do círculo de raio 1 com centro na origem.

Pré-exercício

Dada uma coordenada x, y, diga se o ponto correspondente está dentro do círculo de raio 1 com centro na origem.

```
main <- function() {









}
main()
```

Pré-exercício

Dada uma coordenada x, y, diga se o ponto correspondente está dentro do círculo de raio 1 com centro na origem.

```
main <- function() {
    x <- as.numeric(readline("Informe x: "))









}
main()
```

Pré-exercício

Dada uma coordenada x, y, diga se o ponto correspondente está dentro do círculo de raio 1 com centro na origem.

```r
main <- function() {
    x <- as.numeric(readline("Informe x: "))
    y <- as.numeric(readline("Informe y: "))









}
main()
```

Pré-exercício

Dada uma coordenada x, y, diga se o ponto correspondente está dentro do círculo de raio 1 com centro na origem.

```
main <- function() {
    x <- as.numeric(readline("Informe x: "))
    y <- as.numeric(readline("Informe y: "))

    if (              )




}
main()
```

Pré-exercício

Dada uma coordenada x, y, diga se o ponto correspondente está dentro do círculo de raio 1 com centro na origem.

```
main <- function() {
    x <- as.numeric(readline("Informe x: "))
    y <- as.numeric(readline("Informe y: "))

    if (              ) {
        cat("O ponto está dentro do círculo", "\n")



    }
}
main()
```

Pré-exercício

Dada uma coordenada x, y, diga se o ponto correspondente está dentro do círculo de raio 1 com centro na origem.

```
main <- function() {
    x <- as.numeric(readline("Informe x: "))
    y <- as.numeric(readline("Informe y: "))

    if (              ) {
        cat("O ponto está dentro do círculo", "\n")
    } else {
        cat("O ponto está fora do círculo", "\n")
    }
}
main()
```

Pré-exercício

Dada uma coordenada x, y, diga se o ponto correspondente está dentro do círculo de raio 1 com centro na origem.

```
main <- function() {
    x <- as.numeric(readline("Informe x: "))
    y <- as.numeric(readline("Informe y: "))

    if (distancia <= 1) {
        cat("O ponto está dentro do círculo", "\n")
    } else {
        cat("O ponto está fora do círculo", "\n")
    }
}
main()
```

Pré-exercício

Dada uma coordenada x, y, diga se o ponto correspondente está dentro do círculo de raio 1 com centro na origem.

```r
main <- function() {
    x <- as.numeric(readline("Informe x: "))
    y <- as.numeric(readline("Informe y: "))
    distancia <- sqrt(x^2 + y^2)
    if (distancia <= 1) {
        cat("O ponto está dentro do círculo", "\n")
    } else {
        cat("O ponto está fora do círculo", "\n")
    }
}
main()
```

Pré-exercício

Dada uma coordenada x, y, diga se o ponto correspondente está dentro do círculo de raio 1 com centro na origem.

```r
main <- function() {
    x <- as.numeric(readline("Informe x: "))
    y <- as.numeric(readline("Informe y: "))

    if (x^2 + y^2 <= 1) {
        cat("O ponto está dentro do círculo", "\n")
    } else {
        cat("O ponto está fora do círculo", "\n")
    }
}
main()
```

Método de Monte Carlo

- **Alguns problemas são difíceis de resolver de maneira determinística, mas resultados aproximados podem ser aceitáveis**

Método de Monte Carlo

- **Alguns problemas são difíceis de resolver de maneira determinística, mas resultados aproximados podem ser aceitáveis**
- **Uma maneira de abordar alguns desses problemas é o método de Monte Carlo: chutar** 😂

Método de Monte Carlo

- **Alguns problemas são difíceis de resolver de maneira determinística, mas resultados aproximados podem ser aceitáveis**
- **Uma maneira de abordar alguns desses problemas é o método de Monte Carlo: chutar** 😂
  - Chuta valores de entrada e calcula os resultados correspondentes

Método de Monte Carlo

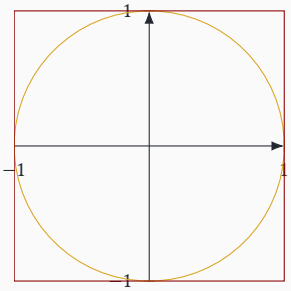
- **Alguns problemas são difíceis de resolver de maneira determinística, mas resultados aproximados podem ser aceitáveis**
- **Uma maneira de abordar alguns desses problemas é o método de Monte Carlo: chutar** 😂
  - Chuta valores de entrada e calcula os resultados correspondentes
  - Faz análise estatística dos resultados

Método de Monte Carlo

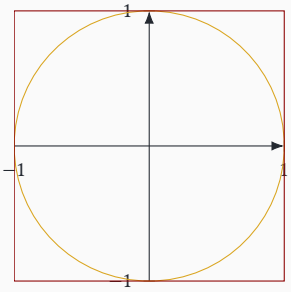
- **Alguns problemas são difíceis de resolver de maneira determinística, mas resultados aproximados podem ser aceitáveis**
- **Uma maneira de abordar alguns desses problemas é o método de Monte Carlo: chutar** 😂
  - Chuta valores de entrada e calcula os resultados correspondentes
  - Faz análise estatística dos resultados

```r
cat(runif(1), "\n") # Número aleatório no intervalo [0,1]
```

Calculando π pelo método de Monte Carlo



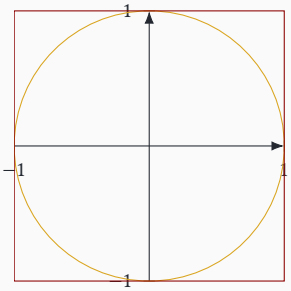
Calculando π pelo método de Monte Carlo



$$\text{área do quadrado} = 2 \cdot 2 = 4$$

$$\text{área do círculo} = \pi \cdot 1^2 = \pi$$

Calculando π pelo método de Monte Carlo

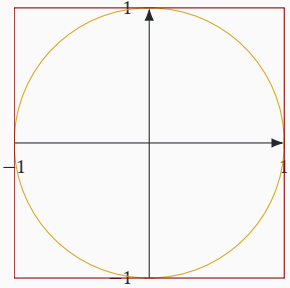


$$\text{área do quadrado} = 2 \cdot 2 = 4$$

$$\text{área do círculo} = \pi \cdot 1^2 = \pi$$

- **Sorteando um ponto dentro do retângulo, a probabilidade de ele estar dentro do círculo é $\frac{\pi}{4}$**

Calculando π pelo método de Monte Carlo

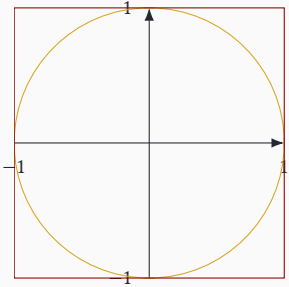


$$\text{área do quadrado} = 2 \cdot 2 = 4$$

$$\text{área do círculo} = \pi \cdot 1^2 = \pi$$

- **Sorteando um ponto dentro do retângulo, a probabilidade de ele estar dentro do círculo é $\frac{\pi}{4}$**
- **Se sortearmos vários pontos, podemos estimar o valor de π com base na proporção de pontos dentro do círculo**

Calculando π pelo método de Monte Carlo

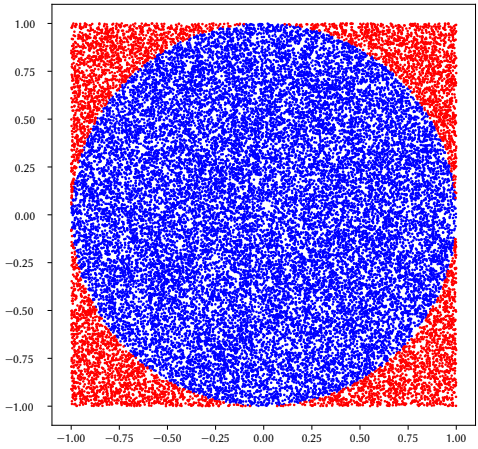


$$\text{área do quadrado} = 2 \cdot 2 = 4$$

$$\text{área do círculo} = \pi \cdot 1^2 = \pi$$

- **Sorteando um ponto dentro do retângulo, a probabilidade de ele estar dentro do círculo é $\frac{\pi}{4}$**
- **Se sortearmos vários pontos, podemos estimar o valor de π com base na proporção de pontos dentro do círculo**
  - $\pi = 4 \cdot$ probabilidade

Calculando π pelo método de Monte Carlo



Calculando π pelo método de Monte Carlo

Calculando π pelo método de Monte Carlo

```
main <- function() {









}
main()
```

Calculando π pelo método de Monte Carlo

```
main <- function() {
    chutes <- as.integer(readline("Quantas amostras? "))









}
main()
```

Calculando π pelo método de Monte Carlo

```
main <- function() {
    chutes <- as.integer(readline("Quantas amostras? "))









    cat("Pi é aproximadamente", 4*dentro/chutes, "\n")
}
main()
```

Calculando π pelo método de Monte Carlo

```
main <- function() {
    chutes <- as.integer(readline("Quantas amostras? "))

    i <- 1
    while (i <= chutes) {









    }
    cat("Pi é aproximadamente", 4*dentro/chutes, "\n")
}
main()
```

Calculando π pelo método de Monte Carlo

```
main <- function() {
    chutes <- as.integer(readline("Quantas amostras? "))

    i <- 1
    while (i <= chutes) {









        i <- i + 1
    }
    cat("Pi é aproximadamente", 4*dentro/chutes, "\n")
}
main()
```

Calculando π pelo método de Monte Carlo

```
main <- function() {
    chutes <- as.integer(readline("Quantas amostras? "))

    i <- 1
    while (i <= chutes) {
        x <- runif(1, -1, 1)
        y <- runif(1, -1, 1)




        i <- i + 1
    }
    cat("Pi é aproximadamente", 4*dentro/chutes, "\n")
}
main()
```

Calculando π pelo método de Monte Carlo

```
main <- function() {
    chutes <- as.integer(readline("Quantas amostras? "))

    i <- 1
    while (i <= chutes) {
        x <- runif(1, -1, 1)
        y <- runif(1, -1, 1)
        if (x^2 + y^2 <= 1) { # dentro do círculo

        }
        i <- i + 1
    }
    cat("Pi é aproximadamente", 4*dentro/chutes, "\n")
}
main()
```

Calculando π pelo método de Monte Carlo

```
main <- function() {
    chutes <- as.integer(readline("Quantas amostras? "))

    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
        }
        i <- i + 1
    }
    cat("Pi é aproximadamente", 4*dentro/chutes, "\n")
}
main()
```

Calculando π pelo método de Monte Carlo

```
main <- function() {
    chutes <- as.integer(readline("Quantas amostras? "))
    dentro <- 0
    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
        }
        i <- i + 1
    }
    cat("Pi é aproximadamente", 4*dentro/chutes, "\n")
}
main()
```