Curso R

Programar é preciso

Alexandre Adalardo de Oliveira

Ecologia- IBUSP maio 2017

Noções de Programação

Desafios

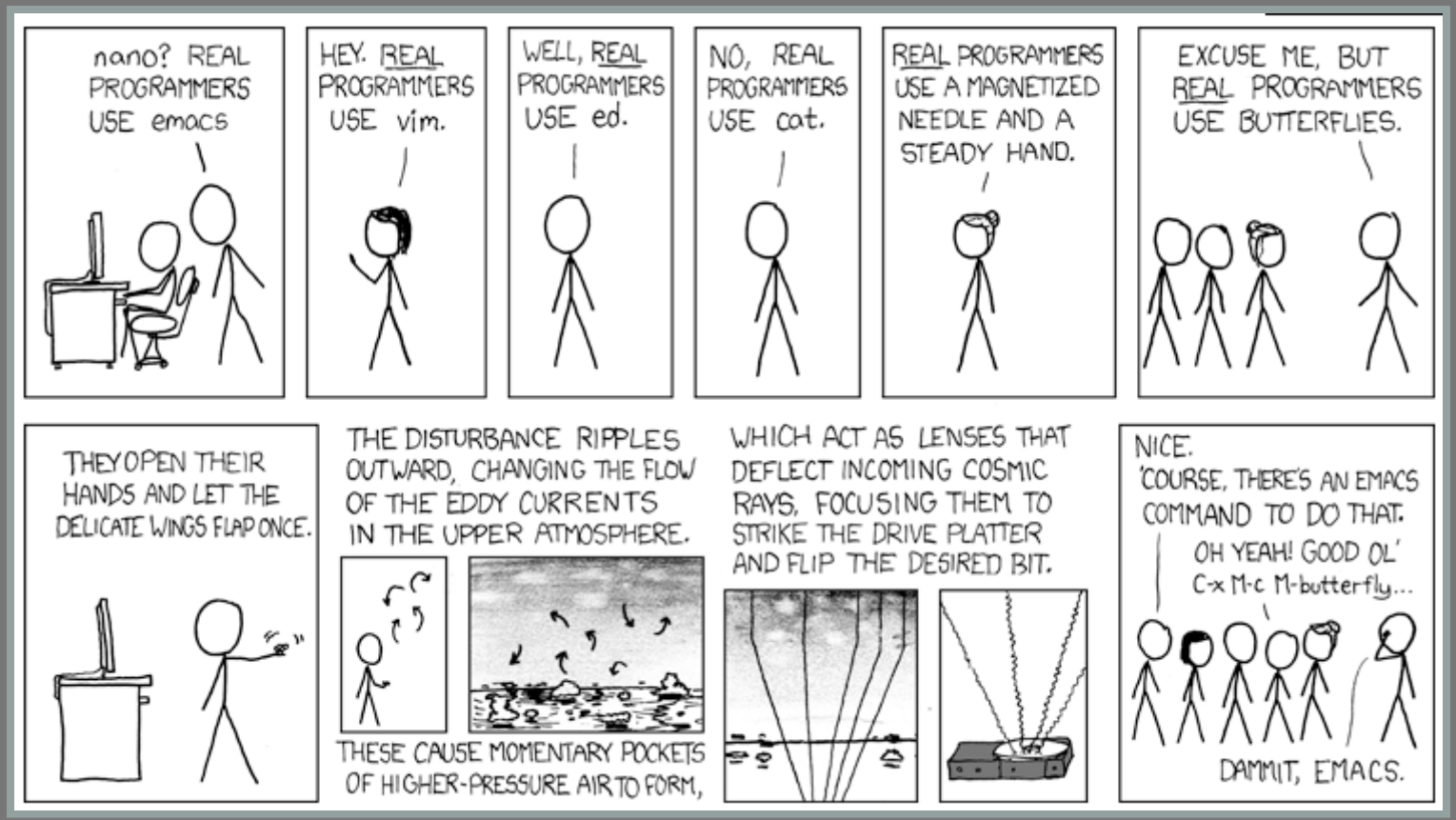
CONQUISTAR O MUNDO!

Primeiros passos no R!

Depois da primeira função!

Conquistar o mundo!

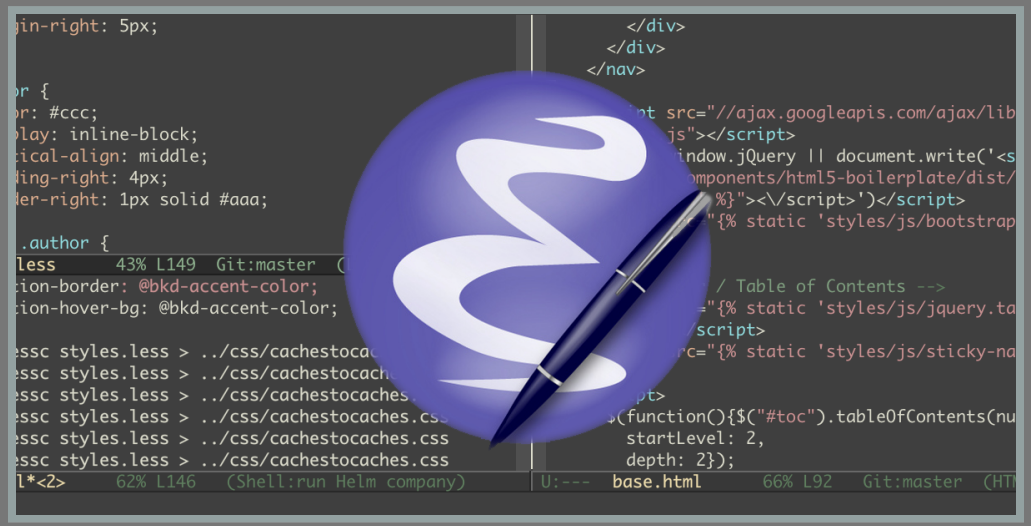
Editor de texto:



RStudio

Ambiente de Desenvolvimento Integrado (IDE)

Emacs + ESS



Pogramação

Todo usuário é prograd or

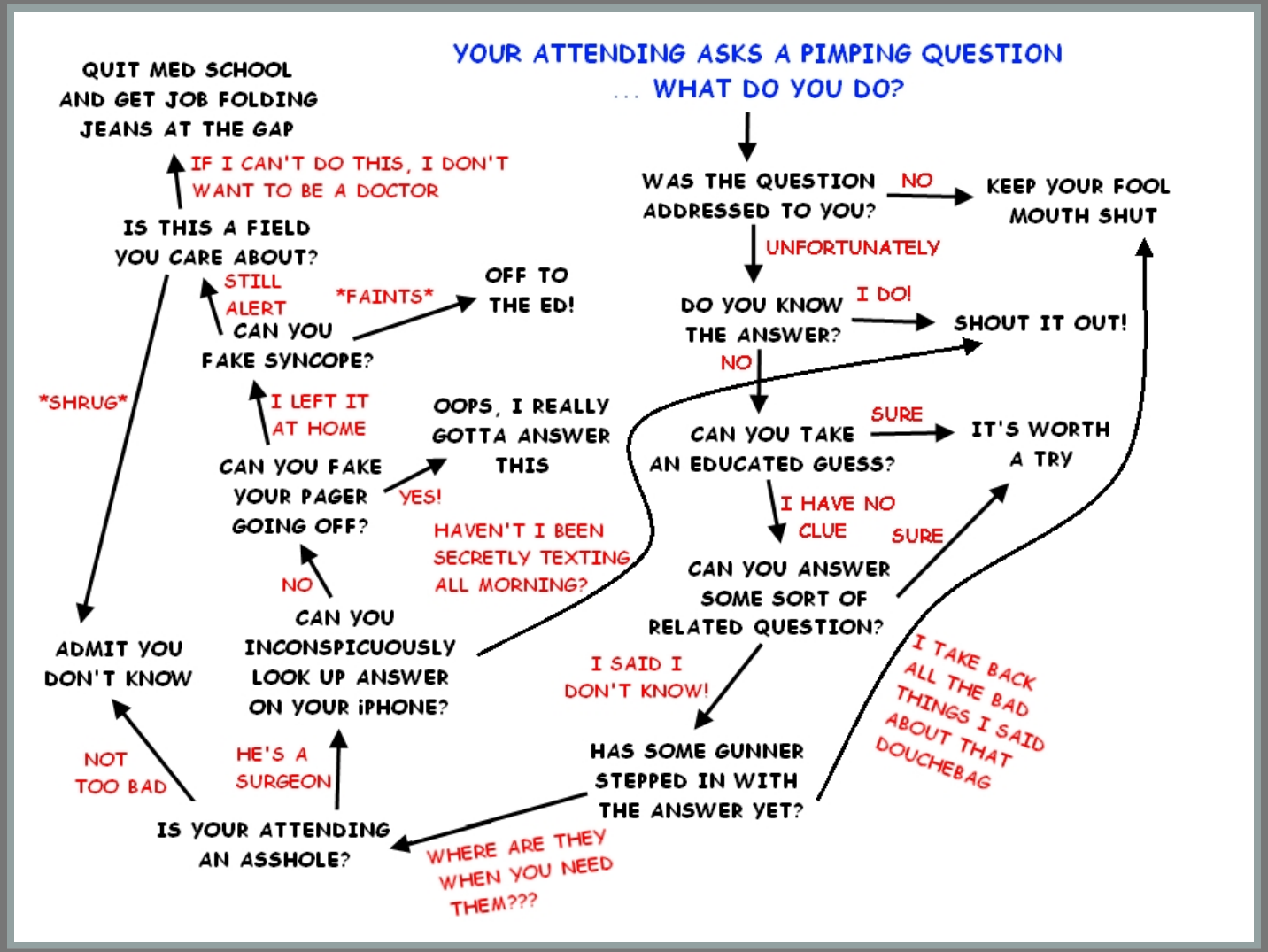
```
funfar <- function(){}
class(funfar)
```

```
## [1] "function"
```

- Função:
  - LINHA DE MONTAGEM conduzida por um ALGORITMO
- Algoritmo:
  - sequência lógica de ações para resolver um problema
  - comandos sequenciais arranjados por um fluxo de tarefas

Algoritmo:

- Fluxo de comandos conduzidos por uma sequência lógica para realizar uma tarefa ou resolver um problema


YOUR ATTENDING ASKS A PIMPING QUESTION
... WHAT DO YOU DO?
WAS THE QUESTION ADDRESSED TO YOU?
NO
KEEP YOUR FOOL MOUTH SHUT
UNFORTUNATELY
DO YOU KNOW THE ANSWER?
I DO!
SHOUT IT OUT!
NO
CAN YOU TAKE AN EDUCATED GUESS?
SURE
IT'S WORTH A TRY
I HAVE NO CLUE
CAN YOU ANSWER SOME SORT OF RELATED QUESTION?
SURE
I TAKE BACK ALL THE BAD THINGS I SAID ABOUT THAT DOUCHEBAG
I SAID I DON'T KNOW!
HAS SOME GUNNER STEPPED IN WITH THE ANSWER YET?
WHERE ARE THEY WHEN YOU NEED THEM???
IS YOUR ATTENDING AN ASSHOLE?
HE'S A SURGEON
NOT TOO BAD
ADMIT YOU DON'T KNOW
NO
CAN YOU INCONSPICUOUSLY LOOK UP ANSWER ON YOUR iPHONE?
HAVEN'T I BEEN SECRETLY TEXTING ALL MORNING?
YES!
CAN YOU FAKE YOUR PAGER GOING OFF?
I LEFT IT AT HOME
OOPS, I REALLY GOTTA ANSWER THIS
SHRUG
CAN YOU FAKE SYNCOPE?
STILL ALERT
FAINTS
OFF TO THE ED!
IS THIS A FIELD YOU CARE ABOUT?
IF I CAN'T DO THIS, I DON'T WANT TO BE A DOCTOR
QUIT MED SCHOOL AND GET JOB FOLDING JEANS AT THE GAP

Algorítmo: Petit Gateau

1. Bater as gemas, ovos e açúcar até quadruplicar de volume
2. Em outro recipiente, derreter a manteiga e o chocolate. Acrescentar o rum e misturar bem
3. Esperar o chocolate esfriar um pouco, e mistura à mistura de ovos e açúcar. Misturar bem, mas com cuidado e sem bater
4. Juntar a farinha e misturar com cuidado
5. Deixar na geladeira por no mínimo 4 h, máximo 2 dias.
6. Untar generosamente as forminhas com manteiga. Colocar no freezer para a manteiga endurecer, untar com uma segunda camada de manteiga, e polvilhar com farinha ou cacau em pó.
7. Tirar a massa da geladeira e misturar um pouco, caso tenha ficado muito dura.
8. Preencher 2/3 das forminhas com a massa, assar no forno quente e pre-áquecido (ver OBS).
9. Assim que formar uma fina casquinha, desenformar em um prato e servir imediatamente com sorvete de creme.

Algoritmo: Script e Função

- script: código organizado para realizar a tarefa (receita)
- função: automatiza a tarefa (maquina de fazer petit gateau)

Funções: tipo de tarefa

1. recorrente
2. não banal
3. generalizável
4. útil

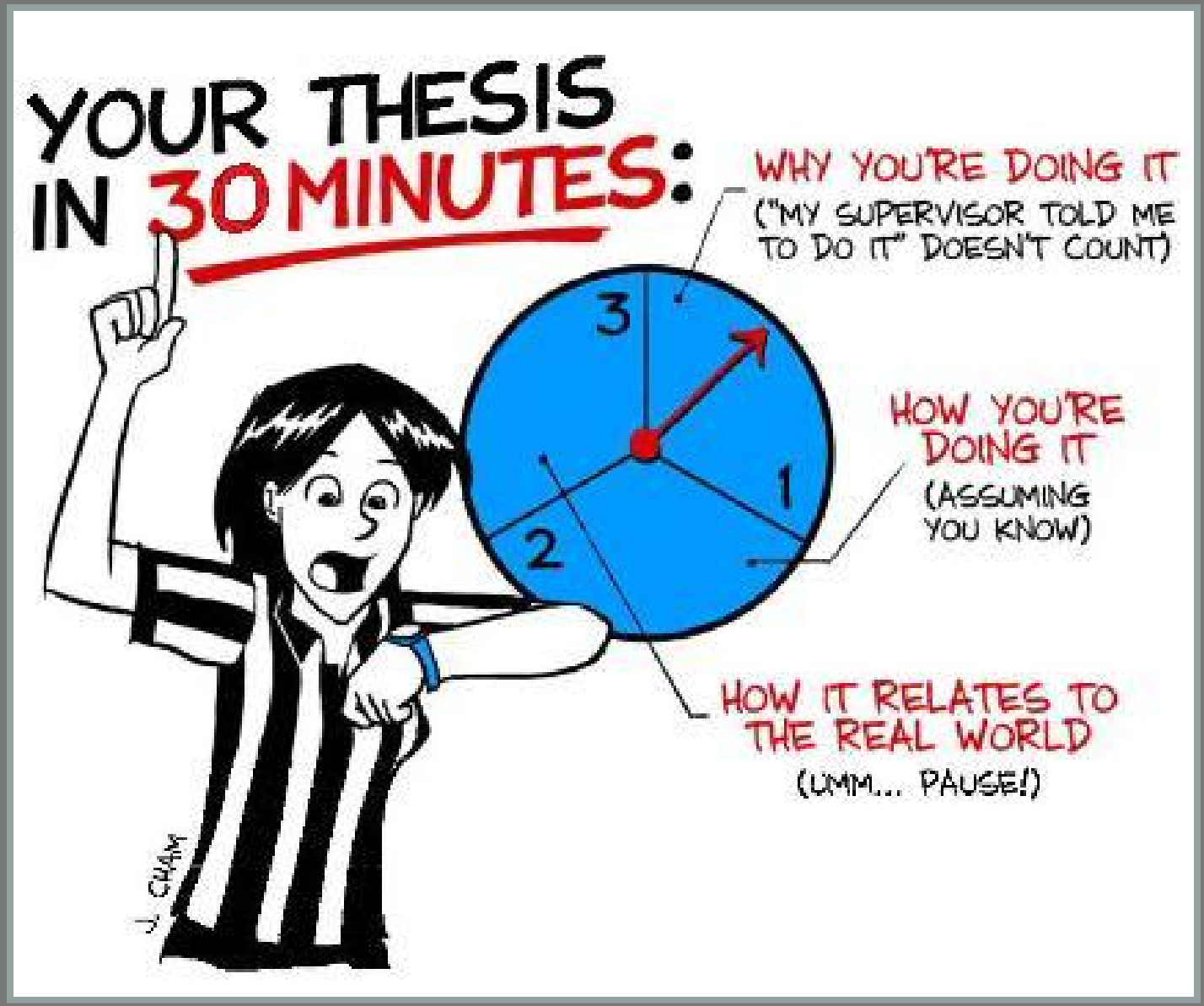
Funções:

Não necessária:

1. tarefa específica
2. tarefa banal
3. não recorrente
4. manipula apenas meu dado
5. não há flexibilidade
6. tarefa de um evento
7. tarefa que não sabe realizar

Atenção:

- análise dos dados da sua tese não é uma boa função!!!


YOUR THESIS IN 30 MINUTES:
WHY YOU'RE DOING IT
("MY SUPERVISOR TOLD ME TO DO IT" DOESN'T COUNT)
3
HOW YOU'RE DOING IT
(ASSUMING YOU KNOW)
1
2
HOW IT RELATES TO THE REAL WORLD
(UMM... PAUSE!)
CHANY

Função

Passos para a construção:

1. construa um script que realize a tarefa;
2. estabeleça quais opções poderão ser manipuladas;
3. represente as opções por estados de argumentos;
4. crie fluxos de trabalhos distintos para cada opção (controle de fluxo, ex. "if" "else");
5. represente a entrada de dados por um argumento (p.ex. "dados", "X", "diretory");
6. retorne o resultado da tarefa.

Funções: nome

Tarefa realizada:

```
summary( )
```

Revela objetivo:

```
plot( )
```

Acrônimo

```
lm(); dp()
```

Memorável:

```
sunflowerplot()
```

Não usual!: caso do pi

Funções: argumentos

- objetos: dados ou outros objetos para realizar a tarefa, funcionam como um avatar, uma cópia com nome definido;
- controladores de fluxo:
  - lógico
    - controle de fluxo binário
    - teste de viabilidade
  - caracter
    - fluxo multientrada
- valores:
  - ciclagens
  - definição de parâmetros

Funções: estrutura básica

```
myfun <- function(fun0, arg1, ...)
{
    fun1 <- comando1(x)
    fun2 <- comando2(fun1)
    ...
    funN <- comandoN(funN-1)
    return(funN)
}
```

Funções: carregando

funciona como um script

- um arquivo texto (script)
- carrega na sua área de trabalho (objeto)

```
source("myfun.r")
```

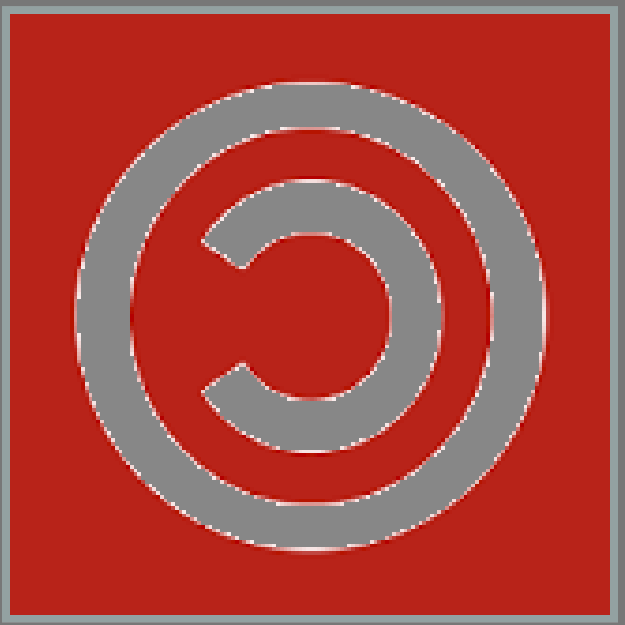
- Alternativa:
  - copiar e colar no console do R

Funções: executando

```
myfun()
```

Código aberto

- código aberto e livre (copyleft)
- linguagem interpretada (não compilada)
- usar, modificar, distribuir



Modificando uma função

```
read.csv
```

```
## function (file, header = TRUE, sep = ",", quote =
##     fill = TRUE, comment.char = "", ...)
## read.table(file = file, header = header, sep = sep
##     dec = dec, fill = fill, comment.char = comment
## <bytecode: 0x3dd2ae0>
## <environment: namespace:utils>
```

Função: read.ale

```
read.ale <-  function (file, header = TRUE, sep = "\t
           quote = "\"", dec = ".", as.is = TRUE, ...
{
    read.table(file = file, header = header, sep = se
           quote = quote, dec = dec, as.is = as.is, .
}

ls(pattern = "ale")
```

```
## [1] "read.ale"
```

```
read.ale
```

```
## function (file, header = TRUE, sep = "\t",
##           quote = "\"", dec = ".", as.is = TRUE,
## {
##     read.table(file = file, header = header, sep =
##           quote = quote, dec = dec, as.is = as.is
## }
```

Função: read.ale

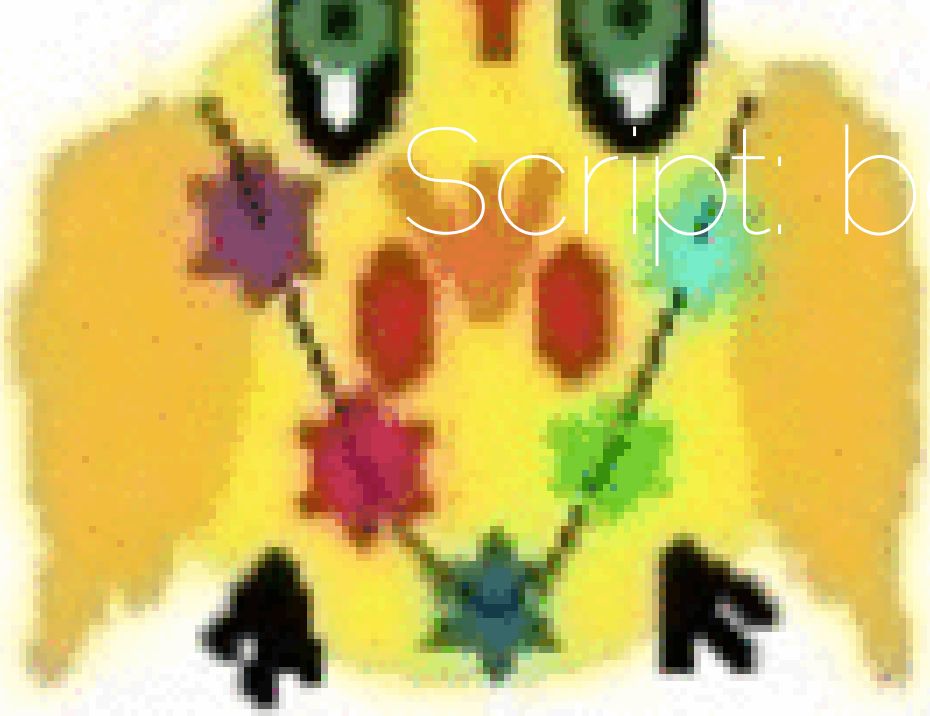
```
davis <- read.ale(file = "data/davis.csv")
str(davis)
```

```
## 'data.frame':    199 obs. of  3 variables:
##  $ sex   : chr  "M" "F" "F" "M" ...
##  $ weight: int  77 58 53 68 59 76 76 69 71 65 ...
##  $ height: int  182 161 161 177 157 170 167 186 17
```

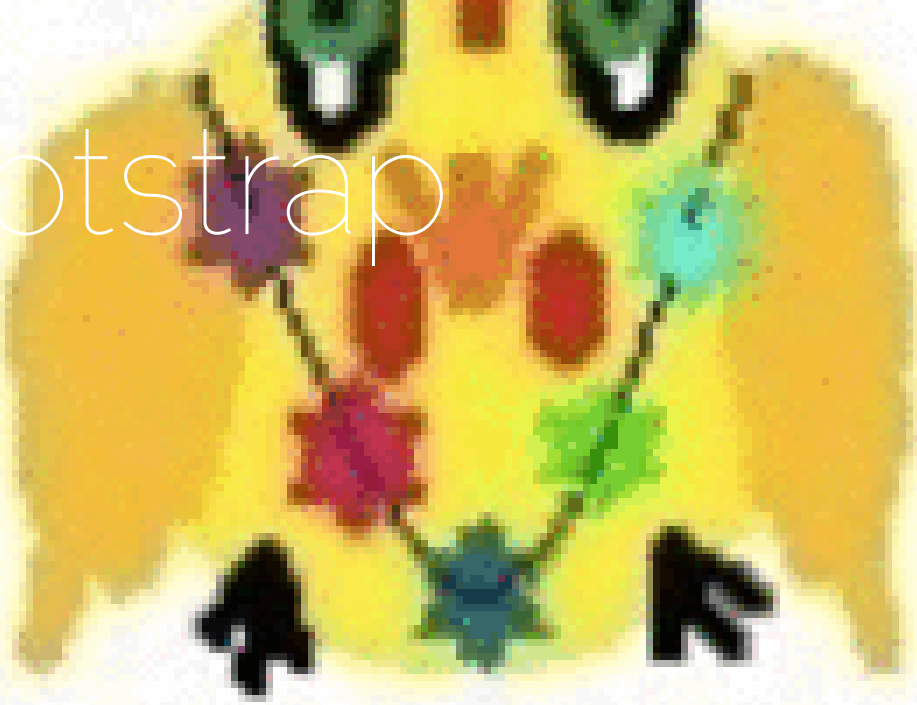
```
head(davis)
```

```
##   sex weight height
## 1   M     77    182
## 2   F     58    161
## 3   F     53    161
## 4   M     68    177
## 5   F     59    157
## 6   M     76    170
```


R



Script: bootstrap

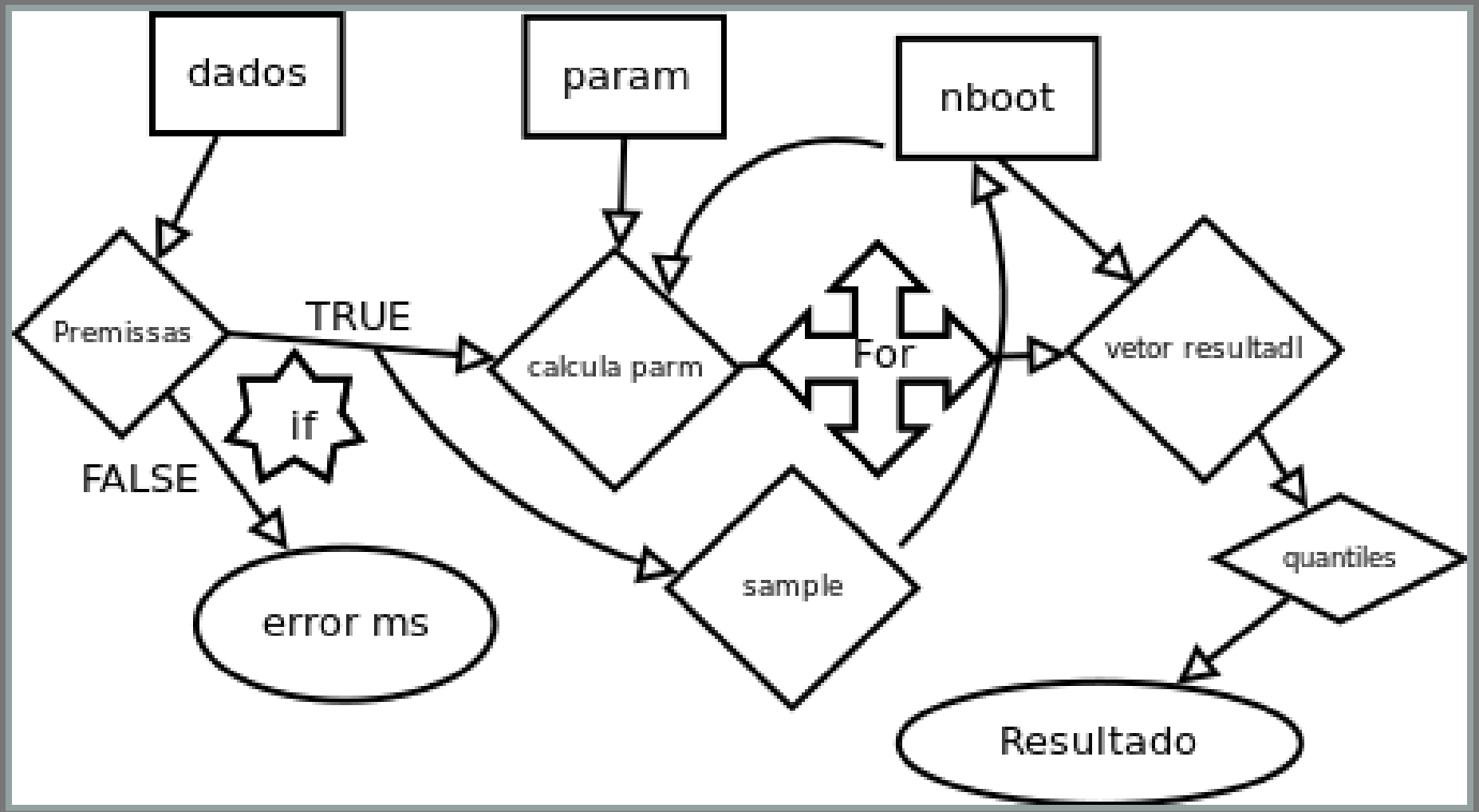


Função bootR: narrativa

1. Tarefa a ser executada: calcular o intervalo de confiança bootstrap
2. Entrada (argumentos) dados: um vetor com variáveis numéricas param: parâmetro de uma distribuição (função) nboot: número de reamostragens qmin: quantil inferior qmax: quantil superior
3. Testa as premissas: dados: is.vetor qmin: between 0 - 1 qmax: between qmin - 1
4. Cria o vetor de resultado da reamostragem fun5. Cria o ciclo
5. Retorna o intervalo um vetor com 3 valores
    - Intervalo inferior
    - Estimativa do parâmetro
    - Intervalo superior

TODO: permitir a estimativa de parâmetros de dados bivariados (inclinação da reta) e lidar com NA

Fluxograma do algoritmo

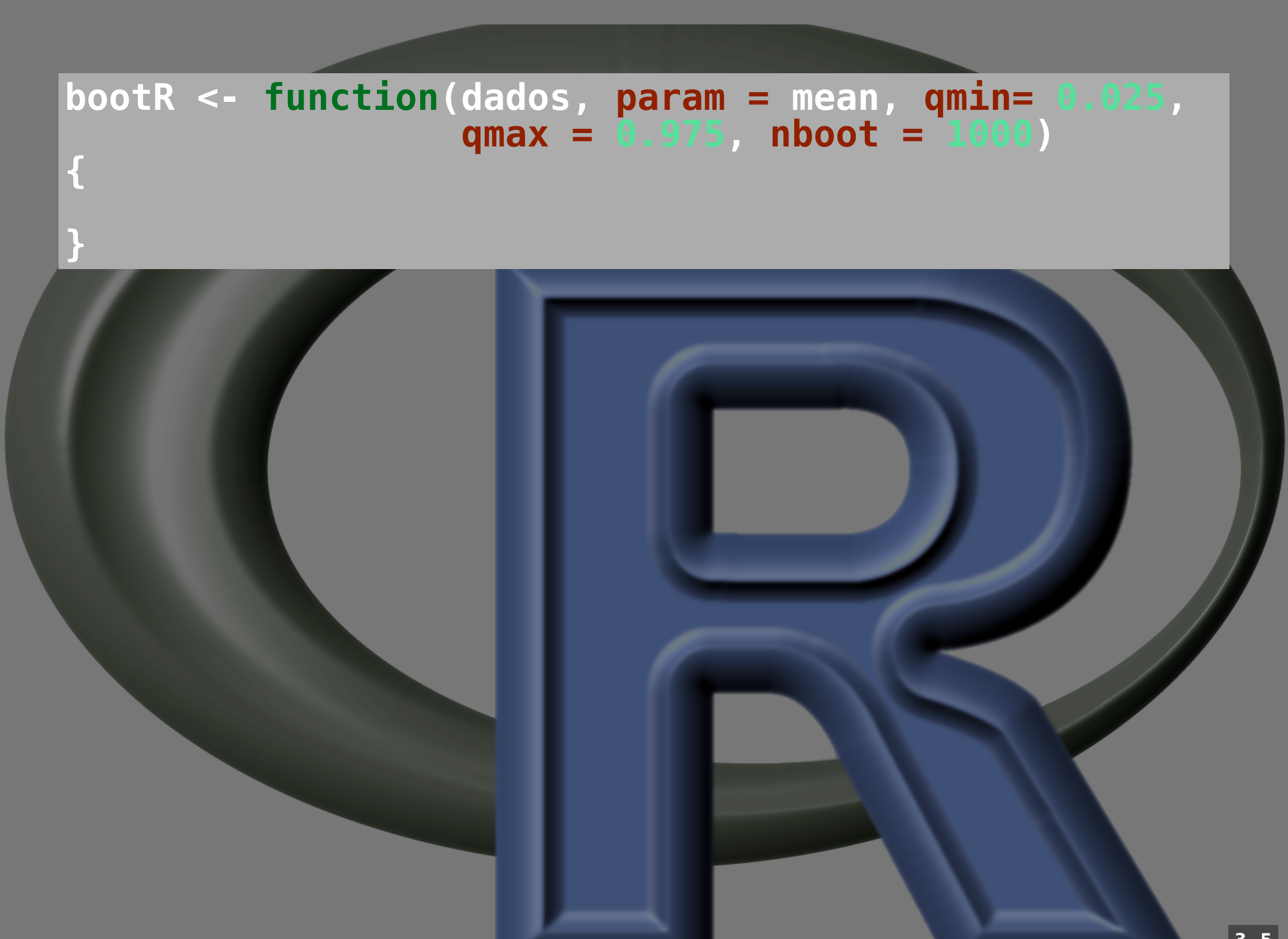


Pseudo-código

- INICIO
- PARAMETRO
  - dados; nboot; qmin; qmax
- SE classe de *dados*
  - PARA
  - escreve: "não é vetor"
- SE *qmin* >1 OU *qmax* <1 OU *qmax* > *qmin*
  - ESCREVE "CI 95%"
- CALCULA *parma* de *dados* -> *param0*
- CRIA *resboot* <- VETOR *nboot* <- NA
- PARA I de 1 ATE *nboot* : resboot NA *nboot*
  - CALCULA *parm* de REAMOSTRA COM REPOSICAO *dados*
  - ARMAZENA em *resboot* POSICAO i
- CALCULA QUANTIL
  - q1 <- *qmin*
  - q2 <- *qmin*
- RETORNA : q1; parma0; q2 FIM

```
bootR <- function(dados, param = mean, qmin= 0.025,
                  qmax = 0.975, nboot = 1000)
{

}
```



Função: bootR

```r
bootR <- function(dados, param = mean, qmin= 0.025,
                  qmax = 0.975, nboot = 1000)
{
    param0 <- param(dados)
    resboot <- rep(NA, times = nboot)
    for(i in 1:nboot)
    {
        resboot[i] <- param(sample(dados, replace=TRU
    }
    qt <- quantile(resboot, prob=c(qmin, qmax))
    res <- c(qt[1], param0, qt[2])
    names(res) <- c("ICinf", "Param", "ICsup")
    return(res)
}
```

Testando: bootR

```
norm10 <- rnorm(100, mean=10, sd=2)
bootR(dados = norm10)
```

```
    ICinf     Param     ICsup
 9.376268  9.794084 10.223575
```

```
bootR(dados = norm10, param = sd)
```

```
   ICinf    Param    ICsup
1.865114 2.130332 2.383955
```

```
bootR(dados = norm10, param = var)
```

```
   ICinf    Param    ICsup
3.506580 4.538316 5.670827
```

Controle do fluxo: if else

Interrompe o fluxo para um teste lógico

- TRUE:
  - executa o que está no if{ } e continua
- FALSE:
  - pula par else{ } e continua

```
dados <- rnorm(100)
if(!(class(dados) == "numeric" | class(dados) == "int
{
    stop( "o objeto não é um vetor numérico")
}else
{
    cat(paste("objeto numérico com", length(dados), "
}
```

```
objeto numérico com 100 observações
```

Controle do fluxo: if else

- stop( ):
  - apresenta mensagem de erro e para a processo
- message( ):
  - apresenta mensagem e continua execução
- warning( ):
  - armazena mensagem e continua execução

```
qmin <- 0.025
qmax <- 0.975

if(qmin < 0 | qmax > 1 | qmax < qmin)
{
    message( "quantils incorretos, calculando \n
Calculando o Intervalo de Confiança bootstrap de  95%
    qmin = 0.025
    qmax = 0.975
}else
{
    cat(paste("Calculando quantils:", qmin, "e", qmax
}
```

```
Calculando quantils: 0.025 e 0.975
```

```r
bootR <- function(dados, param = mean, qmin= 0.025,
                  qmax = 0.975, nboot = 1000)
{
    if(class(dados) != "numeric")
    {
        stop( "o objeto não é um vetor numérico")
    }else
    {
    cat(paste("objeto numérico com", length(dados), "
    }
    if(qmin < 0 | qmax > 1 | qmax < qmin)
    {
        cat( "Quantils Incorretos \n
Calculando o Intervalo de Confiança bootstrap de  95%
        qmin = 0.025
        qmax = 0.975
    }else
    {
        cat(paste("Calculando quantils:", qmin, "e",
    }
    param0 <- param(dados)
    resboot <- rep(NA, times = nboot)
    for(i in 1:nboot)
    {
```

```
    resboot[i] <- param(sample(dados, replace=TRU
}
qt <- quantile(resboot, prob=c(qmin, qmax))
```

Testando: bootR

```
dfnorm100 <- data.frame(r1 = rnorm(100, mean=10, sd=2
bootR(dados = dfnorm100$r1, qmin=-1)
```

```
objeto numérico com 100 observações
```

```
    ICinf     Param     ICsup
 9.333247  9.692195 10.074800
```

```
bootR(dados = dfnorm100$r1 , qmax = 0.025, qmin = 0.9
```

```
objeto numérico com 100 observações
```

```
    ICinf     Param     ICsup
 9.327022  9.692195 10.078398
```

```
bootR(dados = dfnorm100)

##Error in bootR(dados = dfnorm100) : o objeto não é
```

Testando: NA

```
rnormNA <-  c(rnorm(100, mean=10, sd=2), NA)
```

```
bootR(dados = rnormNA)
## objeto numérico com 101 observações
## Calculando quantils: 0.025 e 0.975
## Error in quantile.default(nboot, prob = c(qmin, qm
##   missing values and NaN's not allowed if 'na.rm'
```

```r
bootR <- function(dados, param = mean, qmin= 0.025,
                  qmax = 0.975, nboot = 1000, ...)
{
    if(class(dados) != "numeric")
    {
        stop( "o objeto não é um vetor numérico")
    }else
    {
    cat(paste("objeto numérico com", length(dados), "
    }
    if(qmin < 0 | qmax > 1 | qmax < qmin)
    {
        cat( "Quantils Incorretos \n
Calculando o Intervalo de Confiança bootstrap de  95%
        qmin = 0.025
        qmax = 0.975
    }else
    {
        cat(paste("Calculando quantils:", qmin, "e",
    }
    param0 <- param(dados, ...)
    resboot <- rep(NA, times = nboot)
    for(i in 1:nboot)
    {
```

```
    resboot[i] <- param(sample(dados, replace=TRU
}
qt <- quantile(resboot, prob=c(qmin, qmax))
```

Testando: bootR NA

```
bootR(dados = rnormNA, na.rm=TRUE)
```

```
objeto numérico com 101 observações
Calculando quantils: 0.025 e 0.975
```

```
    ICinf     Param     ICsup
 9.658231  9.987859 10.372415
```

CONQUISTAMOS O MUNDO!


R

Resumo: função

A tarefa:

- repetitiva ou recorrente
- útil
- inédita (seu código)
- não trivial
- um desafio factível

A narrativa

- objeto de entrada
- argumentos (flexibilidade)
- passos da tarefa
- controles de fluxo
  - teste de pressupostos
  - ciclos
- objeto de resultado

Resumo: função

Fluxograma

Pseudo-código

Script com a tarefa

```
    - crie objetos com nome dos argumentos
                                (substitutos)
    - teste cada linha construída
    - comente cada linha descrevendo a sua tarefa
- Teste os ciclos:
    - crie um objeto com o nome do contador
    - associe a ele um valor do ciclo
- Verifique se resultado está sendo preenchido
```

Encapsule o script em um função

```
- nomeie os argumentos
- coloque o script na função
- ajuste os "substitutos" (argumentos)
- teste a tarefa
```

Resumo: função

Teste a função básica

```
debug(bootR)
undebug(bootR)
```

Inclua teste de pressupostos

USE O FÓRUM


DON'T PANIC

FUNÇÃO FIM

Atividades da Tarde

Atividades da Tarde

- Tutorial
- Apostila
- Exercícios

Elabore sua proposta o quanto antes!!

FIM DO CURSO