

Introdução ao R

Wilson Freitas

O que é o R?

O que é o R?

R é um ambiente integrado para análise de dados e criação de visualização computacional, que oferece entre outras coisas:

- uma grande coleção de pacotes para estatística e análise de dados
- uma linguagem para expressar modelos estatísticos e ferramentas para desenvolver modelos lineares e não lineares
- diversas funcionalidades para a criação de gráficos
- uma linguagem de programação orientada a objetos que pode facilmente ser estendida pela comunidade e integrada com outras linguagens

Uma olhadinha no R

```
2+3
```

```
[1] 5
```

```
sqrt(3/4)/(1/3 - 2/pi^2)
```

```
[1] 6.626513
```

Uma olhadinha no R

```
library(MASS)  
mean(chem)
```

```
[1] 4.280417
```

```
m <- mean(chem); v <- var(chem)/length(chem)  
m/sqrt(v)
```

```
[1] 3.958487
```

Pedindo ajuda

```
help(mean)
```

```
?mean
```

```
help(" ")
```

```
? " "
```

A linguagem R

Características da linguagem

- A linguagem R é uma linguagem de manipulação de objetos com diversas características de programação funcional
- Todas as entidades em R são objetos
- R é uma linguagem interativa (pode ser executada através de um prompt) assim como uma linguagem de uso geral com:
 - IO: arquivos, Internet, bancos de dados
 - estruturas de dados: `list`, `vector`, `matrix`, ...
 - estruturas de controle: `if`, `else`, `while`, `for`, ...
 - administração de memória
 - interfaces com outras linguagens: C/C++, Java, Python, ...

Convenção de nomes

Nomes para objetos no R seguem as seguintes regras:

- são *case sensitive*: abc é diferente de Abc
- Nomes são formados por: letras, números, . e _
- Nomes não podem iniciar com: números, _
- Nomes podem iniciar com . apenas quando forem seguidos de letras, nomes com .42 são números decimais
- O . é importante na convenção de nomes para a criação de objetos (*veremos mais adiante*)

Recomendo dar uma olhada no *style guide* do Hadley Wickham <http://r-pkgs.had.co.nz/style.html>

Convenção de nomes

- Nomes fora do padrão podem ser utilizados entre crases ou aspas

```
"repl<-" <- function(x, val) structure(x, value=val)  
`repl<-`
```

```
function(x, val) structure(x, value=val)
```

```
`+`
```

```
function (e1, e2) .Primitive("+")
```

Outras características

- O código R é formado por comandos que são expressões ou atribuições
- Os comandos são separados por `new-line` ou `;`
- Qualquer código em uma linha após `#` é tratado como comentário

Ambiente R

Criando variáveis (I)

Variáveis são criadas atribuindo valores com o operador `<-`

```
a <- 1
```

`ls` lista as variáveis criadas no ambiente

```
ls()
```

```
[1] "a"      "m"      "repl<-" "v"
```

Criando Variáveis (2)

A atribuição também pode ser feita com =

```
b = 2
```

e também com o operador ->

```
3 -> a
```

além da atribuição múltipla

```
b <- a <- 5  
5 -> a -> b
```

Removendo variáveis

rm remove

```
ls()
```

```
[1] "a"      "b"      "m"      "repl<-" "v"
```

```
rm('a')
```

```
ls()
```

```
[1] "b"      "m"      "repl<-" "v"
```

Variáveis ocultas

Variáveis que iniciam com . são ocultas

```
.hid <- 3  
ls()
```

```
[1] "b"      "m"      "repl<-" "v"
```

```
ls(all.names=TRUE)
```

```
[1] ".First" ".hid"   "b"      "m"      "repl<-" "v"
```


Limpando o Ambiente

Para limpar o ambiente `rm(list=ls())`

```
rm(list=ls())  
ls(all.names=TRUE)
```

```
[1] ".First" ".hid"
```

ainda mantém as ocultas

```
rm(list=ls(all.names=TRUE))  
ls(all.names=TRUE)
```

```
character(0)
```

Investigando variáveis (I)

Como o R executa um *shell* interativo, uma forma de investigar variáveis é imprimindo as

```
print("Wilson")
```

```
[1] "Wilson"
```

```
print(print)
```

```
function (x, ...)
  UseMethod("print")
<bytecode: 0x7f835bcd3e40>
<environment: namespace:base>
```

Investigando variáveis (2)

Imprimir funções em R retorna o código

```
print(uniroot)
```

```
function (f, interval, ..., lower = min(interval), upper = max(interval),
  f.lower = f(lower, ...), f.upper = f(upper, ...), extendInt = c("no",
    "yes", "downX", "upX"), check.conv = FALSE, tol = .Machine$double.eps^0.25,
  maxiter = 1000, trace = 0)
{
  if (!missing(interval) && length(interval) != 2L)
    stop("'interval' must be a vector of length 2")
  if (!is.numeric(lower) || !is.numeric(upper) || lower >=
    upper)
    stop("lower < upper is not fulfilled")
  if (is.na(f.lower))
    stop("f.lower = f(lower) is NA")
  if (is.na(f.upper))
    stop("f.upper = f(upper) is NA")
  Sig <- switch(match.arg(extendInt), yes = NULL, downX = -1,
    no = 0, upX = 1, stop("invalid 'extendInt'; please report"))
  truncate <- function(x) pmax.int(pmin(x, .Machine$double.xmax),
    -.Machine$double.xmax)
  f.low. <- truncate(f.lower)
  f.upp. <- truncate(f.upper)
  doX <- (is.null(Sig) && f.low. * f.upp. > 0 || is.numeric(Sig) &&
    (Sig * f.low. > 0 || Sig * f.upp. < 0))
  if (doX) {
    if (trace)
      cat(sprintf("search in [%g,%g]%s", lower, upper,
```

```

        if (trace >= 2)
            "\n"
        else " ... ")
Delta <- function(u) 0.01 * pmax(1e-04, abs(u))
it <- 0L
if (is.null(Sig)) {
    delta <- Delta(c(lower, upper))
    while (isTRUE(f.lower * f.upper > 0) && any(iF <- is.finite(c(lower,
upper)))) {
        if ((it <- it + 1L) > maxiter)
            stop(gettextf("no sign change found in %d iterations",
                it - 1), domain = NA)
        if (iF[1]) {
            ol <- lower
            of <- f.lower
            if (is.na(f.lower <- f(lower <- lower - delta[1],
                ...))) {
                lower <- ol
                f.lower <- of
                delta[1] <- delta[1]/4
            }
        }
        if (iF[2]) {
            ol <- upper
            of <- f.upper
            if (is.na(f.upper <- f(upper <- upper + delta[2],
                ...))) {
                upper <- ol
                f.upper <- of
                delta[2] <- delta[2]/4
            }
        }
    }
    if (trace >= 2)
        cat(sprintf(" .. modified lower,upper: (%15g,%15g)\n",
            lower, upper))
}

```

```

        delta <- 2 * delta
    }
}
else {
    delta <- Delta(lower)
    while (isTRUE(Sig * f.lower > 0)) {
        if ((it <- it + 1L) > maxiter)
            stop(gettextf("no sign change found in %d iterations",
                it - 1), domain = NA)
        f.lower <- f(lower <- lower - delta, ...)
        if (trace >= 2)
            cat(sprintf(" .. modified lower: %g\n", lower))
        delta <- 2 * delta
    }
    delta <- Delta(upper)
    while (isTRUE(Sig * f.upper < 0)) {
        if ((it <- it + 1L) > maxiter)
            stop(gettextf("no sign change found in %d iterations",
                it - 1), domain = NA)
        f.upper <- f(upper <- upper + delta, ...)
        if (trace >= 2)
            cat(sprintf(" .. modified upper: %g\n", upper))
        delta <- 2 * delta
    }
}
if (trace && trace < 2)
    cat(sprintf("extended to [%g, %g] in %d steps\n",
        lower, upper, it))
}
if (!isTRUE(as.vector(sign(f.lower) * sign(f.upper) <= 0)))
    stop(if (doX)
        "did not succeed extending the interval endpoints for f(lower) * f(upper) <= 0"
        else "f() values at end points not of opposite sign")
if (check.conv) {
    val <- tryCatch(.External2(C_zeroIn2, function(arg) f(arg,

```

```

        ...), lower, upper, f.lower, f.upper, tol, as.integer(maxiter)),
        warning = function(w) w)
    if (inherits(val, "warning"))
        stop("convergence problem in zero finding: ", conditionMessage(val))
    }
    else {
        val <- .External2(C_zero2, function(arg) f(arg, ...),
            lower, upper, f.lower, f.upper, tol, as.integer(maxiter))
    }
    iter <- as.integer(val[2L])
    if (iter < 0) {
        (if (check.conv)
            stop
            else warning)(sprintf(ngettext(maxiter, "_NOT_ converged in %d iteration",
                "_NOT_ converged in %d iterations"), maxiter), domain = NA)
        iter <- maxiter
    }
    if (doX)
        iter <- iter + it
    else it <- NA_integer_
    list(root = val[1L], f.root = f(val[1L], ...), iter = iter,
        init.it = it, estim.prec = val[3L])
}
<bytecode: 0x7f835e2a4ed8>
<environment: namespace:stats>

```

Investigando variáveis (3)

Outra forma de investigar as variáveis é com a função `str`

```
str("Wilson")
```

```
chr "Wilson"
```

```
str(print)
```

```
function (x, ...)
```

Investigando variáveis (4)

Para investigar funções a função `args`

```
args(c)
```

```
function (... , recursive = FALSE)  
NULL
```

```
args(sd)
```

```
function (x, na.rm = FALSE)  
NULL
```


Onde é que eu estou?

A partir do instante em que foi inicializado o R está no *working directory* (wd) e para descobri-lo use `getwd()`:

```
getwd()
```

```
[1] "/Users/wilson/dev/Curso-Intro-R"
```

para mudar `setwd()`:

```
setwd('~')  
getwd()
```

```
[1] "/Users/wilson"
```