

Curso de R para meteorología IAG/USP

Sergio Ibarra-Espinosa e possivelmente outros (u r invited to collaborate)

2018-04-29

Contents

1	Pre-requisitos do sistema	5
1.1	Pacotes usados neste curso	5
1.2	Colaborar	6
2	Intro	7
3	R!	9
3.1	Objetos de R	9
3.2	Classe	9
3.3	Vetores	9
3.4	Convertir objetos com <code>as</code>	10
3.5	Matrizes e a função <code>matrix</code>	10
3.6	Array	10
3.7	<code>list</code>	12
3.8	Tempo e Data	12
3.9	Fatores	13
3.10	<code>data.frames</code>	14
4	Methods	17
5	Applications	19
5.1	Example one	19
5.2	Example two	19
6	Final Words	21

Chapter 1

Pre-requisitos do sistema

Em Windows, instale o R, Rtools <https://cran.r-project.org/bin/windows/Rtools/>

Em MAC instale netcdf e:

```
brew unlink gdal
brew tap osgeo/osgeo4mac && brew tap --repair
brew install proj
brew install geos
brew install udunits
brew install gdal2 --with-armadillo --with-complete --with-libkml --with-unsupported
brew link --force gdal2
```

Em Ubuntu:

```
- sudo add-apt-repository ppa:ubuntugis/ubuntugis-unstable --yes
- sudo apt-get --yes --force-yes update -qq
# install tmap dependencies
- sudo apt-get install --yes libprotobuf-dev protobuf-compiler libv8-3.14-dev
# install tmap dependencies; for 16.04 libjq-dev this ppa is needed:
- sudo add-apt-repository -y ppa:opencpu/jq
- sudo apt-get --yes --force-yes update -qq
- sudo apt-get install libjq-dev
# units/udunits2 dependency:
- sudo apt-get install --yes libudunits2-dev
# sf dependencies:
- sudo apt-get install --yes libproj-dev libgeos-dev libgdal-dev libnetcdf-dev netcdf-bin gdal-bin
```

1.1 Pacotes usados neste curso

Para fazer este curso instale os seguintes pacotes como se indica:

```
install.packages("devtools")
devtools::install_github("tidyverse/tidyverse")
devtools::install_github("r-spatial/sf")
devtools::install_github("r-spatial/mapview")
devtools::install_github("r-spatial/stars")
install.packages(c("raster", "sp", "rgdal", "maptools", "ncdf4"))
install.packages("cptcity")
```

- devtools é um pacote para instalar pacotes de diferentes repositórios
- tidyverse é universo de pacotes do Hadley Wickham. A instalação tem que ser usando devtools pois precisamos plotar os objetos espaciais sf usando geom_sf.
- sf e mapview, stars, raster, sp, rgdal e maptools são para a parte espacial. Lembrar que os objetos em meteorologias são espaço-temporais.
- ncdf4 é um pacote para manipular arquivos NetCDF.
- cptcity é um pacote que tem 7140 paletas de cores do arquivo web cpt-city (<http://soliton.vm.bytemark.co.uk/pub/cpt-city/index.html>).

1.2 Colaborar

A forma preferida de colaboração é com pull-requests em <https://github.com/ibarraespinosa/cursoR/pull/new/master>. Lembre de aplicar a Guia de Estilo de R de Google (<https://google.github.io/styleguide/Rguide.xml>) ou com o formato de formatR <https://yihui.name/formatr/>. Em poucas palavras, lembre que seu código vai ser lido por seres humanos.

Chapter 2

Intro

Este curso é para pos, então vamos ver conteúdo rapidamente e se não dá tempo, este curso está online no site <https://github.com/atmoschem/cursorIAG>.

Eu tento usar BASE sempre que posso, e se não dá aí vou para outros paradigmas.

Outros pacotes de BASE: `utils`, `stats`, `datasets`, `graphics`, `grDevices`, `grid`, `methods`, `tools`, `parallel`, `compiler`, `splines`, `tcltk`, `stats4`.

Veja outros pacotes.

Este curso está baseado no livro *R Programming for Data Science*.

Vamos usar Rstudio

Dica:

- Se não sabe como usar uma função, escreva: `?função`.
- As funções têm argumentos, use **TAB** para ver eles numa função.

Vamos lá!

Chapter 3

R!

- Quase em qualquer sistema operacional mas eu vou focar em Linux.
- Muita documentação:
- Intro.
- I/O.
- Quer fazer um pacote? Veja, aqui e aqui.
- Stackoverflow provides a great source of resources.

3.1 Objetos de R

- Character a
- numeric 1
- integer 1
- complex 0+1i
- logical TRUE

3.2 Classe

`class` função permite ver a classe dos objetos

3.3 Vetores

- `c("A", "C", "D")`
- `1:5 = c(1, 2, 3, 4, 5)`
- `c(TRUE, FALSE)`
- `c(1i, -1i)`
- `c(1, "C", "D")` qual é a classe???
- `c(1, NA, "D")` qual é a classe???
- `c(1, NA, NaN)` qual é a classe???

3.4 Converter objetos com as

```
as.numeric(c(1, "C", "D"))
```

```
## Warning: NAs introduzidos por coerção
```

```
## [1] 1 NA NA
```

3.5 Matrizes e a função matrix

[linhas, colunas]

- permitidos elementos **da mesma classe!**

vamos ver os argumentos da função `matrix`

```
args(matrix)
```

```
## function (data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
```

```
## NULL
```

usando TAB

```
(m <- matrix(data = 0, nrow = 4, ncol = 4))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  0   0   0   0
## [2,]  0   0   0   0
## [3,]  0   0   0   0
## [4,]  0   0   0   0
```

```
(m1 <- matrix(data = 1:(4*4), nrow = 4, ncol = 4))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  1   5   9  13
## [2,]  2   6  10  14
## [3,]  3   7  11  15
## [4,]  4   8  12  16
```

```
dim(m1)
```

```
## [1] 4 4
```

```
(m2 <- matrix(data = 1:(4*4), nrow = 4, ncol = 4, byrow = TRUE))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  1   2   3   4
## [2,]  5   6   7   8
## [3,]  9  10  11  12
## [4,] 13  14  15  16
```

3.6 Array

É como uma matriz de matrizes de matrizes de matrizes..... and so on.

```
args(array)
```

```
## function (data = NA, dim = length(data), dimnames = NULL)
## NULL
```

lembre usar TAB

```
(a <- array(data = 0, dim = c(1,1)))
```

```
##      [,1]
## [1,]    0
```

```
class(a)
```

```
## [1] "matrix"
```

```
(a <- array(data = 0, dim = c(1,1,1)))
```

```
## , , 1
##
##      [,1]
## [1,]    0
```

```
class(a)
```

```
## [1] "array"
```

```
(a <- array(data = 0, dim = c(2,2,2)))
```

```
## , , 1
##
##      [,1] [,2]
## [1,]    0    0
## [2,]    0    0
```

```
##
## , , 2
##
##      [,1] [,2]
## [1,]    0    0
## [2,]    0    0
```

```
(a <- array(data = 0, dim = c(2,4,4)))
```

```
## , , 1
##
##      [,1] [,2] [,3] [,4]
## [1,]    0    0    0    0
## [2,]    0    0    0    0
```

```
##
## , , 2
##
##      [,1] [,2] [,3] [,4]
## [1,]    0    0    0    0
## [2,]    0    0    0    0
```

```
##
## , , 3
##
##      [,1] [,2] [,3] [,4]
## [1,]    0    0    0    0
## [2,]    0    0    0    0
##
```

```
## , , 4
##
##      [,1] [,2] [,3] [,4]
## [1,]    0    0    0    0
## [2,]    0    0    0    0

dim(a)

## [1] 2 4 4

(a <- array(data = 0, dim = c(2, 2,2,2)))
```

3.7 list

As listas são como sacolas, e dentro delas, tu pode colocar mais sacolas... então, tu pode ter sacolas, dentro de sacolas, dentro de sacolas... ou

```
list(list(list(list(1)))
```

```
## [[1]]
## [[1]][[1]]
## [[1]][[1]][[1]]
## [[1]][[1]][[1]][[1]]
## [1] 1
```

a diferença das matrizes, tu pode colocar qualquer coisa nas listas, por exemplo: funções, characters, etc.

```
(x <- list(1, "a", TRUE, 1 + 4i))
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] 1+4i
```

3.8 Tempo e Data

R tem classes de tempo e data:

```
(a <- ISOdate(year = 2018, month = 4, day = 5))
```

```
## [1] "2018-04-05 12:00:00 GMT"
```

```
class(a)
```

```
## [1] "POSIXct" "POSIXt"
```

```
(b <- ISOdate(year = 2018, month = 4, day = 5, tz = "Americas/Sao_Paulo"))
```

```
## [1] "2018-04-05 12:00:00 Americas"
```

tempo

```
(d <- ISOdatetime(year = 2018, month = 4, day = 5, hour = 0, min = 0, sec = 0,
  tz = "Americas/Sao_Paulo"))
```

```
## [1] "2018-04-05 Americas"
```

O pacote nanotime permite trabalhar com nano segundos.

Da pra fazer sequencias:

```
hoje <- Sys.time()
(a <- seq.POSIXt(from = hoje, by = 3600, length.out = 24))
```

```
## [1] "2018-04-29 12:56:01 -03" "2018-04-29 13:56:01 -03"
## [3] "2018-04-29 14:56:01 -03" "2018-04-29 15:56:01 -03"
## [5] "2018-04-29 16:56:01 -03" "2018-04-29 17:56:01 -03"
## [7] "2018-04-29 18:56:01 -03" "2018-04-29 19:56:01 -03"
## [9] "2018-04-29 20:56:01 -03" "2018-04-29 21:56:01 -03"
## [11] "2018-04-29 22:56:01 -03" "2018-04-29 23:56:01 -03"
## [13] "2018-04-30 00:56:01 -03" "2018-04-30 01:56:01 -03"
## [15] "2018-04-30 02:56:01 -03" "2018-04-30 03:56:01 -03"
## [17] "2018-04-30 04:56:01 -03" "2018-04-30 05:56:01 -03"
## [19] "2018-04-30 06:56:01 -03" "2018-04-30 07:56:01 -03"
## [21] "2018-04-30 08:56:01 -03" "2018-04-30 09:56:01 -03"
## [23] "2018-04-30 10:56:01 -03" "2018-04-30 11:56:01 -03"
```

funções bacana: **weekdays**, **month**, **julian**

```
weekdays(a)
```

```
## [1] "domingo" "domingo" "domingo" "domingo" "domingo" "domingo" "domingo"
## [8] "domingo" "domingo" "domingo" "domingo" "domingo" "segunda" "segunda"
## [15] "segunda" "segunda" "segunda" "segunda" "segunda" "segunda" "segunda"
## [22] "segunda" "segunda" "segunda"
```

```
months(a)
```

```
## [1] "abril" "abril" "abril" "abril" "abril" "abril" "abril" "abril"
## [9] "abril" "abril" "abril" "abril" "abril" "abril" "abril" "abril"
## [17] "abril" "abril" "abril" "abril" "abril" "abril" "abril" "abril"
```

```
julian(a) #olha ?julian... dias desde origin
```

```
## Time differences in days
## [1] 17650.66 17650.71 17650.75 17650.79 17650.83 17650.87 17650.91
## [8] 17650.96 17651.00 17651.04 17651.08 17651.12 17651.16 17651.21
## [15] 17651.25 17651.29 17651.33 17651.37 17651.41 17651.46 17651.50
## [22] 17651.54 17651.58 17651.62
## attr(,"origin")
## [1] "1970-01-01 GMT"
```

olha https://en.wikipedia.org/wiki/Julian_day:

3.9 Fatores

Os **factors** podem ser um pouco infernais. Olha R INFERNO

Usados para representar categorias, ejemplo clasico para nos, dias da semana.

```
a <- seq.POSIXt(from = hoje, by = 3600, length.out = 24*7)
aa <- weekdays(a)
class(aa)

## [1] "character"

factor(aa)

## [1] domingo domingo domingo domingo domingo domingo domingo domingo
## [9] domingo domingo domingo domingo segunda segunda segunda segunda
## [17] segunda segunda segunda segunda segunda segunda segunda segunda
## [25] segunda segunda segunda segunda segunda segunda segunda segunda
## [33] segunda segunda segunda segunda terça terça terça terça
## [41] terça terça terça terça terça terça terça terça
## [49] terça terça terça terça terça terça terça terça
## [57] terça terça terça terça quarta quarta quarta quarta
## [65] quarta quarta quarta quarta quarta quarta quarta quarta
## [73] quarta quarta quarta quarta quarta quarta quarta quarta
## [81] quarta quarta quarta quarta quinta quinta quinta quinta
## [89] quinta quinta quinta quinta quinta quinta quinta quinta
## [97] quinta quinta quinta quinta quinta quinta quinta quinta
## [105] quinta quinta quinta quinta sexta sexta sexta sexta
## [113] sexta sexta sexta sexta sexta sexta sexta sexta
## [121] sexta sexta sexta sexta sexta sexta sexta sexta
## [129] sexta sexta sexta sexta sábado sábado sábado sábado
## [137] sábado sábado sábado sábado sábado sábado sábado sábado
## [145] sábado sábado sábado sábado sábado sábado sábado sábado
## [153] sábado sábado sábado sábado domingo domingo domingo domingo
## [161] domingo domingo domingo domingo domingo domingo domingo domingo
## Levels: domingo quarta quinta sábado segunda sexta terça
```

olha os **Levels**

Então:

```
ab <- factor(x = aa,
             levels = c("Monday", "Tuesday", "Wednesday", "Thursday",
                        "Friday", "Saturday", "Sunday"))
levels(ab)

## [1] "Monday"    "Tuesday"   "Wednesday" "Thursday"  "Friday"    "Saturday"
## [7] "Sunday"
```

3.10 data.frames

lembre ?data.frame

São como planilha EXCEL.... mais o menos

É uma classe bem especial, tem elementos de matriz mas o modo é lista

```
(df <- data.frame(a = 1:3))

## a
## 1 1
## 2 2
## 3 3
```

```
names(df)
```

```
## [1] "a"
```

```
class(df)
```

```
## [1] "data.frame"
```

```
mode(df)
```

```
## [1] "list"
```

Então

```
nrow(df)
```

```
## [1] 3
```

```
ncol(df)
```

```
## [1] 1
```

```
dim(df)
```

```
## [1] 3 1
```


Chapter 4

Methods

We describe our methods in this chapter.

Chapter 5

Applications

Some *significant* applications are demonstrated in this chapter.

5.1 Example one

5.2 Example two

Chapter 6

Final Words

We have finished a nice book.