

Curso de R para meteorologia IAG/USP

*Sergio Ibarra-Espinosa, Amanda Rehbein, Daniel Schuch, e possivelmente outros (u r
invited to collaborate)*

2018-04-30

Contents

1	Pre-requisitos do sistema	5
1.1	Pacotes usados neste curso	5
1.2	Colaborar	6
1.3	Aportar com dados	6
2	Intro	7
2.1	IMPORTANTE	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	Importando e exportando dados em R	17
4.1	<code>data.frames</code>	17
4.2	Processando nossa data-frame	21
4.3	<code>aggregate</code>	23
4.4	<code>subset</code>	23
4.5	<code>data.table</code> , <code>read_xl</code> e mais	23
4.6	<code>NetCDF</code>	23
4.7	Binarios	40
5	Applications	41
5.1	Example one	41
5.2	Example two	41
6	Final Words	43

Chapter 1

Pre-requisitos do sistema

Em Windows, instale além do 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 indicado:

```
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(c("cptcity", "data.table", "openair"))
```

- devtools é um pacote para instalar pacotes de diferentes repositórios
- tidyverse é o 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>).
- openair é um pacote para trabalhar com dados de qualidade do ar e meteorologia.

Se faltarem dependências de sistema, instale elas e instale os pacotes.

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. Se quiser tem acesso no repositório deste curso, me contate. Tem um botão para editar qualquer página.

1.3 Aportar com dados

Se você tem dados para fazer este curso mais legal, por favor, edite este arquivo e com pull request, eu vou fazer um merge para poder.

1. NCEP: ftp://nomads.ncdc.noaa.gov/GFS/analysis_only/
- 2.
- 3.

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 tem argumentos, use **TAB** para ver eles numa função.

2.1 IMPORTANTE

teu novo melhor amigo, best friend, BFF, parceiro, mano, tabarish, komrade, compaheiro, colega, business partner and whatever meaningful is

- **TAB** no **RSTUDIO**.

Esta combinação é tão boa, como o café com leite, pizza e abacaxi, vitamina de acabate com amendoim Manaus, a melhor combinação.



Porque quando se tu não lembra os argumentos da função, e não quer ver o help ? de cada função, so clica **TAB** e RSTUDIO te mostrara a lista de argumentos.

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] "2018-04-30 18:31:58 -03" "2018-04-30 19:31:58 -03"
## [3] "2018-04-30 20:31:58 -03" "2018-04-30 21:31:58 -03"
## [5] "2018-04-30 22:31:58 -03" "2018-04-30 23:31:58 -03"
## [7] "2018-05-01 00:31:58 -03" "2018-05-01 01:31:58 -03"
## [9] "2018-05-01 02:31:58 -03" "2018-05-01 03:31:58 -03"
## [11] "2018-05-01 04:31:58 -03" "2018-05-01 05:31:58 -03"
## [13] "2018-05-01 06:31:58 -03" "2018-05-01 07:31:58 -03"
## [15] "2018-05-01 08:31:58 -03" "2018-05-01 09:31:58 -03"
## [17] "2018-05-01 10:31:58 -03" "2018-05-01 11:31:58 -03"
## [19] "2018-05-01 12:31:58 -03" "2018-05-01 13:31:58 -03"
## [21] "2018-05-01 14:31:58 -03" "2018-05-01 15:31:58 -03"
## [23] "2018-05-01 16:31:58 -03" "2018-05-01 17:31:58 -03"
```

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

```
weekdays(a)
```

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

```
months(a)
```

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

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

```
## Time differences in days
## [1] 17651.90 17651.94 17651.98 17652.02 17652.06 17652.11 17652.15
## [8] 17652.19 17652.23 17652.27 17652.31 17652.36 17652.40 17652.44
## [15] 17652.48 17652.52 17652.56 17652.61 17652.65 17652.69 17652.73
## [22] 17652.77 17652.81 17652.86
## 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] segunda segunda segunda segunda segunda segunda terça terça
## [9] terça terça terça terça terça terça terça terça
## [17] terça terça terça terça terça terça terça terça
## [25] terça terça terça terça terça terça quarta quarta
## [33] quarta quarta quarta quarta quarta quarta quarta quarta
## [41] quarta quarta quarta quarta quarta quarta quarta quarta
## [49] quarta quarta quarta quarta quarta quarta quinta quinta
## [57] quinta quinta quinta quinta quinta quinta quinta quinta
## [65] quinta quinta quinta quinta quinta quinta quinta quinta
## [73] quinta quinta quinta quinta quinta quinta sexta sexta
## [81] sexta sexta sexta sexta sexta sexta sexta sexta
## [89] sexta sexta sexta sexta sexta sexta sexta sexta
## [97] sexta sexta sexta sexta sexta sexta sábado sábado
## [105] sábado sábado sábado sábado sábado sábado sábado sábado
## [113] sábado sábado sábado sábado sábado sábado sábado sábado
## [121] sábado sábado sábado sábado sábado sábado domingo domingo
## [129] domingo domingo domingo domingo domingo domingo domingo domingo
## [137] domingo domingo domingo domingo domingo domingo domingo domingo
## [145] domingo domingo domingo domingo domingo domingo segunda segunda
## [153] segunda segunda segunda segunda segunda segunda segunda segunda
## [161] segunda segunda segunda segunda segunda segunda segunda segunda
## 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

Importando e exportando dados em R

4.1 data-frames

Probavelmente um dos primeiros objetos que vamos usar quando começamos usar R. Pensa num data-frame como uma planilha de Libreoffice (o excel). Os data-frame pode ser criados como foi visto na seção anterior. O principal, é que temos varias funções para ler data-frames no R, entre elas

- `read.csv`
- `read.csv2`
- `read.table`

Agora vamos a ler dados do repositório usando `read.table`, mas primeiro vamos lembrar que se tu precisar ver a ajuda da função, tem que escrever no R `?read.table`. Então, agora vamos ver os argumentos da função:

```
args(read.table)
```

```
## function (file, header = FALSE, sep = ",", quote = "\"", dec = ".",
##     numerals = c("allow.loss", "warn.loss", "no.loss"), row.names,
##     col.names, as.is = !stringsAsFactors, na.strings = "NA",
##     colClasses = NA, nrow = -1, skip = 0, check.names = TRUE,
##     fill = !blank.lines.skip, strip.white = FALSE, blank.lines.skip = TRUE,
##     comment.char = "#", allowEscapes = FALSE, flush = FALSE,
##     stringsAsFactors = default.stringsAsFactors(), fileEncoding = "",
##     encoding = "unknown", text, skipNul = FALSE)
## NULL
```

Aqui vem-se os valores default dos argumentos da função `read.table`. O terceiro argumento é `sep`, com valores por default = “,”.

```
df <- read.table("https://raw.githubusercontent.com/ibarraespinosa/cursor/master/dados/NOXIPEN2014.txt")
```

Agora vamos usar as funções `head` and `tail` para ver as primeiras e as ultimas 6 linhas do data-frame.

```
head(df)
```

```
##   TipodeRede TipodeMonitoramento      Tipo      Data Hora
## 2 Automático          CETESB Dados Primários 01/01/2014 01:00
## 3 Automático          CETESB Dados Primários 01/01/2014 02:00
## 4 Automático          CETESB Dados Primários 01/01/2014 03:00
## 5 Automático          CETESB Dados Primários 01/01/2014 04:00
## 6 Automático          CETESB Dados Primários 01/01/2014 05:00
## 7 Automático          CETESB Dados Primários 01/01/2014 06:00
```

```
##      CodigoEstação      NomeEstação      NomeParâmetro
## 2      95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 3      95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 4      95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 5      95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 6      95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 7      95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
##      UnidadeMedida MediaHoraria MediaMovel Valido
## 2      ppb      9      -      Não
## 3      ppb      9      -      Sim
## 4      ppb      5      -      Sim
## 5      ppb      4      -      Sim
## 6      ppb      5      -      Sim
## 7      ppb      5      -      Sim
```

```
tail(df)
```

```
##      TipodeRede TipodeMonitoramento      Tipo      Data      Hora
## 8577 Automático      CETESB Dados Primários 01/01/2015 19:00
## 8578 Automático      CETESB Dados Primários 01/01/2015 20:00
## 8579 Automático      CETESB Dados Primários 01/01/2015 21:00
## 8580 Automático      CETESB Dados Primários 01/01/2015 22:00
## 8581 Automático      CETESB Dados Primários 01/01/2015 23:00
## 8582 Automático      CETESB Dados Primários 01/01/2015 24:00
##      CodigoEstação      NomeEstação      NomeParâmetro
## 8577      95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 8578      95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 8579      95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 8580      95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 8581      95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 8582      95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
##      UnidadeMedida MediaHoraria MediaMovel Valido
## 8577      ppb      3      -      Sim
## 8578      ppb      8      -      Sim
## 8579      ppb      11     -      Sim
## 8580      ppb      11     -      Sim
## 8581      ppb      16     -      Sim
## 8582      ppb      NA     -      Sim
```

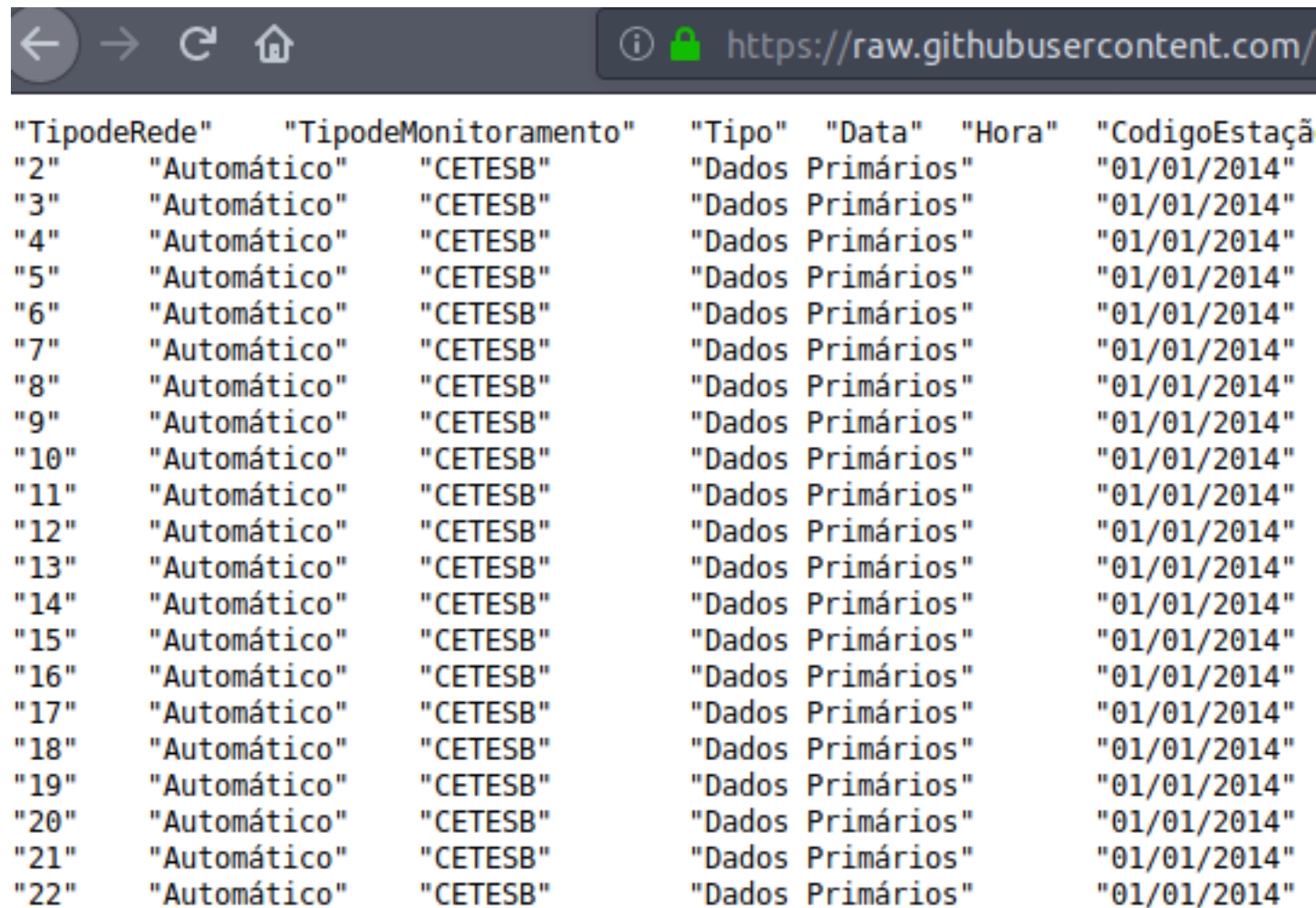
Agora vamos ler os mesmos dados com outro formato e testar se `read.table` funciona do mesmo jeito

```
df2 <- read.table("https://raw.githubusercontent.com/ibarraespinosa/cursoR/master/dados/NOXIPEN2014v2.t
# Error in scan(file = file, what = what, sep = sep, quote = quote, dec = dec, :
# linha 1 não tinha 6 elementos
```

Vemos a mensagem de error, mas o que quer dizer.

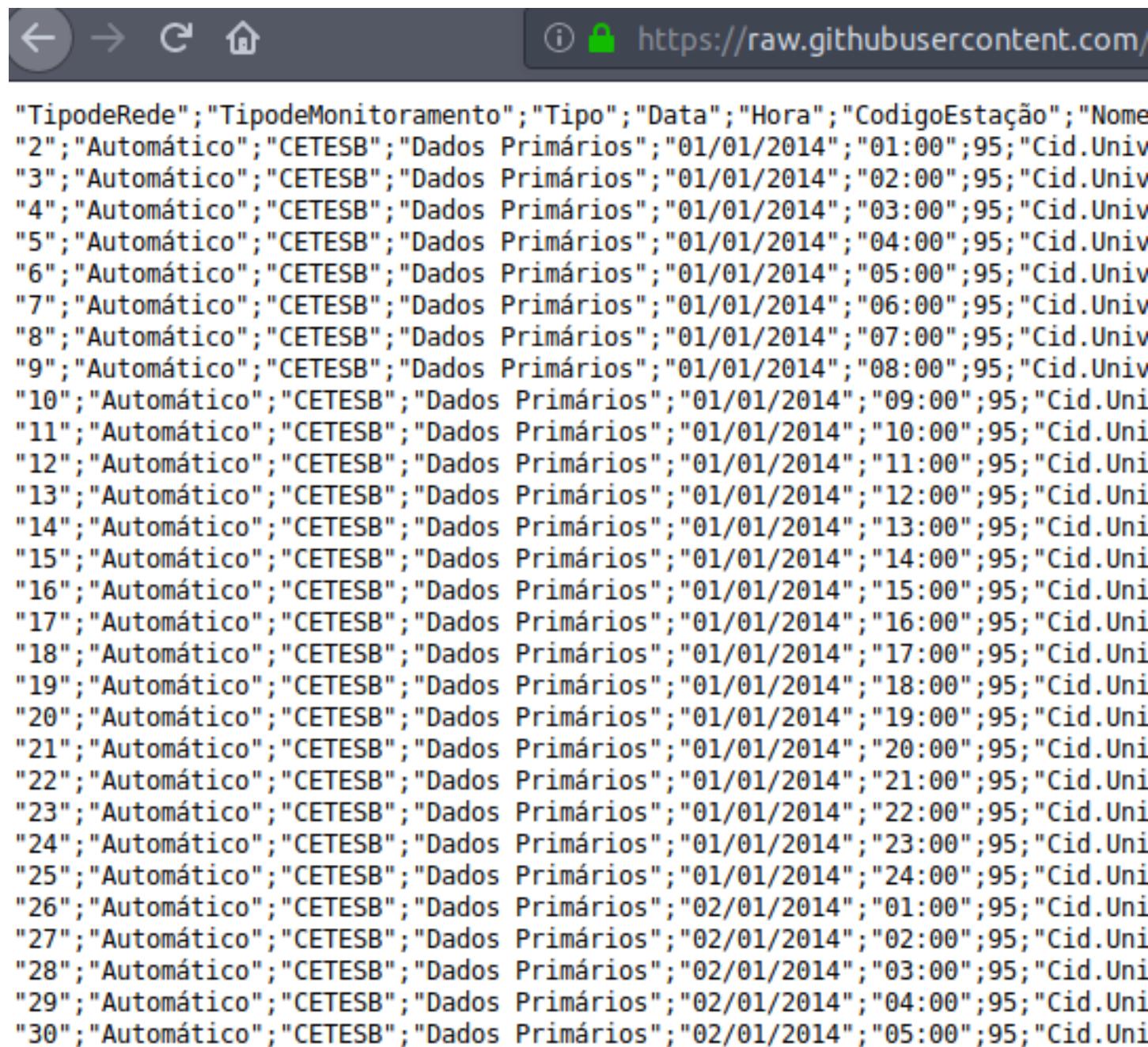
Se tu receber um banco de dados tipo .txt e quer abrir no R... ABRE ELE COM BLOCO DE NOTAS PRIMEIRO!!!

O primeiro arquivo:



"TipodeRede"	"TipodeMonitoramento"	"Tipo"	"Data"	"Hora"	"CodigoEstação"
"2"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"3"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"4"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"5"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"6"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"7"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"8"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"9"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"10"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"11"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"12"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"13"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"14"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"15"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"16"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"17"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"18"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"19"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"20"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"21"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"
"22"	"Automático"	"CETESB"	"Dados Primários"		"01/01/2014"

O segundo arquivo é:



```

"TipodeRede";"TipodeMonitoramento";"Tipo";"Data";"Hora";"CodigoEstação";"Nome
"2";"Automático";"CETESB";"Dados Primários";"01/01/2014";"01:00";95;"Cid.Univ
"3";"Automático";"CETESB";"Dados Primários";"01/01/2014";"02:00";95;"Cid.Univ
"4";"Automático";"CETESB";"Dados Primários";"01/01/2014";"03:00";95;"Cid.Univ
"5";"Automático";"CETESB";"Dados Primários";"01/01/2014";"04:00";95;"Cid.Univ
"6";"Automático";"CETESB";"Dados Primários";"01/01/2014";"05:00";95;"Cid.Univ
"7";"Automático";"CETESB";"Dados Primários";"01/01/2014";"06:00";95;"Cid.Univ
"8";"Automático";"CETESB";"Dados Primários";"01/01/2014";"07:00";95;"Cid.Univ
"9";"Automático";"CETESB";"Dados Primários";"01/01/2014";"08:00";95;"Cid.Univ
"10";"Automático";"CETESB";"Dados Primários";"01/01/2014";"09:00";95;"Cid.Uni
"11";"Automático";"CETESB";"Dados Primários";"01/01/2014";"10:00";95;"Cid.Uni
"12";"Automático";"CETESB";"Dados Primários";"01/01/2014";"11:00";95;"Cid.Uni
"13";"Automático";"CETESB";"Dados Primários";"01/01/2014";"12:00";95;"Cid.Uni
"14";"Automático";"CETESB";"Dados Primários";"01/01/2014";"13:00";95;"Cid.Uni
"15";"Automático";"CETESB";"Dados Primários";"01/01/2014";"14:00";95;"Cid.Uni
"16";"Automático";"CETESB";"Dados Primários";"01/01/2014";"15:00";95;"Cid.Uni
"17";"Automático";"CETESB";"Dados Primários";"01/01/2014";"16:00";95;"Cid.Uni
"18";"Automático";"CETESB";"Dados Primários";"01/01/2014";"17:00";95;"Cid.Uni
"19";"Automático";"CETESB";"Dados Primários";"01/01/2014";"18:00";95;"Cid.Uni
"20";"Automático";"CETESB";"Dados Primários";"01/01/2014";"19:00";95;"Cid.Uni
"21";"Automático";"CETESB";"Dados Primários";"01/01/2014";"20:00";95;"Cid.Uni
"22";"Automático";"CETESB";"Dados Primários";"01/01/2014";"21:00";95;"Cid.Uni
"23";"Automático";"CETESB";"Dados Primários";"01/01/2014";"22:00";95;"Cid.Uni
"24";"Automático";"CETESB";"Dados Primários";"01/01/2014";"23:00";95;"Cid.Uni
"25";"Automático";"CETESB";"Dados Primários";"01/01/2014";"24:00";95;"Cid.Uni
"26";"Automático";"CETESB";"Dados Primários";"02/01/2014";"01:00";95;"Cid.Uni
"27";"Automático";"CETESB";"Dados Primários";"02/01/2014";"02:00";95;"Cid.Uni
"28";"Automático";"CETESB";"Dados Primários";"02/01/2014";"03:00";95;"Cid.Uni
"29";"Automático";"CETESB";"Dados Primários";"02/01/2014";"04:00";95;"Cid.Uni
"30";"Automático";"CETESB";"Dados Primários";"02/01/2014";"05:00";95;"Cid.Uni

```

qual é a diferença?

Como vemos o segundo arquivo tem separação de “;”, então, temos que lero arquivo assim:

```
df2 <- read.table("https://raw.githubusercontent.com/ibarraespinosa/cursorR/master/dados/NOXIPEN2014v2.t
head(df2)
```

```
##   TipodeRede TipodeMonitoramento      Tipo      Data Hora
## 2 Automático          CETESB Dados Primários 01/01/2014 01:00
## 3 Automático          CETESB Dados Primários 01/01/2014 02:00
## 4 Automático          CETESB Dados Primários 01/01/2014 03:00
## 5 Automático          CETESB Dados Primários 01/01/2014 04:00
## 6 Automático          CETESB Dados Primários 01/01/2014 05:00
```

```
## 7 Automático          CETESB Dados Primários 01/01/2014 06:00
##   CodigoEstação      NomeEstação      NomeParâmetro
## 2          95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 3          95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 4          95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 5          95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 6          95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 7          95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
##   UnidadedeMedida MediaHoraria MediaMovel Valido
## 2          ppb          9          -      Não
## 3          ppb          9          -      Sim
## 4          ppb          5          -      Sim
## 5          ppb          4          -      Sim
## 6          ppb          5          -      Sim
## 7          ppb          5          -      Sim
```

```
tail(df2)
```

```
##   TipodeRede TipodeMonitoramento      Tipo      Data Hora
## 8577 Automático          CETESB Dados Primários 01/01/2015 19:00
## 8578 Automático          CETESB Dados Primários 01/01/2015 20:00
## 8579 Automático          CETESB Dados Primários 01/01/2015 21:00
## 8580 Automático          CETESB Dados Primários 01/01/2015 22:00
## 8581 Automático          CETESB Dados Primários 01/01/2015 23:00
## 8582 Automático          CETESB Dados Primários 01/01/2015 24:00
##   CodigoEstação      NomeEstação      NomeParâmetro
## 8577          95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 8578          95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 8579          95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 8580          95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 8581          95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
## 8582          95 Cid.Universitária-USP-Ipen NOx (Óxidos de Nitrogênio)
##   UnidadedeMedida MediaHoraria MediaMovel Valido
## 8577          ppb          3          -      Sim
## 8578          ppb          8          -      Sim
## 8579          ppb         11          -      Sim
## 8580          ppb         11          -      Sim
## 8581          ppb         16          -      Sim
## 8582          ppb         NA          -      Sim
```

4.1.1 Qua dificuldades tu já enfrentou importando dados?

4.2 Processando nossa data-frame

Tem numeroas formas e pacotes para ordenar, arranger (Arrange), mutar e cambiar as data-frames. As mais conhecidas são provavelmente do universe *tidyverse* com o famoso pacote *dplyr*. Mas, nesta curso vamos focar em **base**.

Vamos então revisar a classe de cada columna do nosso data-frame com a função **sapply**, apresentada em outro capítulo, mas se quiser, da uma olhada em **?sapply**.

```
sapply(df, class)
```

```
##   TipodeRede TipodeMonitoramento      Tipo
```

```
##          "factor"          "factor"          "factor"
##          Data             Hora             CódigoEstação
##          "factor"          "factor"          "integer"
##          NomeEstação       NomeParâmetro      UnidadeMedida
##          "factor"          "factor"          "factor"
##          MediaHoraria      MediaMovel         Valido
##          "integer"         "factor"          "factor"
```

Quando nos trabalhamos com series de tempo, é importante ter a variavel de tempo reconhecida como “tempo”, especificamente como classe “POSIXct”. Mas, a classe de Data é “factor” e de Hora também “factor”, o que é ruim. Então, vamos criar uma variavel de tempo mais standard com formato 2018-04-30 18:32:01.

Para isso temos que grudar as variavel Data e Hora. Faremos isso numa nova varaibel chamada `tempo_char`, adicionando ela diretamente no `df` com o cifrão DOLLAR \$. O grude pode ser feito com as funções `paste` ou `paste0`.

```
df$tempo_char <- paste(df$Data, df$Hora)
head(df$tempo_char)

## [1] "01/01/2014 01:00" "01/01/2014 02:00" "01/01/2014 03:00"
## [4] "01/01/2014 04:00" "01/01/2014 05:00" "01/01/2014 06:00"

class(df$tempo_char)
```

```
## [1] "character"
```

Esta melhorando mas ainda tem classe character.

Para converter a nossa classe POSIXct podemos usar a função `as.POSIXct` (olha `as.POSIXct`). Seus argumentos são:

```
args(as.POSIXct)

## function (x, tz = "", ...)
## NULL
```

Então, vamos criar outra variavel tempo o formato POSIXct

```
df$tempo <- as.POSIXct(x = df$tempo_char, tz = "Americas/Sao_Paulo",
                       format = "%d/%m/%Y %H:%M")
head(df$tempo)

## [1] "2014-01-01 01:00:00 Americas" "2014-01-01 02:00:00 Americas"
## [3] "2014-01-01 03:00:00 Americas" "2014-01-01 04:00:00 Americas"
## [5] "2014-01-01 05:00:00 Americas" "2014-01-01 06:00:00 Americas"

class(df$tempo)

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

Agora, vamos a extraer os dias da semana do tempo, mes e dia juliano:

```
df$weekdays <- weekdays(df$tempo)
head(df$weekdays)

## [1] "quarta" "quarta" "quarta" "quarta" "quarta" "quarta"

df$mes <- months(df$tempo)
head(df$mes)

## [1] "janeiro" "janeiro" "janeiro" "janeiro" "janeiro" "janeiro"
```



```
df$diajuliano <- julian(df$tempo)
head(df$diajuliano)
```

```
## Time differences in days
## [1] 16071.04 16071.08 16071.12 16071.17 16071.21 16071.25
```

4.3 aggregate

4.4 subset

4.5 data.table, read_xl e mais

data.table é um pacote que apresenta a classe `data.table`, que é como uma versão melhorada da classe `data.frame`. O termo específico é que `data-table` tem herencia (inherits) da classe `data.frame`.

Vamos ver como funciona data.table lendo o dois arquivos e comparar quanto tempo demoram cada um.

```
df1 <- print(system.time(read.table("https://raw.githubusercontent.com/ibarraespinoza/cursor/master/dados/NOXIPEN2014.txt")))
```

```
##      user  system elapsed
##    0.207   0.011   1.222
```

```
library(data.table)
```

```
df2 <- print(system.time(fread("https://raw.githubusercontent.com/ibarraespinoza/cursor/master/dados/NOXIPEN2014.txt")))
```

```
## Warning in fread("https://raw.githubusercontent.com/ibarraespinoza/
## cursor/master/dados/NOXIPEN2014.txt", : Starting data input on line 2
## and discarding line 1 because it has too few or too many items to be
## column names or data: "TipodeRede" "TipodeMonitoramento" "Tipo" "Data"
## "Hora" "CodigoEstação" "NomeEstação" "NomeParâmetro" "UnidadedeMedida"
## "MediaHoraria" "MediaMovel" "Valido"
```

```
##      user  system elapsed
##    0.025   0.004   0.089
```

olha que estamos usando a função `fread`.

`read_xl` é mais uma função do universo tidyverse que permite importar excel no R, diretamente e inteligentemente.

4.6 NetCDF

O NetCDF (Network Common Data Form) é um conjunto de bibliotecas de software e formatos de dados independentes de máquina e autodescritivos com suporte para criação, acesso e compartilhamento de dados científicos orientados a matrizes. Arquivos NetCDF (criado por essa biblioteca ou por programas que utilizam essa biblioteca) são arquivos com dados, atributos e metadados.

O pacote `ncdf4` pode ser usado com o R para ler e escrever estes arquivos, os comandos abaixo instalam e carregam o pacote:

```
install.packages("ncdf4")
nc_version() # que retorna a versão da biblioteca que o R está utilizando
```

Um arquivo então pode ser acessado por:

```
library("ncdf4")
download.file("https://github.com/ibarraespinosa/cursoR/raw/master/dados/met_em.d03.2016-01-10.nc", des
wrf <- nc_open("dados/met_em.d03.2016-01-10.nc")
```

A partir de agora o objeto `wrf`, contém algumas informações sobre o arquivo, por exemplo um `print(wrf)` ou simplesmente `wrf` mostra o conteúdo do arquivo:

```
wrf
```

```
## File dados/met_em.d03.2016-01-10.nc (NC_FORMAT_64BIT):
##
##      92 variables (excluding dimension variables):
##      char Times[DateStrLen,Time]
##      float PRES[west_east,south_north,num_metgrid_levels,Time]
##          FieldType: 104
##          MemoryOrder: XYZ
##          units:
##          description:
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float SOIL_LAYERS[west_east,south_north,num_st_layers,Time]
##          FieldType: 104
##          MemoryOrder: XYZ
##          units:
##          description:
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float SM[west_east,south_north,num_sm_layers,Time]
##          FieldType: 104
##          MemoryOrder: XYZ
##          units:
##          description:
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float ST[west_east,south_north,num_st_layers,Time]
##          FieldType: 104
##          MemoryOrder: XYZ
##          units:
##          description:
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float GHT[west_east,south_north,num_metgrid_levels,Time]
##          FieldType: 104
##          MemoryOrder: XYZ
##          units: m
##          description: Height
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float HGTTROP[west_east,south_north,Time]
##          FieldType: 104
```



```

##          MemoryOrder: XY
##          units: m
##          description: Height of tropopause
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float TTROP[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: K
##          description: Temperature at tropopause
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float PTROPNN[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: Pa
##          description: PTROP, used for nearest neighbor interp
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float PTROP[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: Pa
##          description: Pressure of tropopause
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float VTROP[west_east,south_north_stag,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: m s-1
##          description: V                                at tropopause
##          stagger: V
##          sr_x: 1
##          sr_y: 1
##      float UTROP[west_east_stag,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: m s-1
##          description: U                                at tropopause
##          stagger: U
##          sr_x: 1
##          sr_y: 1
##      float HGTMAXW[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: m
##          description: Height of max wind level
##          stagger: M
##          sr_x: 1
##          sr_y: 1

```

```

##      float TMAXW[west_east,south_north,Time]
##      FieldType: 104
##      MemoryOrder: XY
##      units: K
##      description: Temperature at max wind level
##      stagger: M
##      sr_x: 1
##      sr_y: 1
##      float PMAXWNN[west_east,south_north,Time]
##      FieldType: 104
##      MemoryOrder: XY
##      units: Pa
##      description: PMAXW, used for nearest neighbor interp
##      stagger: M
##      sr_x: 1
##      sr_y: 1
##      float PMAXW[west_east,south_north,Time]
##      FieldType: 104
##      MemoryOrder: XY
##      units: Pa
##      description: Pressure of max wind level
##      stagger: M
##      sr_x: 1
##      sr_y: 1
##      float VMAXW[west_east,south_north_stag,Time]
##      FieldType: 104
##      MemoryOrder: XY
##      units: m s-1
##      description: V                      at max wind
##      stagger: V
##      sr_x: 1
##      sr_y: 1
##      float UMAXW[west_east_stag,south_north,Time]
##      FieldType: 104
##      MemoryOrder: XY
##      units: m s-1
##      description: U                      at max wind
##      stagger: U
##      sr_x: 1
##      sr_y: 1
##      float SNOWH[west_east,south_north,Time]
##      FieldType: 104
##      MemoryOrder: XY
##      units: m
##      description: Physical Snow Depth
##      stagger: M
##      sr_x: 1
##      sr_y: 1
##      float SNOW[west_east,south_north,Time]
##      FieldType: 104
##      MemoryOrder: XY
##      units: kg m-2
##      description: Water equivalent snow depth
##      stagger: M

```

```

##          sr_x: 1
##          sr_y: 1
##      float SKINTEMP[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: K
##          description: Skin temperature
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float SOILHGT[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: m
##          description: Terrain field of source analysis
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float LANDSEA[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: proprtn
##          description: Land/Sea flag (1=land, 0 or 2=sea)
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float SEAICE[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: proprtn
##          description: Ice flag
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float ST100200[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: K
##          description: T 100-200 cm below ground layer (Bottom)
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float ST040100[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: K
##          description: T 40-100 cm below ground layer (Upper)
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float ST010040[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: K

```

```

##          description: T 10-40 cm below ground layer (Upper)
##          stagger: M
##          sr_x: 1
##          sr_y: 1
## float ST000010[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: K
##          description: T 0-10 cm below ground layer (Upper)
##          stagger: M
##          sr_x: 1
##          sr_y: 1
## float SM100200[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: fraction
##          description: Soil Moist 100-200 cm below gr layer
##          stagger: M
##          sr_x: 1
##          sr_y: 1
## float SM040100[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: fraction
##          description: Soil Moist 40-100 cm below grn layer
##          stagger: M
##          sr_x: 1
##          sr_y: 1
## float SM010040[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: fraction
##          description: Soil Moist 10-40 cm below grn layer
##          stagger: M
##          sr_x: 1
##          sr_y: 1
## float SM000010[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: fraction
##          description: Soil Moist 0-10 cm below grn layer (Up)
##          stagger: M
##          sr_x: 1
##          sr_y: 1
## float PSFC[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: Pa
##          description: Surface Pressure
##          stagger: M
##          sr_x: 1
##          sr_y: 1
## float RH[west_east,south_north,num_metgrid_levels,Time]
##          FieldType: 104

```

```

##          MemoryOrder: XYZ
##          units: %
##          description: Relative Humidity
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float VV[west_east,south_north_stag,num_metgrid_levels,Time]
##          FieldType: 104
##          MemoryOrder: XYZ
##          units: m s-1
##          description: V
##          stagger: V
##          sr_x: 1
##          sr_y: 1
##      float UU[west_east_stag,south_north,num_metgrid_levels,Time]
##          FieldType: 104
##          MemoryOrder: XYZ
##          units: m s-1
##          description: U
##          stagger: U
##          sr_x: 1
##          sr_y: 1
##      float TT[west_east,south_north,num_metgrid_levels,Time]
##          FieldType: 104
##          MemoryOrder: XYZ
##          units: K
##          description: Temperature
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float PMSL[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: Pa
##          description: Sea-level Pressure
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float URB_PARAM[west_east,south_north,z-dimension0132,Time]
##          FieldType: 104
##          MemoryOrder: XYZ
##          units: dimensionless
##          description: Urban_Parameters
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float LAKE_DEPTH[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: meters MSL
##          description: Topography height
##          stagger: M
##          sr_x: 1
##          sr_y: 1

```

```

##      float VAR_SSO[west_east,south_north,Time]
##      FieldType: 104
##      MemoryOrder: XY
##      units: meters2 MSL
##      description: Variance of Subgrid Scale Orography
##      stagger: M
##      sr_x: 1
##      sr_y: 1
##      float OL4[west_east,south_north,Time]
##      FieldType: 104
##      MemoryOrder: XY
##      units: whoknows
##      description: something
##      stagger: M
##      sr_x: 1
##      sr_y: 1
##      float OL3[west_east,south_north,Time]
##      FieldType: 104
##      MemoryOrder: XY
##      units: whoknows
##      description: something
##      stagger: M
##      sr_x: 1
##      sr_y: 1
##      float OL2[west_east,south_north,Time]
##      FieldType: 104
##      MemoryOrder: XY
##      units: whoknows
##      description: something
##      stagger: M
##      sr_x: 1
##      sr_y: 1
##      float OL1[west_east,south_north,Time]
##      FieldType: 104
##      MemoryOrder: XY
##      units: whoknows
##      description: something
##      stagger: M
##      sr_x: 1
##      sr_y: 1
##      float OA4[west_east,south_north,Time]
##      FieldType: 104
##      MemoryOrder: XY
##      units: whoknows
##      description: something
##      stagger: M
##      sr_x: 1
##      sr_y: 1
##      float OA3[west_east,south_north,Time]
##      FieldType: 104
##      MemoryOrder: XY
##      units: whoknows
##      description: something
##      stagger: M

```

```

##          sr_x: 1
##          sr_y: 1
##      float OA2[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: whoknows
##          description: something
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float OA1[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: whoknows
##          description: something
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float VAR[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: whoknows
##          description: something
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float CON[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: whoknows
##          description: something
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float SLOPECAT[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: category
##          description: Dominant category
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float SNOALB[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: percent
##          description: Maximum snow albedo
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float LAI12M[west_east,south_north,z-dimension0012,Time]
##          FieldType: 104
##          MemoryOrder: XYZ
##          units: m^2/m^2

```

```

##         description: MODIS LAI
##         stagger: M
##         sr_x: 1
##         sr_y: 1
## float GREENFRAC[west_east,south_north,z-dimension0012,Time]
##         FieldType: 104
##         MemoryOrder: XYZ
##         units: fraction
##         description: MODIS FPAR
##         stagger: M
##         sr_x: 1
##         sr_y: 1
## float ALBEDO12M[west_east,south_north,z-dimension0012,Time]
##         FieldType: 104
##         MemoryOrder: XYZ
##         units: percent
##         description: Monthly surface albedo
##         stagger: M
##         sr_x: 1
##         sr_y: 1
## float SCB_DOM[west_east,south_north,Time]
##         FieldType: 104
##         MemoryOrder: XY
##         units: category
##         description: Dominant category
##         stagger: M
##         sr_x: 1
##         sr_y: 1
## float SOILCBOT[west_east,south_north,z-dimension0016,Time]
##         FieldType: 104
##         MemoryOrder: XYZ
##         units: category
##         description: 16-category bottom-layer soil type
##         stagger: M
##         sr_x: 1
##         sr_y: 1
## float SCT_DOM[west_east,south_north,Time]
##         FieldType: 104
##         MemoryOrder: XY
##         units: category
##         description: Dominant category
##         stagger: M
##         sr_x: 1
##         sr_y: 1
## float SOILCTOP[west_east,south_north,z-dimension0016,Time]
##         FieldType: 104
##         MemoryOrder: XYZ
##         units: category
##         description: 16-category top-layer soil type
##         stagger: M
##         sr_x: 1
##         sr_y: 1
## float SOILTEMP[west_east,south_north,Time]
##         FieldType: 104

```



```

##          MemoryOrder: XY
##          units: Kelvin
##          description: Annual mean deep soil temperature
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float HGT_M[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: meters MSL
##          description: GMTED2010 30-arc-second topography height
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float LU_INDEX[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: category
##          description: Dominant category
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float LANDUSEF[west_east,south_north,z-dimension0024,Time]
##          FieldType: 104
##          MemoryOrder: XYZ
##          units: category
##          description: 24-category USGS landuse
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float COSALPHA_V[west_east,south_north_stag,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none
##          description: Cosine of rotation angle on V grid
##          stagger: V
##          sr_x: 1
##          sr_y: 1
##      float SINALPHA_V[west_east,south_north_stag,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none
##          description: Sine of rotation angle on V grid
##          stagger: V
##          sr_x: 1
##          sr_y: 1
##      float COSALPHA_U[west_east_stag,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none
##          description: Cosine of rotation angle on U grid
##          stagger: U
##          sr_x: 1
##          sr_y: 1

```

```

##      float SINALPHA_U[west_east_stag,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none
##          description: Sine of rotation angle on U grid
##          stagger: U
##          sr_x: 1
##          sr_y: 1
##      float XLONG_C[west_east_stag,south_north_stag,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: degrees longitude
##          description: Longitude at grid cell corners
##          stagger: CORNER
##          sr_x: 1
##          sr_y: 1
##      float XLAT_C[west_east_stag,south_north_stag,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: degrees latitude
##          description: Latitude at grid cell corners
##          stagger: CORNER
##          sr_x: 1
##          sr_y: 1
##      float LANDMASK[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none
##          description: Landmask : 1=land, 0=water
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float COSALPHA[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none
##          description: Cosine of rotation angle
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float SINALPHA[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none
##          description: Sine of rotation angle
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float F[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: -
##          description: Coriolis F parameter
##          stagger: M

```

```

##          sr_x: 1
##          sr_y: 1
##      float E[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: -
##          description: Coriolis E parameter
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float MAPFAC_UY[west_east_stag,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none
##          description: Mapfactor (y-dir) on U grid
##          stagger: U
##          sr_x: 1
##          sr_y: 1
##      float MAPFAC_VY[west_east,south_north_stag,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none
##          description: Mapfactor (y-dir) on V grid
##          stagger: V
##          sr_x: 1
##          sr_y: 1
##      float MAPFAC_MY[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none
##          description: Mapfactor (y-dir) on mass grid
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float MAPFAC_UX[west_east_stag,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none
##          description: Mapfactor (x-dir) on U grid
##          stagger: U
##          sr_x: 1
##          sr_y: 1
##      float MAPFAC_VX[west_east,south_north_stag,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none
##          description: Mapfactor (x-dir) on V grid
##          stagger: V
##          sr_x: 1
##          sr_y: 1
##      float MAPFAC_MX[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none

```

```

##          description: Mapfactor (x-dir) on mass grid
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##    float MAPFAC_U[west_east_stag,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none
##          description: Mapfactor on U grid
##          stagger: U
##          sr_x: 1
##          sr_y: 1
##    float MAPFAC_V[west_east,south_north_stag,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none
##          description: Mapfactor on V grid
##          stagger: V
##          sr_x: 1
##          sr_y: 1
##    float MAPFAC_M[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: none
##          description: Mapfactor on mass grid
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##    float CLONG[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: degrees longitude
##          description: Computational longitude on mass grid
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##    float CLAT[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: degrees latitude
##          description: Computational latitude on mass grid
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##    float XLONG_U[west_east_stag,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: degrees longitude
##          description: Longitude on U grid
##          stagger: U
##          sr_x: 1
##          sr_y: 1
##    float XLAT_U[west_east_stag,south_north,Time]
##          FieldType: 104

```

```

##          MemoryOrder: XY
##          units: degrees latitude
##          description: Latitude on U grid
##          stagger: U
##          sr_x: 1
##          sr_y: 1
##      float XLONG_V[west_east,south_north_stag,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: degrees longitude
##          description: Longitude on V grid
##          stagger: V
##          sr_x: 1
##          sr_y: 1
##      float XLAT_V[west_east,south_north_stag,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: degrees latitude
##          description: Latitude on V grid
##          stagger: V
##          sr_x: 1
##          sr_y: 1
##      float XLONG_M[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: degrees longitude
##          description: Longitude on mass grid
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##      float XLAT_M[west_east,south_north,Time]
##          FieldType: 104
##          MemoryOrder: XY
##          units: degrees latitude
##          description: Latitude on mass grid
##          stagger: M
##          sr_x: 1
##          sr_y: 1
##
##      13 dimensions:
##          Time Size:1   *** is unlimited ***
##          DateStrLen Size:19
##          west_east Size:51
##          south_north Size:51
##          num_metgrid_levels Size:27
##          num_st_layers Size:4
##          num_sm_layers Size:4
##          south_north_stag Size:52
##          west_east_stag Size:52
##          z-dimension0132 Size:132
##          z-dimension0012 Size:12
##          z-dimension0016 Size:16
##          z-dimension0024 Size:24
##

```

```

##      76 global attributes:
##      TITLE: OUTPUT FROM METGRID V3.9.1
##      SIMULATION_START_DATE: 2016-01-10_00:00:00
##      WEST-EAST_GRID_DIMENSION: 52
##      SOUTH-NORTH_GRID_DIMENSION: 52
##      BOTTOM-TOP_GRID_DIMENSION: 27
##      WEST-EAST_PATCH_START_UNSTAG: 1
##      WEST-EAST_PATCH_END_UNSTAG: 51
##      WEST-EAST_PATCH_START_STAG: 1
##      WEST-EAST_PATCH_END_STAG: 52
##      SOUTH-NORTH_PATCH_START_UNSTAG: 1
##      SOUTH-NORTH_PATCH_END_UNSTAG: 51
##      SOUTH-NORTH_PATCH_START_STAG: 1
##      SOUTH-NORTH_PATCH_END_STAG: 52
##      GRIDTYPE: C
##      DX: 1000
##      DY: 1000
##      DYN_OPT: 2
##      CEN_LAT: -23.5996932983398
##      CEN_LON: -46.6294555664062
##      TRUELAT1: -23
##      TRUELAT2: -24
##      MOAD_CEN_LAT: -23.6000061035156
##      STAND_LON: -45
##      POLE_LAT: 90
##      POLE_LON: 0
##      corner_lats: -23.8218078613281
##      corner_lats: -23.3720855712891
##      corner_lats: -23.3771743774414
##      corner_lats: -23.826904296875
##      corner_lats: -23.8217391967773
##      corner_lats: -23.3720245361328
##      corner_lats: -23.3772277832031
##      corner_lats: -23.8269424438477
##      corner_lats: -23.826286315918
##      corner_lats: -23.3675918579102
##      corner_lats: -23.372673034668
##      corner_lats: -23.8314056396484
##      corner_lats: -23.8262329101562
##      corner_lats: -23.3675231933594
##      corner_lats: -23.3727111816406
##      corner_lats: -23.8314437866211
##      corner_lons: -46.8780517578125
##      corner_lons: -46.8716430664062
##      corner_lons: -46.3817138671875
##      corner_lons: -46.3864440917969
##      corner_lons: -46.8829650878906
##      corner_lons: -46.8765258789062
##      corner_lons: -46.3768005371094
##      corner_lons: -46.3815307617188
##      corner_lons: -46.8781127929688
##      corner_lons: -46.87158203125
##      corner_lons: -46.3816528320312
##      corner_lons: -46.386474609375

```

```
##      corner_lons: -46.8830261230469
##      corner_lons: -46.87646484375
##      corner_lons: -46.3767700195312
##      corner_lons: -46.3815612792969
##      MAP_PROJ: 1
##      MMINLU: USGS
##      NUM_LAND_CAT: 24
##      ISWATER: 16
##      ISLAKE: -1
##      ISICE: 24
##      ISURBAN: 1
##      ISOILWATER: 14
##      grid_id: 3
##      parent_id: 2
##      i_parent_start: 35
##      j_parent_start: 33
##      i_parent_end: 51
##      j_parent_end: 49
##      parent_grid_ratio: 3
##      sr_x: 1
##      sr_y: 1
##      NUM_METGRID_SOIL_LEVELS: 4
##      FLAG_METGRID: 1
##      FLAG_EXCLUDED_MIDDLE: 0
##      FLAG_SOIL_LAYERS: 1
##      FLAG_SNOW: 1
##      FLAG_PSFC: 1
##      FLAG_SMO00010: 1
##      FLAG_SMO10040: 1
##      FLAG_SMO40100: 1
##      FLAG_SM100200: 1
##      FLAG_ST000010: 1
##      FLAG_ST010040: 1
##      FLAG_ST040100: 1
##      FLAG_ST100200: 1
##      FLAG_SLP: 1
##      FLAG_SNOWH: 1
##      FLAG_SOILHGT: 1
##      FLAG_UTROP: 1
##      FLAG_VTROP: 1
##      FLAG_TTROP: 1
##      FLAG_PTROP: 1
##      FLAG_PTROPNN: 1
##      FLAG_HGTTROP: 1
##      FLAG_UMAXW: 1
##      FLAG_VMAXW: 1
##      FLAG_TMAXW: 1
##      FLAG_PMAXW: 1
##      FLAG_PMAXWNN: 1
##      FLAG_HGTMAXW: 1
##      FLAG_MF_XY: 1
##      FLAG_LAI12M: 1
##      FLAG_LAKE_DEPTH: 1
```

Entre a informação mostrada na tela estão o nome do arquivo (e versão da biblioteca usada para criar), numero de variáveis (92 no arquivo de exemplo), uma descrição de cada variável (incluindo atributos) as dimensões (13 para esse arquivo) e os atributos globais.

Agora vamos abrir alguma variável:

```
names(wrf$var)           # print no nome de cada variavel
```

## [1]	"Times"	"PRES"	"SOIL_LAYERS"	"SM"	"ST"
## [6]	"GHT"	"HGT Trop"	"TTrop"	"PTropNN"	"PTrop"
## [11]	"VTrop"	"UTrop"	"HGTMAXW"	"TMAXW"	"PMAWNN"
## [16]	"PMAW"	"VMAW"	"UMAXW"	"SNOWH"	"SNOW"
## [21]	"SKINTemp"	"SOILHGT"	"LANDSEA"	"SEAICE"	"ST100200"
## [26]	"ST040100"	"ST010040"	"ST000010"	"SM100200"	"SM040100"
## [31]	"SM010040"	"SM000010"	"PSFC"	"RH"	"VV"
## [36]	"UU"	"TT"	"PMSL"	"URB_PARAM"	"LAKE_DEPTH"
## [41]	"VAR_SS0"	"OL4"	"OL3"	"OL2"	"OL1"
## [46]	"OA4"	"OA3"	"OA2"	"OA1"	"VAR"
## [51]	"CON"	"SLOPECAT"	"SNOALB"	"LAI12M"	"GREENFRAC"
## [56]	"ALBEDO12M"	"SCB_DOM"	"SOILCBOT"	"SCT_DOM"	"SOILCTOP"
## [61]	"SOILTEMP"	"HGT_M"	"LU_INDEX"	"LANDUSEF"	"COSALPHA_V"
## [66]	"SINALPHA_V"	"COSALPHA_U"	"SINALPHA_U"	"XLONG_C"	"XLAT_C"
## [71]	"LANDMASK"	"COSALPHA"	"SINALPHA"	"F"	"E"
## [76]	"MAPFAC_UY"	"MAPFAC_VY"	"MAPFAC_MY"	"MAPFAC_UX"	"MAPFAC_VX"
## [81]	"MAPFAC_MX"	"MAPFAC_U"	"MAPFAC_V"	"MAPFAC_M"	"CLONG"
## [86]	"CLAT"	"XLONG_U"	"XLAT_U"	"XLONG_V"	"XLAT_V"
## [91]	"XLONG_M"	"XLAT_M"			

```
ST <- ncvar_get(wrf, "ST") # escolho você picachu
```

ncvar_get Read data from a netCDF file

ncatt_get Get attribute from netCDF file ncatt_put Put an attribute into a netCDF file

ncvar_add Add New netCDF Variable to Existing File ncvar_change_missval Change the Missing Value For a netCDF Variable ncvar_def Define a netCDF Variable ncvar_put Write data to a netCDF file ncvar_rename Rename an Existing Variable in a netCDF File

nc_version Report version of ncdf4 library nc_create Create a netCDF File ncdim_def Define a netCDF Dimension nc_enddef Takes a netCDF file out of define mode nc_redef Puts a netCDF file back into define mode

nc_close Close a netCDF File nc_sync Synchronize (flush to disk) a netCDF File

Para salvar toda informação e liberar o acesso ao arquivo use a função `nc_close` (ou a função `nc_sync` que sincroniza o NetCDF carregado na memória com o NetCDF no Disco)

```
nc_close(wrf)
```

4.7 Binarios

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.