

Curso de R para Meteorologia IAG/USP

*Sergio Ibarra-Espinosa, Amanda Rehbein, Daniel Schuch, Camila Lopes, Isabela Siqueira, e
possivelmente outros (você está convidado para colaborar)*

2018-05-22

Contents

1	Pré-requisitos do sistema	5
1.1	Pacotes usados neste curso	5
1.2	Colaborar	6
1.3	Compartilhar dados	6
2	Intro	7
2.1	IMPORTANTE	7
3	R!	9
3.1	Objetos do R	9
3.2	Classe	9
3.3	Vetores	9
3.4	Converter objetos	10
3.5	Matrizes e a função <code>matrix</code>	10
3.6	Array	11
3.7	<code>list</code>	12
3.8	Tempo e Data	12
3.9	Fatores	14
3.10	Data.frames	15
4	Importando e exportando dados em R	17
4.1	data.frames	17
4.2	Exportando texto com <code>write.table</code>	21
4.3	Exportando objetos com <code>save</code>	22
4.4	Exportando objetos com <code>saveRDS</code>	22
4.5	Processando nossa data-frame	23
4.6	<code>aggregate</code>	24
4.7	<code>subset</code>	25
4.8	<code>data.table</code> , <code>read_xl</code> e mais	26
4.9	<code>NetCDF</code>	27
5	Plotando	63
5.1	<code>plot</code> (base)	63
5.2	<code>ggplot</code> (<code>ggplot2</code>)	66
6	Estruturas de control	77
6.1	<code>if-else</code>	77
6.2	<code>for</code>	77
6.3	<code>while</code>	77
6.4	<code>repeat</code>	77
6.5	<code>lapply</code>	77
6.6	<code>sapply</code>	77

6.7	split	77
6.8	tapply	77
6.9	apply	77
6.10	mapply	77
7	De scripts a funções e de funções a pacotes	79
8	Geo Spatial: raster, sf e stars	81

Chapter 1

Pré-requisitos do sistema

Antes de instalar o R na sua plataforma de interesse, verifique se há recomendações abaixo:

Windows

A princípio não há pré-requisitos! Caso fique entusiasmado com o R e queira desenvolver os próprios pacotes, instale o Rtools <https://cran.r-project.org/bin/windows/Rtools/>

MacOS

Instale NetCDF4 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
```

Linux (Ubuntu e derivados)

```
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")
install.packages("tidyverse")
install.packages("reshape2")
```

```
install.packages("sf")
install.packages("maptools")
install.packages("mapview")
install.packages("fields")
install.packages("raster")
install.packages("sp")
install.packages("rgdal")
install.packages("ncdf4")
install.packages("data.table")
install.packages("openair")
install.packages("cptcity")
```

- devtools permite a instalação de versões de desenvolvimento de pacotes de diferentes repositórios
- tidyverse é o universo de pacotes do Hadley Wickham para tratamento e visualização de dados
 - Se você quiser plotar os objetos espaciais sf com o pacote ggplot2 (que faz parte do tidyverse), ele precisa ser instalado usando o devtools (`devtools::install_github("tidyverse/ggplot2")`), pois a função `geom_sf` ainda não está disponível na versão oficial
- sf, mapview, raster, sp, rgdal, maptools e fields tratam dados espaciais. Lembre-se que os objetos em Meteorologia 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
- openair é um pacote para trabalhar com dados de qualidade do ar e Meteorologia

Preste atenção na instalação dos pacotes pois eles podem precisar de dependências do sistema.

1.2 Colaborar

A melhor forma de colaboração é com *pull requests* em <https://github.com/ibarraespinosa/cursorR/pull/new/master>. Aplique o Guia de Estilo de R do Google ou o formato formatR. Em poucas palavras, lembre que seu código vai ser lido por seres humanos. É possível editar qualquer página usando um dos botões acima.

1.3 Compartilhar dados

Se você conhece alguma fonte de dados para deixar este curso mais legal, edite este arquivo e faça um *pull request*.

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

Chapter 2

Intro

Este curso é voltado para os alunos de pós-graduação, dessa forma, veremos os conceitos rapidamente. Caso não haja tempo, o conteúdo ficará online no link: <https://github.com/atmoschem/cursorIAG>.

Sempre que tiver uma dúvida, tente utilizar: `BASE`.

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

Acesse outros para a lista de pacotes disponíveis.

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

Neste curso iremos utilizar `Rstudio`

Dica:

- Se não souber usar uma função, escreva: `?função`.
- As funções tem argumentos, use **TAB** para vê-los numa função.

2.1 IMPORTANTE

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

Isso te ajudará a evitar coisas como: grafia errada da função, verificar se a função existe, verificar argumentos, etc... Use sempre!



Vamos começar!

Chapter 3

R!

- Iremos focar no Linux, mas R e RStudio estão disponíveis para Windows e Mac também.
- Documentação:
- Intro.
- I/O.
- Quer fazer um pacote? Veja, aqui e aqui.
- Stackoverflow te ajudará em horas de desespero.

3.1 Objetos do R

- Character a
- Numeric 1
- Integer 1
- Complex 0+1i
- Logical TRUE

3.2 Classe

`class` essa 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

3.4.1 as

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

## Warning: NAs introduzidos por coerção
## [1]  1 NA NA
```

3.4.2 merge e melt

3.5 Matrizes e a função matrix

[linhas, colunas]

- Só são permitidos elementos da mesma classe!

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

Permite armazenar diversos elementos, com diversas dimensões. Dessa forma, um array com duas dimensões é o mesmo que uma matriz, dessa forma, podemos armazenar diversas matrizes dentro de um array, mas suas dimensões são pré-estabelecidas.

```
args(array)
```

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

Não esqueça do 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

Já as listas permitem que você armazene qualquer tipo de objeto, independente da classe, dessa forma, podemos colocar numa lista: número, caracteres, argumentos lógicos, ou que você quiser:

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

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

Isso faz com que elas sejam bastante versáteis e sirvam para armazenar o que você precisar, mas elas só podem ter uma dimensão, como uma fila.

```
(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 também 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"
```

Caso você precise, o pacote nanotime permite trabalhar com nano segundos.

É possível fazer sequências:

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

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

funções úteis: **weekdays**, **month**, **julian**

```
weekdays(a)

## [1] "terça" "terça" "quarta" "quarta" "quarta" "quarta" "quarta"
## [8] "quarta" "quarta" "quarta" "quarta" "quarta" "quarta" "quarta"
## [15] "quarta" "quarta" "quarta" "quarta" "quarta" "quarta" "quarta"
## [22] "quarta" "quarta" "quarta"

months(a)

## [1] "maio" "maio" "maio" "maio" "maio" "maio" "maio" "maio" "maio" "maio"
## [11] "maio" "maio" "maio" "maio" "maio" "maio" "maio" "maio" "maio" "maio"
## [21] "maio" "maio" "maio" "maio"

julian(a) #dia Juliano*

## Time differences in days
## [1] 17674.05 17674.09 17674.13 17674.17 17674.22 17674.26 17674.30
## [8] 17674.34 17674.38 17674.42 17674.47 17674.51 17674.55 17674.59
## [15] 17674.63 17674.67 17674.72 17674.76 17674.80 17674.84 17674.88
## [22] 17674.92 17674.97 17675.01
```

```
## attr("origin")
## [1] "1970-01-01 GMT"
```

*Para mais informações: https://en.wikipedia.org/wiki/Julian_day:

3.9 Fatores

Os factors podem ser um pouco infernais. Dê uma olhada em R INFERNO

São variáveis que representam categorias, como por exemplo, dias da semana.

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

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

```
factor(aa)
```

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

São muito úteis para regressões, plotes e resumos estatísticos.

Olhe 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

Lembram uma planilha EXCEL... Mais ou 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"
```

Podemos utilizar para armazenar dados, sendo que um data.frame é sempre composto por vetores com comprimento IGUAL

```
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

Provavelmente 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
```

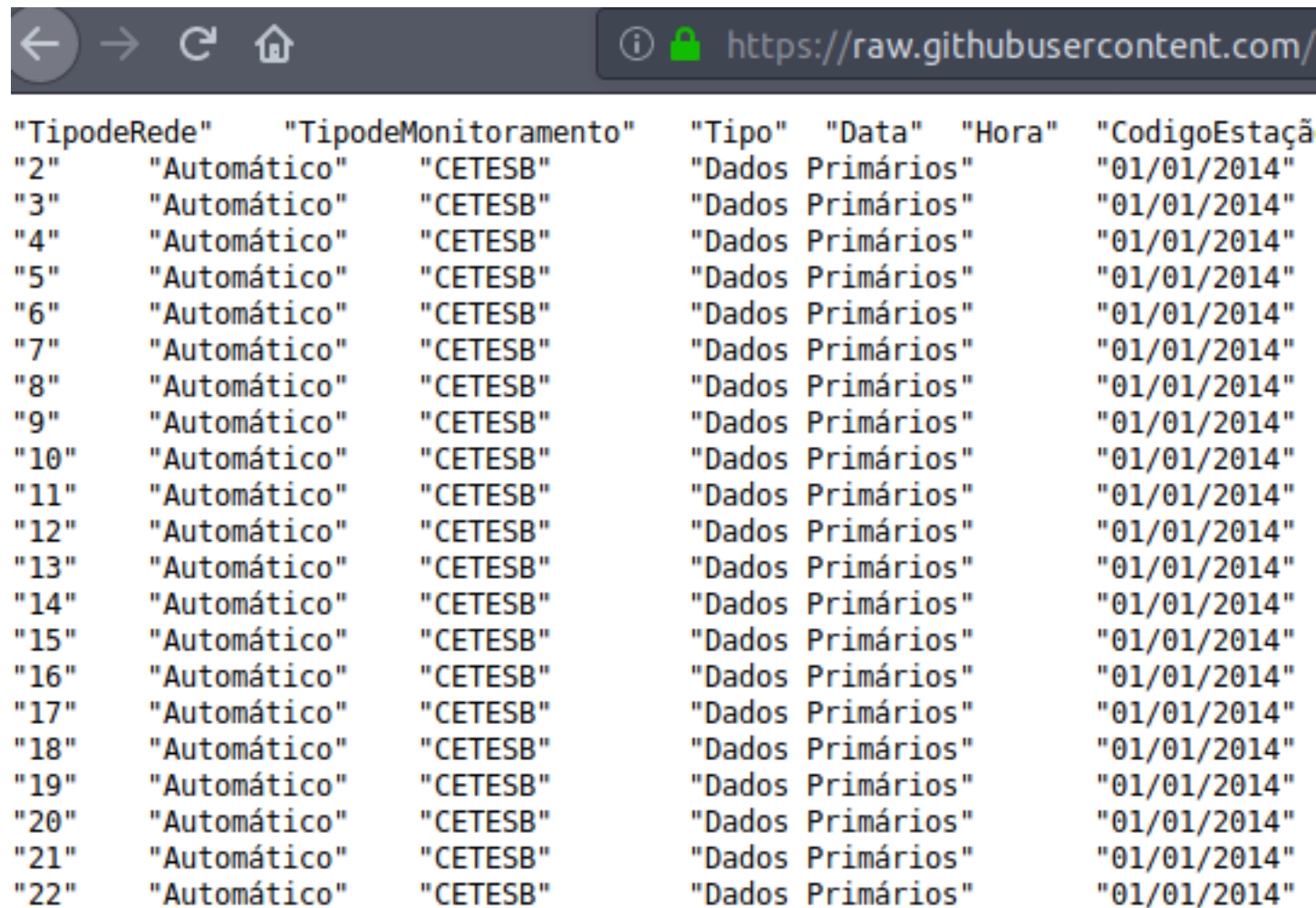
Agora vamos ler os mesmos dados com outro formato e testar se `read.table` funciona do mesmo jeito

```
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.

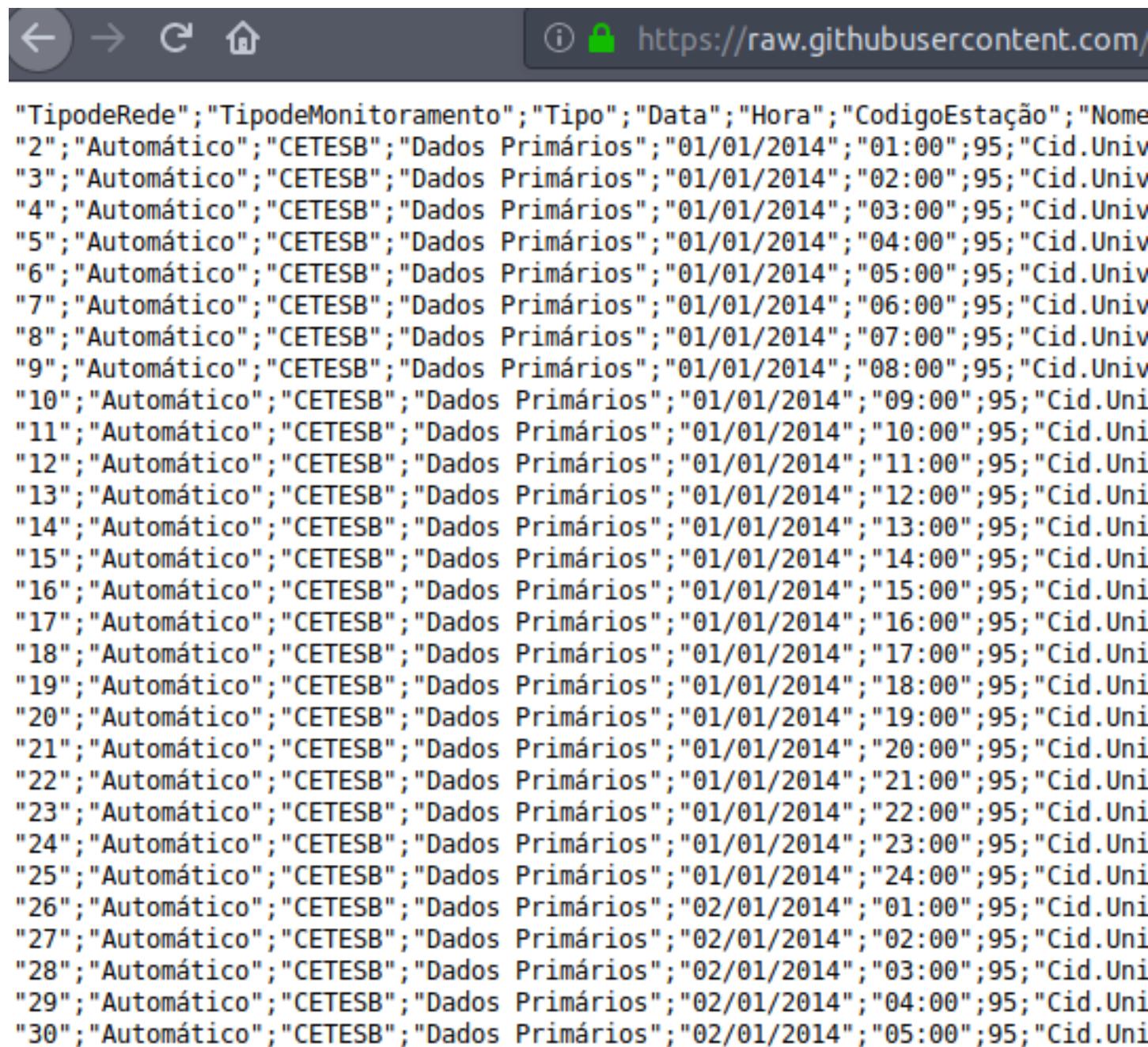
Se tu receber um banco de dados tipo .txt e quer abrir no R... ABRE ELE COM BLOCO DE NOTAS PRIMEIRO!!!

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 ler o 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 Exportando texto com `write.table`

Exportar é bem facil, mas se sabemos os argumentos das funções, vai ser mais eficiente ainda. Vamos `write.table`

```
args(write.table)
```

```
## function (x, file = "", append = FALSE, quote = TRUE, sep = " ",
##      eol = "\n", na = "NA", dec = ".", row.names = TRUE, col.names = TRUE,
##      qmethod = c("escape", "double"), fileEncoding = "")
## NULL
```

Se temos um data-frame com colunas de classe character, `quote = TRUE` quer dizer que o arquivo de texto resultante vai ter aspas nas colunas de character.

`sep` é como vão ser separadas as colunas. Se tu quer abrir o arquivo com Excel, poderia separar com “,”, “;”, “ ”, “^”... Depende como tu quer.

`eol` quer dizer *end of line*, e é para ver a forma de colocar o “end of line”

`row.names..` esta TRUE mas SEMPRE SEMPRE SEMPRE COLOCA:

`row.names = FALSE`. Se não, R vai adicionar uma coluna com os indices das linhas....

`col.names` se tu quer o nome nas colunas...

PRATICA!

4.3 Exportando objetos com save

```
args(save)
```

```
## function (... , list = character(), file = stop("'file' must be specified"),
##      ascii = FALSE, version = NULL, envir = parent.frame(), compress = isTRUE(!ascii),
##      compression_level, eval.promises = TRUE, precheck = TRUE)
## NULL
```

`save` salva o objeto com a extensão `.rda`. Para carregar de volta o objeto, tem que ser feito com a função `load`

```
args(load)
```

```
## function (file, envir = parent.frame(), verbose = FALSE)
## NULL
```

O que pode ser ruim, porque as vezes tu esqueceu o nome do objeto no ambiente de R. Por exemplo, tu salvou o arquivo

```
save(frenteFria, file = "FrenteQuente.rda")
```

logo tu carrega

```
load("FrenteQuente.rda")
```

acreditando que vai ter tua frente quente, mas o nome do objeto no ambiente de R é `frenteDria`... então, tem que ficar de olho, e como somos imperfeito, vai dar merda....

O melhor da função é que permite salvar com tipos de compressão, por exemplo `compress = "xz"`.

4.4 Exportando objetos com saveRDS

Esta é uma das minhas funções favoritas no R

```
args(saveRDS)
```

```
## function (object, file = "", ascii = FALSE, version = NULL, compress = TRUE,
##      refhook = NULL)
## NULL
```

e


```
args(readRDS)
```

```
## function (file, refhook = NULL)
## NULL
```

Tu consegue salvar o objeto R de forma serializada e compactada com o argumento `compress` mas o melhor é quando vai chamar o objeto de volta ao R. Agora tu usa o `readRDS` e coloca o nome que tu quiser.

```
saveRDS(FrenteQuente, "FrenteQuente.rds")
```

```
frenteQ <- readRDS("FrenteQuente.rds")
```

4.5 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          CodigoEstação
##          "factor"          "factor"          "integer"
##          NomeEstação          NomeParâmetro          UnidadedeMedida
##          "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 2018-05-22 22:10:09.

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 clase 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 variável 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 extrair os dias da semana do tempo, mês e dia juliano:

```
df$weekdays <- format(df$tempo, "%A")
head(df$weekdays)
```

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

```
df$mes <- format(df$tempo, "%B")
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
```

```
df$ano <- format(df$tempo, "%Y")
```

4.6 aggregate

Vamos a carregar a nossa data.frame. Primeiro uma olhada

```
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
##   CodigoeEstação      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      tempo_char
```



```
## 2          ppb          9          -      Não 01/01/2014 01:00
## 3          ppb          9          -      Sim 01/01/2014 02:00
## 4          ppb          5          -      Sim 01/01/2014 03:00
## 5          ppb          4          -      Sim 01/01/2014 04:00
## 6          ppb          5          -      Sim 01/01/2014 05:00
## 7          ppb          5          -      Sim 01/01/2014 06:00
##          tempo weekdays      mes      diajuliano ano
## 2 2014-01-01 01:00:00   quarta janeiro 16071.04 days 2014
## 3 2014-01-01 02:00:00   quarta janeiro 16071.08 days 2014
## 4 2014-01-01 03:00:00   quarta janeiro 16071.12 days 2014
## 5 2014-01-01 04:00:00   quarta janeiro 16071.17 days 2014
## 6 2014-01-01 05:00:00   quarta janeiro 16071.21 days 2014
## 7 2014-01-01 06:00:00   quarta janeiro 16071.25 days 2014
```

Poderíamos calcular a media horaria por dia da semana. Então:

```
dff <- aggregate(df$MediaHoraria, by = list(df$weekdays), sum, na.rm = T)
dff
```

```
##   Group.1      x
## 1 domingo 20327
## 2 quarta 40180
## 3 quinta 41199
## 4 sábado 32298
## 5 segunda 34057
## 6 sexta 42558
## 7 terça 37904
```

```
names(dff) <- c("dias", "MediaHoraria")
```

```
dff$sd <- aggregate(df$MediaHoraria,
                    by = list(df$weekdays),
                    sum, na.rm = T)$x
dff
```

```
##      dias MediaHoraria      sd
## 1 domingo      20327 20327
## 2 quarta      40180 40180
## 3 quinta      41199 41199
## 4 sábado      32298 32298
## 5 segunda      34057 34057
## 6 sexta      42558 42558
## 7 terça      37904 37904
```

4.7 subset

Como poderíamos escolher só o mes de janeiro??

```
#[      LINHAS      , COLUNAS  ]
head(df[df$mes == "janeiro", ]) #TODAS AS COLUNAS
```

```
##   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      tempo_char
## 2          ppb          9          -   Não 01/01/2014 01:00
## 3          ppb          9          -   Sim 01/01/2014 02:00
## 4          ppb          5          -   Sim 01/01/2014 03:00
## 5          ppb          4          -   Sim 01/01/2014 04:00
## 6          ppb          5          -   Sim 01/01/2014 05:00
## 7          ppb          5          -   Sim 01/01/2014 06:00
##           tempo weekdays      mes      diajuliano  ano
## 2 2014-01-01 01:00:00   quarta janeiro 16071.04 days 2014
## 3 2014-01-01 02:00:00   quarta janeiro 16071.08 days 2014
## 4 2014-01-01 03:00:00   quarta janeiro 16071.12 days 2014
## 5 2014-01-01 04:00:00   quarta janeiro 16071.17 days 2014
## 6 2014-01-01 05:00:00   quarta janeiro 16071.21 days 2014
## 7 2014-01-01 06:00:00   quarta janeiro 16071.25 days 2014
```

Mes janeiro pero solo o valor mediahoraria, que retorna um vetor numerico

```
names(df)
```

```
## [1] "TipodeRede"          "TipodeMonitoramento" "Tipo"
## [4] "Data"              "Hora"                "CodigoEstação"
## [7] "NomeEstação"       "NomeParâmetro"       "UnidadeMedida"
## [10] "MediaHoraria"      "MediaMovel"          "Valido"
## [13] "tempo_char"        "tempo"               "weekdays"
## [16] "mes"               "diajuliano"          "ano"
```

```
head(df[df$mes == "janeiro", 10])
```

```
## [1] 9 9 5 4 5 5
```

```
head(df[df$mes == "janeiro", "MediaHoraria"])
```

```
## [1] 9 9 5 4 5 5
```

```
class(df[df$mes == "janeiro", "MediaHoraria"])
```

```
## [1] "integer"
```

Mas vamos salvar o nosso “df”

```
saveRDS(df, "df.rds")
```

4.8 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/ibarraespinosa/cursor/master/dados/NO2_2016-01-10.nc")))

##      user      system elapsed 
##    0.107      0.008      1.275 

library(data.table)
df2 <- print(system.time(fread("https://raw.githubusercontent.com/ibarraespinosa/cursor/master/dados/NO2_2016-01-10.nc")))

##      user      system elapsed 
##    0.024      0.000      0.081
```

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.

4.9 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 compostos por dados, atributos e metadados.

O pacote `ncdf4` pode ser usado para acessar a essa biblioteca, os comandos abaixo instalam e carregam esse pacote:

```
#install.packages("ncdf4") # instala o pacote
library("ncdf4")           # carrega o pacote
nc_version()               # que retorna a versão da biblioteca
```

```
## [1] "ncdf4_1.16_20170401"
```

Um exmplo de NetCDF:

```
download.file("https://github.com/ibarraespinosa/cursor/raw/master/dados/met_em.d03.2016-01-10.nc", destfile="met_em.d03.2016-01-10.nc")

wrf <- ncdf4::nc_open("~/met_em.d03.2016-01-10.nc")
```

O objeto `wrf` contém algumas informações sobre o conteúdo do arquivo, com um `print(wrf)` ou simplesmente `wrf` visualizamos o conteúdo do arquivo:

```
class(wrf)

## [1] "ncdf4"

wrf

## File ~/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 HGTROP[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_SM000010: 1
##      FLAG_SM010040: 1
##      FLAG_SM040100: 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
```

que mostra o nome do arquivo (e versão da biblioteca usada para criar), número 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"        "HGTTROP"   "TTROP"       "PTROPNN" "PTROP"
## [11] "VTROP"      "UTROP"     "HGTMAXW"     "TMAXW"   "PMAXWNN"
## [16] "PMAXW"      "VMAXW"     "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"
```

```
TEMP <- ncdf4::ncvar_get(wrf, "TT") # escolho você picachu
class(TEMP)
```

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

Como o NetCDF é organizado para guardar matrizes (arrays), só sabemos que a variável ST é um array

```
ncatt_get(wrf, "TT") # ou ncatt_get(wrf, "TT", verbose = T)
```

```
## $FieldType
## [1] 104
##
## $MemoryOrder
## [1] "XYZ"
##
## $units
## [1] "K"
##
## $description
## [1] "Temperature"
##
## $stagger
## [1] "M"
##
## $sr_x
## [1] 1
##
## $sr_y
## [1] 1
```

```
dim(TEMP)
```

```
## [1] 51 51 27
```

praticamente a mesma informação do print anterior:

```
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
```

como temos apenas 1 tempo essa dimensão é desconsiderada para simplificar.

A latitude de cada ponto de grade, assim como longitude níveis e tempo podem ser extraídas:

```
lat <- ncvar_get(wrf, "XLAT_M")
lon <- ncvar_get(wrf, "XLONG_M")
time <- ncvar_get(wrf, "Times")
```

O metadado de Longitude:

```
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
```

Latitude:

```
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
```

e a altura:

```
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
```

Da mesma forma com que podemos acessar variáveis e atributos com `ncvar_get` e `ncatt_get`, podemos modificar estes valores com `ncvar_put` e `ncatt_put`. Outras operações como renomear (`ncvar_rename`) e trocar o valor de missval (`ncvar_change_missval`) também estão disponíveis.

DICA: `ncatt_get` e `ncatt_put` acessam e alteram os atributos de variáveis e também atributos globais do NetCDF usando o argumento `varid=0`.

Para salvar as alterações e/ou liberar o acesso ao arquivo use a função `nc_close` (ou a função `nc_sync` que sincroniza o NetCDF mas não fecha a conexão com o arquivo).

```
nc_close(wrf) # ou nc_sync(wrf)
```

Novas dimensões e também novas variáveis podem ser criadas com `ncvar_def` e `ncvar_add` em um arquivo aberto com permissão de leitura, como por exemplo:

```
wrf      <- nc_open("~/met_em.d03.2016-01-10.nc", write=TRUE)
extrema <- ncvar_def(name = "Tex",
                    units = "K",
                    dim = list(wrf$dim$west_east,
                              wrf$dim$south_north,
                              wrf$dim$Time),
                    missval = -999,
                    longname = "temperatura extrema")
ncvar_add(wrf, extrema)
names(wrf$var)
nc_close(wrf)
```

Se esse arquivo for aberto novamente vai conter 93 variáveis junto com a variável `Tex` da forma que definimos, caso queria os mesmos atributos que as demais é só usar a função `ncatt_get` na variável.

```
wrf <- nc_open("~/met_em.d03.2016-01-10.nc",write=T)
print(wrf)

## File ~/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_SS0[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

```

O pacote possui ainda funções mais específicas para a criação de arquivos em NetCDF como `nc_create`, funções que definem dimensões como `ncdim_def` e funções para colocar e tirar o arquivo de modo de definição `nc_redef` e `nc_enddef`.

DICA: o NetCDF no R funciona de forma parecida com ouma lista ou data frame, podemos “ver” ou selecionar suas sub-partes (sub-sub-partes...) com “\$” e TAB.

Plotando

Exemplo: Dados de qualidade do ar

```

##          TipodeRede      TipodeMonitoramento      Tipo
## Automático:8581      CETESB:8581      Dados Primários:8581
##
##
##
##
##
##
##
##          Data          Hora          CodigoEstação
## 01/01/2014:  24      16:00  : 363      Min.      :95
## 01/01/2015:  24      12:00  : 361      1st Qu.:95
## 01/02/2014:  24      14:00  : 361      Median   :95
## 01/04/2014:  24      18:00  : 361      Mean     :95
## 01/05/2014:  24      13:00  : 360      3rd Qu.:95
## 01/06/2014:  24      17:00  : 360      Max.     :95
## (Other)      :8437      (Other):6415
##          NomeEstação          NomeParâmetro
## Cid.Universitária-USP-Ipen:8581      NOx (Óxidos de Nitrogênio):8581
##
##
##
##
##
##
##
##  UnidadedeMedida  MediaHoraria  MediaMovel Valido      tempo_char
## ppb:8581          Min.      : 0.00  -:8581      Não: 907      Length:8581
##                  1st Qu.:  9.00                Sim:7674      Class :character
##                  Median   : 18.00                Mode  :character
##                  Mean     : 29.87
##                  3rd Qu.: 34.00
##                  Max.     :306.00

```

```
##          NA's      :260
##      tempo          weekdays          mes
##  Min.   :2014-01-01 01:00:00 Length:8581 Length:8581
## 1st Qu.:2014-04-05 14:00:00 Class :character Class :character
## Median :2014-07-05 23:00:00 Mode  :character Mode  :character
## Mean   :2014-07-04 20:22:55
## 3rd Qu.:2014-10-03 10:00:00
## Max.   :2015-01-02 00:00:00
##
##      diajuliano      ano
## Length:8581      Length:8581
## Class :difftime   Class :character
## Mode  :numeric    Mode  :character
##
##
##
```

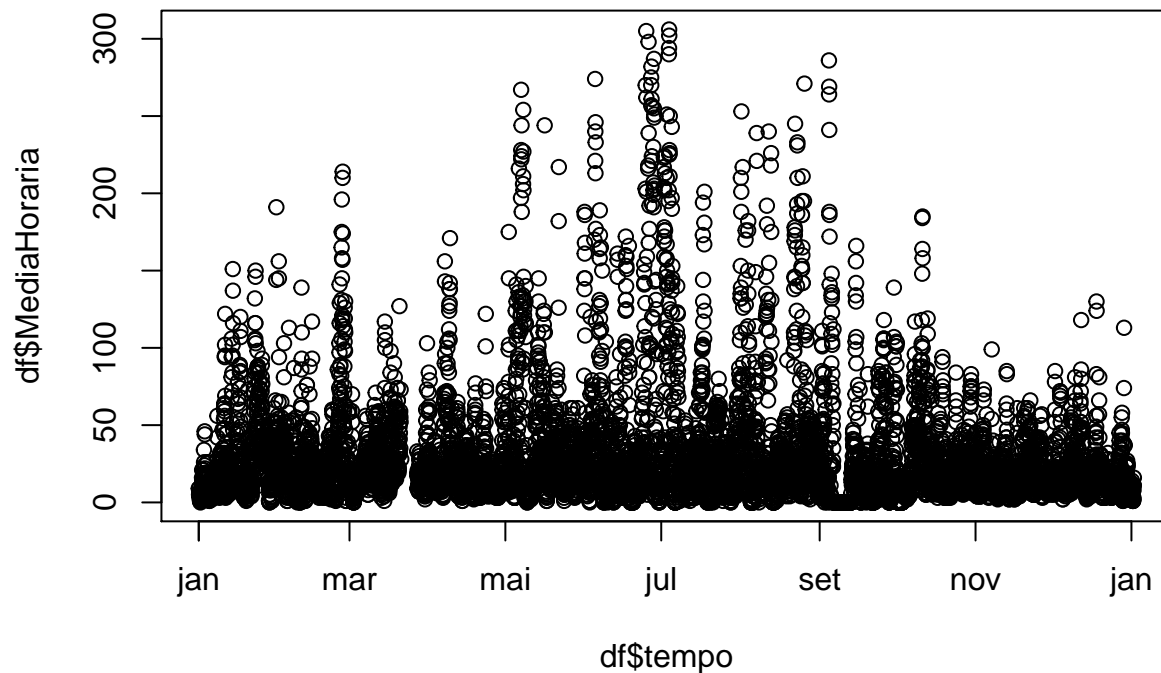
A função `plot` precisa dos seguintes argumentos:

```
args(plot)
```

```
## function (x, y, ...)
## NULL
```

Então, a forma mais fácil de plotar uma variável em função do tempo é:

```
plot(x = df$tempo, y = df$MediaHoraria)
```



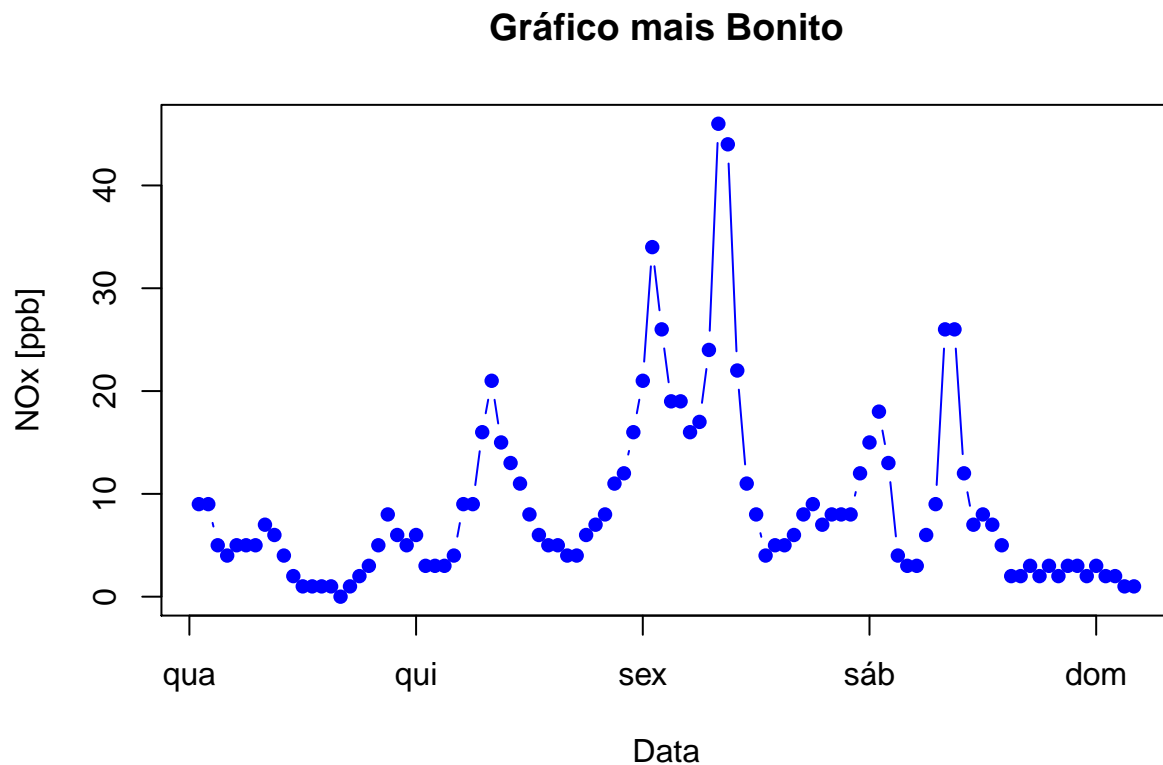
Feio, né?

Tentando deixar mais bonito...

```
plot(x = df$tempo[1:100], y = df$MediaHoraria[1:100], #-- Selecionando uma parte do df!
     pch = 16, #-- Forma do ponto (círculo preenchido)
     type = "b", #-- Tipo de gráfico ("b" = both, ponto e linha))
```



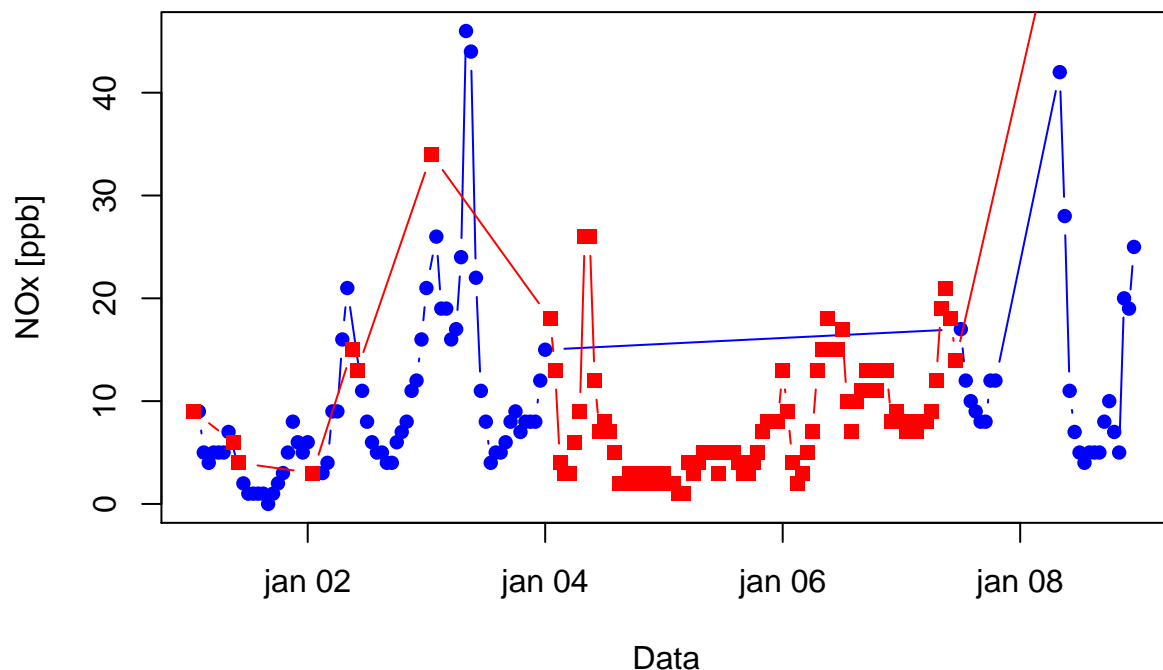
```
col = "blue", #-- Cor do elemento (definido pelo type)
xlab = "Data", ylab = "NOx [ppb]", #-- Nome dos eixos x e y
main = "Gráfico mais Bonito") #-- Título do gráfico
```



Colocando **DOIS** elementos no mesmo gráfico:

```
df_parcial <- df[1:180,] #-- Selecionando uma parte do df!
plot(x = df_parcial$tempo[df_parcial$Valido == "Sim"],
     y = df_parcial$MediaHoraria[df_parcial$Valido == "Sim"],
     pch = 16, type = "b", col = "blue",
     xlab = "Data", ylab = "NOx [ppb]",
     main = "Dados Válidos e Inválidos")
lines(x = df_parcial$tempo[df$Valido == "Não"],
      y = df_parcial$MediaHoraria[df$Valido == "Não"],
      pch = 15, type = "b", col = "red")
```

Dados Válidos e Inválidos



Desafio: Coloque uma legenda na figura especificando que os dados válidos estão em azul e os inválidos em vermelho

A função `plot` cumpre bem o papel de gerar um gráfico simples, e até permite algumas customizações, mas ela exige cada vez mais linhas de código e argumentos dentro das funções para deixar o gráfico “mais bonito” - ao cumprir o desafio, você irá perceber como uma coisa “simples” como colocar uma legenda pode exigir muito mais do que parece!

5.2 ggplot (ggplot2)

A função `ggplot` funciona de um jeito um pouco diferente. Veja a figura abaixo:

Em vez de uma única função, o gráfico é formado por camadas, sendo que cada camada é um elemento (`geom_...` ou `stat_...`) ou configuração (`scale_...`, `coord_...`, `theme` ou `theme_...`, `guides`, `labs`, etc). Consulte a maioria das opções disponíveis em Data Visualization Cheatsheet.

Que tal refazermos os gráficos da seção anterior?

```
##-- Não esqueça de carregar o pacote!
library(ggplot2)
```

```
ggplot(df, aes(x = tempo, y = MediaHoraria)) +
  geom_point(pch = 1)
```

```
## Warning: Removed 260 rows containing missing values (geom_point).
```

Complete the template below to build a graph.

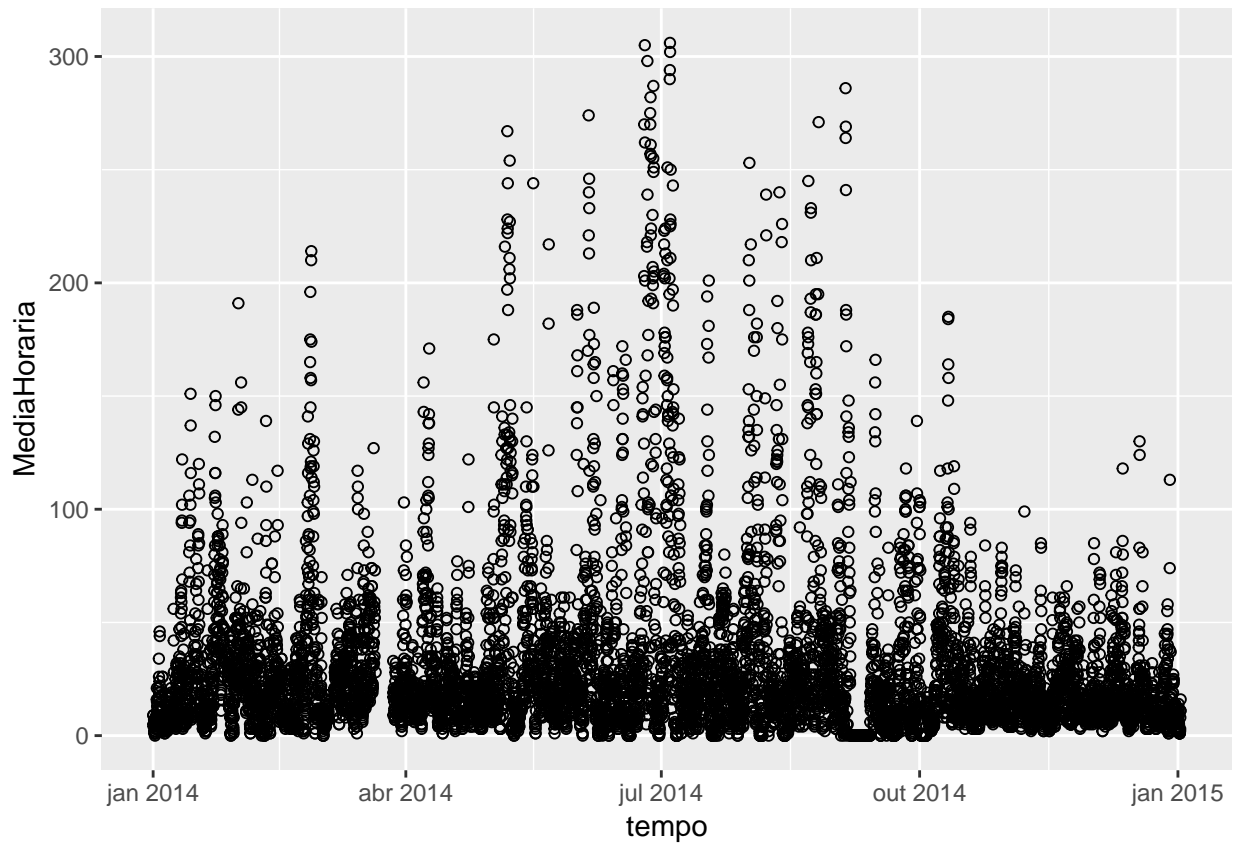
```
ggplot (data = <DATA>) +  
  <GEOM_FUNCTION> (mapping = aes(<MAPPINGS>),  
  stat = <STAT> , position = <POSITION> ) +  
  <COORDINATE_FUNCTION> +  
  <FACET_FUNCTION> +  
  <SCALE_FUNCTION> +  
  <THEME_FUNCTION>
```

required

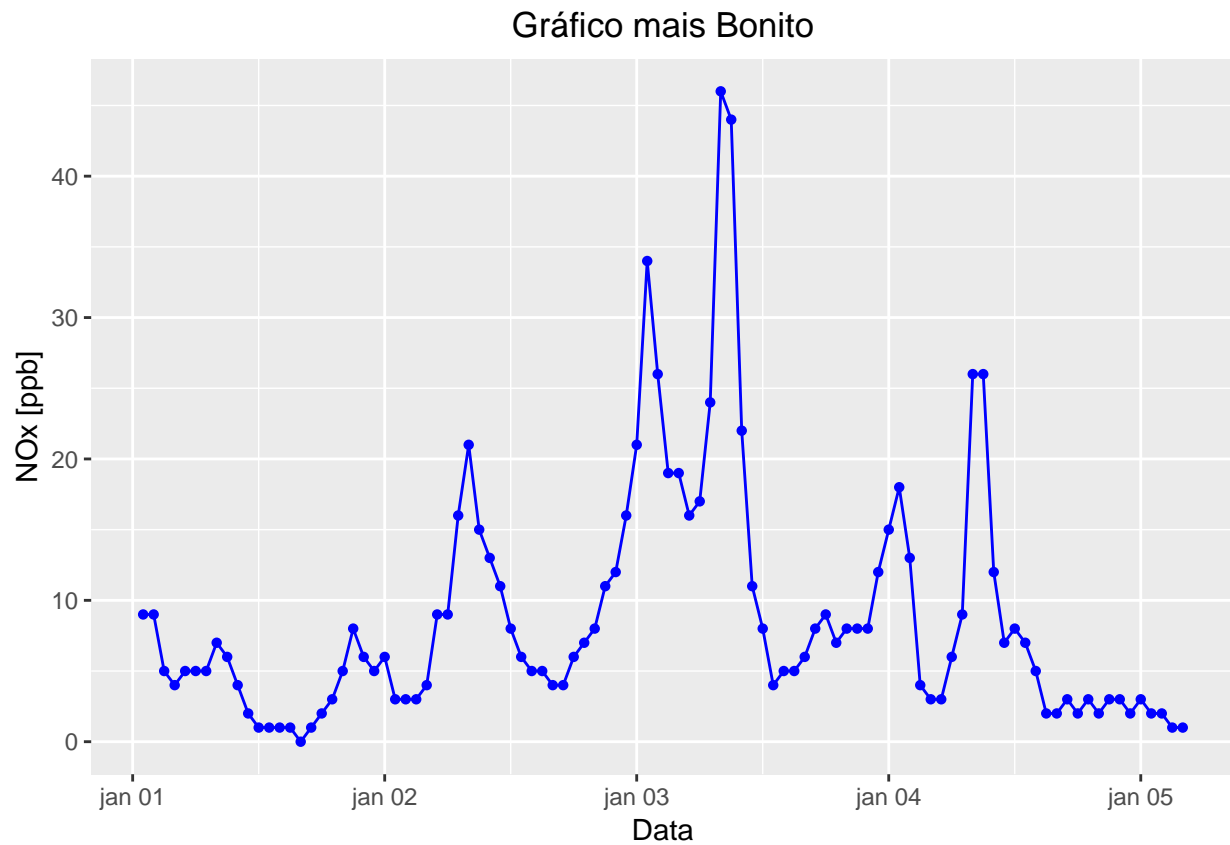
Not required, sensible defaults supplied

ggplot(data = mpg, **aes**(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

Figure 5.1: Fonte: <https://github.com/rstudio/cheatsheets/raw/master/data-visualization-2.1.pdf>

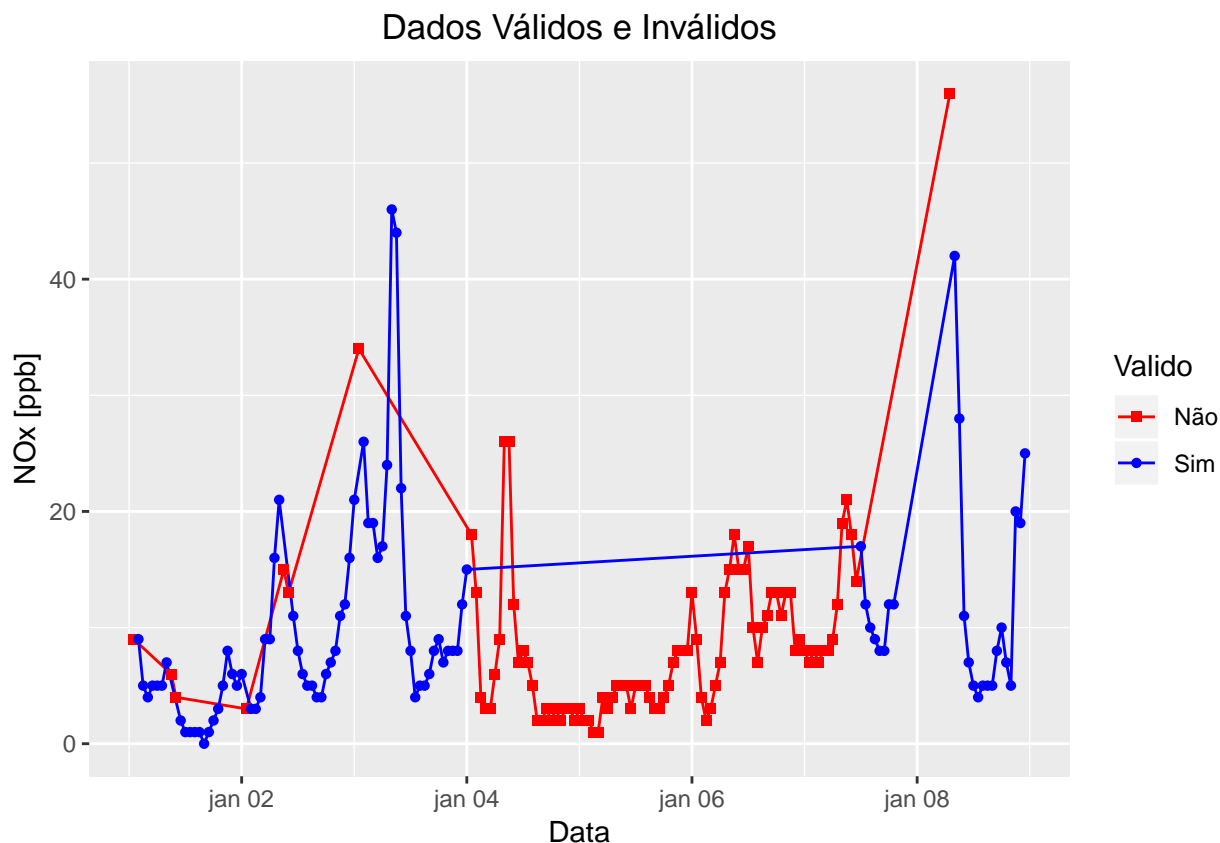


```
ggplot(df[1:100,], aes(x = tempo, y = MediaHoraria)) +
  geom_line(color = "blue") + #-- Linhas...
  geom_point(color = "blue", pch = 16) + #-- ... com pontos
  labs(title = "Gráfico mais Bonito", x = "Data", y = "NOx [ppb]") + #-- Títulos
  theme(plot.title = element_text(hjust = 0.5)) #-- Centralizando o título
```



Agora o mais interessante:

```
ggplot(df[1:180,], aes(x = tempo, y = MediaHoraria)) +
  geom_line(aes(color = Valido)) +
  geom_point(aes(color = Valido, shape = Valido)) +
  labs(title = "Dados Válidos e Inválidos", x = "Data", y = "NOx [ppb]") +
  scale_color_manual(values = c("red", "blue")) + #-- Definindo as cores manualmente
  scale_shape_manual(values = c(15, 16)) + #-- Definindo as formas manualmente
  theme(plot.title = element_text(hjust = 0.5))
```



Pergunta: Qual a principal diferença entre o código acima e o código usando `plot`?

A função `ggplot` plota apenas data frames, pois ela mapeia as variáveis por nomes de colunas. Assim, é preciso converter matrizes ou arrays em data frames.

Uma vantagem de trabalharmos com data frames, como já vimos antes, é poder manipular esses dados de muitas formas possíveis antes de plotá-los.

Continuação do Exemplo: Extraindo algumas informações sobre os dados

Vamos analisar o ano de 2014:

- Em média, como o NOx varia ao longo do dia?
 - E para cada dia da semana?
 - E para cada mês?

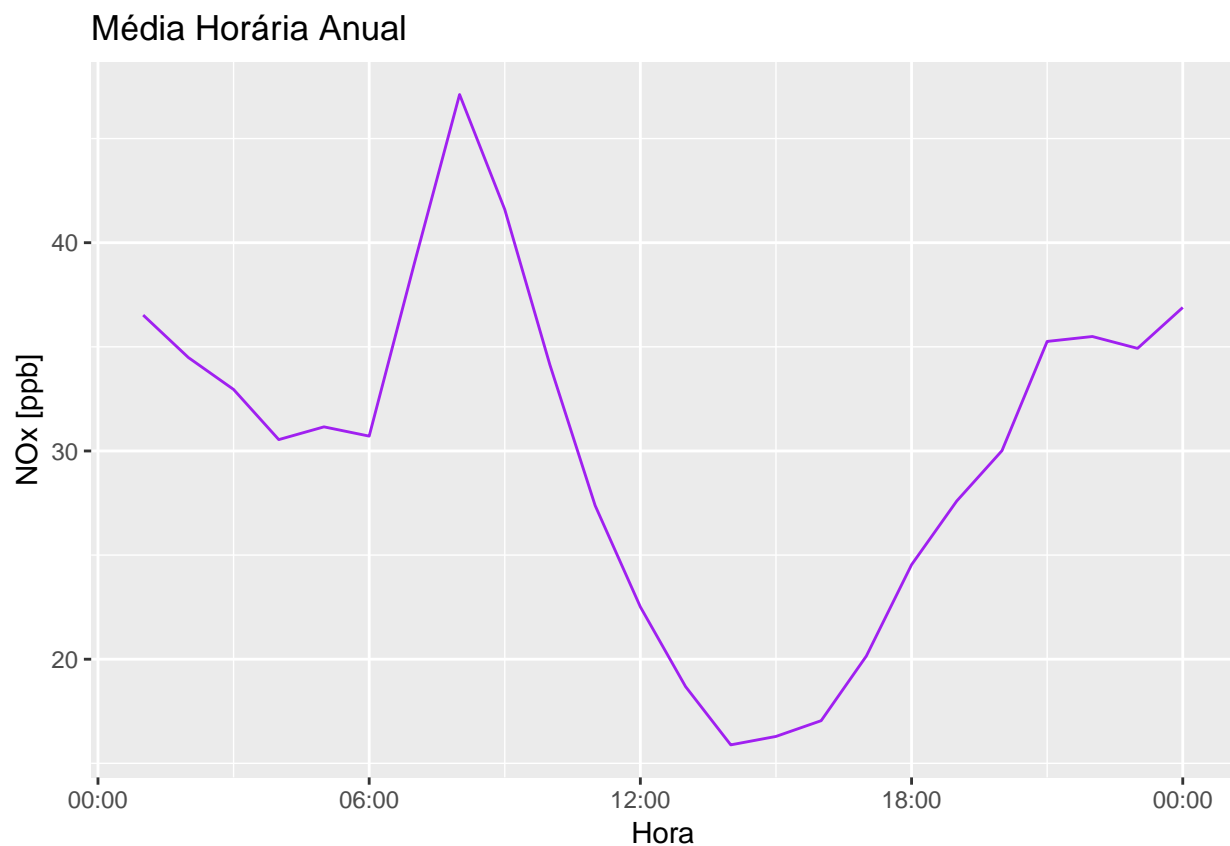
Usando algumas funções dentro do pacote **tidyverse**, como o *pipe* (`%>%`):

```
library(tidyverse)
library(scales)

df_2014 <- filter(df, ano == "2014")
df_2014_hour <- df_2014 %>% #-- A partir do data frame df_2014...
  group_by(Hora) %>% #-- ... agrupe os dados pela coluna hora...
  summarise(Media = mean(MediaHoraria, na.rm = T)) %>% #-- ... E calcule as médias, salvando em uma col...
  mutate(Hora = as.POSIXct(strptime(Hora, "%H:%M"))) %>% #-- Transformando em data
  ungroup() #-- Desagrupando

ggplot(df_2014_hour) +
  scale_x_datetime(date_labels = "%H:%M") + #-- Formato de data que aparecerá no eixo x
```

```
geom_line(aes(x = Hora, y = Media, group = 1), color = "purple") +
labs(title = "Média Horária Anual", y = "NOx [ppb]")
```

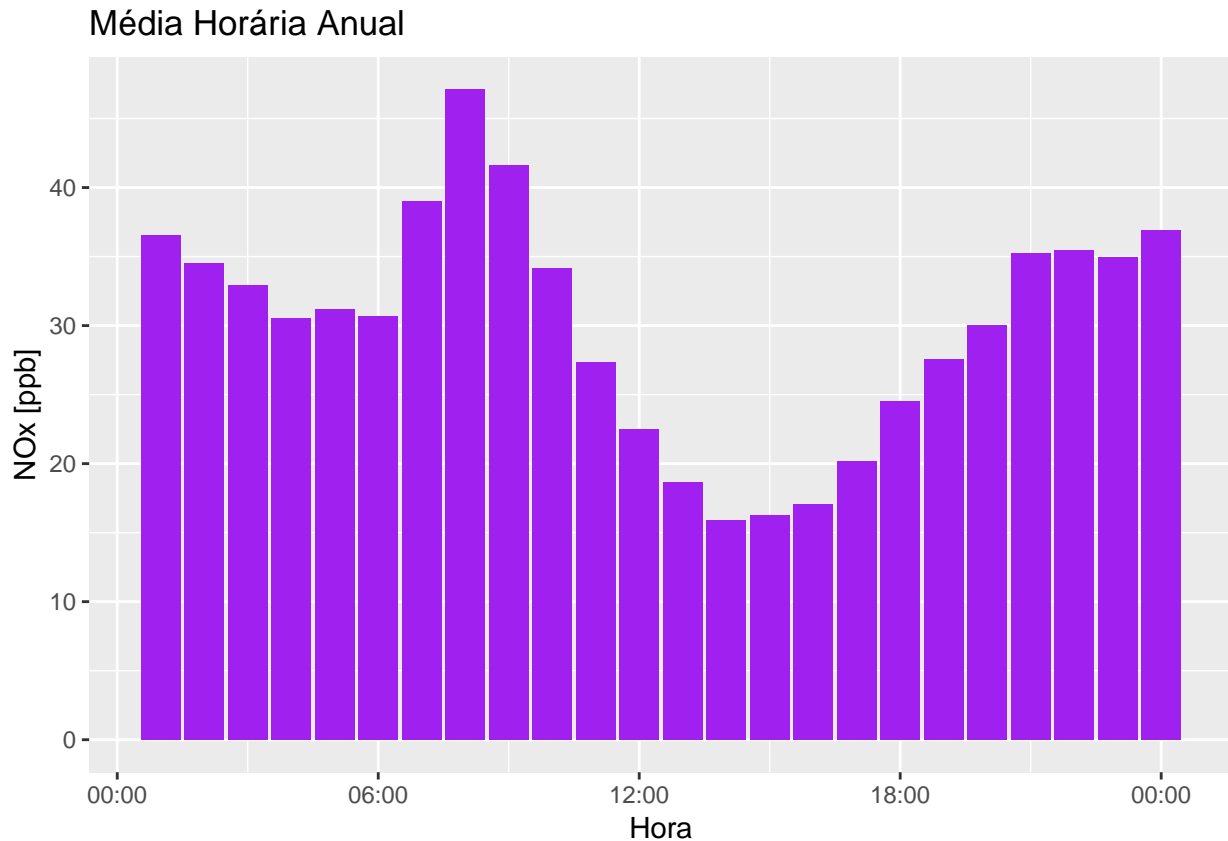


```
library(tidyverse)

df_2014 <- filter(df, ano == "2014")
df_2014_hourly <- df_2014 %>% #-- A partir do data frame df_2014...
  group_by(Hora) %>% #-- ... agrupe os dados pela coluna Hora...
  summarise(Media = mean(MediaHoraria, na.rm = T)) %>% #-- ... E calcule as médias,
  #-- salvando em uma coluna nova

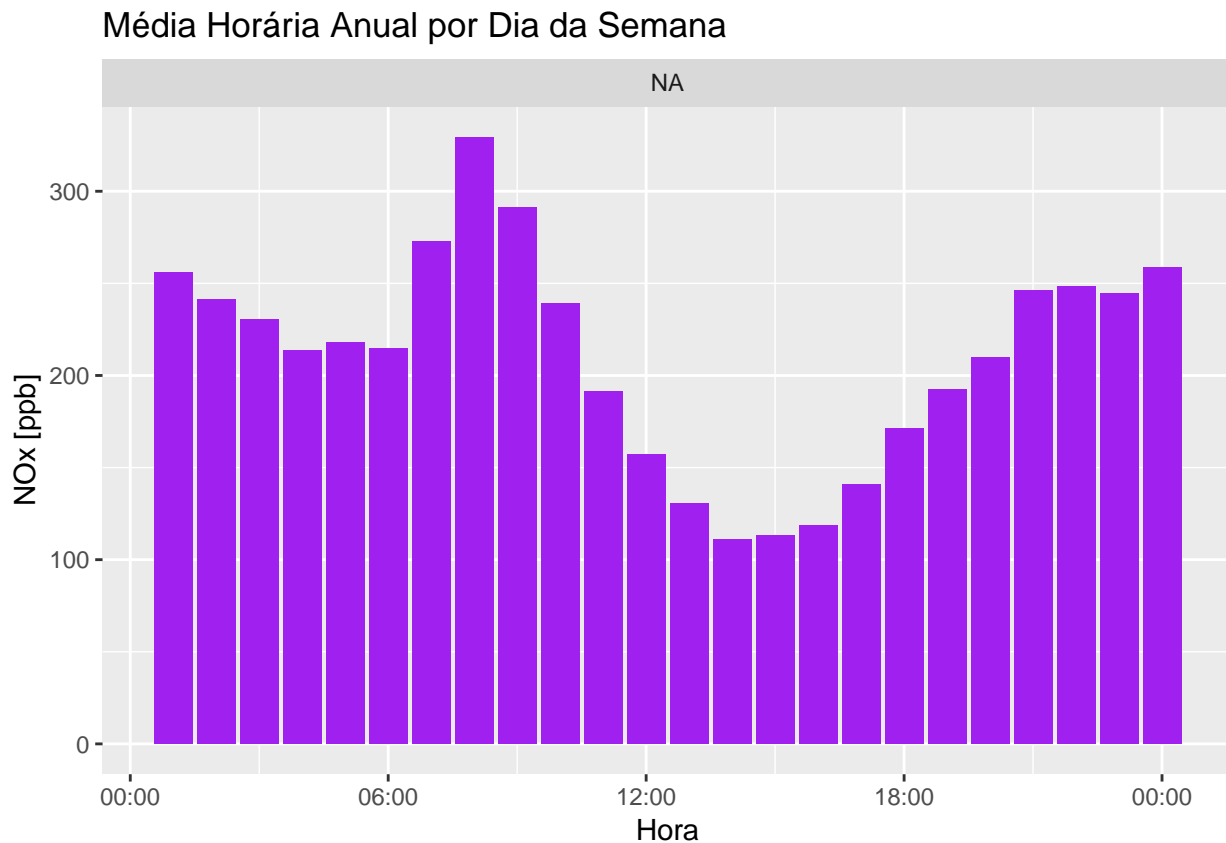
  ungroup() %>% #-- Desagrupando
  mutate(Hora = as.POSIXct(strptime(Hora, "%H:%M"))) #-- Transformando em data

ggplot(df_2014_hourly) +
  scale_x_datetime(date_labels = "%H:%M") + #-- Formato de data que aparecerá no eixo x
  geom_col(aes(x = Hora, y = Media), fill = "purple") +
  labs(title = "Média Horária Anual", y = "NOx [ppb]")
```



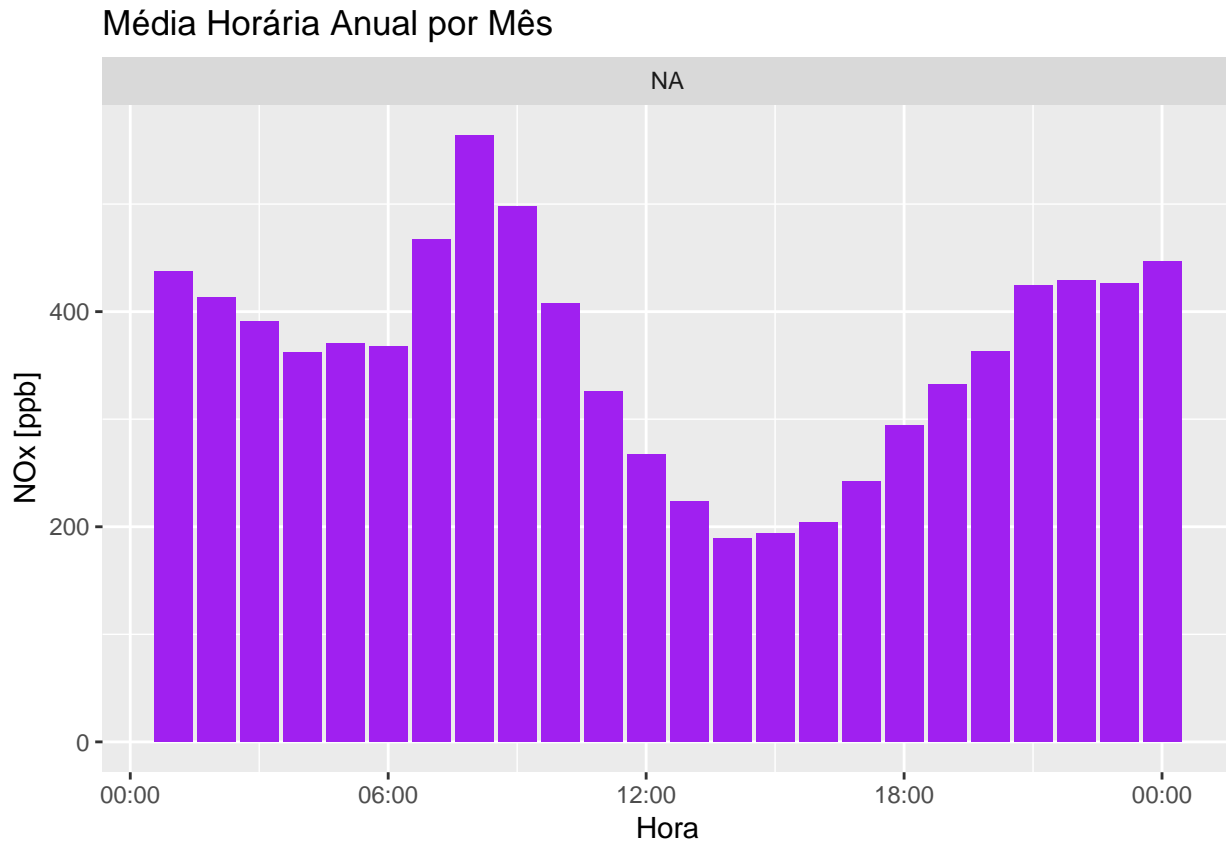
```
df_2014_weekly <- df_2014 %>% #--
  group_by(Hora, weekdays) %>% #-- Agrupando os dados pelas colunas Hora e weekdays
  summarise(Media = mean(MediaHoraria, na.rm = T)) %>%
  ungroup() %>%
  mutate(Hora = as.POSIXct(strptime(Hora, "%H:%M"))) %>%
  mutate(weekdays = factor(weekdays, levels = c("Monday", "Tuesday", "Wednesday",
                                                    "Thursday", "Friday", "Saturday",
                                                    "Sunday"))) #-- Ordenando os dias da semana

ggplot(df_2014_weekly) +
  scale_x_datetime(date_labels = "%H:%M") +
  geom_col(aes(x = Hora, y = Media), fill = "purple") +
  labs(title = "Média Horária Anual por Dia da Semana", y = "NOx [ppb]") +
  facet_wrap(~ weekdays) #-- Criando painéis em função do dia da semana
```

```
df_2014_monthly <- df_2014 %>%
  group_by(Hora, mes) %>% #-- Agrupando os dados pelas colunas Hora e mes
  summarise(Media = mean(MediaHoraria, na.rm = T)) %>%
  ungroup() %>%
  mutate(Hora = as.POSIXct(strptime(Hora, "%H:%M"))) %>%
  mutate(mes = factor(mes, levels = c("January", "February", "March", "April", "May",
    "June", "July", "August", "September", "October",
    "November", "December"))) #-- Ordenando os meses

ggplot(df_2014_monthly) +
  scale_x_datetime(date_labels = "%H:%M") +
  geom_col(aes(x = Hora, y = Media), fill = "purple") +
  labs(title = "Média Horária Anual por Mês", y = "NOx [ppb]") +
  facet_wrap(~ mes) #-- Criando painéis em função do mês
```



Exercício: Em média, como os dados válidos de NOx variam mensalmente ao longo do ano de 2014? Faça um gráfico.

Desafio: Ainda é possível melhorar os gráficos acima! Pesquise como:

- Diminuir a quantidade de horários no eixo x
- Separar por dias da semana e meses a partir da coluna “tempo”, não precisando usar as colunas de caracteres e consequentemente ordená-las manualmente

5.2.1 Explorando outras escalas de cores e temas

Pacotes **veinreport** e **cptcity**

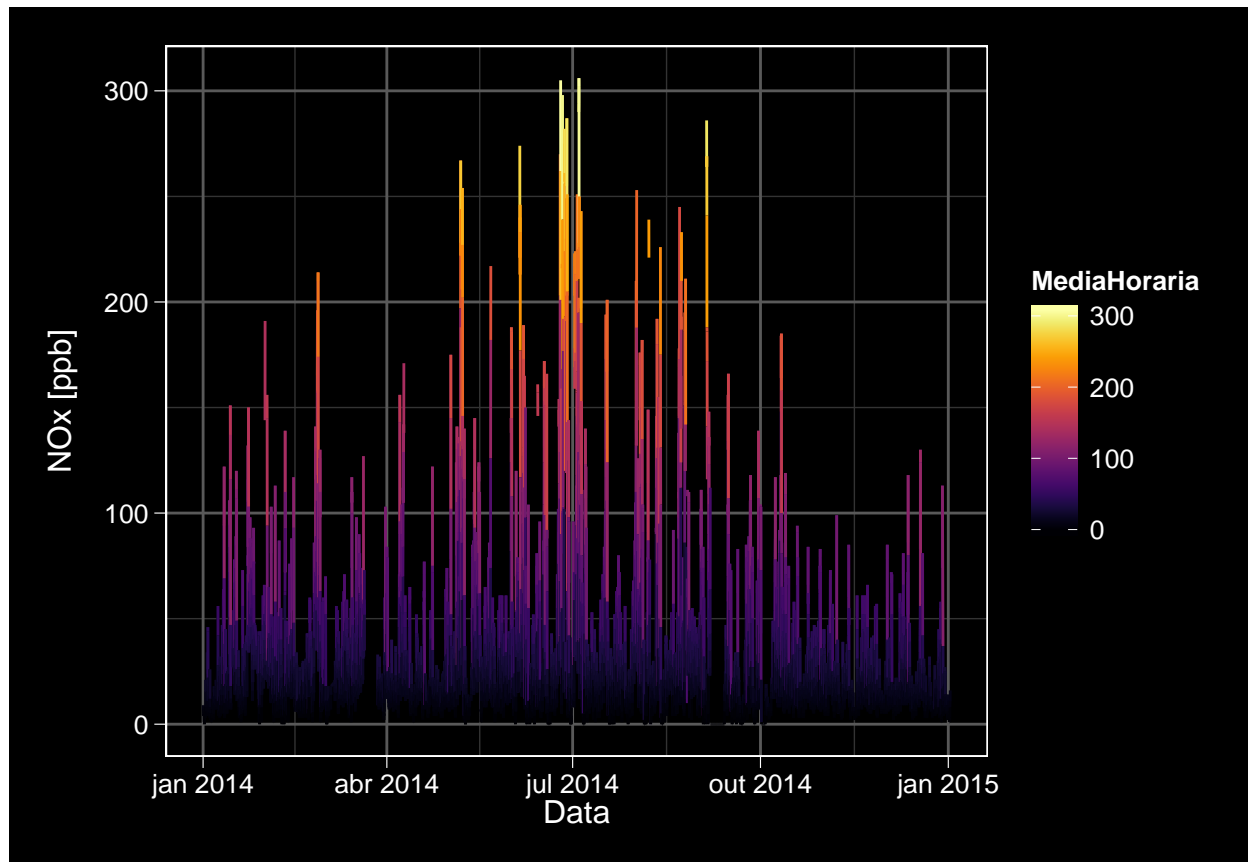
```
devtools::install_github("atmoschem/veinreport")
```

```
library(veinreport)
library(cptcity)
```

Refazendo alguns gráficos:

```
ggplot(df, aes(x = tempo, y = MediaHoraria)) +
  geom_line(aes(color = MediaHoraria)) +
  labs(x = "Data", y = "NOx [ppb]") +
  scale_color_gradientn(colours = cpt()) + #-- Definindo as cores com uma escala gradiente
  theme_black()
```

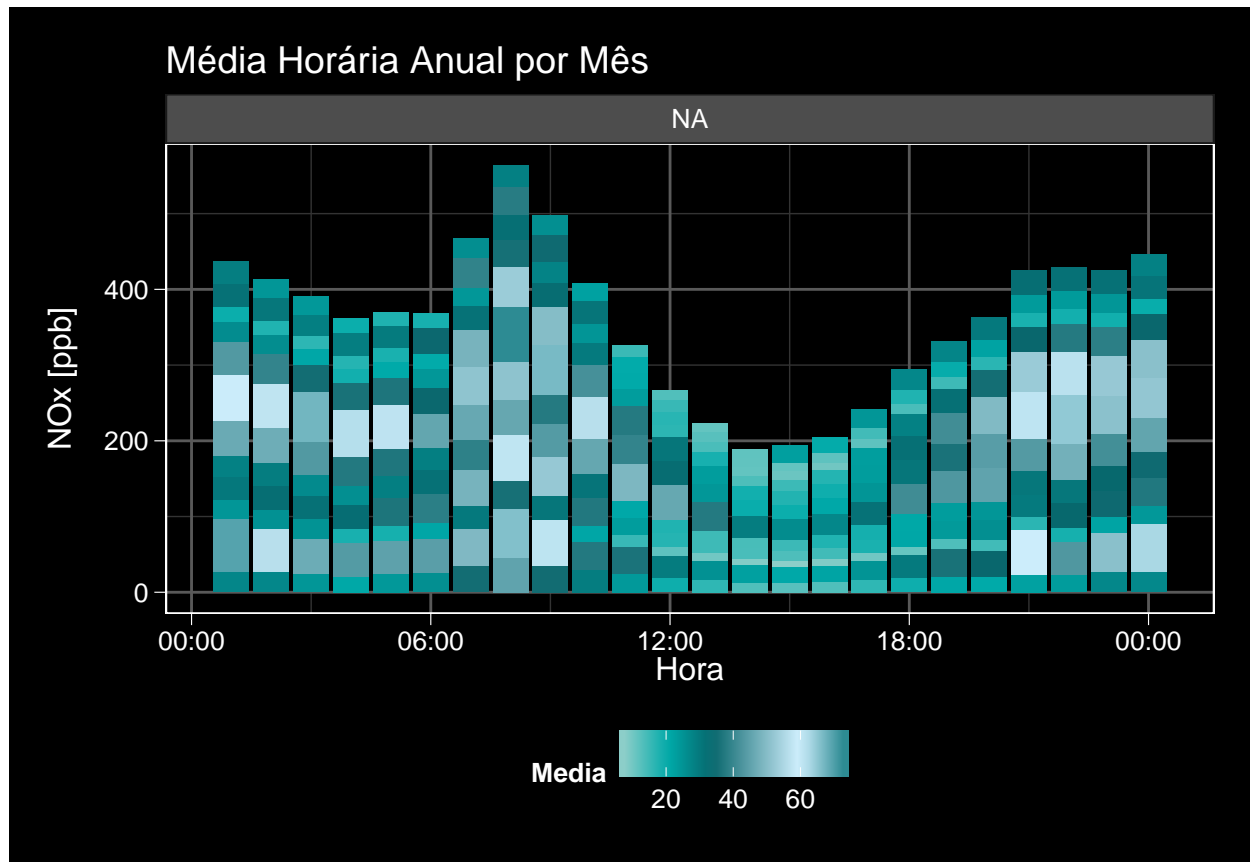
```
## Warning: Removed 1 rows containing missing values (geom_path).
```



Experimentando escalas de cores com a função lucky:

```
ggplot(df_2014_monthly) +
  scale_x_datetime(date_labels = "%H:%M") +
  geom_col(aes(x = Hora, y = Media, fill = Media)) +
  labs(title = "Média Horária Anual por Mês", y = "NOx [ppb]") +
  scale_fill_gradientn(colors = lucky()) + #-- Definindo as cores com uma escala gradiente aleatória
  theme_black() +
  theme(legend.position = "bottom", legend.direction = "horizontal") + #-- Colocando a legenda na parte
  facet_wrap(~ mes) #-- Criando painéis em função do mês
```

```
## Colour gradient: es_ocean_breeze_es_ocean_breeze_099, number: 1601
```



Este é só o começo! Veja aqui um pouco mais das muitas aplicações do `ggplot`.

Chapter 6

Estruturas de control

6.1 if-else

6.2 for

6.3 while

6.4 repeat

6.5 lapply

6.6 sapply

6.7 split

6.8 tapply

6.9 apply

6.10 mapply

Chapter 7

De scripts a funções e de funções a pacotes

Coming soon

Chapter 8

Geo Spatial: raster, sf e stars

Coming soon