Introdução à Linguagem de Programação em R para tratamento de dados de poluição do ar

openair e R do dia dia

Mario Gavidia-Calderón, Rafaela Squizzato, Thiago Nogueira

20/02/2025

Universidade de São Paulo

Dúvidas aula passada

Média móvel

Combinar data frames

Transformar tipos de objetos

Boas praticas do R

Mais recursos.

e agora?

- Nesta aula vamos falar sobre situações que acontecem quando trabalhamos com dados de qualidade do ar.
- Também falaremos sobre boas praticas de R.
- E resolver as suas dúvidas.

Dúvidas aula passada

Média das 2 pm até as 2 pm do dia seguinte

```
library(openair)

pin <- readRDS("../../data/pin_example_23_24.rds")
pin_day <- timeAverage(pin, avg.time = "day")
```

Média das 2 pm até as 2 pm do dia seguinte

```
head(pin_day)
```

```
## # A tibble: 6 x 5
##   date                   wd  pm25    o3    ws
##   <dttm>              <dbl> <dbl> <dbl> <dbl>
## 1 2023-01-01 00:00:00 169.      NaN  53.0  1.44
## 2 2023-01-02 00:00:00   0.656   NaN  44.4  1.71
## 3 2023-01-03 00:00:00   2.90    NaN  27.3  1.85
## 4 2023-01-04 00:00:00 356.      NaN  17.5  1.7
## 5 2023-01-05 00:00:00 158.      NaN  26.8  1.69
## 6 2023-01-06 00:00:00 146.      NaN  22.9  2.53
```

Média das 2 pm até as 2 pm do dia seguinte

- A estrategia é modificar o data frame `pin` para a primeira linha seja às 14:00, depois vamos *enganar* ao R para que ache que são às 00:00

```
library(openair)

# Desde a linha 15 ate última linha - 11
pin_2pm <- pin[15:(nrow(pin) - 11),  ]
# Trocamos a coluna `date`
names(pin_2pm)[1] <- "date_original"
```

Média das 2 pm até as 2 pm do dia seguinte

```r
# Criamos uma nova coluna de data usando seq.
pin_2pm$date <- seq(as.POSIXct("2023-01-01 00:00"),
                    by = "hour",
                    length.out = nrow(pin_2pm))
head(pin_2pm)
```

```
##          date_original       aqs  wd pm25  o3  ws                   d
## 15 2023-01-01 14:00:00 Pinheiros 184   NA  76 1.7 2023-01-01 00:00
## 16 2023-01-01 15:00:00 Pinheiros 148   NA  97 3.0 2023-01-01 01:00
## 17 2023-01-01 16:00:00 Pinheiros 158   NA 119 2.3 2023-01-01 02:00
## 18 2023-01-01 17:00:00 Pinheiros 160   NA 108 2.3 2023-01-01 03:00
## 19 2023-01-01 18:00:00 Pinheiros 149   NA  89 2.7 2023-01-01 04:00
## 20 2023-01-01 19:00:00 Pinheiros 142   NA  83 2.9 2023-01-01 05:00
```

Média das 2 pm até as 2 pm do dia seguinte

```
pin_day_2pm <- timeAverage(pin_2pm, avg.time = "day",
                           vector.ws = T)
head(pin_day_2pm)
```

```
## # A tibble: 6 x 6
##   date                date_original          wd  pm25    o3    ws
##   <dttm>              <dttm>              <dbl> <dbl> <dbl> <dbl>
## 1 2023-01-01 00:00:00 2023-01-02 01:30:00 139.    NaN  52.7 0.431
## 2 2023-01-02 00:00:00 2023-01-03 01:30:00  12.4   NaN  41.6 0.907
## 3 2023-01-03 00:00:00 2023-01-04 01:30:00 358.    NaN  22.9 1.85
## 4 2023-01-04 00:00:00 2023-01-05 01:30:00 159.    NaN  21.5 1.21
## 5 2023-01-05 00:00:00 2023-01-06 01:30:00 152.    NaN  28.9 2.21
## 6 2023-01-06 00:00:00 2023-01-07 01:30:00 144.    NaN  17   2.64
```

Média das 2 pm até as 2 pm do dia seguinte

- Um jeito mais complicado (mas funciona) está neste link.

Dia da semana e final de semana

```
weekdays <- selectByDate(pin, day = "weekday")
head(weekdays)
```

```
##                    date       aqs  wd pm25 o3  ws
## 25 2023-01-02 00:00:00 Pinheiros 165   NA 41 1.3
## 26 2023-01-02 01:00:00 Pinheiros 177   NA NA 1.4
## 27 2023-01-02 02:00:00 Pinheiros 326   NA NA 0.9
## 28 2023-01-02 03:00:00 Pinheiros 326   NA 24 0.7
## 29 2023-01-02 04:00:00 Pinheiros 359   NA 25 0.8
## 30 2023-01-02 05:00:00 Pinheiros  NA   NA 13 0.0
```

Dia da semana e final de semana

```
weekends <- selectByDate(pin, day = "weekend")
head(weekends)
```

```
##                  date       aqs  wd pm25 o3  ws
## 1 2023-01-01 00:00:00 Pinheiros  NA   NA NA  NA
## 2 2023-01-01 01:00:00 Pinheiros 313   NA NA 0.9
## 3 2023-01-01 02:00:00 Pinheiros 173   NA NA 0.7
## 4 2023-01-01 03:00:00 Pinheiros  NA   NA  2 0.0
## 5 2023-01-01 04:00:00 Pinheiros  NA   NA  1 0.0
## 6 2023-01-01 05:00:00 Pinheiros  NA   NA  0 0.0
```

Lag e exposição

- Neste trabalho foi usado *Distributed Lag Nonlinear Models*, the `dlnm` package.

Média móvel

Média móvel

- O padrão qualidade do ar em São Paulo de O_3 é a média móvel de 8 horas.
- Outro padrão da **WHO** é a MDA8 (*average of daily maximun 8-hour*) e também Peak season.
- `openair` conta com a função `rollingMean()` para fazer esses padrões.

Média móvel

- Vamos calcular o padrão de qualidade do ar de O_3 para CETESB no 2021.

```
library(openair)
# Lendo o arquivo
pin <- readRDS('../../data/pin_openair_ex.rds')
pin_2021 <- selectByDate(pin, year = 2021)
# Média móvel
pin_2021 <- rollingMean(
  pin_2021,
  pollutant = 'o3',
  width = 8,
  new.name = "o3_8h",
  data.thresh = 0.75)
```

Média móvel

- Foi criada a coluna **o3_8h**.

```
##                  date       aqs  wd pm25 no no2 o3  ws o3_8h
## 1 2021-01-01 00:00:00 Pinheiros 346   NA  0   1 27 1.7    NA
## 2 2021-01-01 01:00:00 Pinheiros 331   NA  0   0 28 1.2    NA
## 3 2021-01-01 02:00:00 Pinheiros 338   NA  2  10 24 1.1    NA
## 4 2021-01-01 03:00:00 Pinheiros  NA   NA  0  11 18 0.0 24.25
## 5 2021-01-01 04:00:00 Pinheiros 272   NA  1  17 13 0.8 22.00
## 6 2021-01-01 05:00:00 Pinheiros 359   NA  1  12 20 1.2 23.25
```

Exercício 1

A MDA8 é a média móvel **máxima diária**. Como seria calculada?

Script do exercício 1: MDA8

```r
library(openair)
pin_mda8 <- timeAverage(pin_2021[c("date", "o3_8h")],
                        avg.time = "day", # Diária
                        statistic = "max") # Máxima
head(pin_mda8)
```

```
## # A tibble: 6 x 2
##   date                 o3_8h
##   <dttm>               <dbl>
## 1 2021-01-01 00:00:00  52.4
## 2 2021-01-02 00:00:00  49.6
## 3 2021-01-03 00:00:00  62.4
## 4 2021-01-04 00:00:00  67.9
```

Exercício 2

Quantos dias foi superado o padrão de O_3 na estação Pinheiros?

Script exercício 2

Podemos usar `subset`. O padrão é 130 μgm^{-3}.

```
dias_o3 <- subset(pin_mda8,
                  subset = o3_8h >= 130)
print(
  paste("O padrão foi superado ", nrow(dias_o3), "dias")
)
```

```
## [1] "O padrão foi superado  3 dias"
```

Combinar data frames

Combinar data frames

- Muitas vezes precisamos combinar duas tabelas que tem uma coluna comun. Por exemplo se temos uma tabela com a média anual de O_3 das estações e outra tabela com a média anual de $PM_{2.5}$, para isso usamos `merge()`.

Combinar data frames

```
o3 <- data.frame(aqs = c("pin", "ibu", "usp", "fsp"),
                 o3 = runif(4, 0, 160))
pm25 <- data.frame(aqs = c("pin", "ibu", "usp", "fsp"),
                   pm25 = runif(4, 10, 60))
dados <- merge(o3, pm25)
dados
```

```
##   aqs          o3     pm25
## 1 fsp  22.11558908 58.99931
## 2 ibu 152.19325729 58.00175
## 3 pin   0.01679815 18.39779
## 4 usp  97.24250678 41.14414
```

Completar dados faltantes

- Podemos usar o `merge` para completar com `NA` se um data frame não tem uma linha de dados.
- Precisamos adicionar o argumento `all = TRUE`.
- Serve para detectar quantos dados faltantes existem na nossa base de dados.

Completar dados faltantes

```
o3 <- data.frame(aqs = c("pin", "ibu", "usp", "fsp"),
                 o3 = runif(4, 0, 160))
pm25 <- data.frame(aqs = c("pin", "ibu", "usp"),
                   pm25 = runif(3, 10, 60))
dados <- merge(o3, pm25, all = TRUE)
dados
```

```
##   aqs       o3     pm25
## 1 fsp 147.7678       NA
## 2 ibu 127.9199 45.10363
## 3 pin 156.0687 14.55590
## 4 usp 116.8377 49.94713
```

Transformar tipos de objetos

`character` para `numeric`

- Às vezes quando usamos `read.table` ou `read_excel`, se uma coluna tem o dado faltante como um character especial (e.g "M"). Para poder operar precisamos forçar a transformação para `numeric`.

character para numeric

```
o3 <- c(90, 89, 76, 83, "M")
class(o3)
```

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

```
o3 <- as.numeric(o3) # Atualizamos o valor do vetor o3
class(o3)
```

```
## [1] "numeric"
```

```
o3
```

```
## [1] 90 89 76 83 NA
```

Transformar zona horária

- Fontes de dados globais podem ter a zona horária em UTC. Você precisa trazer para horário local.

```
# Criamos uma base de dados
date <- seq(as.POSIXct("2024-01-01 00:00", tz = "UTC"),
            length.out = 30 * 24, by = "hour")
o3 <- runif(length(date), 100, 160)
df <- data.frame(date, o3)
head(df$date)
```

```
## [1] "2024-01-01 00:00:00 UTC" "2024-01-01 01:00:00 UTC"
## [3] "2024-01-01 02:00:00 UTC" "2024-01-01 03:00:00 UTC"
## [5] "2024-01-01 04:00:00 UTC" "2024-01-01 05:00:00 UTC"
```

Transformar zona horária

```
attributes(df$date)$tzone <- "America/Sao_Paulo"
head(df$date)
```

```
## [1] "2023-12-31 21:00:00 -03" "2023-12-31 22:00:00 -03"
## [3] "2023-12-31 23:00:00 -03" "2024-01-01 00:00:00 -03"
## [5] "2024-01-01 01:00:00 -03" "2024-01-01 02:00:00 -03"
```

Boas praticas do R

Boas praticas do R

- Para cada trabalho usar RStudio Projects:
  - Mais fácil de compartilhar. Não precisa definir o caminho dos inputs.
- É bom olhar guias de estilo do R.
  - Google R guide:
    - Usar <- para definir variavéis.
    - Usar = para os argumntos.
    - Usar espaços para separar argumentos e =.
    - Usar nome significativos das variavéis.

Mais recursos.

Base de dados

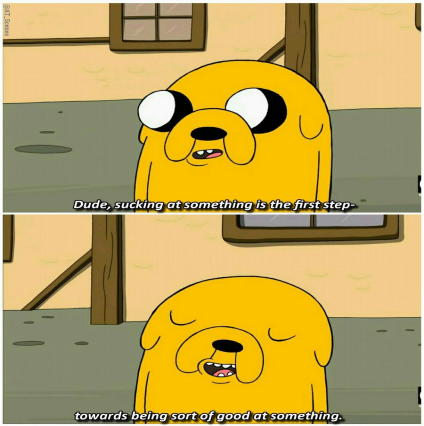
- Dados dos aeroportos: `riem` package
- Dados do brazil geobr `geobr` package
- Dados qualidade do ar São Paulo: `qualR` package
- Dados DATASUS: `DATASUS` package
- Dados DATASUS: `microdatasus` package
- Dados de qualidade do ar: *openaq*

Sobre R

- Funções do R: **R reference card**
- Galeria de figuras no R: **The R Graph Gallery**
- RStudio folha de dicas: **RStudio cheat sheet**

e agora?

e agora?


Dude, sucking at something is the first step-
towards being sort of good at something.

e agora?

- Tem que começar usar R (em vez de Excel)
- Procurar no google *R package <o que eu preciso>*
- Podem explorar **tidyverse**.
- Podem também nos escrever!

Muito Obrigado e boa prática!