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

Mario Gavidia-Calderón, Rafaela Squizzato, Thiago Nogueira

06/02/2024

Universidade de São Paulo

openair

1. Introdução

2. Instalar e desinstalar pacotes

3 Importando os dados

4. Organizando os dados

5. Exportando os dados

6. Exportando as figuras

7. Funções do pacote openair

openair

Manual do openair

1. Introdução

1.1 Software R

- Software gratuito, muito utilizado entre a comunidade científica;
- Muito material disponível na internet;
- Ótimo para armazenar e manipular dados; realizar cálculos e testes estatísticos; produção de gráficos etc;
- Ótima opção para quem trabalha com banco de dados grandes; otimização do tempo.

1.2 R ou Rstudio?

- Rstudio - interface mais amigável e funcional;
- Para usar o Rstudio é necessário ter instalado no computador o software R.
- `https://posit.co/download/rstudio-desktop/`

1.3 RStudio

Relembrando:

- Menu de ajuda: `?sum`, `help(sum)`
- R é sensível a maiúscula e minúsculas, ou seja, “a”≠“A”.
- Nunca deixe espaços entre nomes de objetos no R (ex: o3 noturno), coloque símbolos no lugar (ex: o3_noturno).
- Não é possível usar um número isolado para nomear objetos.
- Evite usar acentos, cedilhas, apóstrofes, aspas, etc. em nomes de objetos no R.

2. Instalar e desinstalar pacotes

2 Instalar e desinstalar pacotes

```
install.packages("openair")
remove.packages("openair")
rm(a) #remove o data.frame "a"
rm(list=ls()) #remove todos os data.frames no ambiente
```

2.1 Para que o pacote funcione ele precisa ser carregado

```
library(openair)
library(readxl) #para ler arquivos em excel (.xls)
```

3 Importando os dados

3.1 Em .xls

```
file <- 'G:/Meu Drive/QUALAMET_2023/QUALAMET_FMUSP_gases.xlsx'
dado <- read_excel(file)
```

OBS:

Inverter a barra \ para /.

3.2 Em .csv

```
# Definir diretório trabalho
setwd("G:/Meu Drive/QUALAMET_2023/NH3")
dado <- read.csv("nh3_all_hour.csv", header = TRUE)
#ou
dado <-
  read.csv("G:/Meu Drive/QUALAMET_2023/NH3/nh3_all_hour.csv",
           header = TRUE)
getwd() # mostra o diretório de trabalho atual
```

OBS:

O argumento `header = FALSE` indica que os dados não contém cabeçalho. Caso contrário, use `header = TRUE`.

3.3 Usando o qualR

```
install.packages('qualR',
      repos = c('https://ropensci.r-universe.dev',
                'https://cloud.r-project.org'))

library(qualR)
cetesb_aqs #nome,código, lat lon
cetesb_param #parâmetro, unidade e código

my_user_name <- "e-mail"
my_password <- "senha"
start_date <- "01/01/2020"
end_date <- "31/12/2021"
```

1° Exercício

1) Baixar a partir do pacote qualR as variáveis O3, PM2.5, VV e DV para a estação Pinheiros, para os anos de 2020 e 2021.

3.4 Reemplazando dados

```
dado$wd<- replace(dado$wd,  dado$wd > 360, NA)
```

4. Organizando os dados

4. Organizando os dados

```
dado$date <- as.POSIXct(
  strptime(dado$date, format = "%Y-%m-%d %H:%M:%OS", tz = "UTC"))
dado$hour <- as.numeric((dado$hour)) #transformando a coluna em numéric
names(dado) <- c("date","estacao","wd", "pm25","o3", "ws") #trocar o no
dado["hour"]<- format(dado$date, "%H") #criando coluna só com hora
novo <- dado[ ,c(1,5)] #novo dataframe com as variáveis selecionadas
dado$aqs <- "Pinheiros" # criou nova coluna

cbind # faz a junção dos data.frames pela coluna
rbind # faz a junção dos data.frames pela linha

df4 <- cbind(df1,df2,df3)
```

4.1 Transformar os dados horários em diários

```
diario <- timeAverage(dado, avg.time = "day")
```

OBS:

avg.time = "sec", "min", "hour", "day", "2 week", "week", "month", "year", "15 min"

4.2 Selecionando um período

```
periodo <- selectByDate(dado, year = 2020, month = 6:8, hour = c(8:18))
# também pode ser inserido no meio das funções
```

4.3 Trocando valores de calibração por NA's

```
dado$O3<- replace(dado$O3, dado$O3 <= 0, NA)
dado$O3 <- replace(dado$O3, dado$hour == 6, NA)
```

5. Exportando os dados

5. Exportando os dados

```
write.csv(dado, "C:/Users/Fulano/R files/dado_teste.csv",
          row.names = FALSE)
```

6. Exportando as figuras

6. Exportando as figuras

```
setwd(" C:/Users/Fulano/R files ")
png("O3.png", width = 9 * 300, height = 5 * 300, res = 300)
timeVariation (dado)
dev.off()
```

2° Exercício

2) Transformar a média horária em média diária dos dados e exportar o arquivo em csv.

```
diario <- timeAverage(dado, avg.time = "day")
write.csv(dado,
          "C:/Users/Rafaela/Dropbox/POSDOC/curso rstudio/dado_teste.csv
          row.names = FALSE)
```

7. Funções do pacote openair

7. Funções do pacote openair

- As funções presentes no pacote Openair são dedicadas à análise de dados de poluição atmosférica.
- As funções usam como padrão uma forma de análise mais simples e rápida, porém análises mais detalhadas também são possíveis.
- Entre as principais funções, temos: `summaryPlot`, `timePlot`, `calendarPlot`, `timeVariation`, `windRose`, `percentilRose`, `polarPlot`, `polarAnnulus`, `scatterPlot`, `corPlot.`

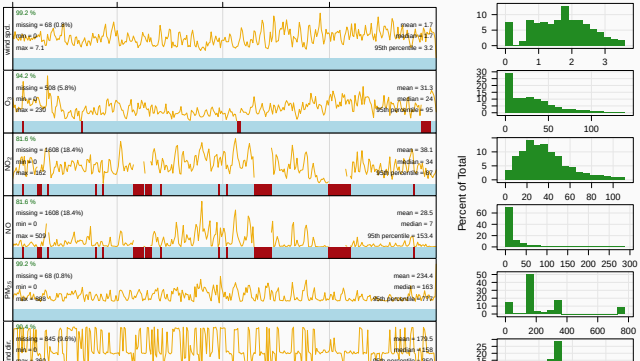
7.1 summaryPlot

- Interessante para monitoramento, pois resume rapidamente aspectos importantes dos dados, apresentando resumos estatísticos (mín, máx, média, mediana etc).

Comando simples: `summaryPlot(dado)`

7.1 summaryPlot

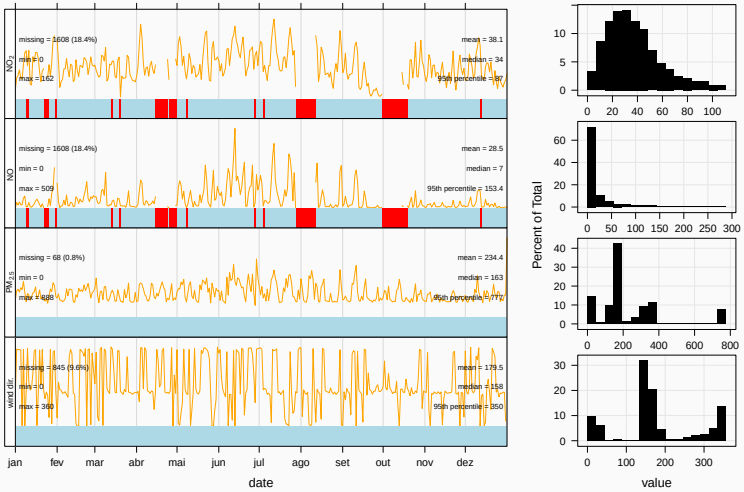
```
library(openair)
dado <- readRDS("../../data/pin_openair_ex.rds")
dado$wd<- replace(dado$wd,  dado$wd > 360, NA)
summaryPlot(dado)
```



7.1 summaryPlot

```
summaryPlot (dado[, c(1,3,4,5,6)], #selecionando as variáveis que desej
             period = "months", #melhora a visualização do dado para os
             print.datacap = FALSE, #traz o percentual por período
             date.breaks = 10, #número de intervalos principais do eixo
             avg.time = "day", #dados horários/diários
             date.format = "%b-%Y", #formato do eixo x
             col.data = "lightblue", #cor da barra de dados disponíveis
             col.trend = "orange", #cor da linha
             col.hist = "black", #cor do histograma ou gráfico de densi
             col.mis= "red") #cor a ser usada para mostrar dados ausent
```

7.1 summaryPlot



7.2 timePlot

- Para representar rapidamente séries temporais de dados ao mesmo tempo (vários poluentes ou variáveis). Possibilita visualizar o comportamento dos dados (picos de concentração, falhas nos dados, etc), auxiliando-o na seleção dos dados.
- Comando simples: `timePlot(dado)`

OBS:

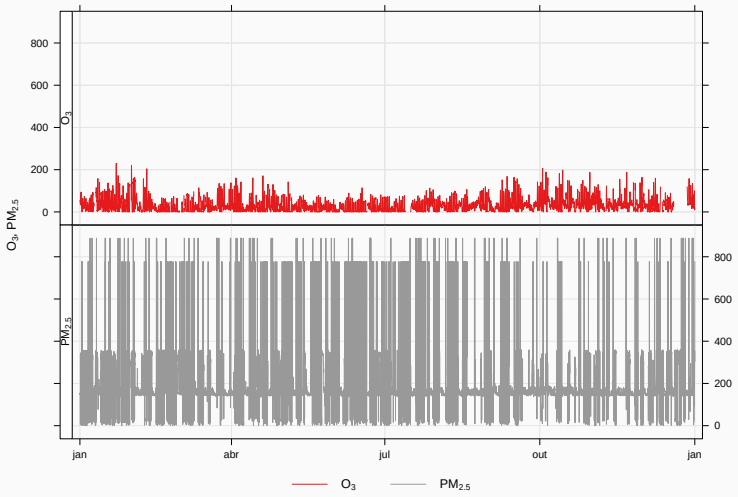
Pode ser usado o ‘selectByDate‘ nessa função

7.2 timePlot

```
timePlot(dado,
         pollutant = c("o3","pm25"))

timePlot(selectByDate(dado, year = "2019"), #selecionando o período
         pollutant = c("o3","pm25"), #selecionando o(s) poluente(s)
         y.relation = "free", #escala varia de acordo com o poluente
         name.pol = c ("O3 (ppb)","PM25 (ug/m3)"), #insere o nome dese
         date.format = "%b-%Y", #o formato da data
         avg.time = "month", #período de tempo da média
         date.breaks = (10),
         windflow = list(scale = 0.05, lwd =2, col = "gray"), #insere
         ref.y = list(h = 15, lty = 5, col = ("red"))) #insere linha ho
```

7.2 timePlot

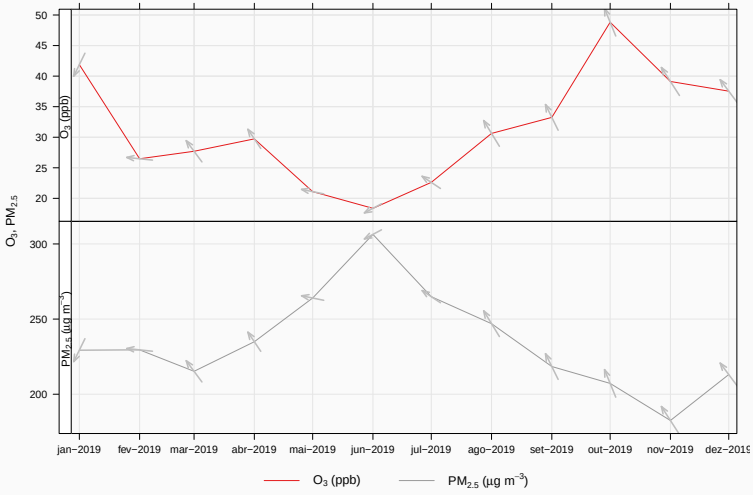
```
timePlot(dado, pollutant = c("o3","pm25"))
```



7.2 timePlot

```
timePlot(selectByDate(dado, year = "2019"), #selecionando o período
         pollutant = c("o3","pm25"), #selecionando o(s) poluente(s)
         y.relation = "free", #escala varia de acordo com o poluente
         name.pol = c ("O3 (ppb)","PM25 (ug/m3)"), #insere o nome dese
         date.format = "%b-%Y", #o formato da data
         avg.time = "month", #período de tempo da média
         date.breaks = (10),
         windflow = list(scale = 0.05, lwd =2, col = "gray"), #insere
         ref.y = list(h = 15, lty = 5, col = ("red"))) #insere linha ho

#obs: date.format (%d/%m/%Y; %b/%Y; %b-%Y; %d-%m; %m-%d; %B-%Y )
```

7.2 timePlot


O3, PM2.5
O3 (ppb)
PM2.5 (µg m−3)
50
45
40
35
30
25
20
300
250
200
jan−2019
fev−2019
mar−2019
abr−2019
mai−2019
jun−2019
jul−2019
ago−2019
set−2019
out−2019
nov−2019
dez−2019
O3 (ppb)
PM2.5 (µg m−3)

3° Exercício

3) Fazer um timeplot com as concentrações diárias de PM2.5 para os meses de junho, julho e agosto, mudando cor/espessura de linha e inserir uma linha referente ao limite do poluente (WHO).

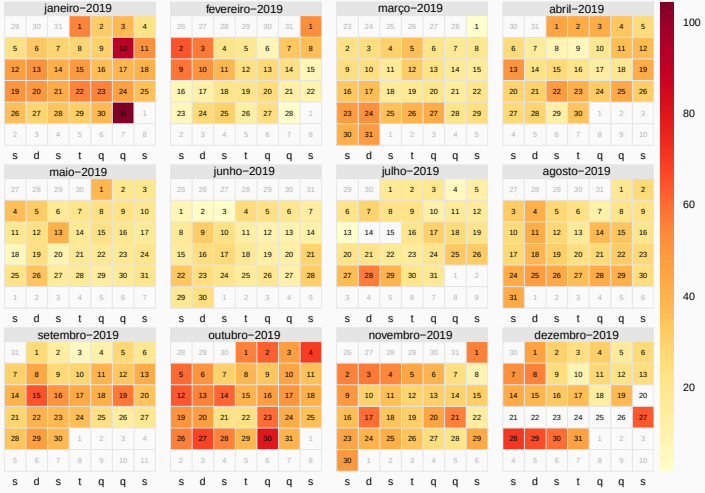
```
timePlot(selectByDate(dado, year = "2019", month = c("jun", "jul", "ago
         pollutant = c("pm25"),
         date.format = "%b-%Y",
         avg.time = "day",
         lwd = 2,
         ylab = "PM2.5 (ug/m3)",
         cols= "blue",
         ref.y = list(h = 15, lty = 5, col = ("red")))
```

7.3 calendarPlot

- Representação dos dados na escala de dias ou meses, mostrando as concentrações diárias dispostas em formato de calendário.
- O calendário pode dispor: os dias ou as concentrações. Além disso, é possível plotar setas que indicarão a direção do vento e velocidade, para observar a influência da meteorologia nas concentrações dos poluentes.
- Comando simples: `calendarPlot(dado)`

7.3 calendarPlot

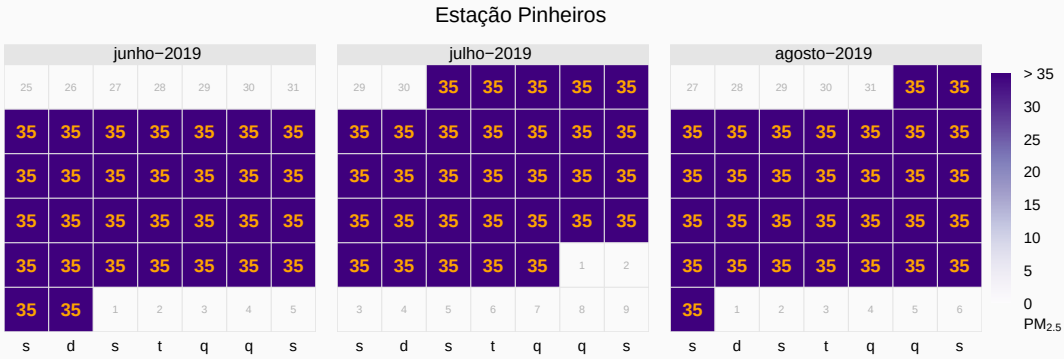
```
calendarPlot(dado, pollutant = "o3", year = 2019)
```



7.3 calendarPlot

```
calendarPlot(selectByDate(dado,
             year = 2019, month = c("jun","jul","ago")), #seleciona o a
             pollutant = "pm25", #seleciona o poluente
             annotate = "value", #o que estará no interior do quadrado
             statistic= "mean", #seleciona a análise estatística
             limits = c(0,35), #altera o limite da escala
             lim = 25, #destaca o poluente que ultrapassar o limite
             cols = c("Purples"), # muda a escala de cores: "increment"
             col.lim = c("black", "orange"), #
             layout = c(3, 1), #como estará disposta a figura
             main = "Estação Pinheiros", #título
             key.footer = "PM2.5") #texto na legenda: key.header = "PM2
```

7.3 calendarPlot

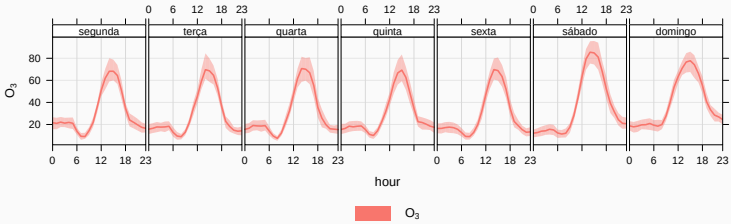


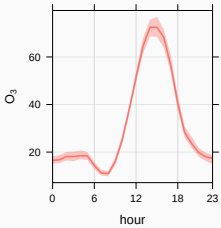
7.4 timeVariation

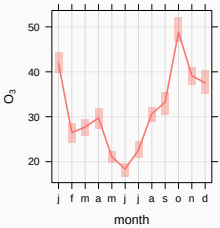
- Essa função produz quatro gráficos: a variação das concentrações do dia ao longo da semana, variação média da hora do dia, um gráfico mensal (média) e variação das concentrações ao longo da semana (média). Também é mostrado nos gráficos o intervalo de confiança de 95% na média.
- Na poluição atmosférica, a variação de um poluente por hora do dia e dia da semana pode revelar informações úteis sobre as fontes prováveis.

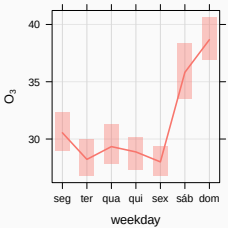
7.4 timeVariation

```
timeVariation(dado, pollutant = c("o3"))
```







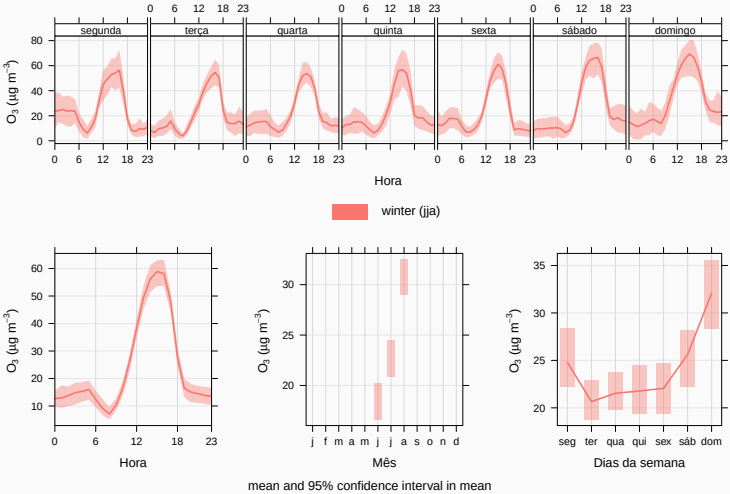


mean and 95% confidence interval in mean

7.4 timeVariation

```
timeVariation(selectByDate(dado, year = 2019, month = c("jun", "julho",
              pollutant = c("o3"),
              group = "season",
              ylab = " O3 (ug m-3)",
              hemisphere = "southern",
              xlab = c("Hora", "Hora", "Mês", "Dias da semana"))
#group = "season", "weekend", "month"
```

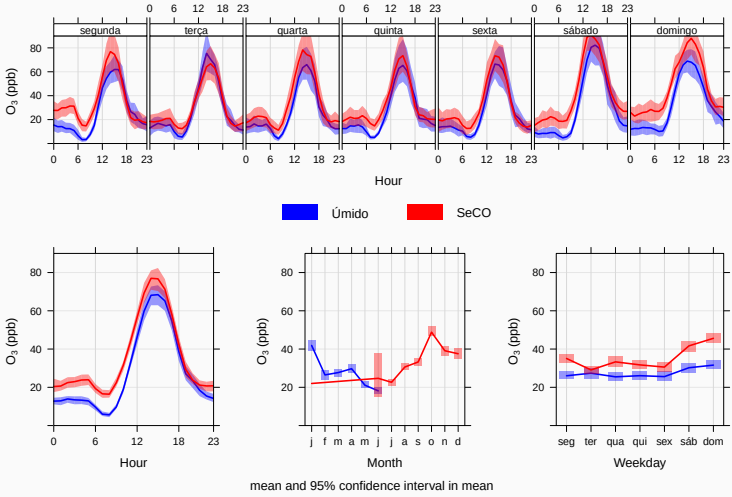
7.4 timeVariation



7.4 timeVariation

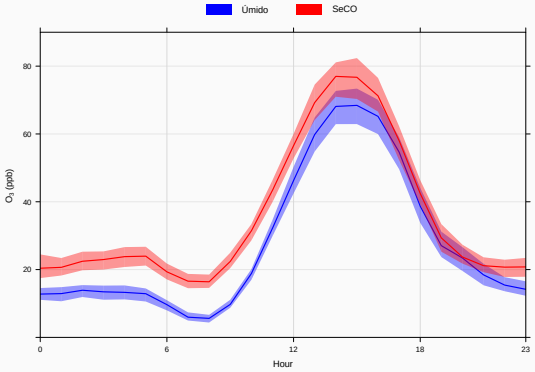
```
tv <- splitByDate(dado, dates = "30/06/2019",
                  labels = c("Úmido", "Seco"))

o3 <-timeVariation(tv, pollutant = c("o3"),
                   group = "split.by",
                   ylab = " O3 (ppb)",
                   xlab = c("Hour", "Hour", "Month", "Weekday"),
                   ylim = c(0,90),
                   cols = c("blue", "red"))
plot(o3, subset = "hour") #selecionando apenas 1 das 4 figuras
#subset: month, hour, day, day.hour
```

7.4 timeVariation



7.4 timeVariation

```
#selecionando apenas 1 das 4 figuras
#subset: month, hour, day, day.hour
plot(o3, subset = "hour")
```



4° Exercício

4) Usando o splitByDate, separe os dados em antes (2020) e depois (2021). Use a função timeVariation para plotar o ciclo diurno do O3, separando-a do resto da figura (subset = “hour”).

```
tv <- splitByDate(dado, dates = "31/12/2020",
                  labels = c("2020", "2021"))
o3 <-timeVariation(tv, pollutant = c("o3"),
                   group = "split.by",
                   ylab = " O3 (ppb)",
                   xlab = c("Hour", "Hour", "Month", "Weekday"),
                   ylim = c(0,90),
                   cols = c("blue", "red"))
plot(o3, subset = "hour") #selecionando apenas 1 das 4 figuras
```

7.5 windRose

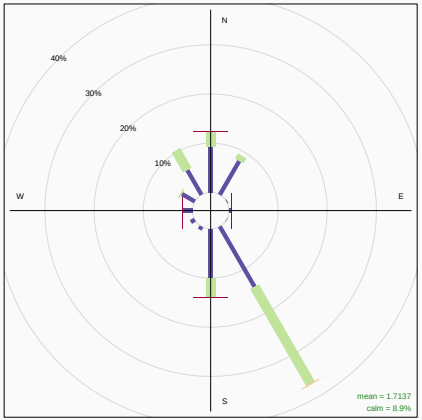
- Utilizada para visualizar dados de velocidade e direção do vento. Com essa função é possível observar como as condições meteorológicas variam com o tempo (mês, ano, estação, etc).
- Comando simples: `windRose(dado)`

OBS:

Deixar como nome das variáveis de ws e wd.

7.5 windRose

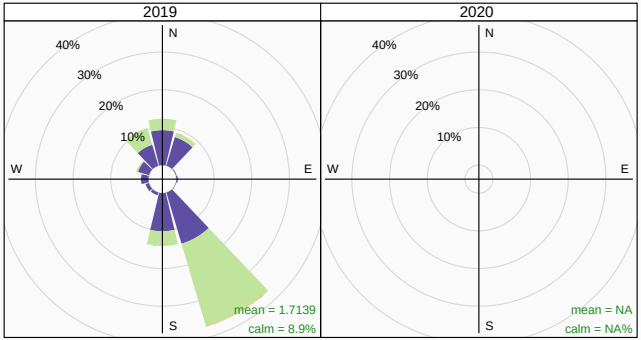
```
windRose(dado)
```



7.5 windRose

```
windRose(dado, type = "year",
         layout = c(3, 1), #layout do gráfico
         width = 2, #ajustar a largura dos intervalos de velocidade do
         paddle = FALSE, #muda o estilo do marcador de velocidade
         key.position =  "right", #muda a posição da escala
         key.header = "velocidade", #nomear legenda
         annotate = TRUE,#tirar os dados de média e calmaria
         angle.scale = 325) #muda a posição da frequência de contagem
#type: “season”, “year”, “weekday”
```

7.5 windRose



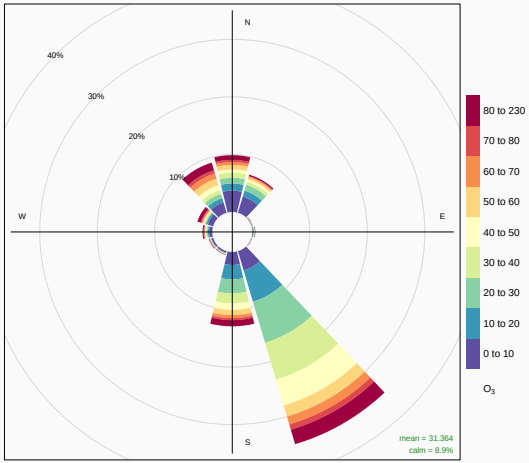
Frequency of counts by wind direction (%)

7.6 pollutionRose

- É uma variante da função “windRose” que é útil para considerar concentrações de poluentes por direção do vento ou, mais especificamente, a porcentagem de tempo em que a concentração está em uma faixa específica.
- Comando simples: `pollutionRose(dado, pollutant = "O3")`

7.6 pollutionRose

```
pollutionRose(dado, pollutant = "o3")
```

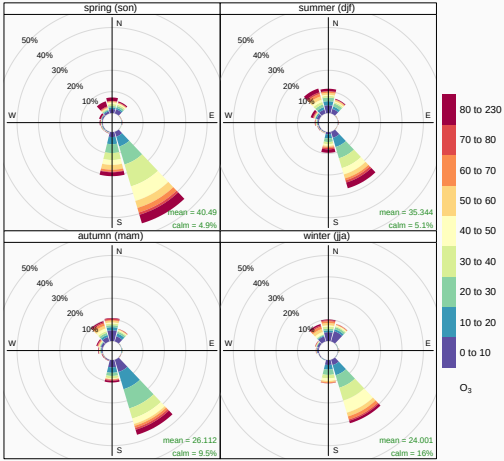


Frequency of counts by wind direction (%)

7.6 pollutionRose

```
pollutionRose (selectByDate(dado, year = 2019),
               pollutant = "o3",
               type = "season",
               hemisphere = "southern",
               layout = c(2,2))
```

7.6 pollutionRose



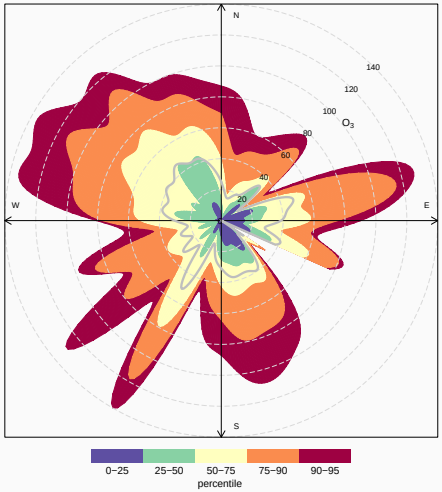
Frequency of counts by wind direction (%)

7.7 percentileRose

- Calcula os níveis percentuais de um poluente e os plota pela direção do vento.
- Comando simples: `percentileRose(dado, pollutant = "O3")`

7.7 percentileRose

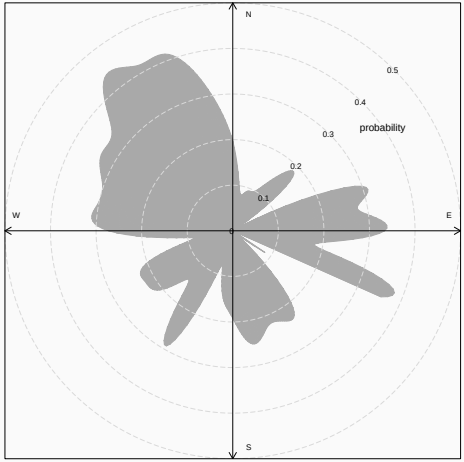
```
percentileRose(dado, pollutant = "o3", smooth = TRUE) #, percentile = N
```



7.7 percentileRose

```
percentileRose(dado,
               pollutant="o3",
               percentile = 75,
               method = "cpf", #função de probabilidade condicional
               col = "darkgray", #cor da figura
               smooth = TRUE) #dados suavizados
```

7.7 percentileRose



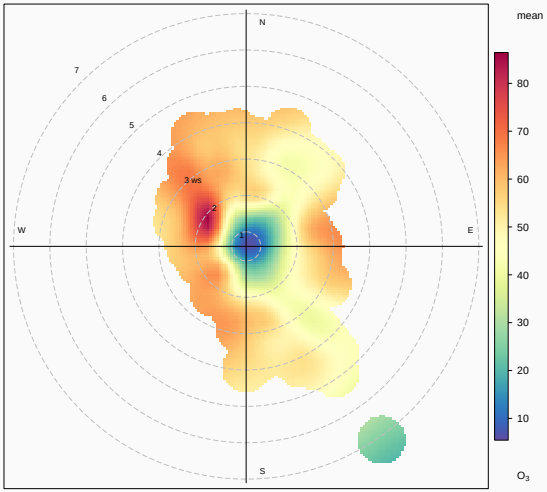
CPF at the 75th percentile (=47)

7.8 polarPlot

- Traça um gráfico polar bivariado de concentrações. As concentrações variam de acordo com a velocidade e a direção do vento.
- Comando simples: `polarPlot(dado, pollutant = "O3")`

7.8 polarPlot

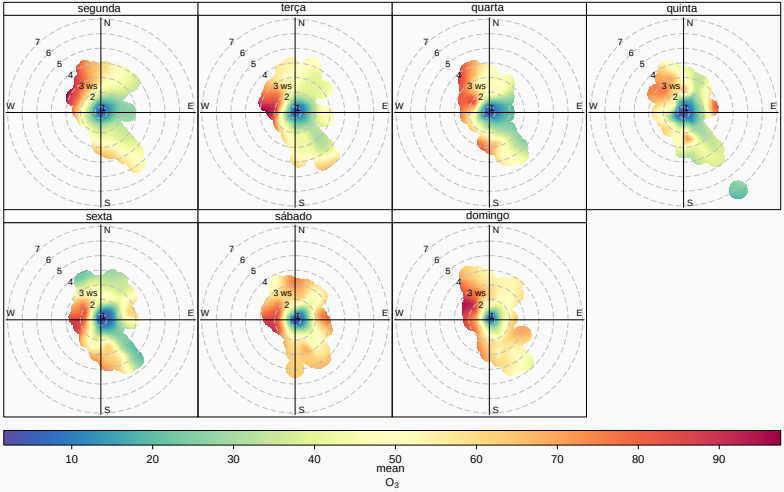
```
polarPlot(dado, pollutant = "o3")
```



7.8 polarPlot

```
polarPlot(dado,
          pollutant = "o3",
          key.footer = "O3", #adiciona texto na escala
          key.position = "bottom", #posição da escala
          type = "weekday", #“season”, “year”
          estatística = "mean")
#obs: statistic = “mean”(default), “median”, “max”(maximum), “frequency
```

7.8 polarPlot



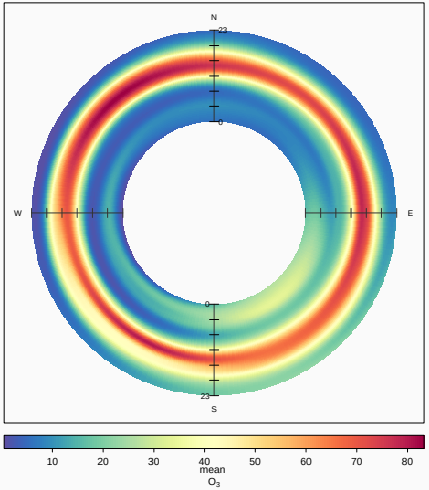
7.9 polarAnnulus

- Considera aspectos temporais de uma concentração de poluentes por direção do vento, visualizando variações diurnas, de dias da semana, sazonais e de tendências.
- Comando simples: `polarAnnulus(dado, pollutant = "O3")`

7.9 polarAnnulus

```
polarAnnulus(dado,
             pollutant = "o3",
             period = "hour",
             exclude.missing = FALSE,
             key.position = "bottom",
             key.footer = "O3")
```

7.9 polarAnnulus



5° Exercício

5) Fazer um polarPlot e um polarAnnulus de todo o período analisado, usando o statistic = “mean”(default). E plotar elas juntas, com as escalas do lado direito.

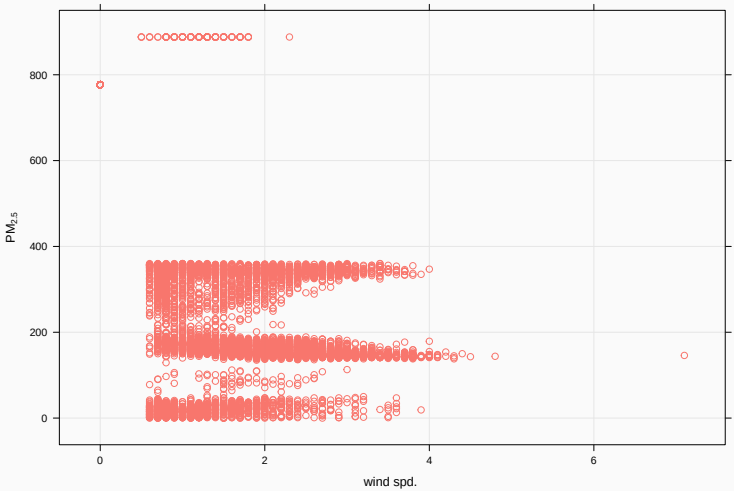
```
library(gridExtra) #plotar figuras juntas
pp <- polarPlot(dado, pollutant = "o3", estatística = "mean",
          key.footer = "O3", #adiciona texto na escala
          key.position = "right") #posição da escala

pa<- polarAnnulus(dado, pollutant = "o3", period = "hour",
             exclude.missing = FALSE, key.position = "right",
             key.footer = "O3", estatística = "mean")
grid.arrange(pp$plot, pa$plot, nrow=1, ncol=2)
```

7.10 scatterPlot

- São gráficos de dispersão. O objetivo da função scatterPlot é tornar simples a consideração de como as variáveis estão relacionadas entre si de uma forma consistente.
- Comando simples: `scatterPlot(dado)`

7.10 scatterPlot

```
scatterPlot(dado, x="ws", y="pm25")
```

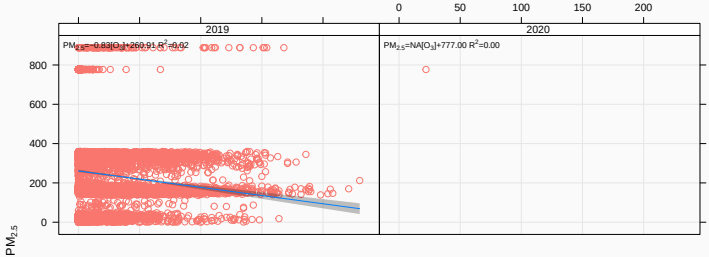


7.10 scatterPlot

```
scatterPlot(dado, x = "o3", y = "pm25",
linear = TRUE, #mostra a equação da reta e o valor de R2
type = "year", #"wd", "season"
layout = c(2, 2))
```

7.10 scatterPlot

```
## Warning in predict.lm(mod, data.frame(x = xseq), interval = "confide
## prediction from a rank-deficient fit may be misleading

## Warning in qt((1 - level)/2, df): NaNs produced
```

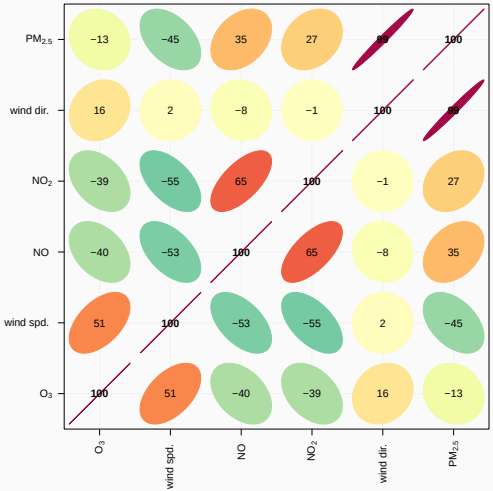


7.11 corPlot

- Mostra a correlação entre duas ou mais variáveis.
- Comando simples: `corPlot(dado)`

7.11 corPlot

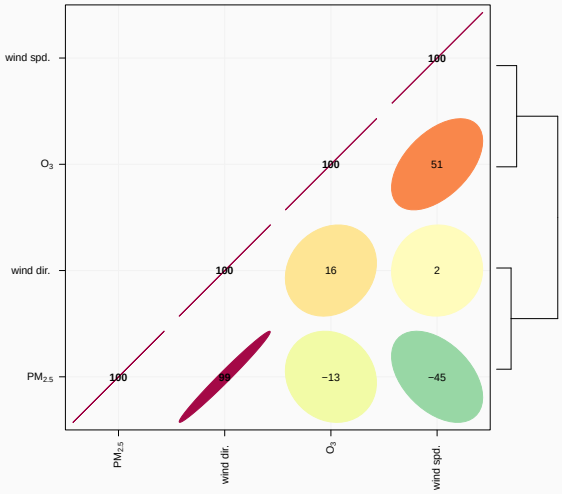
```
corPlot (dado)
```



7.11 corPlot

```
corPlot(dado,
        pollutants =c ( "o3", "pm25", "ws", "wd"),
        dendrogram = TRUE,
        lower = TRUE)
#obs1: type = "default“ -> dendrogram = TRUE
#obs2: type = “season”, “year”, “weekday” etc.
```

7.11 corPlot


wind spd.
O3
wind dir.
PM2.5
100
100
100
51
100
16
2
100
99
−13
−45
PM2.5
wind dir.
O3
wind spd.

Atividade final 1:

Selecionar o período de 01 a 22 de março de 2020 (pré-lockdwon) e de 23 de março a 13 de abril de 2020 (Lockdown). Plotar a série temporal e verificar se ocorreu diferença de concentração entre os períodos.Avaliar também se houve diferença entre as concentrações de O3 e PM2.5 quanto ao perfil diurno e dia da semana.