

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

openair

---

Mario Gavidia-Calderón, Rafaela Squizzato, Thiago Nogueira

19/02/2025

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. Funções do pacote openair

openair

---

# Manual do openair

Manual do openair [https://bookdown.org/david\\_carslaw/openair/](https://bookdown.org/david_carslaw/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)`, `?openair::summaryPlot`
- 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 acento, cedilha, apóstrofo, 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'  
dados <- 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 <- "01/01/2023"
```

```
end_date <- "31/12/2024"
```

# 1º Exercício

- 1) Baixar a partir do pacote qualR as variáveis O3, PM2.5, NO2, VV e DV para a estação Pinheiros, para os anos de 2023 e 2024.



# Script do exercício 1

```
my_user_name <- "e-mail"
my_password <- "senha"

dado <- cetesb_retrieve_param(my_user_name,
                              my_password,
                              c("03", "MP2.5", "VV", "DV"),
                              "Pinheiros",
                              start_date = "01/01/2023",
                              end_date = "31/12/2024")
```

## 4. Organizando os dados

---

## 4.1 Transformar os dados horários em diários

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

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

### OBS:

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

## 4.2 Selecionando um período

```
perodo <- selectByDate(dado,  
                        year = 2024, month = 6:8, hour = c(8:18))  
# também pode ser inserido no meio das funções
```

## 5. Exportando os dados

---

## 5.1 Exportando os dados

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

## 2º Exercício

- 2) Transformar a média horária em média diária. Exportar, em csv, dois arquivos (1 com dados diários e 1 horários).

## Script do exercício 2

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



## 6. Funções do pacote openair

---

## 6. 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`.

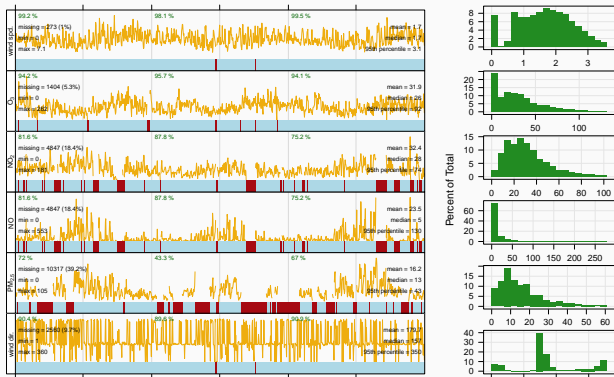
## 6.1 summaryPlot

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

Comando simples: `summaryPlot(dado)`

# 6.1 summaryPlot

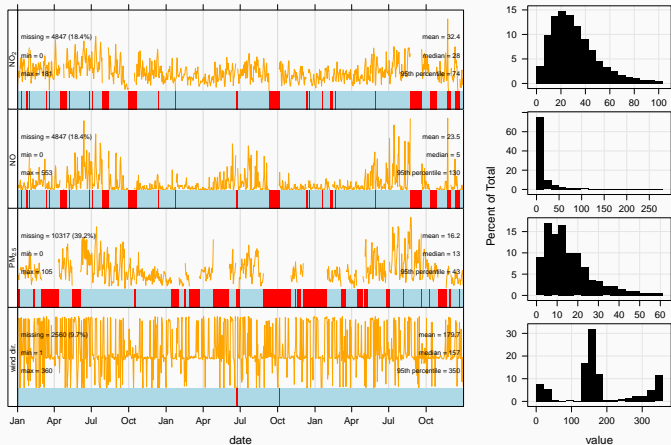
```
library(openair)
dado <- readRDS("../../data/pin_openair_ex.rds")
summaryPlot(dado)
```



## 6.1 summaryPlot

```
summaryPlot (dado[, c(1,3,4,5,6)], #selecionando as variáveis que de  
    period = "months", #melhora a visualização do dado para  
    print.datacap = FALSE, #traz o percentual por período  
    avg.time = "day", #dados horários/diários  
    col.data = "lightblue", #cor da barra de dados disponíveis  
    col.trend = "orange", #cor da linha  
    col.hist = "black", #cor do histograma ou gráfico de densidade  
    col.mis= "red") #cor a ser usada para mostrar dados ausentes
```

## 6.1 summaryPlot



## 6.2 timePlot

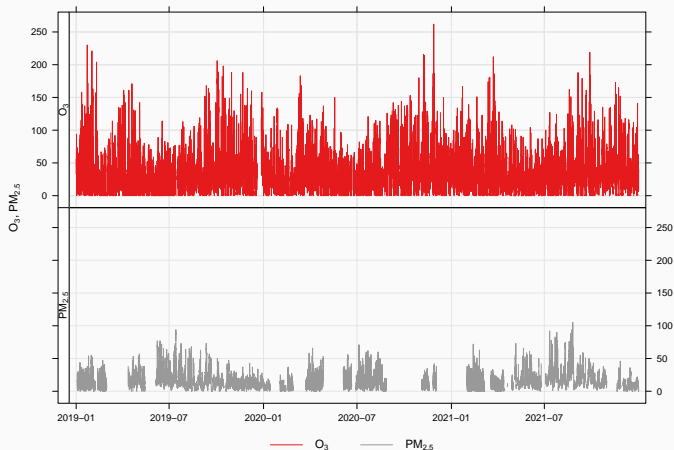
- Para representar rapidamente séries temporais de dados ao mesmo tempo (vários poluentes ou variáveis). Possibilita visualizar o comportamento dos dados (picos de concentração, falhas nos dados, etc), auxiliando-o na seleção dos dados.
- Comando simples: `timePlot(dado, pollutant = c("X"))`

### OBS:

Pode ser usado o 'selectByDate' nessa função

## 6.2 timePlot

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





## 6.2 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 des
  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

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

## 6.2 timePlot



### 3º Exercício

- 3) Fazer um timeplot com as concentrações diárias de PM2.5 para os meses de junho, julho e agosto (ano 2024), mudando cor/espessura de linha e inserir uma linha referente ao limite do poluente (WHO - 15  $\mu\text{g}/\text{m}^3$ ).

## Script do exercício 3

```
timePlot(selectByDate(dado, year = "2024",  
  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")))
```

## 6.3 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, pollutant = "X")`

## 6.3 calendarPlot

```
library(openair)
# calendarPlot(dado, pollutant = "o3", year = 2024)
head(dado)
```

##		date	aqs	wd	pm25	no	no2	o3	ws
## 1	2019-01-01	00:00:00	Pinheiros	NA	NA	NA	NA	NA	NA
## 2	2019-01-01	01:00:00	Pinheiros	147	NA	0	9	56	2.4
## 3	2019-01-01	02:00:00	Pinheiros	155	NA	0	13	47	1.7
## 4	2019-01-01	03:00:00	Pinheiros	153	NA	0	12	45	1.7
## 5	2019-01-01	04:00:00	Pinheiros	149	NA	0	9	45	1.9
## 6	2019-01-01	05:00:00	Pinheiros	148	NA	0	9	45	1.9

## 6.3 calendarPlot

```
calendarPlot(selectByDate(dado,  
  year = 2024, month = c("jun","jul","aug")), #seleciona o ano  
  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 = 15, #destaca o dia que ultrapassar o limite  
  cols = c("Purples"), #muda 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 = "PM2.5"
```

## 6.3 calendarPlot

Estação Pinheiros



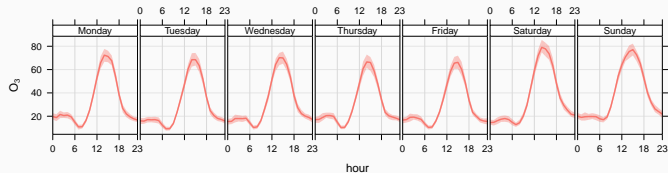


## 6.4 timeVariation

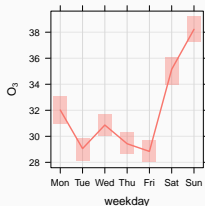
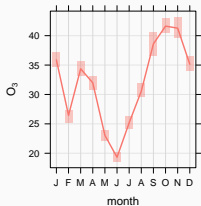
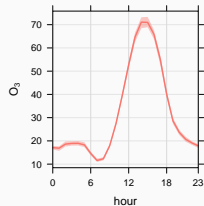
- Essa função produz quatro gráficos: a variação das concentrações do dia ao longo da semana, variação média da hora do dia, um gráfico mensal (média) e variação das concentrações ao longo da semana (média). Também é mostrado nos gráficos o intervalo de confiança de 95% na média.
- Na poluição atmosférica,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

## 6.4 timeVariation

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



■  $O_3$

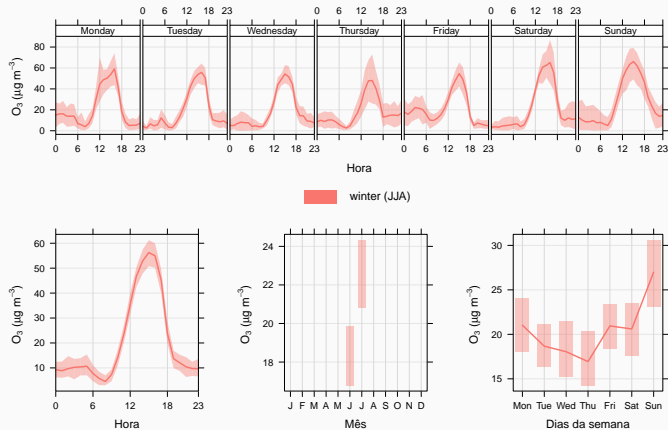


mean and 95% confidence interval in mean

## 6.4 timeVariation

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

## 6.4 timeVariation

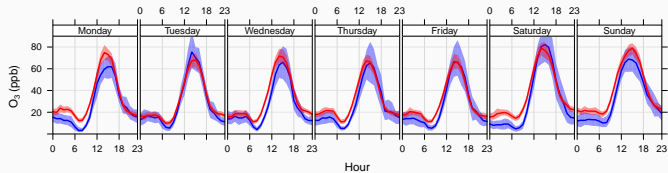


mean and 95% confidence interval in mean

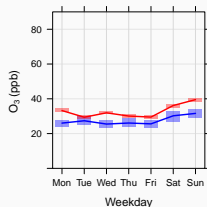
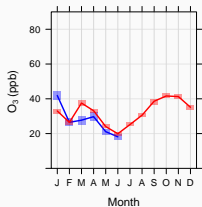
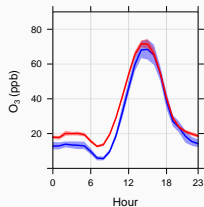
## 6.4 timeVariation

```
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") #seleccionando apenas 1 das 4 figuras  
#subset: month, hour, day, day.hour
```

## 6.4 timeVariation



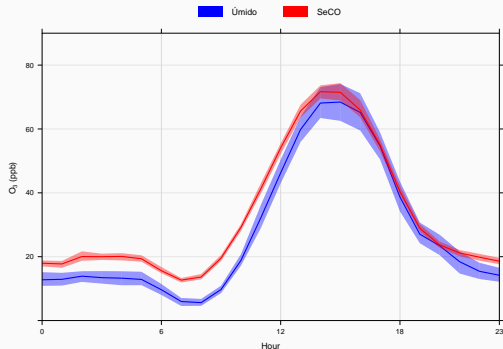
Úmido SeCO



mean and 95% confidence interval in mean

## 6.4 timeVariation

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



## 4º Exercício

- 4) Usando o `splitByDate`, separe os dados em antes (2023) e depois (2024). Use a função `timeVariation` para plotar o ciclo diurno do O3, separando-a do resto da figura (`subset = "hour"`).



## Script do exercício 4

```
tv <- splitByDate(dado, dates = "31/12/2023",  
                 labels = c("2023", "2024"))  
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
```

## 6.5 windRose

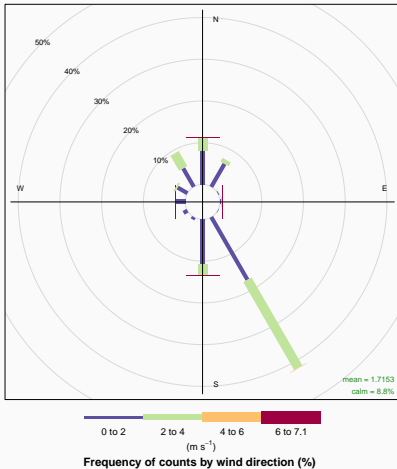
- Utilizada para visualizar dados de velocidade e direção do vento. Com essa função é possível observar como as condições meteorológicas variam com o tempo (mês, ano, estação, etc).
- Comando simples: `windRose(dado)`

### OBS:

Deixar como nome das variáveis de `ws` e `wd`.

## 6.5 windRose

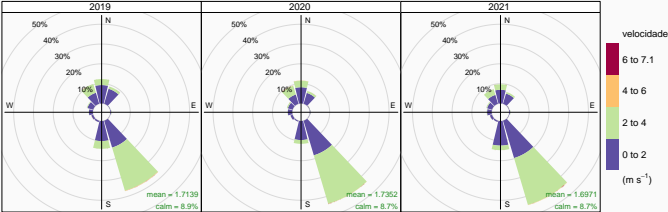
`windRose(dado)`



## 6.5 windRose

```
windRose(dado, type = "year",  
         layout = c(3, 1), #layout do gráfico  
         width = 2, #ajustar a largura dos intervalos de velocidade  
         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"
```

# 6.5 windRose



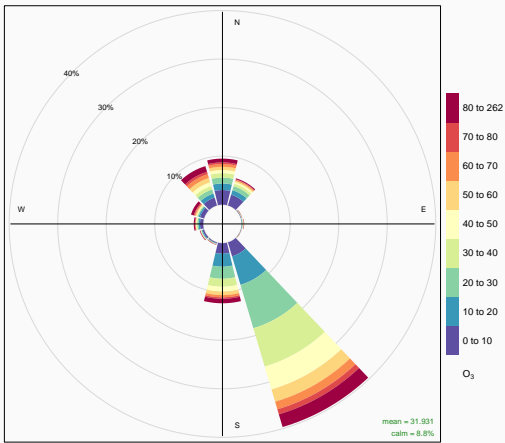
Frequency of counts by wind direction (%)

## 6.6 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 = "X")`

## 6.6 pollutionRose

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

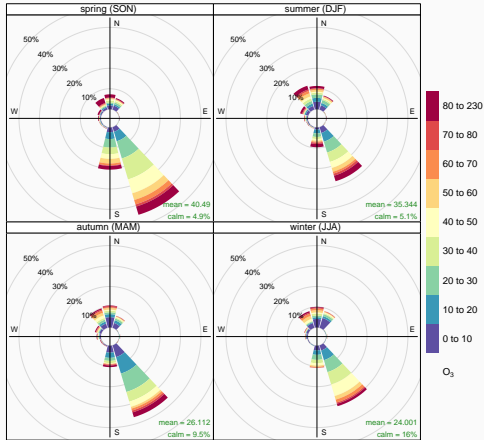


## 6.6 pollutionRose

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



## 6.6 pollutionRose



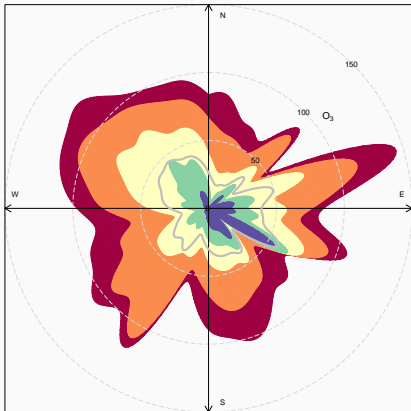
Frequency of counts by wind direction (%)

## 6.7 percentileRose

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

## 6.7 percentileRose

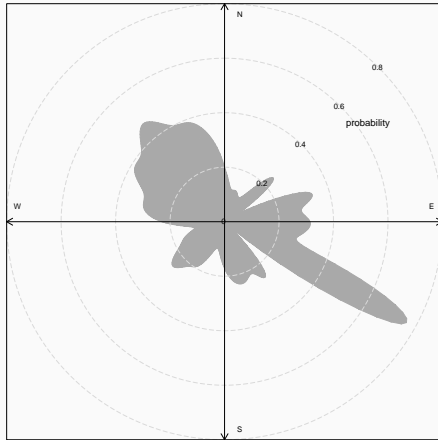
```
percentileRose(dado, pollutant = "o3",  
              smooth = TRUE)
```



## 6.7 percentileRose

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

## 6.7 percentileRose



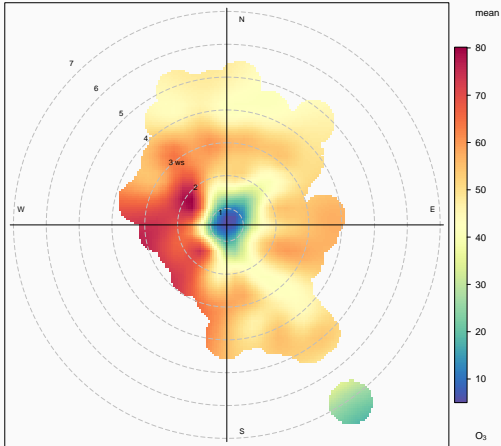
CPF at the 75th percentile (=47)

## 6.8 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 = "X")`

## 6.8 polarPlot

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

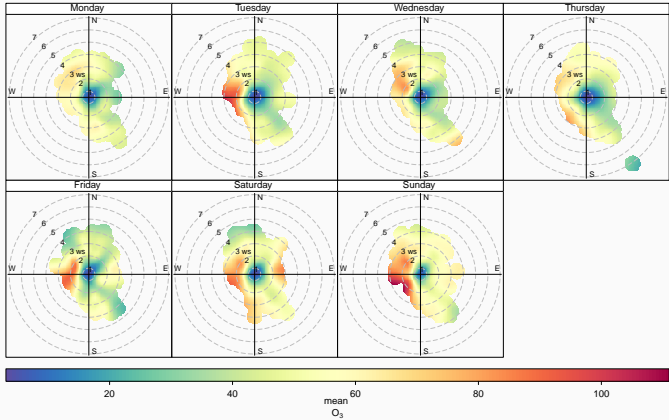


## 6.8 polarPlot

```
polarPlot(dado,  
          pollutant = "o3",  
          key.footer = "03", #adiciona texto na escala  
          key.position = "bottom", #posição da escala  
          type = "weekday", #"season", "year"  
          statistic = "mean")  
#obs: statistic = "mean"(default), "median", "max"(maximum), "frequ
```



## 6.8 polarPlot



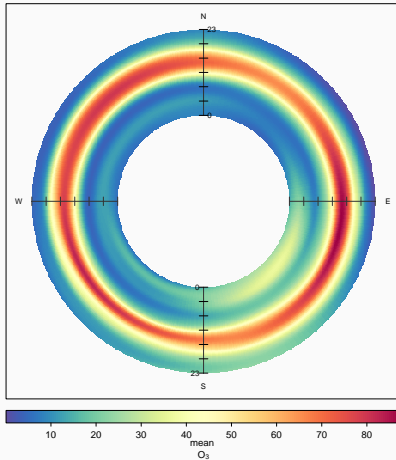
## 6.9 polarAnnulus

- 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 = "X")`

## 6.9 polarAnnulus

```
polarAnnulus(dado,  
              pollutant = "o3",  
              period = "hour",  
              exclude.missing = FALSE,  
              key.position = "bottom",  
              key.footer = "03")
```

## 6.9 polarAnnulus



## 5º Exercício

- 5) Fazer um `polarPlot` e um `polarAnnulus` de todo o período analisado (2023 e 2024), usando o `statistic = "mean"` (default). E plotar elas juntas, com as escalas do lado direito.

## Script do exercício 5

```
library(gridExtra) #plotar figuras juntas
pp <- polarPlot(dado, pollutant = "o3", statistic = "mean",
               key.footer = "03", #adiciona texto na escala
               key.position = "right") #posição da escala

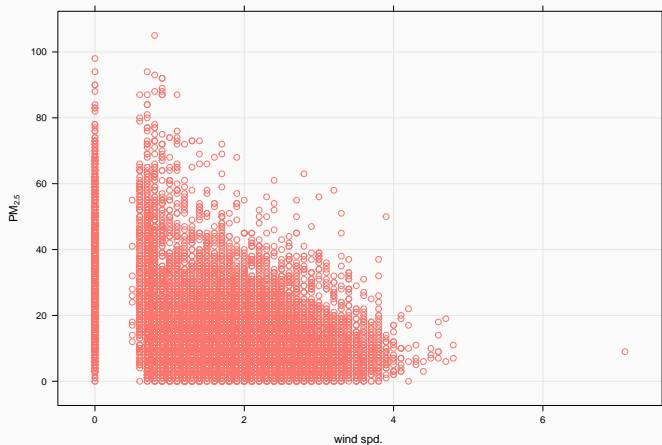
pa<- polarAnnulus(dado, pollutant = "o3", period = "hour",
                 exclude.missing = FALSE, key.position = "right",
                 key.footer = "03", statistic = "mean")
grid.arrange(pp$plot, pa$plot, nrow=1, ncol=2)
```

## 6.10 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, x = "X", y = "Y")`

## 6.10 scatterPlot

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

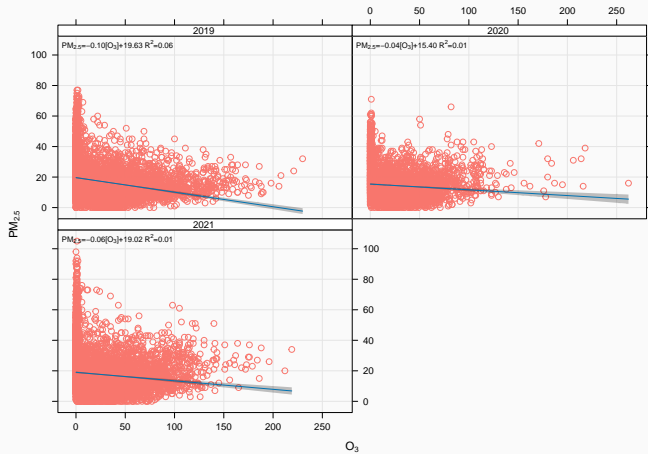




## 6.10 scatterPlot

```
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))
```

## 6.10 scatterPlot

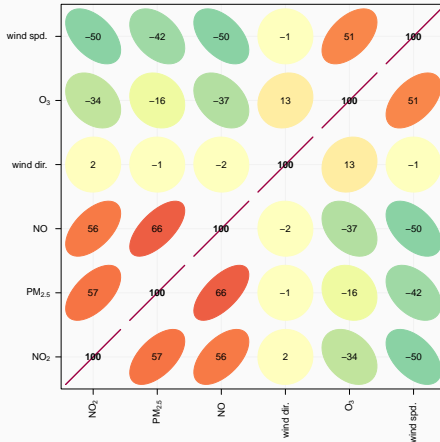


## 6.11 corPlot

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

## 6.11 corPlot

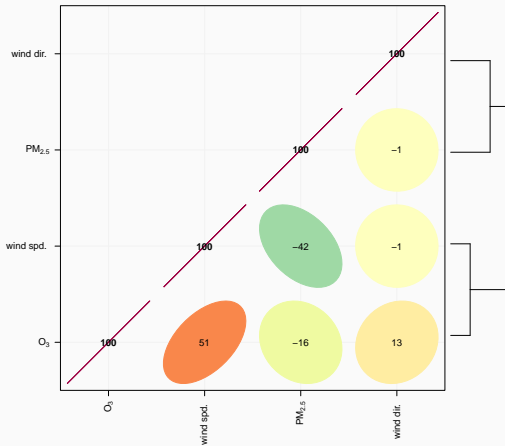
`corPlot(dado)`



## 6.11 corPlot

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

## 6.11 corPlot



## 7. Extras

```
dado$date <- as.POSIXct(  
  strptime(dado$date, format = "%Y-%m-%d %H:%M:%S", tz = "UTC"))  
dado$hour <- as.numeric((dado$hour)) #transformando a coluna em numé  
names(dado) <- c("date","estacao","wd", "pm25","o3", "ws") #trocar o  
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)
```

## 7.1 Trocando valores de calibração por NA's

```
dado$03<- replace(dado$03, dado$03 <= 0, NA)  
dado$03 <- replace(dado$03, dado$hour == "06", NA)
```



## 7.2 Exportando as figuras

```
setwd(" C:/Users/Fulano/R files ")  
png("03.png", width = 9 * 300, height = 5 * 300, res = 300)  
timeVariation (dado, pollutant = "o3")  
dev.off()
```