**MORE ON OPENAIR**

TEMPLATE: Please copy content/questions to your own document.

Your name(s):

**STANDARD AND ADVANCED PLOTS**

Below is a list of *openair* plot functionalities that might be useful for analyzing the data from your site or for supporting your team’s research question. For each function’s page, we have provided information on the function’s purpose and requirements, as well as example code and their plots. Yes, some functions are repeats from the earlier assignment – feel free to explore more of their options.

For each function, try out an example code with the data from your team’s site. Plot those figures as well as the code that produced it.

Afterwards, paste 3 figures that your team believes best supports your research question (on 3rd page). You may include supplementary figures as well, but note these as supplementary. Give reasoning of why your team chose them. These 3 figures do not have to be the final figures your team presents during the Preliminary Presentation or Final Project, but they should help narrow down your choices.

**SET UP**

The following examples require the initial setup code:

**library**(openair);  
**library**(tidyverse);  
  
dat <- importAURN(site = "my1", year = 2000:2005);

*Remember that your plots should be based on the data from your team’s Germany site.*

**OPENAIR PLOT FUNCTIONS**

[WIND ROSE](#_sva2bbkp3ohk)

[POLLUTION ROSE](#_skahvfcd63z)

[POLAR FREQUENCIES](#_kui3t46f2s1)

[PERCENTILE ROSE](#_bj2zoi30cc42)

[POLAR PLOTS](#_h9i2r06vp9w4)

[POLAR ANNULUS](#_c87vnoo68tbs)

[TIME SERIES PLOTS](#_67uw3hhop49h)

[TEMPORAL VARIATION PLOTS](#_tcg6jibq71mp)

[TIME PROP PLOTS](#_eqwzoix6v4vc)

[TREND LEVEL HEAT MAP](#_j6zxu3f9s1zs)

[CALENDAR PLOT](#_zfxsjl9dtyk2)

[THEIL-SEN TRENDS](#_pzhr8xki87e0)

[SMOOTH TREND](#_z5mwn1wirv23)

[SCATTER PLOTS](#_3oww3rkcqw6c)

[FIGURE EXPORTING](#_mx14pjvynhon)

# 

# Paste 3 figures that your team believes best supports your research question. You may include supplementary figures as well, but note these as supplementary. Give reasoning of why your team chose them. These 3 figures do not have to be the final figures your team presents during the Preliminary Presentation or Final Project, but they should help narrow down your choices.

# 

Code 1:

# Plot 1:

# 

# Reasoning 1:

# 

Code 2:

# Plot 2:

# Reasoning 2:

# 

# 

Code 3:

# Plot 3:

# Reasoning 3:

# 

# 

Optional Supplementary Code, Plot(s), and Reasoning(s):

# WIND ROSE

**Uses (and Sources):**

Summarizes the frequencies of winds of various velocities at a single location.

(Air Pollution Control Engineering)

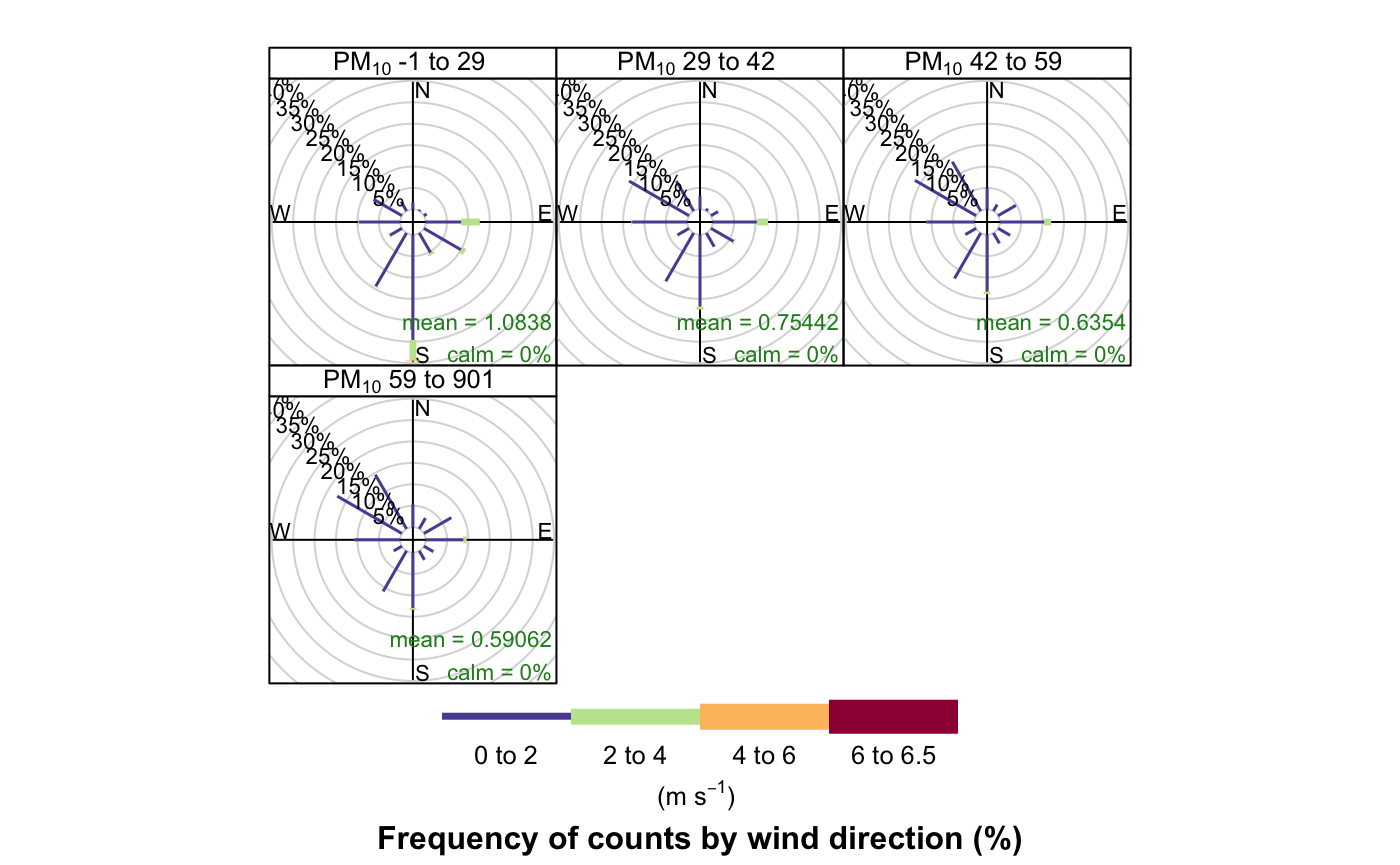
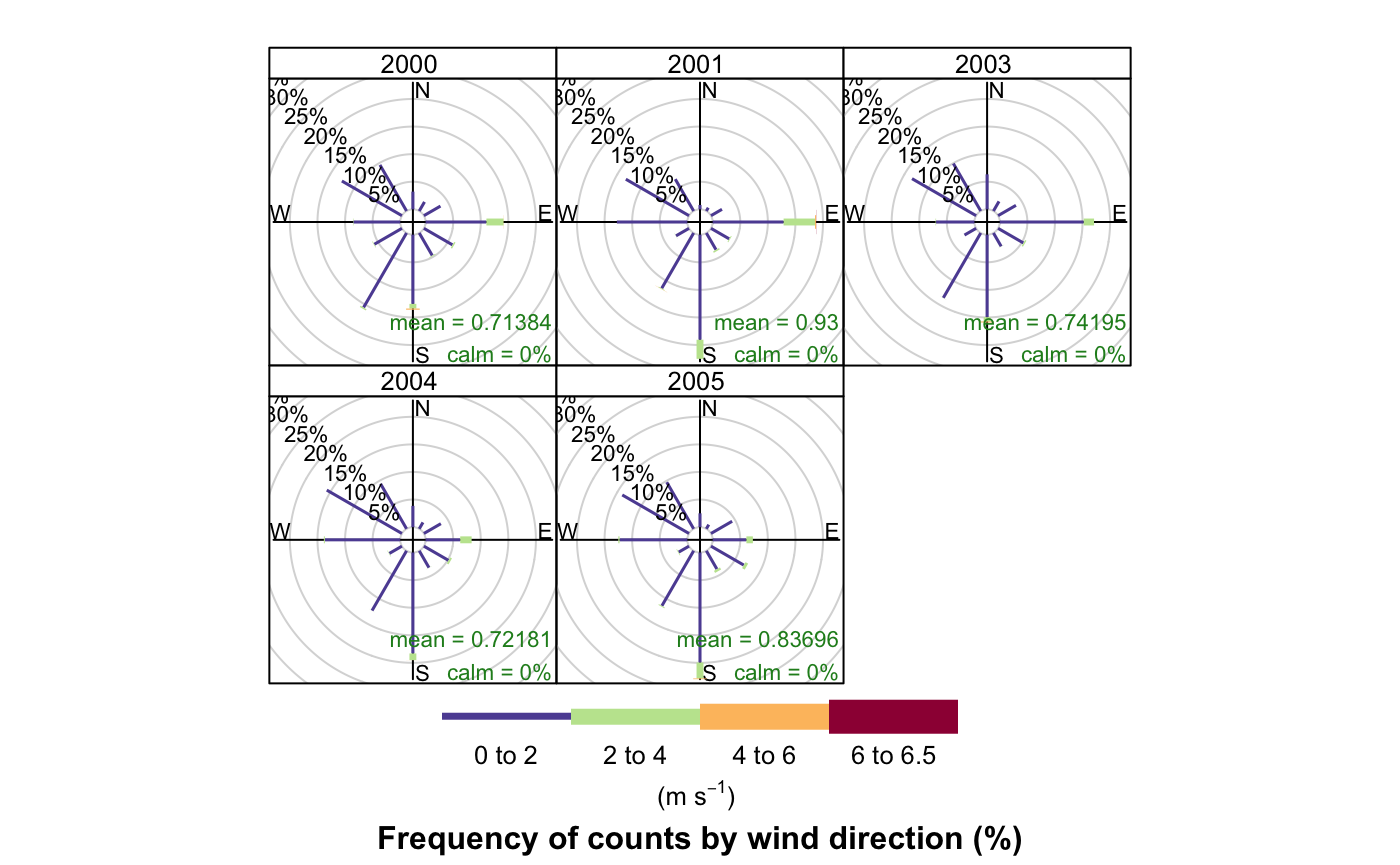
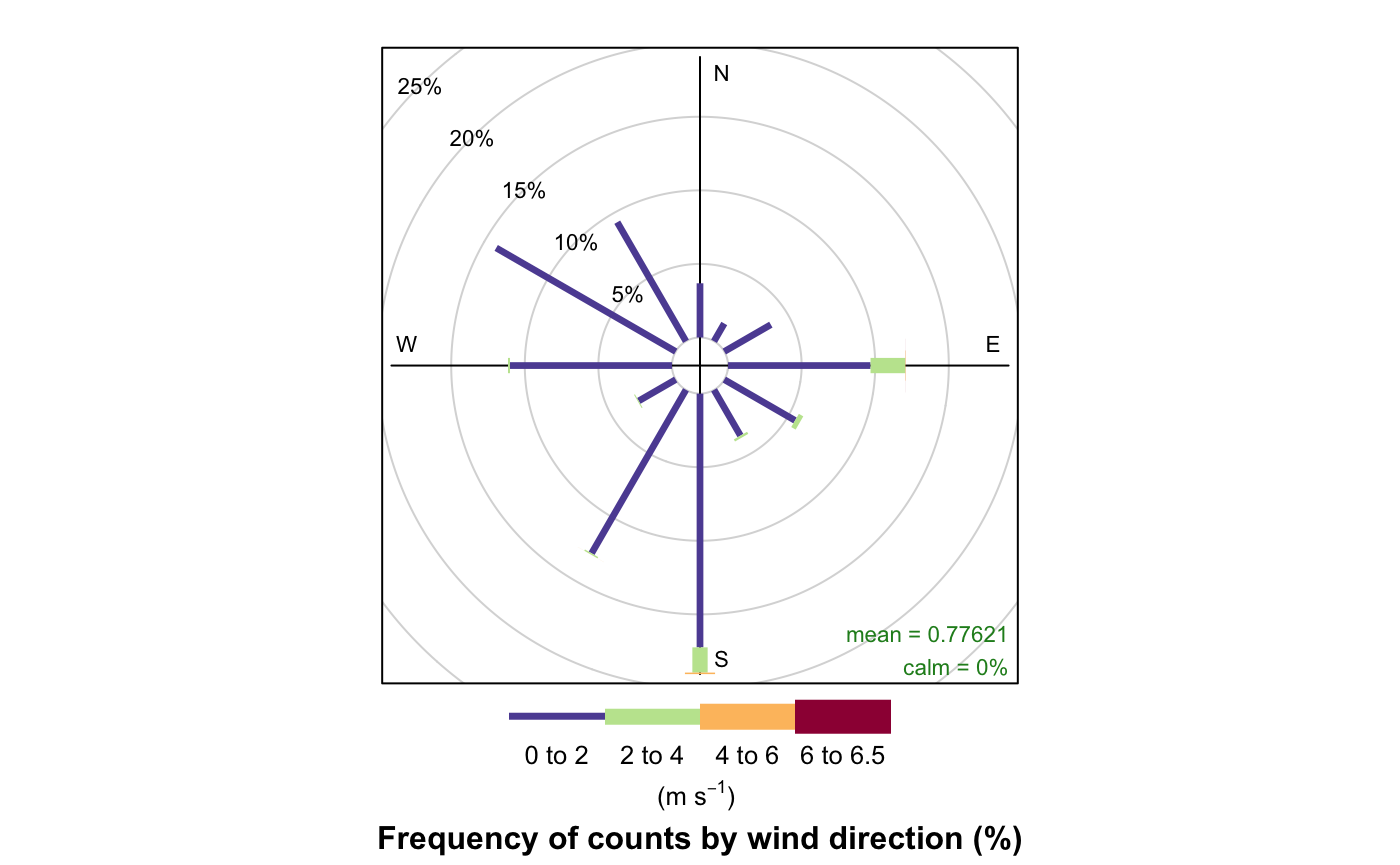
**Data Columns Required:**

* Wind Direction (wd)
* Wind Speed (ws)
* Date/Time (date)

**Example Code:**

| # plot a windrose for everything windRose(dat)  # plot by year windRose(dat, type = "year", layout = c(3, 2))  # plot by pm10 (the type function is useful!) windRose(dat, type = "pm10", layout = c(3, 2)) |
| --- |

**Example Plot Output:**



Your Code:

Your Plot(s):

# 

# POLLUTION ROSE

**Uses (and Sources):**

Summarizes the levels of pollutants associated with certain wind directions.

(Source Region ID using Kernel Smoothing)

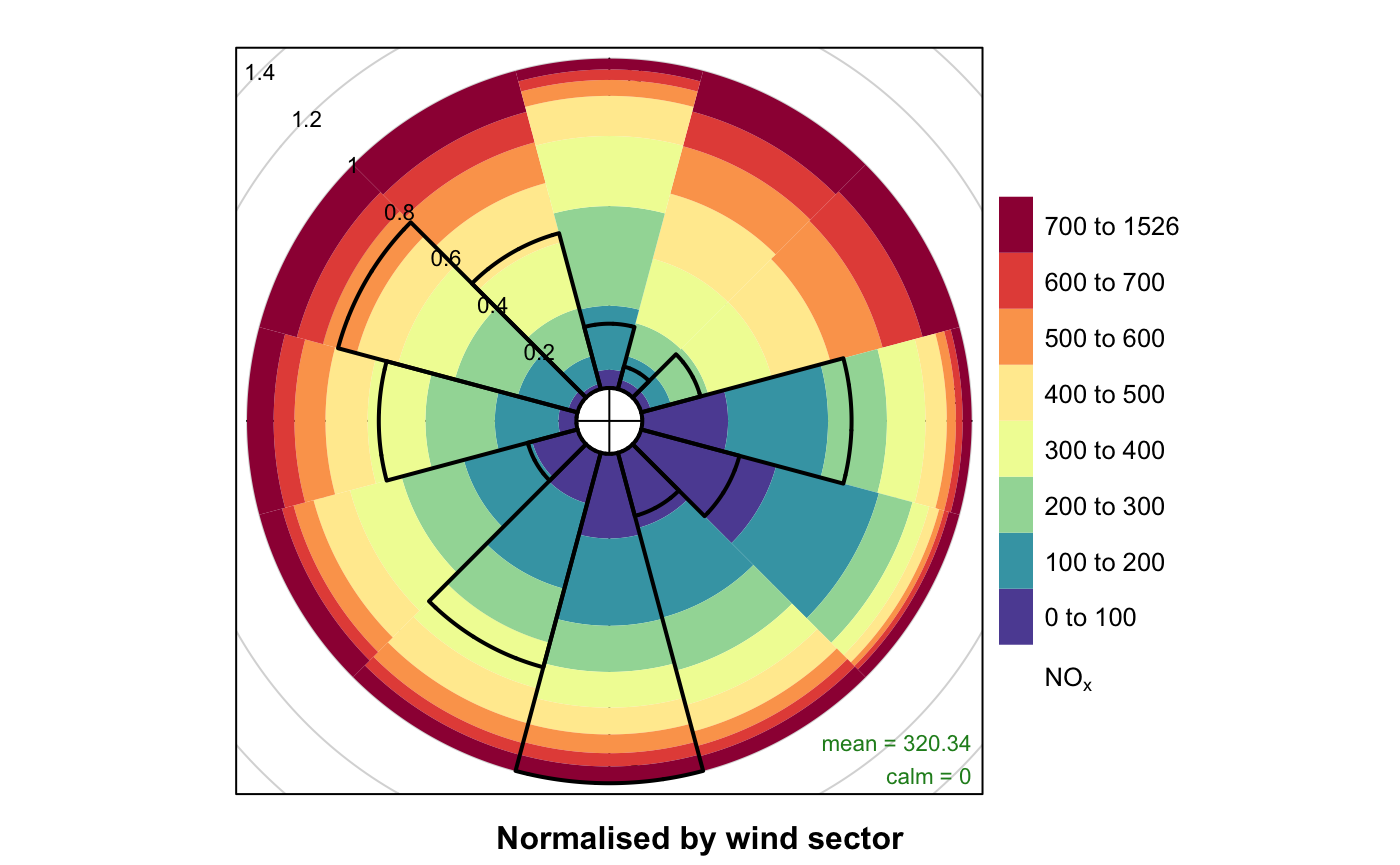
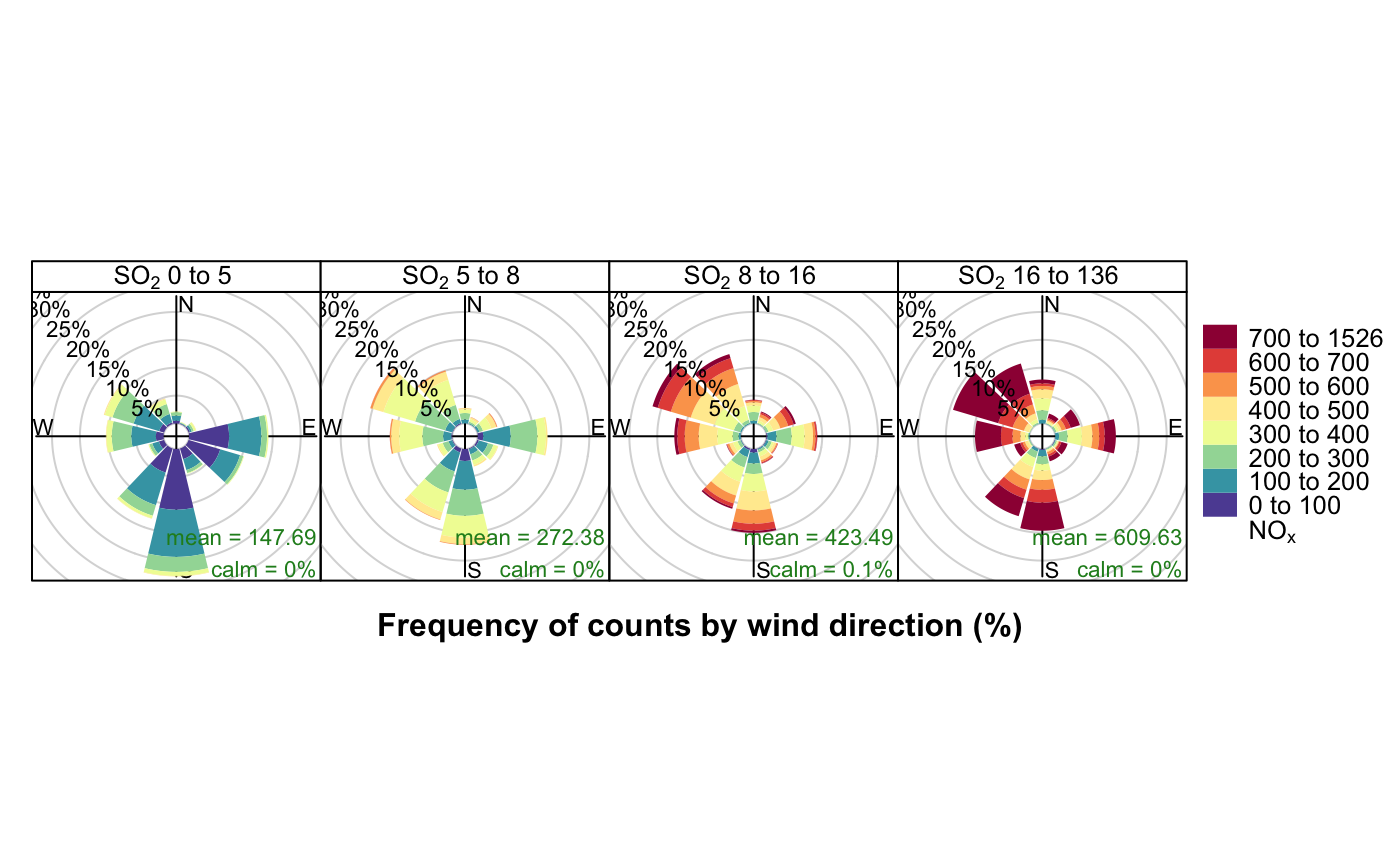
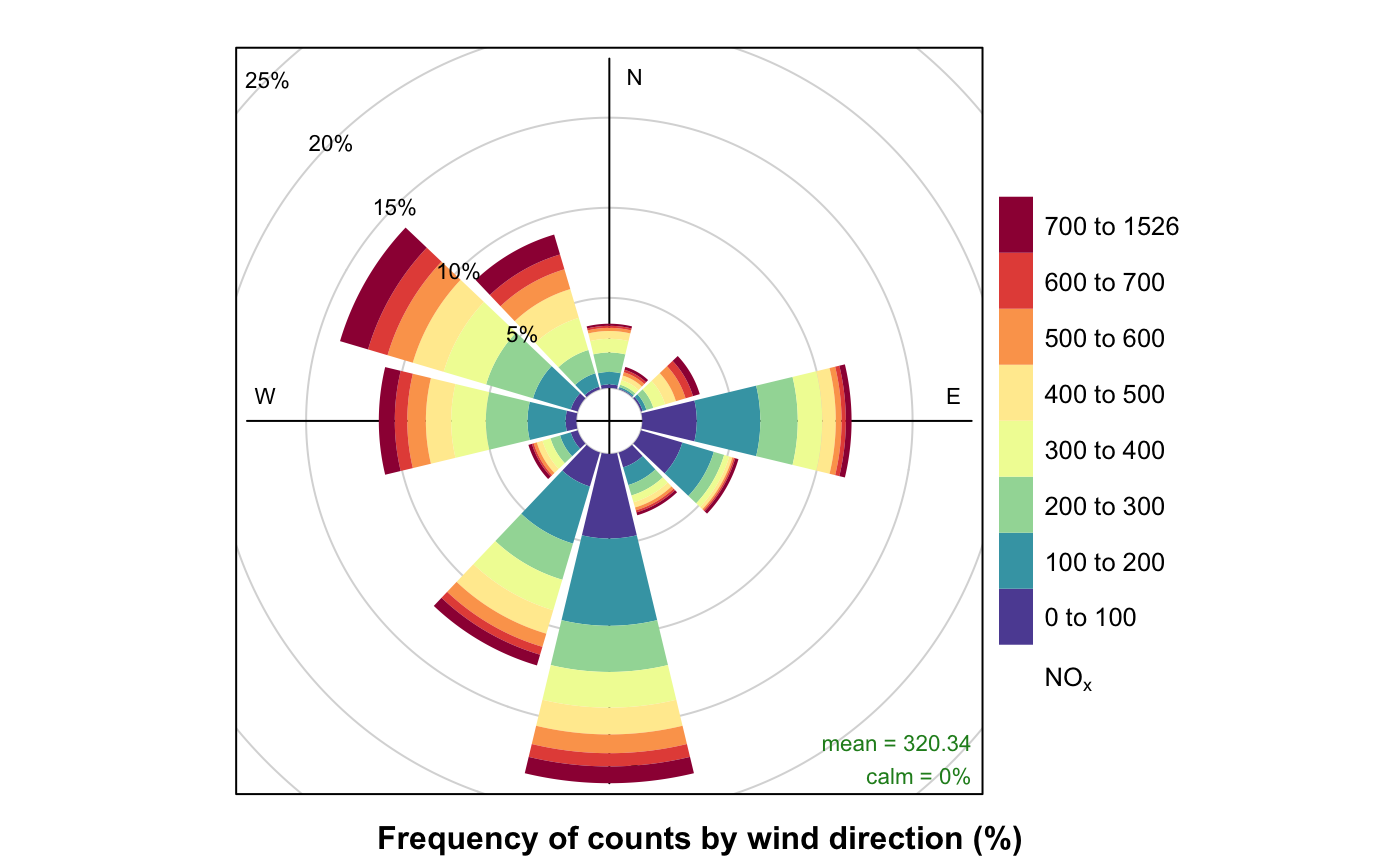
**Data Columns Required:**

* Wind Direction (wd)
* A Pollutant (e.g., pm10)
* Date/Time (date)

**Example Code:**

| # basic example for nox pollutionRose(dat, pollutant = "nox")  # link with s02 using type pollutionRose(dat, pollutant = "nox", type = "so2", layout = c(4, 1))  # segment and normalize # seg = 1 removes the spaces between the bars # normalize makes the length of all bars the same to see pollutant proportions more clearly pollutionRose(dat, pollutant = "nox", seg = 1, normalise = TRUE) |
| --- |

**Example Plot Output:**



Your Code:

Your Plot(s):

# POLAR FREQUENCIES

**Uses (and Sources):**

Summarizes wind speeds and directions by the number of hours a wind is in a certain speed and direction. Can also add pollution data.

(Polar Isopleths)

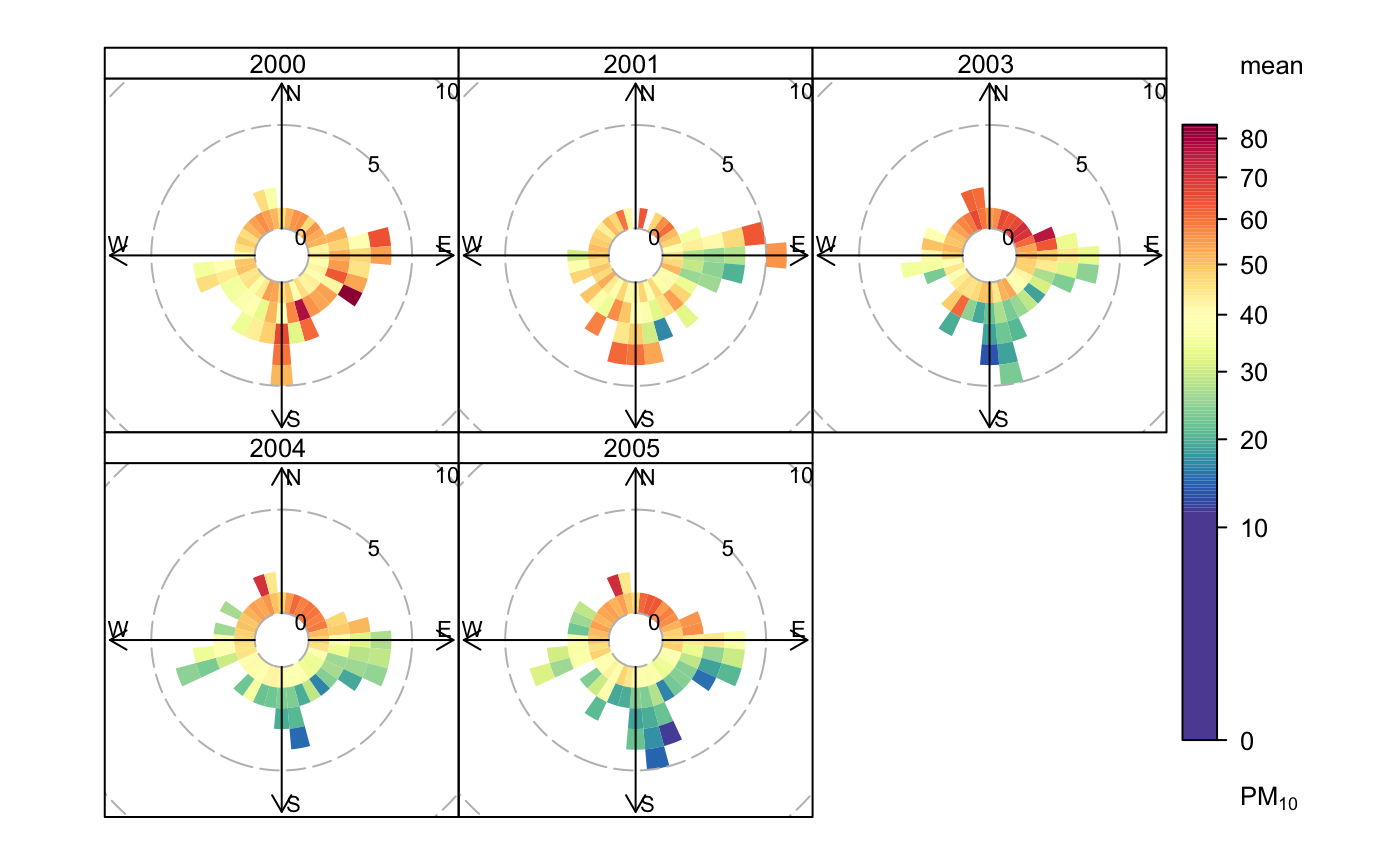
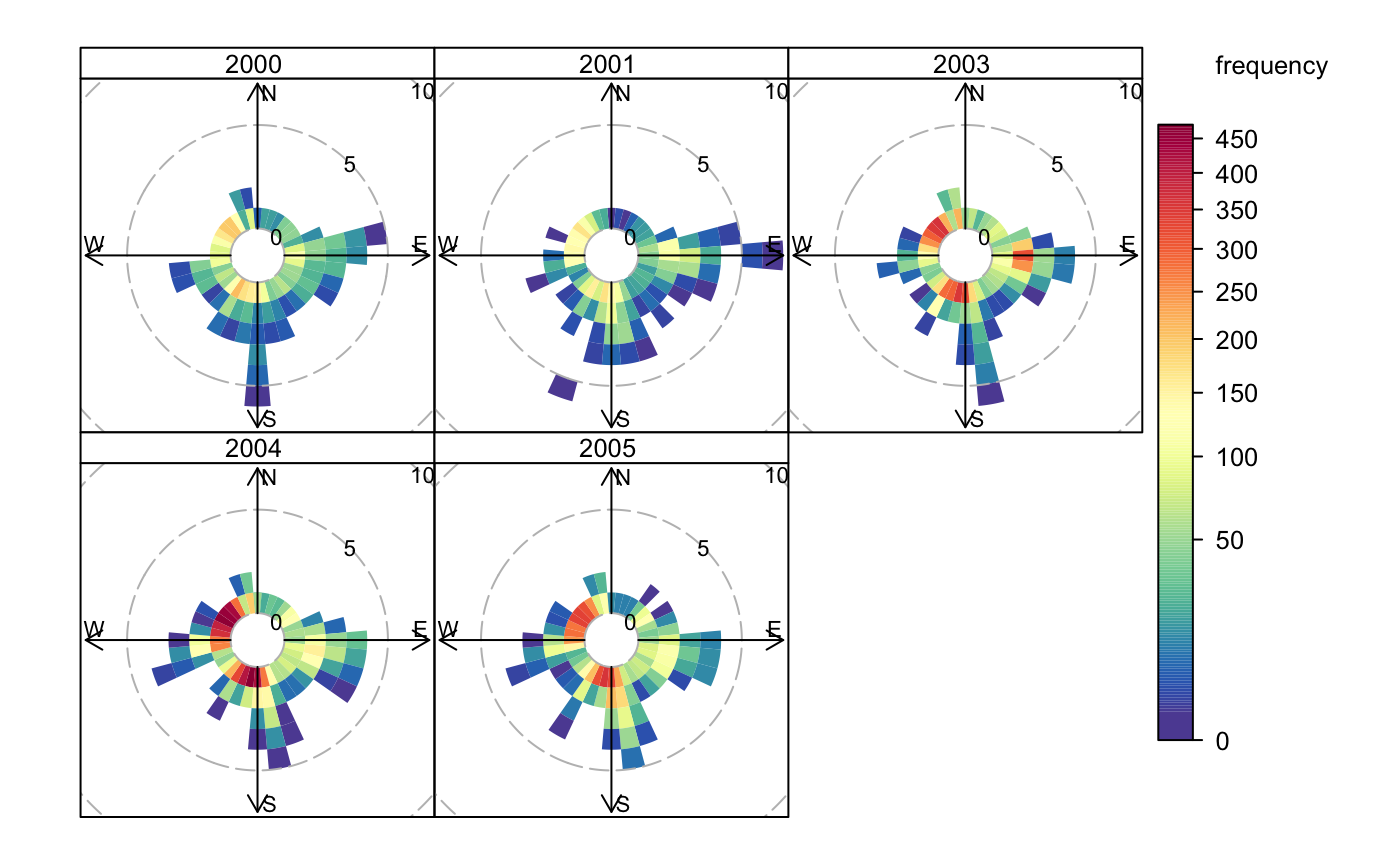
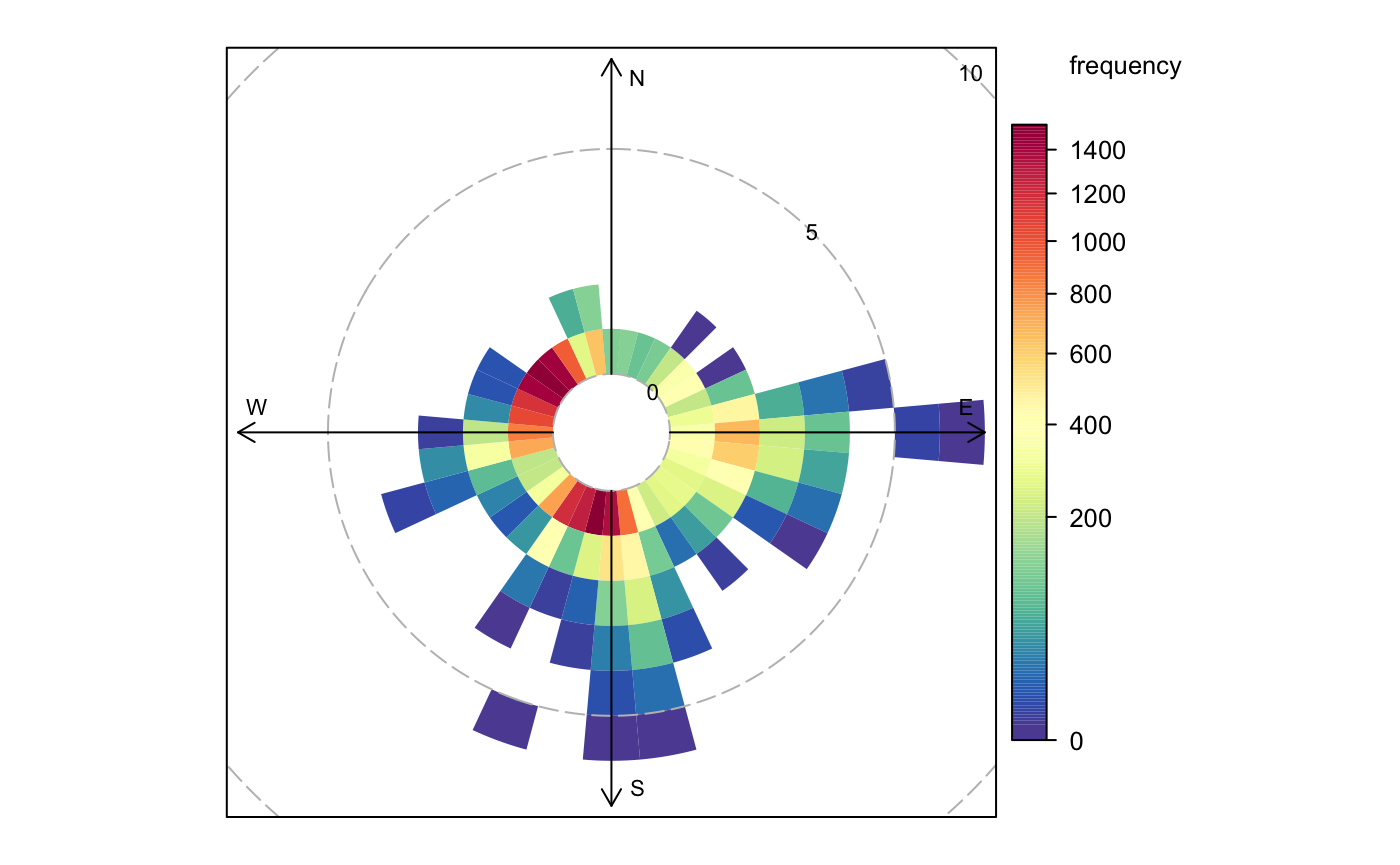
**Data Columns Required:**

* Wind Direction (wd)
* Wind Speed (ws)
* A Pollutant (e.g., pm10)
* Date/Time (date)

**Example Code:**

| # basic example, a more detailed wind description polarFreq(dat)  # bin by year polarFreq(dat, type = "year")  # add pollutant # we can see in detail how high concentrations came from SE in 2000, but moved north and decreased in frequency starting in 2003 polarFreq(dat, type = "year", pollutant = "pm10", statistic = "mean", min.bin = 2) |
| --- |

**Example Plot Output:**



Your Code:

Your Plot(s):

# 

# PERCENTILE ROSE

**Uses (and Sources):**

Summarizes pollutants tied to wind directions, most crucially shows the distribution of concentrations by direction.

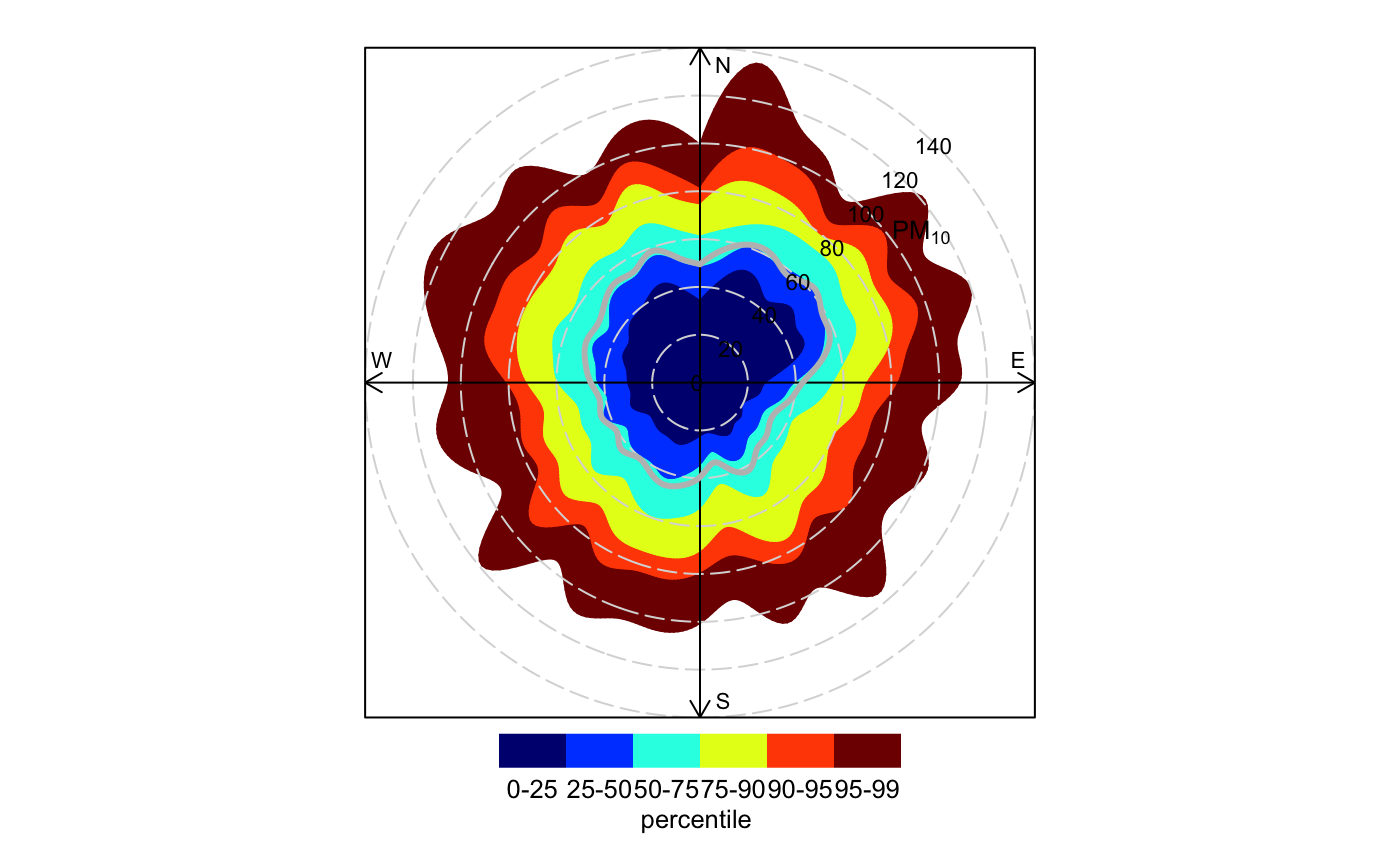
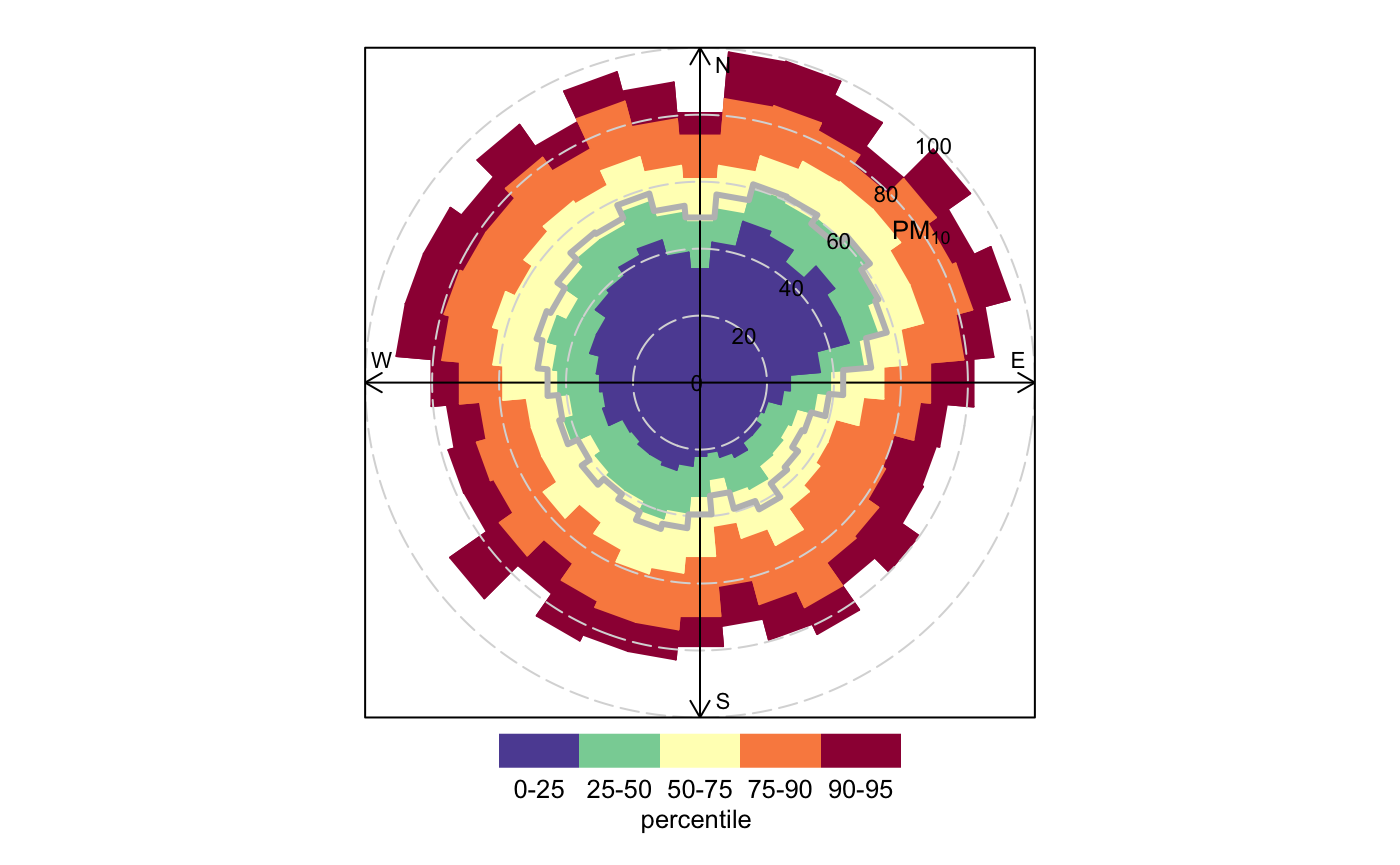
**Data Columns Required:**

* Wind Direction (wd)
* Wind Speed (ws)
* A Pollutant (e.g., pm10)
* Date/Time (date)

**Example Code:**

| #basic example, the concentrations of the pollutant are marked on the rings and the percentiles are shaded percentileRose(dat, pollutant = "pm10")  #can also customize the percentiles and smooth the plot #we see the pm10 is pretty even from all directions percentileRose(dat, pollutant = "pm10", percentile = c(25, 50, 75, 90, 95, 99), col = "jet", smooth = TRUE) |
| --- |

**Example Plot Output:**



Your Code:

Your Plot(s):

# POLAR PLOTS

**Uses (and Sources):**

Shows the concentrations of a pollutant modeled on a wind speed surface, which can then be plotted on a map with a little work. Also can create a ratio variable of two pollutants and plot that. Shows wind speed dependence. (Analysis of air quality in a street canyon…)

<https://www-sciencedirect-com.dartmouth.idm.oclc.org/science/article/pii/S1352231007006863?via%3Dihub>

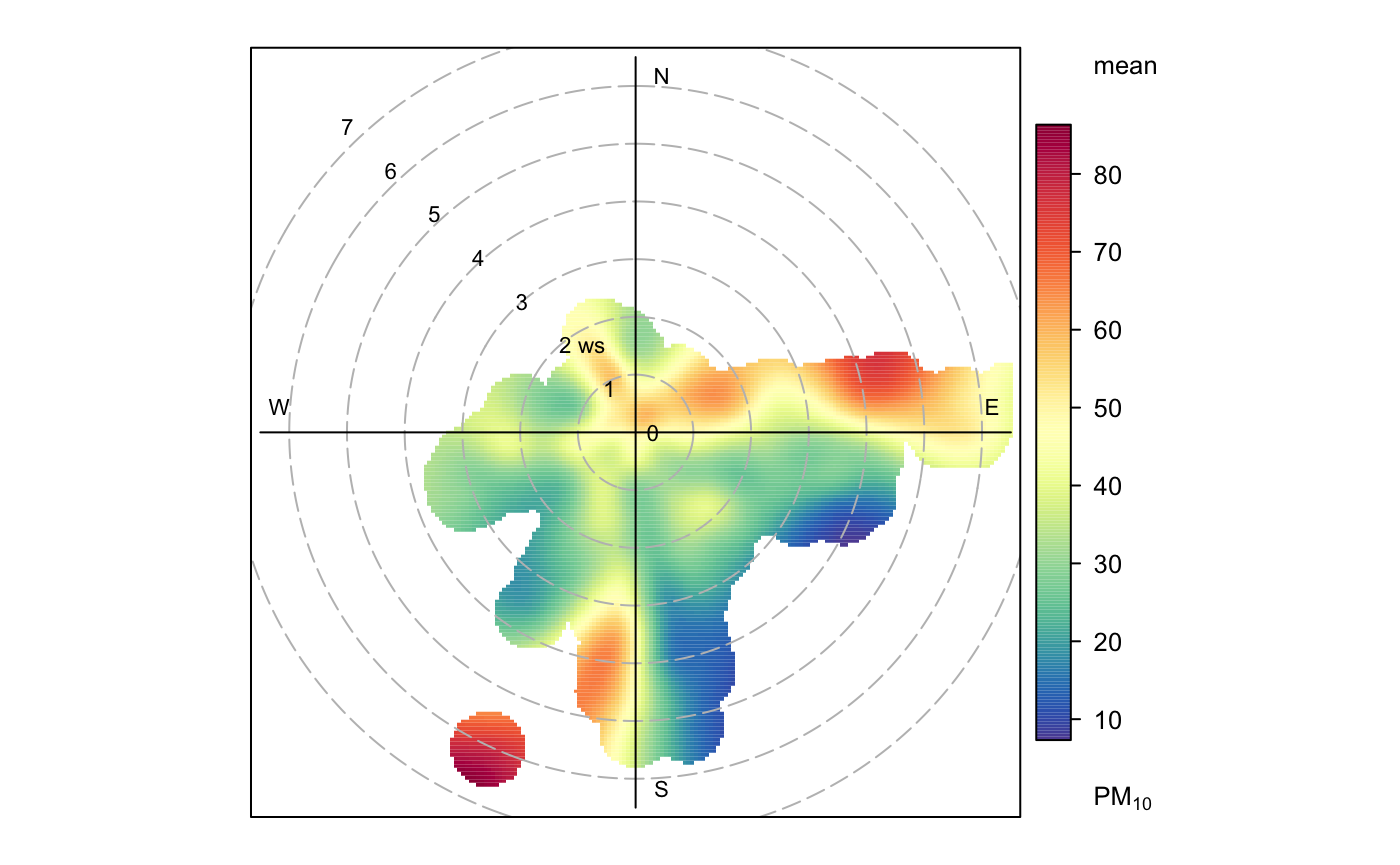
**Data Columns Required:**

* Wind Direction (wd)
* Wind Speed (ws)
* A Pollutant (e.g., pm10)
* Date/Time (date)

**Example Code:**

| # basic polar plot # so we see that there is only a high concentration from the SW when the wind speed is higher than usual polarPlot(dat, pollutant = "pm10") |
| --- |

**Example Plot Output:**



Your Code:

Your Plot(s):

# 

# POLAR ANNULUS

**Uses (and Sources):**

Plots an Annulus that shows changes in pollutant direction over time on different time scales.

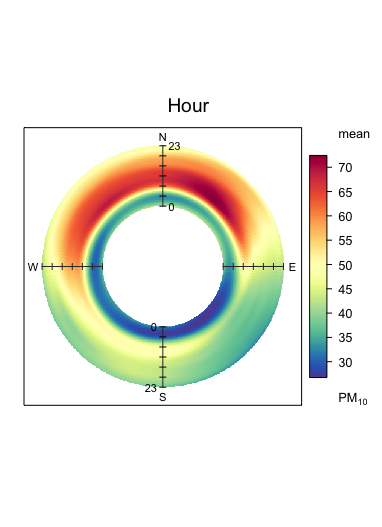
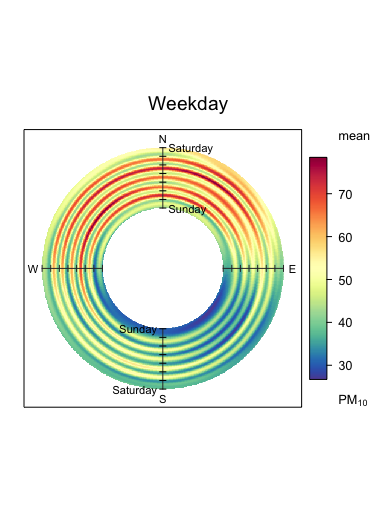
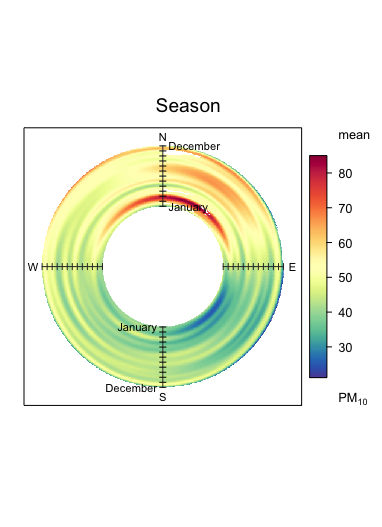
**Data Columns Required:**

* Wind Direction (wd)
* Wind Speed (ws)
* A Pollutant (e.g., pm10)
* Date/Time (date)

**Example Code:**

| polarAnnulus(dat, poll = "pm10", period = "season", main = "Season") polarAnnulus(dat, poll = "pm10", period = "weekday", main = "Weekday") polarAnnulus(dat, poll = "pm10", period = "hour", main = "Hour") |
| --- |

**Example Plot Output:**



Your Code:

Your Plot(s):

# 

# TIME SERIES PLOTS

**Uses (and Sources):**

Simply plots pollution level over time for the requested pollutants in the requested time frame. Averaging of values is also possible. Easily reveals data gaps if present.

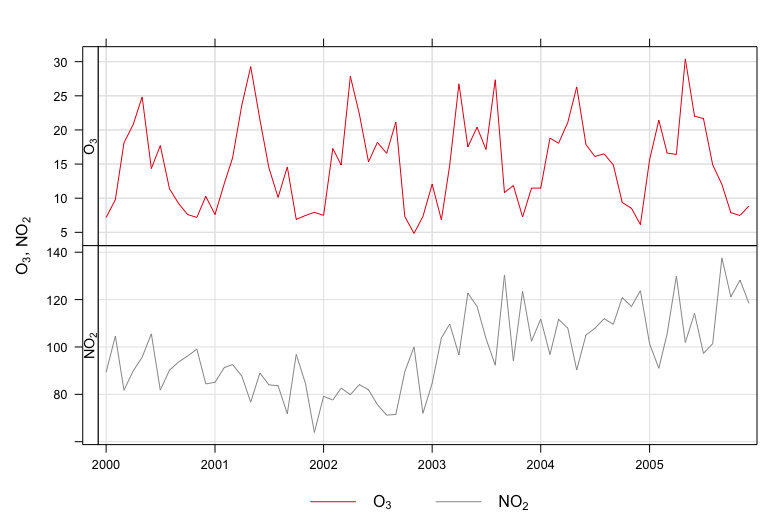
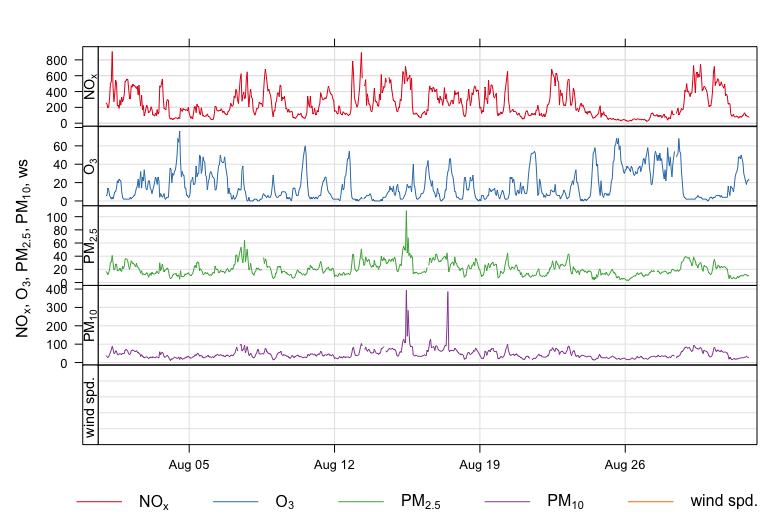
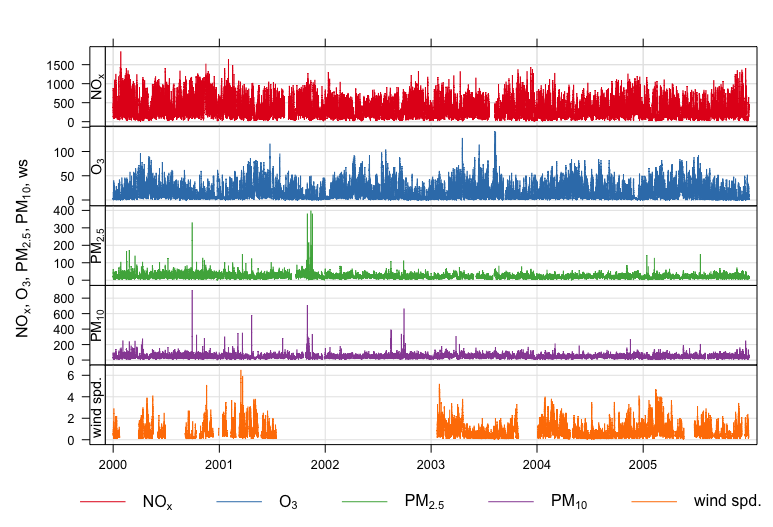
**Data Columns Required:**

* A Pollutant (e.g., pm10)
* Date/Time (date)

**Example Code:** Note the year and month for your data.

| # plot the values of pollutants over time timePlot(dat,  pollutant = c("nox", "o3", "pm2.5", "pm10", "ws"),  y.relation = "free")  # in a more specific time timePlot(selectByDate(dat, year = 2002, month = "aug"),  pollutant = c("nox", "o3", "pm2.5", "pm10", "ws"),  y.relation = "free")  # with averaging timePlot(dat, pollutant = c("o3", "no2"), avg.time = "month", y.relation = "free") |
| --- |

**Example Plot Output:**



Your Code:

Your Plot(s):

# 

# 

# 

# 

# 

# TEMPORAL VARIATION PLOTS

**Uses (and Sources):**

Plots pollution over time on several different scales, including a 95% CI of the mean (pink shading). Use in conjunction with PolarPlot to find a ws/wd of interest and focus on that.

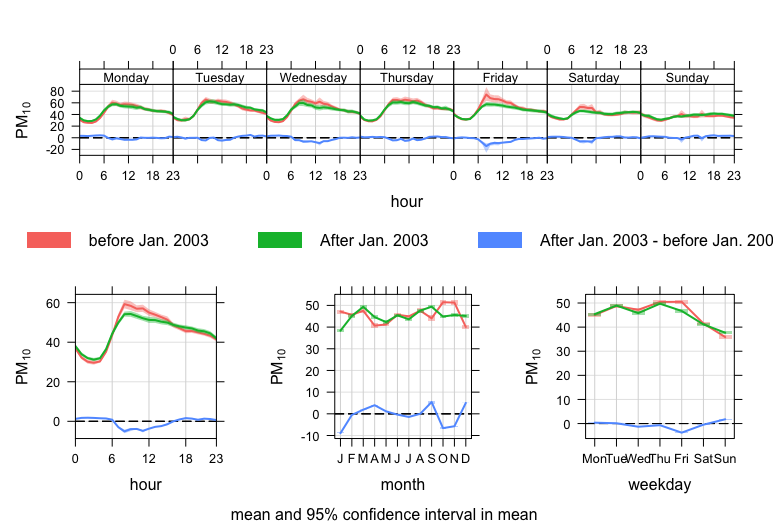
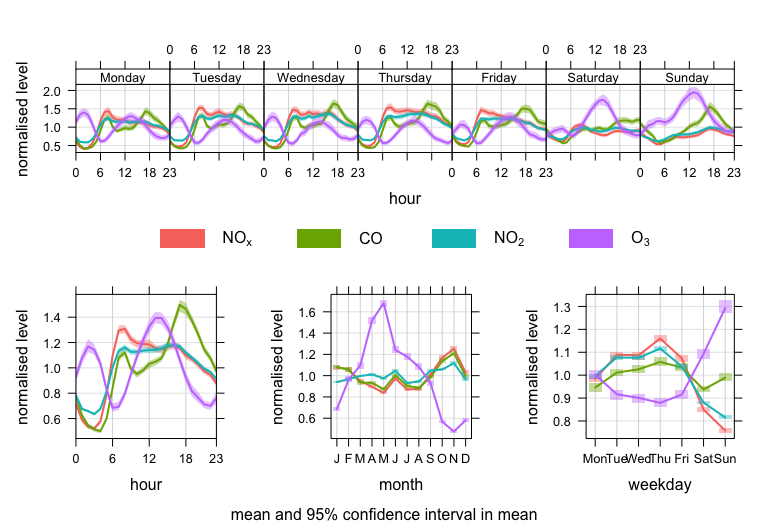
**Data Columns Required:**

* A Pollutant (e.g., pm10)
* Time/Date (date)

**Example Code:** Note the year and month for your data.

| # time var with normalization timeVariation(dat,   pollutant = c("nox", "co", "no2", "o3"),   normalise = TRUE)  # time var splitting by date dat1 <- splitByDate(dat, dates= "1/1/2003",  labels = c("before Jan. 2003", "After Jan. 2003")) timeVariation(dat1, pollutant = "pm10",   group = "split.by",   difference = TRUE)  # Can also make greater CIs or group by a new feature that you determine |
| --- |

**Example Plot Output:**



Your Code:

Your Plot(s):

# 

# 

# TIME PROP PLOTS

**Uses (and Sources):**

Shows the time series plots as stacked bar charts. Time series is averaged and broken down by the proportion contribution of another variable. Difficult to read with more than 3 or 4 contributions.

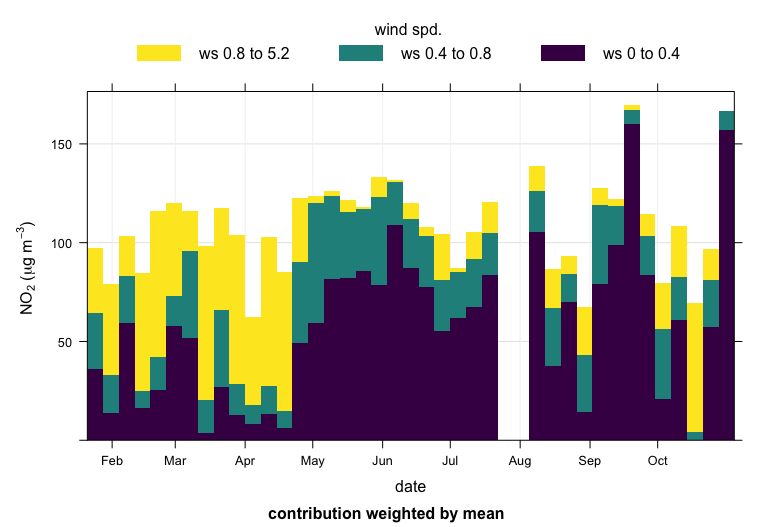
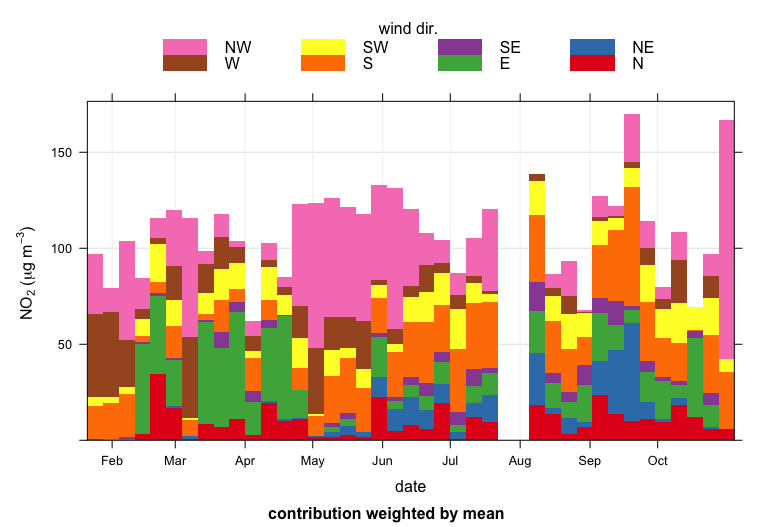
**Data Columns Required:**

* Pollutant (e.g., pm10)
* Date/Time (date)
* Wind Speed (ws)
* Wind Direction (wd)
* Another Factor (...)

**Example Code:** Note the year and month for your data.

| # break down so2 by proportion contribution from wind direction timeProp(selectByDate(dat, year = 2003),  pollutant = "no2", avg.time = "7 day",  proportion = "wd", date.breaks = 10, key.position = "top",  key.columns = 4, ylab = "no2 (ug/m3)")  # break down by wind speed instead timeProp(selectByDate(dat, year = 2003),  pollutant = "no2",  avg.time = "7 day",  n.levels = 3,  cols = "viridis",  proportion = "ws", date.breaks = 10,  key.position = "top", key.columns = 3,  ylab = "no2 (ug/m3)") |
| --- |

**Example Plot Output:**



Your Code:

Your Plot(s):

# 

# 

# TREND LEVEL HEAT MAP

**Uses (and Sources):**

Creates a heat map of a pollutant on a field of two variables x and y, often grouped by year. Defaults to x = month and y = hour. Possible to modify many other ways.

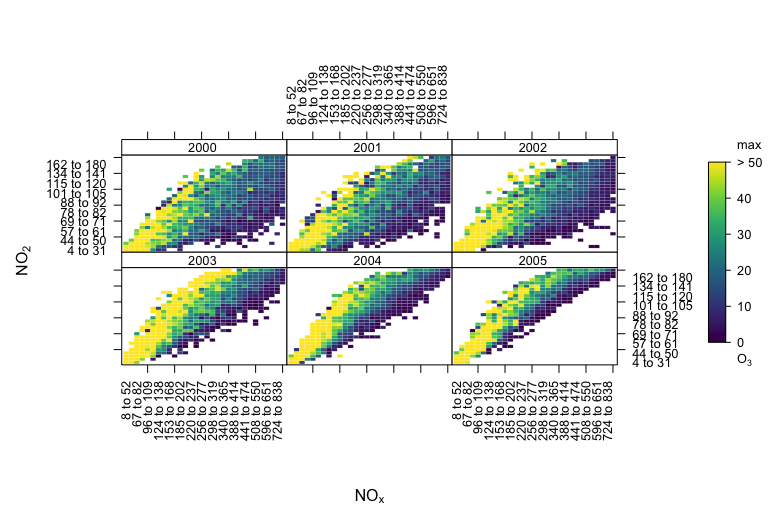
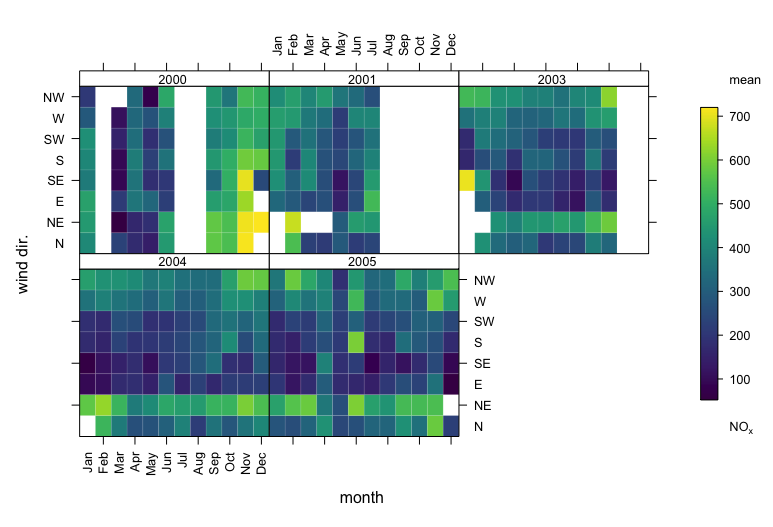
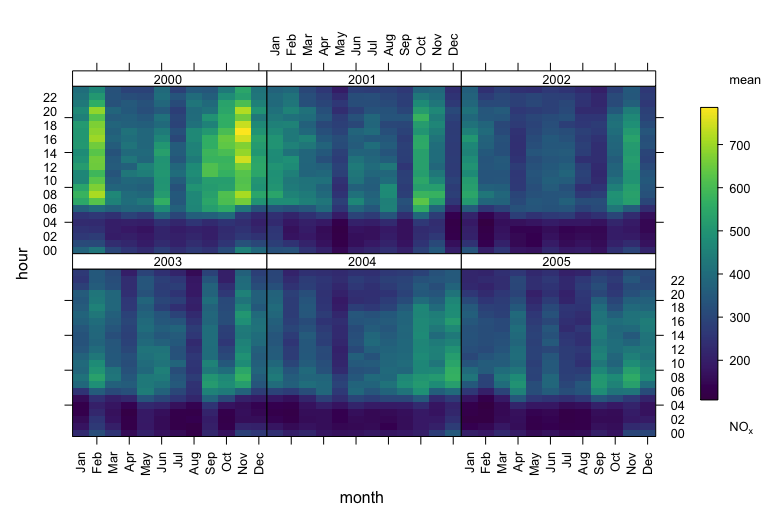
**Data Columns Required:**

* Date/Time (date)
* A Pollutant (e.g., pm10)
* Other factors
  + Ws
  + Wd
  + More pollutants

**Example Code:**

| # Trendlevel makes a heatmap by two variables x and y. By default it is month x and hour y trendLevel(dat, pollutant = "nox", cols = "viridis")  # With wind direction as y trendLevel(dat, pollutant = "nox", y = "wd",   border = "white",   cols = "viridis")  # or with one pollutant as x and another as y trendLevel(dat, x = "nox", y = "no2", pollutant = "o3",   border = "white", cols = "viridis",  n.levels = 30, statistic = "max",   limits = c(0, 50)) |
| --- |

**Example Plot Output:**



Your Code:

Your Plot(s):

# 

# 

# CALENDAR PLOT

**Uses (and Sources):**

Plots the pollutant level directly onto a calendar for clear temporal viewing of peak days.

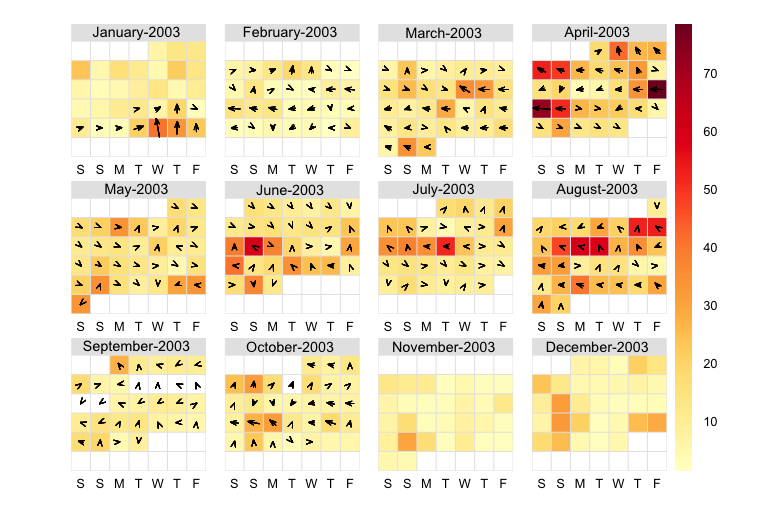
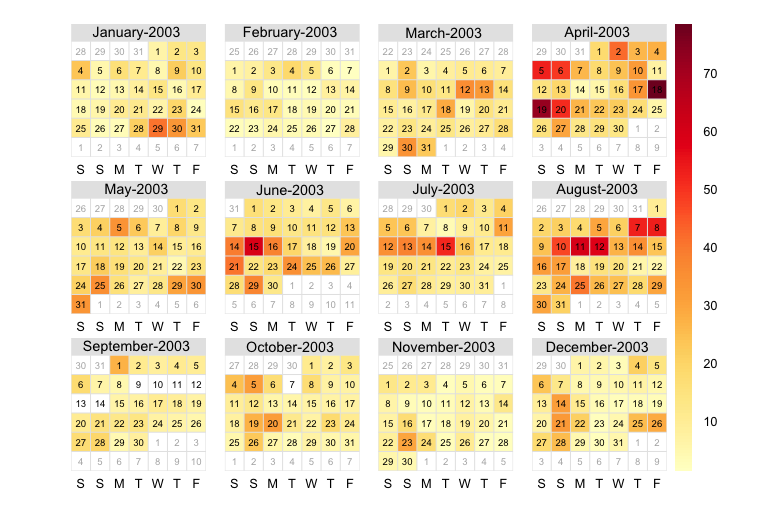
**Data Columns Required:**

* Pollutant (e.g., pm10)
* Time/Date (date)
* Other factors
  + Ws
  + Wd
  + Other pollutant

**Example Code:** Note the year and month for your data.

| # basic calendar plot calendarPlot(dat, pollutant = "o3", year = 2003)  # annotate with the wind direction calendarPlot(dat, pollutant = "o3", year = 2003, annotate = "ws")  # can add averaging functions or specific binning fxns |
| --- |

**Example Plot Output:**



Your Code:

Your Plot(s):

# 

# 

# THEIL-SEN TRENDS

**Uses (and Sources):**

A different type of regression averaging: slope is taken between every pair of two points. This is resistant to non-normal data with a non-constant variance, and to outliers. Solid line is Theil-Sen Estimate and Dashed Lines are 95% CI.

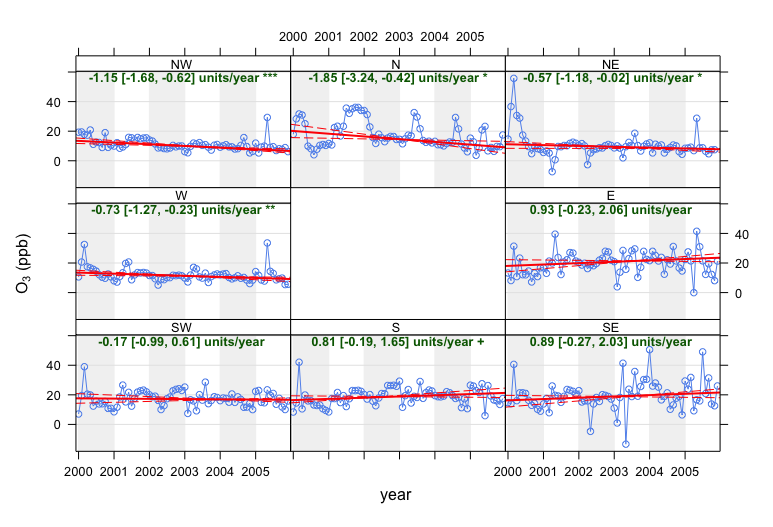
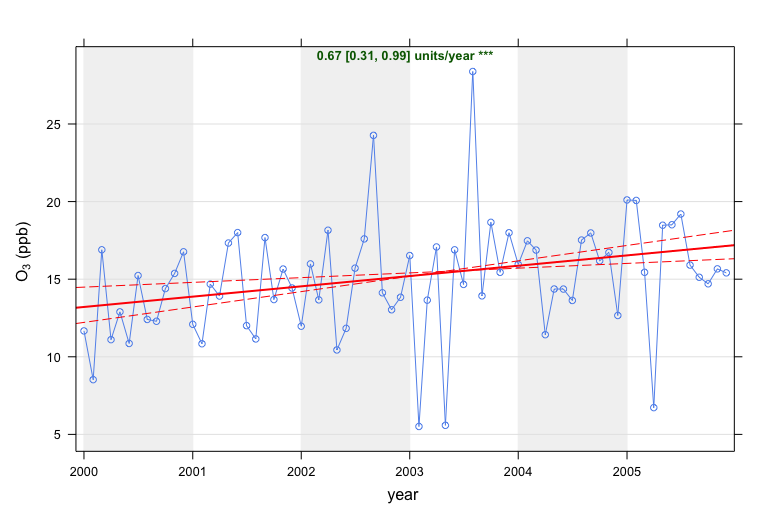
**Data Columns Required:**

* Pollutant (e.g., pm10)
* Time/Date (date)
* Other factors
  + Ws
  + Wd
  + Other pollutant

**Example Code:**

| # basic t\_s TheilSen(dat, pollutant = "o3",   ylab = "ozone (ppb)",   deseason = TRUE,  date.format = "%Y")  # can be typed by wd TheilSen(dat, pollutant = "o3", type = "wd",   deseason = TRUE,  date.format = "%Y",  ylab = "ozone (ppb)") |
| --- |

**Example Plot Output:**



Your Code:

Your Plot(s):

# 

# 

# SMOOTH TREND

**Uses (and Sources):**

Another trend type with 95% CI. It is often optimized to include important data but also be quite smooth. This won't necessarily be a straight line! Data can be deseasonalized to remove large impacts of monthly averages.

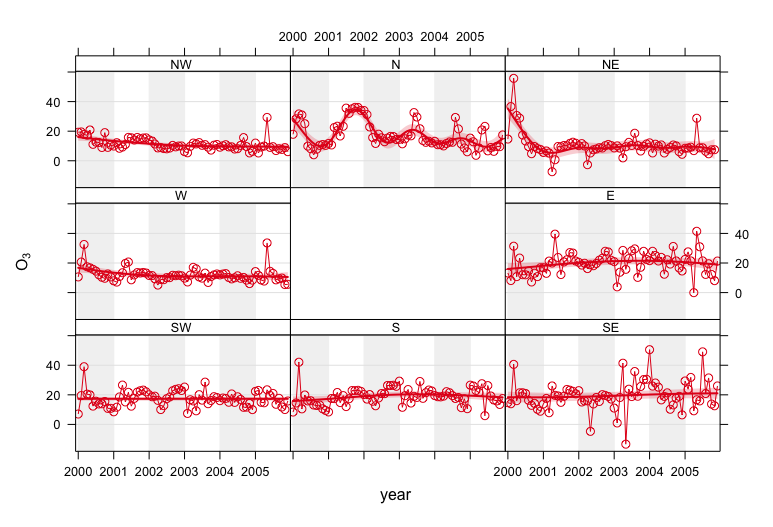
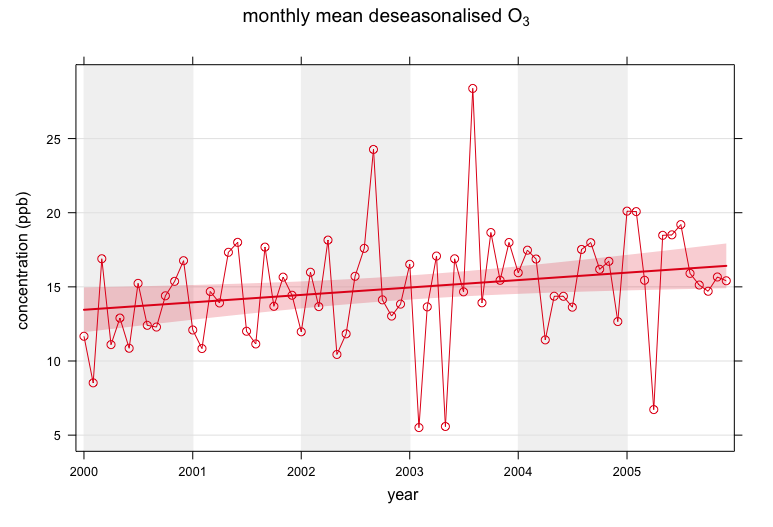
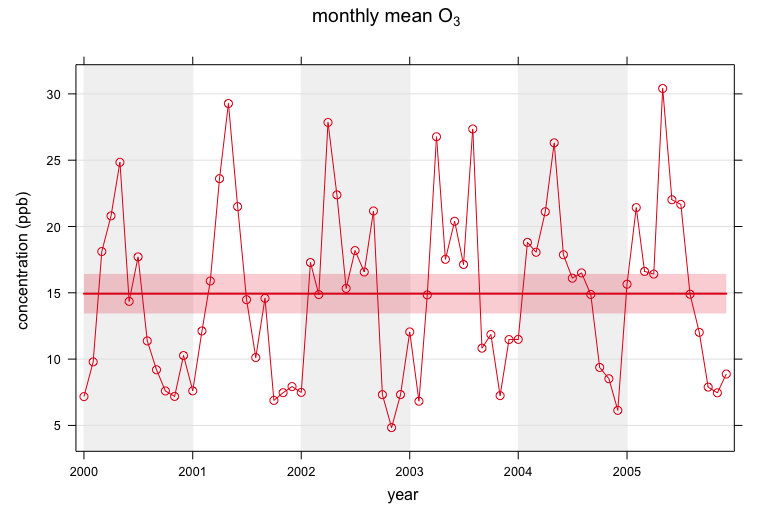
**Data Columns Required:**

* Pollutant (e.g., pm10)
* Time/Date (date)
* Other factors
  + Ws
  + Wd
  + Other pollutant

**Code Example:**

| # smooth trend that might not be a straight line smoothTrend(dat, pollutant = "o3", ylab = "concentration (ppb)",  main = "monthly mean o3")  # deseasonalized (removing the effects of the seasonal cycle) smoothTrend(dat, pollutant = "o3", deseason = TRUE, ylab = "concentration (ppb)",  main = "monthly mean deseasonalised o3")  # with type by wind direction smoothTrend(dat, pollutant = "o3", deseason = TRUE,  type = "wd") |
| --- |

**Plot Output:**



Your Code:

Your Plot(s):

# 

# 

# SCATTER PLOTS

**Uses (and Sources):**

A basic data visualization tool for comparing two continuous variables. Can be made more complex through binning, formatting, and adding/making trends, etc.

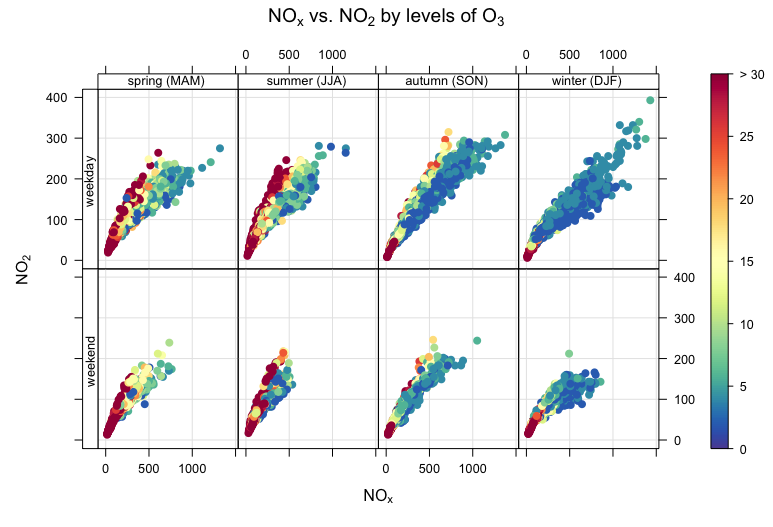
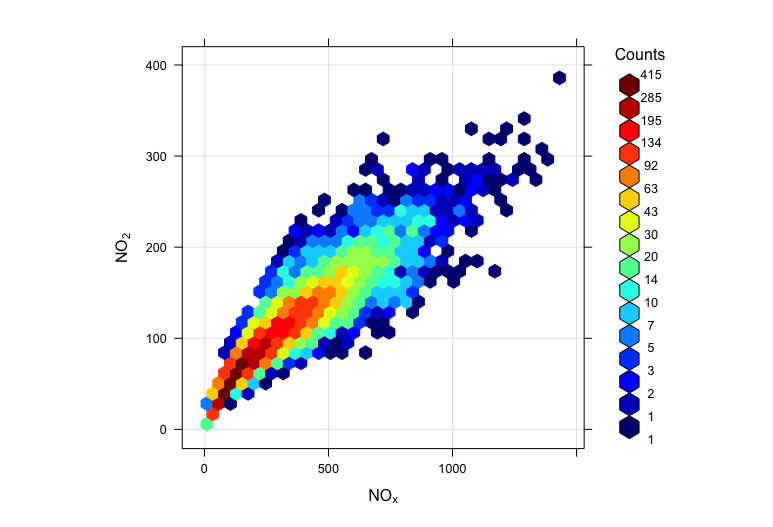
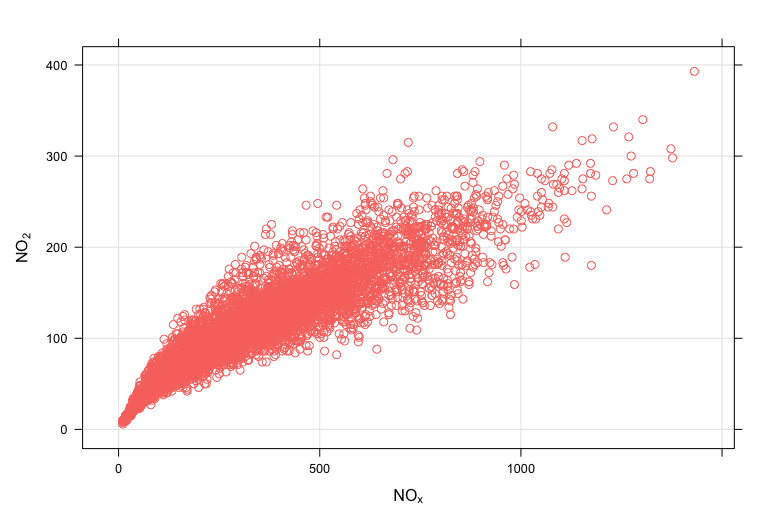
**Data Columns Required:**

* 2 Pollutants (e.g, no2 and nox)
* Other factors

**Code Example:** Note the year and month for your data.

| # creating some basic and higher level scatter plots # this is basic R scatter plot, looks nicer in ggplot2 data2003 <- selectByDate(dat, year = 2003) scatterPlot(data2003, x = "nox", y = "no2")  # hex binning and shading scatterPlot(data2003, x = "nox", y = "no2", method = "hexbin", col= "jet")  # further specs by density, linear fit, and grouping # can also add z variable which is heatmapped  # grouping example. o3 is in shading, x and y are nox and no2, and there are charts for weekend/weekday and also season scatterPlot(data2003,   x = "nox", y = "no2", z = "o3",   type = c("season", "weekend"),  limits = c(0, 30)) |
| --- |

**Plot Output:**



Your Code:

Your Plot(s):

# 

# FIGURE EXPORTING

**Uses:** As you know, you can simply use the copy/paste functions or “save as” buttons to copy a plot over to your document or presentation for use. But what if you want more control than that? Sometimes the viewer window isn’t the right dimensions for your plot and we want custom dimensions.

**Data Required:**

* Just a plot!

**Code Example:**

# Lets say we have a scatter plot we’ve just made

scatterPlot(data2003, x = "nox", y = "no2", method = "hexbin", col= "jet");

# We can run this command alone to get the figure, but we can also save the figure to an object like so:

fig1 <- scatterPlot(data2003, x = "nox", y = "no2", method = "hexbin", col= "jet");

fig1 # this won’t produce a figure until we call the object like this, note that there’s no semicolon at the end to suppress the output

# MAKING A PNG ####

# we can use commands like png(), jpeg(), etc. to export a figure as desired:

png(file = "filename", width = 6, height = 4, units = "in", res = 650); #sets up what you want to create, will save to working directory

fig1 #then produce the figure, either from scratch here (histogram(...)) or by typing the name of a figure you have saved (fig1 <- histogram(...))

dev.off(); #turns off the viewer so you don’t overwrite your plot

# NOTE: this will save your file to wherever your working directory is, so make sure you know where that is with getwd() or set it to what you want with setwd()