

# openair News

The **openair** Project newsletter

Issue 11, January 2012

## Some recent updates to **openair**

While the **openair** project has officially finished as far as NERC funding goes, it continues to be developed and refined. Further funding is being sought to help support its continued development.

This issue considers some recent refinements to the **timeVariation** function that provides more robust and consistent estimates of uncertainties and also a new approach for considering differences between two pollutants. The latter change is particularly flexible and should be useful to many users in a wide range of circumstances. A few examples are given in the newsletter.

The much-used **polarPlot** function has been made much more flexible by allowing *any* numeric variable to be plotted in polar coordinates with

wind direction. Previously only wind speed could be used, but this was a somewhat artificial restriction. The key issue is whether the other numeric variable provides some sort of source discrimination and examples are given of using ambient temperature.

There continue to be many other changes to **openair** and these are listed in this newsletter. Finally, the **openair** manual continues to be improved, which hopefully will be appreciated by many users.

This newsletter was produced using R version 2.14.1 and **openair** version 0.5-16.

<mailto:david.carslaw@kcl.ac.uk>

### Contents

Some recent updates to <b>openair</b> . . . . .	1
Refinements to <b>timeVariation</b> . . . . .	2

More flexibility for <b>polarPlot</b> . . . . .	4
Trends and averages by season . . . . .	6
Other <b>openair</b> developments . . . . .	7

# Refinements to `timeVariation`

Feedback from a wide range of users suggests that the `timeVariation` function is one of the more frequently used functions in `openair`. The function itself tries to do something straightforward: show the important temporal variations in air pollutant concentrations by hour of the day, day of the week and month of the year. The function also estimates the 95% confidence interval in the mean and shows the uncertainties as shaded regions.

While the idea of `timeVariation` is simple enough, the usefulness of the function is probably related to its flexibility. For example, it is possible to consider multiple pollutants, to normalise the data and compare one period of time against another. Also, the flexible ‘type’ option in `openair` makes it easy to carry out analysis in a very wide variety of ways.

Two significant changes have been made to this function. First, bootstrap resampling methods are now used to estimate the uncertainties in the mean. This change should provide more robust estimates of the uncertainties particularly when there are relatively few measurements. The second major change should prove to be very useful: it is now possible to calculate differences between two time series (and the bootstrap uncertainties of the differences in the mean). The difference calculations are possible when two time series are considered.

There should be many situations where considering differences will be useful; here are two, but there are many others:

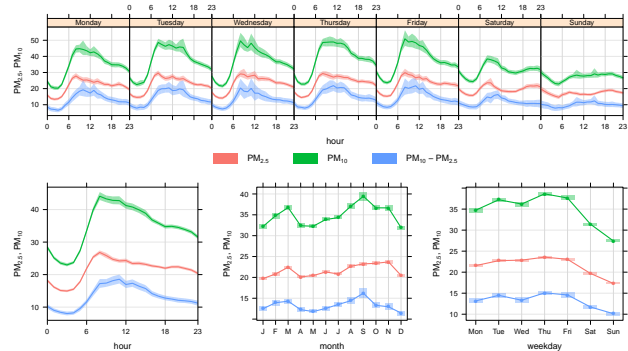
- Comparing two periods in time to see how the temporal nature of the concentrations have changed. These calculations will help reveal the factors controlling a change. For example, is most of an increase in the concentration of a species observed only for weekdays, or only certain hours of the day? Such information can help identify important factors that cause a change.
- For model evaluation it is useful to compare measured against modelled temporal variations to see how well a model agrees with measurements. However, what is the nature of the agreement — is it good for all hours of the day and all seasons, and if not, why not?

The difference calculation can be made in two ways. First, two pollutants can be selected from a

data set (data frame). Second, a grouping variable that identifies exactly two groups can be used. In both cases the second pollutant is subtracted from the first.

In the first example, the coarse fraction of  $PM_{10}$  is calculated as the difference between  $PM_{10}$  and  $PM_{2.5}$  concentrations. We tell the function to show the differences by choosing `difference = TRUE`. Note that the function automatically adds the difference as a new variable ‘ $PM_{10} - PM_{2.5}$ ’.

```
timeVariation(mydata, pollutant = c("pm25",
                                   "pm10"), difference = TRUE)
```



**Figure 1:** Use of the `difference` option in `timeVariation` to show the PM coarse fraction.

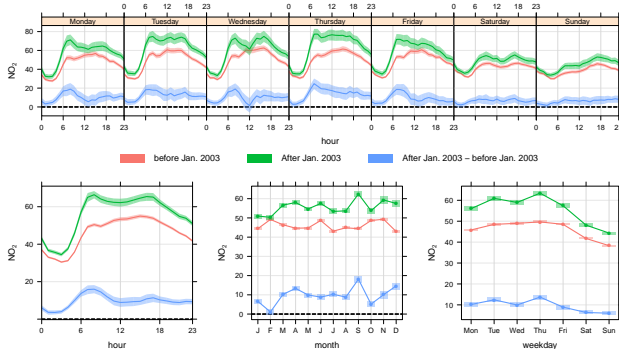
A very powerful and under-used analysis is to compare the temporal components of two different time periods. We use the `splitByDate` function to divide up the data into dates before January 2003 and after January 2003. This time the option `difference` is used to highlight how  $NO_2$  concentrations have changed over these two periods. Note that `splitByDate` returns a column `split.by` by default, which can then be referred to in the `timeVariation` function using the option `group`. This way of using the function therefore differs from the previous example where two pollutants were explicitly named. However, there are still only two unique values for the variable `split.by`: before Jan. 2003 and after Jan. 2003.

There is some indication in this plot that data after 2003 seem to show more of a double peak in the diurnal plots; particularly in the morning rush hour. Also, the difference line does more clearly highlight a more substantial change over weekdays and weekends. Given that cars are approximately constant at this site each day, the change may in-

dicating a change in vehicle emissions from other vehicle types. Given that it is known that primary NO<sub>2</sub> emissions are known to have increased sharply from the beginning of 2003 onwards, this perhaps provides clues as to the principal cause i.e. not heavy goods vehicles where there are far fewer of the at weekends.

```
## split data into two periods
mydata <- splitByDate(mydata,
  dates = "1/1/2003", labels = c("before Jan.
2003",
  "After Jan. 2003"))

timeVariation(mydata, pollutant = "no2",
  group = "split.by", difference = TRUE)
```



**Figure 2:** Example plot using the `timeVariation` function to plot NO<sub>2</sub> concentrations at Marylebone Road. In this plot, the concentrations are shown before and after January 2003.

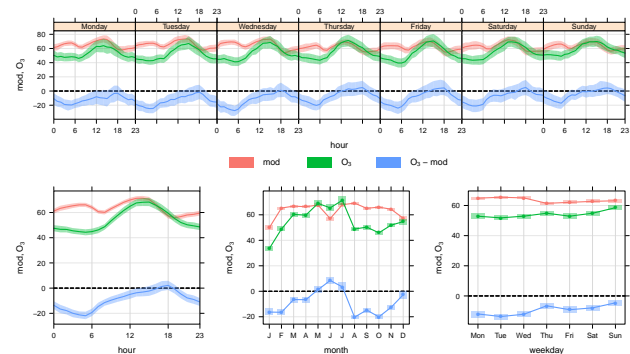
In the next example we will have a look at a model evaluation setting. Results of O<sub>3</sub> predictions from the CMAQ model at Harwell will be considered. These results were kindly made available from some CMAQ test modelling by Sean Beevers and Nutthida Kitwiroon. First the data are loaded:

```
load("~/openair/Data/CMAQozone.RData")
head(CMAQ.KCL)

##      site      date o3
## 1 Aston.Hill 2006-01-01 00:00:00 NA
## 2 Aston.Hill 2006-01-01 01:00:00 74
## 3 Aston.Hill 2006-01-01 02:00:00 72
## 4 Aston.Hill 2006-01-01 03:00:00 72
## 5 Aston.Hill 2006-01-01 04:00:00 70
## 6 Aston.Hill 2006-01-01 05:00:00 66
##      rollingO3Meas  mod rollingO3Mod  group
## 1              NA 92.80              NA CMAQ.KCL
## 2              NA 92.18              NA CMAQ.KCL
## 3              NA 92.14              NA CMAQ.KCL
## 4              NA 91.72              NA CMAQ.KCL
## 5              NA 91.50              NA CMAQ.KCL
## 6              NA 92.28              NA CMAQ.KCL
```

Second, we will only consider the Harwell site and hence use the `subset` command to select it. In this example it can be seen that when `difference = TRUE` a dashed line at  $y = 0$  is shown, which helps to highlight the difference. In this case it can be seen that the model generally over-estimates O<sub>3</sub> concentrations, except late in the day around 6pm and through the summer months. Such differences can help focus on why the model may not be capturing the temporal variations.

```
timeVariation(subset(CMAQ.KCL,
  site == "Harwell"), pollutant = c("mod",
  "o3"), difference = TRUE)
```



**Figure 3:** Example plot using the `timeVariation` function to compare observed and modelled O<sub>3</sub> concentrations using the KCL CMAQ model for Harwell.

# More flexibility for polarPlot

The `openair polarPlot` function is another function that is widely used for source identification and characterisation. While it is already flexible e.g. it can:

- Consider multiple pollutants.
- Can be used with the flexible 'type' option to show plots by season, year, any other numeric variable ...
- Uncertainties can be considered.

Despite the flexibility the function to date only plotted wind speed against wind direction in polar coordinates. The reason why wind speed was used is that it is excellent at discriminating different pollutant sources. Even though considering the wind speed dependence of a source is a simple thing to do; it is very useful. It can for example reveal different source types. Emissions from tall chimney stacks tend to ground during higher wind speed conditions, street canyon recirculation can be revealed in more detail etc.

But equally there are potentially many other numeric variables that could be plotted against wind direction in polar coordinates that could also help discriminate different source types. For this reason, `polarPlot` can now consider any numeric variable through the option `x`. The extent to which this is useful clearly depends on the data available, but in a research setting where numerous species are measured this increased flexibility could be very useful.

As an example, we will consider temperature as a variable to use with wind direction. There are many situations where atmospheric temperature could help separate-out different source types. For example, the emission rate of a species itself may depend on temperature. Increased temperatures will mean increased thermal turbulence in the atmosphere, so it is possible high-level sources could be brought down to ground level.

To illustrate some different situations where temperature might be useful, we consider three different species: SO<sub>2</sub>, ethane and isoprene. The data are stored on a KCL server, but can be loaded directly:

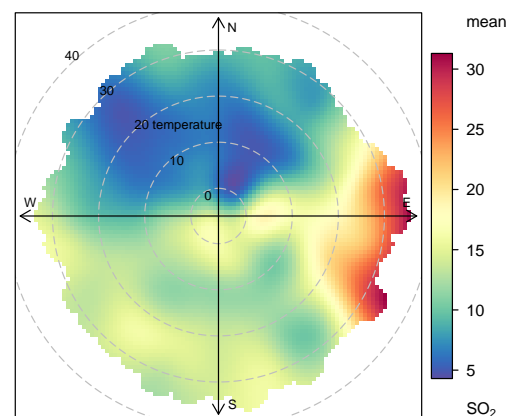
```
load(url("http://www.erg.kcl.ac.uk/downloads/Policy_Reports/
head(thedata)
```

```
##           date so2 nox isoprene
## 1 2000-01-01 00:00:00 16 388      1.22
## 2 2000-01-01 01:00:00 35 886      0.85
## 3 2000-01-01 02:00:00 32 816      1.75
## 4 2000-01-01 03:00:00 24 636      4.36
## 5 2000-01-01 04:00:00 19 483      3.08
## 6 2000-01-01 05:00:00 11 231      2.12
##  ethane benzene  ws  wd temp
## 1  13.54   7.74 2.058 200  7.4
## 2  11.99   5.66 2.058 190  7.5
## 3  14.69  11.38 2.058 190  7.9
## 4  20.38  30.75 1.543 240  8.1
## 5  18.75  19.66 1.029 240  8.2
## 6  16.38  13.13 2.058 280  8.7
```

Note the full path is "[http://www.erg.kcl.ac.uk/downloads/Policy\\_Reports/AQdata/hcData.RData](http://www.erg.kcl.ac.uk/downloads/Policy_Reports/AQdata/hcData.RData)".

First, consider concentrations of SO<sub>2</sub> shown below. This plot shows very clearly that concentrations of SO<sub>2</sub> are highest when the wind is from the east and the temperature is high (greater than ≈25 degrees C). This is a result of plumes from chimney stacks about 25 km to the east being brought down to ground level due to high thermally-induced mixing i.e. unstable atmospheric conditions.

```
polarPlot(thedata, pollutant = "so2",
x = "temp", min.bin = 3)
```



**Figure 1:** Example plot using the `polarPlot` function to consider SO<sub>2</sub> concentrations as a function of wind direction and temperature.

Indeed, the characteristics of SO<sub>2</sub> can be further explored by considering the data by season and plot-

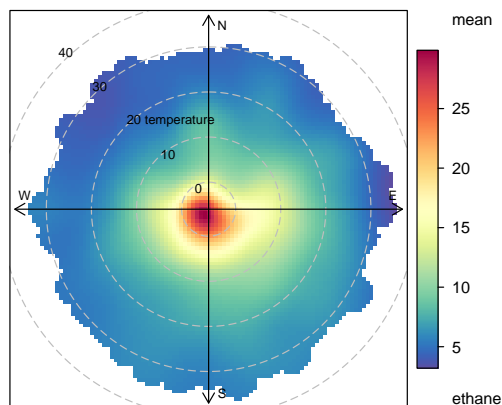
ting the maximum SO<sub>2</sub> concentration — not shown, but plotted by:

```
polarPlot(thedata, pollu = "so2",
          x = "temp", min.bin = 3, type = "season",
          stat = "max")
```

In this case the higher concentrations at high temperatures are still seen in summer, but there is also a peak in winter at low temperatures ( $\approx 5$  degrees). The peak in winter will be due to another mechanism i.e. high wind speeds and *mechanical* mixing, which also brings the plume down to ground level. These plots can therefore highlight different mechanisms for dispersion.

In the next example we consider ethane. Emissions of ethane are dominated by natural gas leakage. These emissions therefore represent a diffuse emission with no buoyancy and which should respond to general atmospheric mixing. The next plot shows this characteristics very well i.e a “bull’s-eye” pattern. At low temperatures atmospheric mixing is reduced and concentrations increase and vice-versa. Indeed, the Figure is an excellent example of the type of pattern one would expect from an area source that is released at ground level and is neutrally buoyant. (Often similar patterns are observed for other pollutants at background sites).

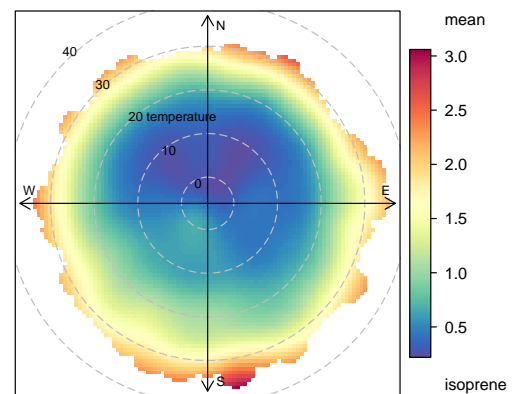
```
polarPlot(thedata, pollutant = "ethane",
          x = "temp", min.bin = 3)
```



**Figure 2:** Example plot using the `polarPlot` function to consider ethane concentrations as a function of wind direction and temperature.

In the final example we consider isoprene. Isoprene is a hydrocarbon that is emitted both by road vehicles but also biogenic sources. Emissions of species such as isoprene will become increasingly important in the future due to their role in regional O<sub>3</sub> formation. But what does a polar plot look like? The plot below shows a very different response to increasing temperature compared with either SO<sub>2</sub> or ethane in that concentrations increase as temperature increases. This increase is due to the temperature response of biogenic isoprene sources. It should be remembered that increased temperatures will also result in increased dilution, so it is likely the concentrations would be even higher if the dilution was ‘fixed’. The other thing to note from this plot is that no one wind direction dominates and this tells us something about the source or time scales for isoprene destruction. If it was destroyed in the atmosphere very quickly then only local sources would tend to feature in the plot.

```
polarPlot(thedata, pollutant = "isoprene",
          x = "temp", min.bin = 3)
```



**Figure 3:** Example plot using the `polarPlot` function to consider isoprene concentrations as a function of wind direction and temperature.

These plots together introduce alternative ways of considering the data, but there will be many more possibilities. Indeed, if readers can provide other examples, it would be good to hear from you.

# Trends and averages by season

Considering trends by season can be very important in air pollution and the atmospheric sciences in general. Recent changes to `openair` make it easier to average data by year-season and plot it.

The `timeAverage` function is used as the basis of averaging data in many flexible ways. Recently a new averaging period “season” has been added to the function to make it easier to calculate means and other statistics by season. The function considers ‘proper’ seasons e.g. winter is Dec/Jan/Feb in the northern hemisphere and therefore straddles two years.

For example, consider the seasonal means of `mydata`:

```
seas <- timeAverage(mydata,
  avg.time = "season")
head(seas)

##           date      ws      wd      nox
## 1 1998-01-30 11:30:00 4.695 220.7 223.1
## 2 1998-04-15 23:30:00 4.106 233.3 163.7
## 3 1998-07-16 23:30:00 4.551 245.5 186.2
## 4 1998-10-16 11:30:00 4.189 235.5 210.4
## 5 1999-01-14 23:30:00 4.806 234.7 220.8
## 6 1999-04-15 23:30:00 4.505 234.4 179.7
##      no2      o3      pm10      so2      co      pm25
## 1 49.84 3.465 34.31 7.293 2.400      NaN
## 2 47.06 7.893 31.39 6.792 1.683 20.99
## 3 46.83 5.748 31.75 6.519 1.968 19.60
## 4 48.36 4.696 32.49 6.954 2.154 22.10
## 5 47.52 5.059 29.39 5.929 1.874 19.23
## 6 45.30 8.034 31.67 4.759 1.608 22.57
```

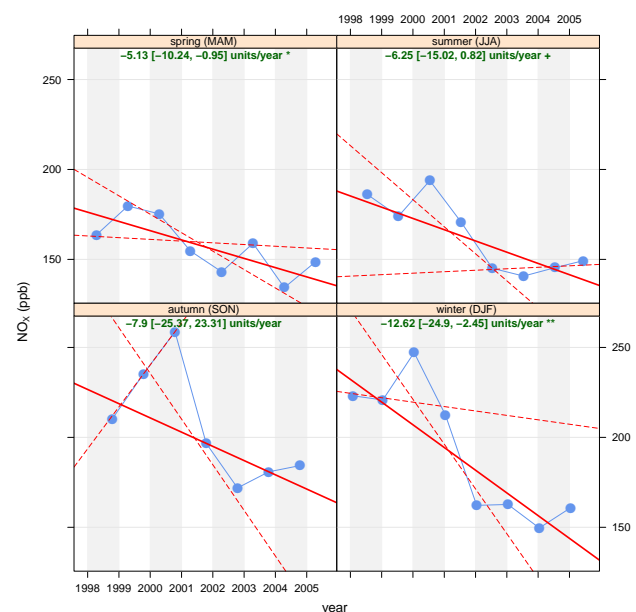
The date returned is also the mean of the entire seasonal period. For example, 1998-01-30 is the mean of the first two months of 1998 (there were no data for 1997) and so on.

This addition is probably most useful in trend analysis. Consider for example the seasonal trends

in  $\text{NO}_x$  at Marylebone Road. What we want are separate panels for each season, but also an averaging time for each season. The plot below also uses a different plotting symbol and size. Now it is easy to see that the trends in the winter months have been greater than other seasons (at least for data between 1998–2005).

```
TheilSen(mydata, pollutant = "nox",
  avg.time = "season", type = "season",
  pch = 16, cex = 1.5, ylab = "nox (ppb)")
```

```
## [1] "Taking bootstrap samples. Please wait."
## [1] "Taking bootstrap samples. Please wait."
## [1] "Taking bootstrap samples. Please wait."
## [1] "Taking bootstrap samples. Please wait."
```



**Figure 1:** Example plot using the `TheilSen` function to consider  $\text{NO}_x$  concentrations plotted by season, and for seasonal means.

## Other **openair** developments

Below is a summary of the developments to **openair** since version 0.5-11.

- Update to **GoogleMapsPlot**. This forces maps to be square until a bug is fixed in the package **RgoogleMaps**.
- Tidy up **calcFno2** plotting.
- Improve speed of **timeAverage** by removing `date.pad`.
- Add vector averaging of wind speed to **timeAverage**.
- Remove any considerations of time zones in **selectByDate**. If 'start' and 'end' are supplied, whole days based on Date format are used to select. Now accepts start/end in the form "YYYY-mm-dd" as well as UK format e.g. dd/mm/YYYY.
- Major update to **polarPlot** allowing variables other than "ws" to be plotted with wind direction.
- Improve documentation for **polarPlot/TaylorDiagram**
- Update **openair** citation information to Journal article
- do not clip **polarPlot** concentrations unless upper is supplied
- new option 'intervals' for **percentileRose**
- add `min.bin` option to **polarAnnulus**
- add Index of Agreement to **modStats**
- new averaging time "season" for **timeAverage**
- better treatment of `avg.time = "season"` and `type = "season"` in **TheilSen** and **smoothTrend**
- use bootstrap methods to calculate 95% confidence intervals in the mean for **timeVariation**

- New option 'difference' in **timeVariation** to show difference between two variables with bootstrap 95% CI in the mean
- Byte-compile package for improved speed
- Place key on right by default in **scatterPlot** to avoid clash with x or y axis labels.
- Include all colour schemes defined in the **RColorBrewer** package, see **openColours**
- Many improvements to the manual. All code associated with plots should appear at the top of each plot.

### **openair** citation information

The main citation for the **openair** package is now:

Carslaw, D.C. and K. Ropkins, (2012). **openair** — an R package for air quality data analysis. *Environmental Modelling & Software*. Volume 27-28, 52-61.

Details of how to cite the manual are given in the manual.

### Use of colours

The package has been updated to allow users to refer directly to those colours in the **RColorBrewer** package. This is a very useful package for choosing decent colour schemes. Users should refer to the help pages in **openColours** for more details, but in summary **RColorBrewer** schemes can be referred to directly by name.

For example,

```
polarPlot(mydata, col = "YlOrBr")
```

```
timeVariation(mydata, pollutant = c("no2",  
  "pm10", "pm25"), col = "Accent")
```