LENGUAJE R APLICADO AL ANÁLISIS DE DATOS DE CALIDAD DEL AIRE

MANUAL BÁSICO
para el tratamiento
de datos de
CALIDAD DEL AIRE
mediante el
lenguaje estadístico R
y paquetes adicionales como
OPENAIR

Elaborado por Fernando Follos Pliego.

Año de redacción Septiembre de 2012.

Este manual de *Openair* y de utilización de *R* ha sido elaborado por el autor como guía básica para *"dummies"* en el uso del lenguaje *R*, y de paquetes adicionales como *Openair*, para el análisis estadístico de datos sobre calidad del aire. El uso y distribución de esta manual es libre, cuenta con la autorización plena de su autor, que le agradece que lo considere como parte de su estudio o consulta y le permite su reproducción total o parcial, siempre y cuando se proceda a mencionar al mismo en las referencias bibliográficas o reproducciones que se hagan de su contenido.

Los actuales apuntes se han realizado en base a la extensa bibliografía ya disponible sobre *R, Openair* o herramientas adicionales como *Rstudio*, sin las cuales no habría sido posible, sirviendo más como un trabajo de traducción, resumen y adaptación esencialmente orientado a los técnicos que desarrollan su trabajo en el área de calidad del aire, y que no tienen conocimientos previos de programación, o conocimientos muy extensos de estadística, como de hecho así ocurre con el propio autor.

Para su elaboración se han utilizado las versiones v.2.15.o de *R* y v.0.6.0 del Paquete *Openair,* como principales herramientas, y se ha trabajado en paralelo con los programas para aplicar el propio manual a la experiencia de desarrollo, por lo que todas las instrucciones y secuencias de trabajo dispuestas en el manual han sido probadas con anterioridad por el autor, formando parte de un *practicum* completo sobre el programa. Así mismo, el presente manual ha sido elaborado en vistas a que se siga una secuencia de aprendizaje lógica del software (de principio a fin), por lo que al principio cualquier instrucción estará mucho más detallada y explicada, mientras que según se avanza y profundiza en el programa se irán resumiendo más y se detallarán sólo aquellos aspectos más fundamentales.

No obstante, el autor reconoce sus amplias limitaciones en cuanto a los temas tocantes a la estadística o la programación, por lo que pide disculpas de antemano por los diversos errores e imprecisiones que puedan existir en el manual y solicita su asistencia en caso de encontrar cualquier fallo o incongruencia en los contenidos.

Se recomienda acudir a las siguientes páginas Web y referencias de URL para adquirir versiones del software y guías con mayor actualización y profusión de datos que lo aquí presentado, si fuese preciso:

R Development Core Team (2011). http://www.R-project.org/
Openair (2012)– Environmental Modelling & Software. http://www.openair-project.org/
Rstudio Inc. (2012) - http://rstudio.org/

En todo caso, el autor espera que estos apuntes resulten de interés para el lector y sirvan para profundizar en el uso para profesionales de *R* y del paquete *Openair*, aprovechando para agradecer su trabajo desinteresado a los verdaderos expertos y cerebros que se encuentran tras el diseño y desarrollo del software libre utilizado en estos apuntes y en especial a David Carslaw, y a su eterna paciencia con el autor, las observaciones, consultas y el oxidado inglés de este último, su visión ha sido esencial a la hora de plantear muchos temas en este manual e incluso su propia estructura.

Por último, recordar que el autor está abierto también a cualquier contribución que se quiera hacer a este manual de libre distribución para su mejora y actualización, ya sea a través de comentarios, apreciaciones u observaciones, que deberán dirigirse a los siguientes datos de contacto y que serán indefectiblemente contestadas y, en su caso, contempladas dentro de su desarrollo.

D. Fernando Follos Pliego
Environmental Risk Management
C/ Federico García Lorca, nº24.
45.280 Olías del Rey. Toledo—Spain.
bael76@hotmail.es

Dedicado a mi hija Alexandra y a mi mujer, que me han buscado un hueco en sus vidas para poder introducirme en el apasionante mundo de R y Openair.

INDICE DE CONTENIDOS:

Referencias y Bibliografía asociada:

- R Developement Core Team (2012). R: A lenguaje and environment for statistical computing. R Foundation for Statistical Computing, Viena, Austrial. ISBN 3-900051-07-0.
- Carslaw, D.C. and Kart Ropkins, (2012) openair—an R package for air quality data análisis. Environmental Modelling & Software. Volume 27-28, 52-61.
- David Carslaw and Kart Ropkins (2012). Openair: Open-Source tools for the análisis of air pollution data. R Package versión 0.6-0.
- PLYR Packaged (2012) Tools for splitting, applying and combining data.—Ver.1.7.1 Hadley Wickman.
- The Openair Manual— open-source tools for analysing air pollution data. Kings College London. David Carslaw. Version: January 2012.
- Introducción a R— R Development Core Team—Traducción de Andrés González y Silvia González.
- R para principiantes— Emmanuel Paradis—Traducción de Jorge A. Ahumada
- Gráficos Estadísticos con R — Juan Carlos Correa y Nefi González. Universidad Nacional. Sede Medellín. Año 2002.
- R news and tutorials contributed by (393) R bloggers - http://www.r-bloggers.com/
- R Graphical Manual - http://rgm2.lab.nig.ac.jp/RGM2/
- Quick-R. Accesing The power of R - http://www.statmethods.net/

Los datos utilizados en este manual para ejemplificar las distintas funciones y operaciones de *R* y *Openair* son datos reales y de carácter público, obtenidos a partir de los datos publicados por las redes de inmisión públicas de las Comunidades Autónomas de Madrid y Castilla-La Mancha en sus respectivas páginas web:

- http://gestiona.madrid.org/azul_internet/run/j/AvisosAccion.icm
- http://pagina.jccm.es/medioambiente/rvca/calidadaire.htm

INSTALACIÓN DE R Y PAQUETES ADICIONALES:

Este manual está pensado como si fuesen unos apuntes básicos para manejar de una forma esencialmente práctica el lenguaje *R*. No obstante, para hacerlo es estrictamente necesario instalar no sólo *R* en el ordenador, sino también el paquete o librería *"Openair*", e incluso otras herramientas de utilidad que se recomienda instalar, aunque no es estrictamente necesario, como *Rstudio*, que permiten un manejo mejorado del lenguaje (algo más gráfico y asequible para los que no formamos parte de la comunidad de programadores informáticos).

R es un software libre que se puede encontrar en la página web dedicada al proyecto *R*, ver referencias bibliográficas. Dentro de esta página deberemos proceder a la descarga del programa en función del software de que disponga nuestro equipo (*windows*, *linux*, etc).

La instalación del paquete *Openair* es aún más sencilla, se puede realizar a través de la propia web de *Openair*, bajando el correspondiente instalador, o solicitando a *R* que instale el correspondiente paquete:

install.packages("openair", dep=TRUE).

Para ponerlo *aún más fácil*, la instalación también se puede llevar a cabo a través del menú superior de la ventana de *R: Paquetes → instalar paquetes*, donde seleccionaremos el paquete *Openair*.

R requerirá que carguemos el paquete *Openair* cada vez que lo iniciemos, salvo que guardemos la sesión. La orden para cargar el paquete *Openair* se verá con posterioridad, cuando conozcamos cómo se trabaja con *R* con mayor profundidad.

Además del paquete *Openair,* es preciso que nos aseguremos de que tenemos instalados y debidamente seleccionados a la hora de trabajar otros paquetes que también utilizaremos en nuestro trabajo habitual como *RgoogleMaps*, *plyr, hexbin, lattice y utils*, entre otros. Muchos de ellos ya vienen preinstalados y seleccionados con *R*, y otros van implícitos en la instalación de *Openair*, pero en cualquier caso es recomendable asegurarse debidamente de que tenemos todos instalados y, en la medida de nuestra programación de trabajo, seleccionados.

En todo caso, conviene indicar que *Openair* llama a estos paquetes cuando alguna de sus funciones o herramientas va a requerir que se utilicen los mismos. Si no estuviesen instalados nos dará un error y nos indicará el paquete que debemos *"bajarnos"* de Internet e instalar en nuestro equipo.

Una vez instalado lo básico, el usuario también puede optar por instalar, aunque no sería estrictamente necesario, un programa como *Rstudio*, que proporciona una consola de fácil manejo de *R* con la que mejorar, simplificar y facilitar el trabajo.

En caso de instalar la Consola de Rstudio, no hace falta iniciar *R*, pues la consola inicia este lenguaje de programación de forma automática, lo que si será necesario, en todo caso, es llamar a la librería: *Openair*, tal y como se verá más adelante.

¿QUÉ ES R Y COMO FUNCIONA? - CONCEPTOS BÁSICOS :

R es en sí mismo un lenguaje de programación creado en un entorno pensado para el análisis estadístico y gráfico de datos, siendo un software libre que se distribuye bajo licencia *GNU GPL*.

R como entorno de programación se desarrolla mediante librerías (también llamadas en *R* como paquetes) que lo que hacen es completar el lenguaje con nuevos desarrollos previstos para distintas áreas del análisis estadístico y gráfico de los datos. Por ejemplo, *Openair* es una de esas librerías, específicamente diseñada para tratar datos de calidad del aire, aunque existen multitud de librerías para distintas áreas de estudio, por lo que conviene visitar la página Web de *R* para ver las múltiples aplicaciones de que dispone este lenguaje estadístico.

A efectos prácticos *R* consiste básicamente en un lenguaje de programación para el estudio estadístico de datos que presenta subconjuntos de lenguaje desarrollados para áreas específicas del análisis de datos, como en nuestro caso la calidad del aire.

Así, el lenguaje *R* es común para *R* y *Openair*, pero el lenguaje de *Openair*, en muchas de sus instrucciones y comandos, es propio del desarrollo de *Openair*, no existiendo previamente para *R* si no se carga la librería de *Openair*.

Como entorno de programación básicamente se trata de una consola (ventana de trabajo) sobre la que se van introduciendo scripts (instrucciones más o menos complejas) que se ejecutan sobre los datos previamente cargados (conjuntos o ventanas de datos).

Los scripts consistirán en una serie de instrucciones modificadas por comandos y variables ejecutadas sobre un conjunto o conjuntos de datos que, además, pueden concatenarse introduciéndose en batería (un script detrás de otro) o en serie (concatenando un script con otro).

En la consola de *R* se pueden iniciar distintas sesiones de trabajo (denominadas áreas de trabajo), que podemos grabar para retomar con posterioridad en el punto que lo dejamos, y sobre las que se van cargando y guardando no sólo los scripts que requerimos a *R,* sino todas aquellos paquetes de datos que previamente hemos cargado o leído, por lo que se convierte en un entorno de trabajo muy funcional y práctico.

Además, no sólo se pueden guardar las áreas de trabajo que vayamos abriendo con *R,* sino que también podemos guardar las ventanas de datos como archivos *R* o los propios scripts que hayamos programado como objetos de *R*, o incluso las gráficas que vayamos generando en formatos gráficos totalmente compatibles. Esto aporta una gran versatilidad al programa.

ATENCIÓN: *R y Openair*, como casi cualquier lenguaje de programación, **requieren que se respeten escrupulosamente sus principios de configuración y escritura de comandos,** por lo tanto, habrá que prestar especial atención a que:

- **Se diferencia entre mayúsculas y minúsculas,** una sola letra puesta al revés en una instrucción o comando y esta deja de funcionar. Las instrucciones contempladas en este manual no están mal escritas, sino que respetan escrupulosamente la disposición de mayúsculas y minúsculas establecida por el autor en la definición de la misma.
- **Las variables no numéricas** definidas por el conjunto de datos, por Openair o por el usuario, **van entre comillas, y las numéricas van sin entrecomillar.**
- **Los distintos comandos** definidos dentro de una instrucción **van separados siempre por comas, y los decimales de los números por puntos.**
- **La definición de los comandos de una instrucción se realizará siempre mediante un signo "=" y la disposición de la variable o variables.**

En todo caso, **LA PRÁCTICA HACE AL EXPERTO,** por lo que se recomienda que se vaya aprendiendo sobre la marcha en el seguimiento de este manual y su trasposición a la práctica en R paso por paso.

IMPORTACION Y EXPORTACIÓN DE ARCHIVOS DE DATOS :

Importar datos es, por lo general, el primer paso que se debe dar para realizar análisis de datos con *Openair* o *R*. Importar datos es, de hecho, uno de los factores más "problemáticos" de *R*, pues este lenguaje tan sólo trabaja con archivos planos de texto, del tipo *.csv*, *.txt* o *.dat*, y por lo general las bases de datos de calidad del aire se encuentran en sistemas más complejos del tipo *MySQL, Oracle*, etc.

El primer paso es exportar los datos desde la base de datos disponible a un archivo del tipo *.txt*, en texto plano. En este caso, si no se dispone de un software específico para la exportación de datos a un archivo de este tipo, lo suyo es hacer una consulta con hoja de cálculo o sistema similar, y guardarla como archivo *.txt*, una vez realizadas las siguientes operaciones básicas que se comentan a continuación:

Disposición de la información: Los datos vendrán en columnas con encabezamientos, disponiéndose de un registro que se denomine "date" que contendrá la fecha por la que se regirá el conjunto de datos.

La fecha: *R* considera el apartado fecha como un todo, mientras que la mayoría de sistemas de control de la calidad del aire interpretan la fecha y luego las horas (o cuartohorarios, o diezminutales, etc.). Es recomendable, a pesar de que se podría resolver con posterioridad en la importación a *R* la existencia de registros separados (fecha/hora), que la fecha y la hora quedasen incluidas en un sólo registro, sumando los mismos, y que el formato respondiese a un estándar claro definido por el usuario que se viese con posterioridad respetado en *R*.

Los datos válidos: *R* no interpreta *flags* de calificación de datos, sino sólo datos. Es preciso que el archivo para exportar datos en *dat* cuente tan sólo con el campo contaminante con los datos válidos, sin *flags* ni mayores complicaciones.

Encabezamientos: los nombres de los campos serán los encabezamiento del archivo y se deberán ubicar en la primera línea del archivo de exportación.

Los nombres de los campos: *R* y *Openair* son sistemas ingleses, y por lo tanto, si se quiere que las gráficas de *Openair* se ejecuten de forma automática, es preciso traducir determinados parámetros del castellano al inglés (Velocidad del viento (*VV*) será entonces *Wind Speed* (*ws*)).

Los datos nulos: Dependiendo del sistema de adquisición de datos que se esté utilizando, los datos nulos pueden aparecer como "-9999", "nulo", "n.a.", etc. Esta variada terminología no es entendida por *R* que, a pesar de disponer de herramientas para convertir este tipo de datos, es conveniente que disponga ya de ellos como un hueco en blanco o con la denominación "NA".

La separación entre datos: Los datos de cada uno de los campo deberán venir separados de alguna forma para que el sistema los interprete. En este sentido *R* permite varias combinaciones, aunque es recomendable utilizar el tabulador (pues permite un mejor repaso de los archivos de datos en caso de precisar ser consultados en bruto).

La calidad de los datos: Es importante garantizar que los datos disponen de una calidad adecuada, para evitar que el tratamiento estadístico de los mismos arroje datos que no se corresponden. Para ello habrá que asegurarse de que el archivo de datos se exporta a partir de datos validados, y que la validación realizada de los datos de calidad del aire es la adecuada. Para ello sería recomendable un tratamiento previo de los datos en función de criterios básicos como la inexistencia de datos nulos, la garantía de que todos los dato se encuentran dentro del rango lógico de medida, etc.

Una vez que tenemos exportados los datos de calidad del aire que correspondan a un archivo de datos plano, del tipo *.txt*, tendremos que importarlos en *R*, para lo cual disponemos de varias opciones:

- La forma más básica es leer los datos en *R* mediante el comando: ***read***.
  Para ello, debemos indicar a *R* la denominación que vamos a datos al fichero o grupo de datos, utilizando el comando *read* para decirle que lea los datos del archivo de texto plano que corresponda.

Ej. **datoscont<-read.txt("C:/misdocumentos/Openair/datoscont.txt", header=TRUE,dec=",")**, donde indicamos a R que los datos de "datoscont" se encuentan en la dirección que entrecomillamos, que se trata de un archivo .txt, que dispone de encabezamiento, y cuyas cifras separan los decimales con comas.

Si llevamos a cabo esta instrucción y el archivo se ha exportado correctamente, en principio, deberíamos de tener ya cargados los datos en nuestro sistema *R* como una ventana o conjunto de datos. Podemos comprobarlo utilizando cualquiera de las instrucciones que se comentan en el apartado 3 de este manual, como por ejemplo ***head***:

Ej. **head(datoscont)**, con esta instrucción deberíamos poder ver los encabezados del archivo que acabamos de leer.

- Si utilizamos algún tipo de consola para el manejo en *R*, del tipo *Rstudio*, la importación de datos se hace más intuitiva, pudiendo acudir a la utilización de los cuadros de diálogo que automatizan y resumen el proceso de lectura e importación de datos. En el caso de *Rstudio*, por ejemplo, se pueden importar datos acudiendo al menú: ***Workspace → Import Datase → From Text File..***.

  Al pulsar sobre este menú, se abrirá un cuadro de diálogo directamente sobre el directorio de trabajo del programa para que seleccionemos el archivo de datos *.txt* que disponga de los datos. Una vez seleccionado el archivo se pulsa en ***Abrir***. En este momento *Rstudio* abre un segundo cuadro de diálogo en el que le debemos indicar las variables que caracterizan el archivo *.txt*:

  - El nombre que le vamos a dar a los datos en *R*: ***Name*** (se recomienda utilizar un nombre corto y en minúsculas, pues luego se va a escribir en todas las operaciones que se quieran realizar con esos datos).
  - Si el archivo dispone o no de encabezamiento: ***Heading*** (yes o no).
  - El tipo de separador que se utiliza entre datos: ***Separator***
  - El tipo de símbolo que se utiliza para los decimales: ***Decimal*** (que evidentemente deberá ser distinto al utilizado como separador)
  - El tipo de símbolo utilizado para delimitar el texto: ***Quote***

  Una vez indicadas las variables al sistema de importación, es tan sencillo como ***Aceptar*** la importación del archivo que, de forma automática aparecerá en *Rstudio*, en su cuadrante de *"Workspace"* (normalmente cuadrante superior derecho), así como en el apartado *"Source"* (normalmente cuadrante superior izquierdo), donde se muestra un resumen de los encabezados con las mil primeras líneas.

Una vez importados los datos de contaminación atmosférica que queremos, es preciso indicar a *R* y esencialmente a *Openair*, cual es el campo que contiene los datos de fecha y qué formato tiene, pues será este campo el que utilice como *index* para la posterior representación gráfica y estudio estadístico de los datos en la mayoría de las funciones a ejecutar. Esto se lleva a cabo mediante la función ***as.POSIXct***, que convierte la fecha al formato adecuado para su manejo en *R*, y la función ***strptime***, que le dice a *R* en qué formato está la fecha que se va a utilizar. Un ejemplo de su uso sería el siguiente:

Ej. **datoscont$date <- as.POSIXct(strptime(datoscont$date, format = "%d/%m/%Y %H:%M", "GMT"))**, donde le indicamos a Openair que la fecha de los datos **datoscont$date**, se corresponderán con la posición fecha, y con el formato día/mes/año y hora:minuto **"%d/%m/%Y %H:%M"**, y se encuentran en GMT, aunque también le podemos decir que están en UTC, si así fuese.

En este sentido, es importante que sigamos un estándar propio a la hora de establecer el formato de la fecha de los distintos conjuntos, puesto que este parámetro se utilizará muy habitualmente en posteriores instrucciones de *R*, y podría complicársenos la vida si cada conjunto de datos que manejemos está en un formato de fecha y hora distinto. El formato de fecha, como hemos visto en el párrafo anterior, lo da ***strptime***, y se basará en los siguientes parámetros básicos de configuración que se adjuntan:

Concepto	Código	Tipo	Formato	Rango
DÍAS				
… del mes	%d	número	nn	01 – 31
… de la semana	%a	texto	Abreviado	Sab – Vie
	%A	texto	Completo	Sábado – Viernes
	%w	número	nn	0 – 6 (0 = domingo)
… del año	%j	número	nnn	001 -366
SEMANAS				
	%U	número	nn	0 – 53 (usa domingo como primer día)
	%W	número	nn	0 – 53 (usa lunes como primer día)
MESES				
	%b	texto	Abreviado	Ene – dic
	%B	texto	Completo	Enero – Diciembre
	%m	número	nn	01 – 12
AÑOS				
	%y	número	nn	00 – 99
	%Y	número	nnnn	1900 – 2999
HORAS				
	%I	número	nn	01 – 12
	%H	número	nn	00 – 23
	%p	texto	Completo	AM/PM (a usar sólo con %I)
MINUTOS				
	%M	número	nn	00 – 61
SEGUNDOS				
	%S	número	nn	00 – 61

Es importante que, una vez asignada y caracterizada la fecha en el archivo, según lo especificado hasta el momento, procedamos a comprobarlo en *R* mediante alguna instrucción del tipo *str* (ver apartado correspondiente a las instrucciones básicas) que nos permita comprobar las características de los distintos campos del archivo. Así, al ejecutar el comando str antes del formateo de la fecha podemos comprobar que el campo de fecha del archivo aparece como:

```
Ej. str(datoscont)
'data.frame':   107376 obs. of  12 variables:
 $ date: Factor w/ 107376 levels "01/01/2000 01:00",..: 1 2 3 4 5 6 7 8 9 10 ...
```

Sin embargo, una vez caracterizado y formateado el campo de fecha *"date"*, al ejecutar un *str* este aparece como un campo *as.POSIXct* con formato de fecha, tal y como aparece a continuación:

```
Ej. str(datoscont)
'data.frame':   107376 obs. of  12 variables:
 $ date: POSIXct, format: "2000-01-01 01:00:00" "2000-01-01 02:00:00" …
```

Si hemos seguido todas las instrucciones previstas en este apartado y hemos sido cuidadosos en nuestro trabajo, ya podemos empezar a manejar este conjunto de datos en *R* y *Openair* sin tener problemas, aunque en primera instancia sería recomendable realizar una serie de actuaciones encaminadas a comprobar de una forma rápida la consistencia de los datos que hemos importado, tal y como se verá en apartados posteriores.

INSTRUCCIONES BÁSICAS DE R Y OPENAIR :

Antes de comenzar a trabajar con el lenguaje *R* y con el paquete *Openair* debemos saber que este lenguaje funciona con órdenes o instrucciones básicas que deben ir acompañadas del objeto de la orden entre paréntesis, tal y como se puede empezar a intuir de los pocos ejemplos propuestos hasta ahora. Si no existe objeto siempre deberán aparecer los paréntesis, aunque sea vacíos.

Las instrucciones que demos a *R* deberán incluir, si es preciso, dentro de esos paréntesis además del objeto al que van dirigidas, las instrucciones vinculadas, comandos o cualquier otra variable que queramos añadir a la instrucción, según se irá viendo poco a poco en los distintos puntos de ejemplo. De hecho *R* y *Openair* son lenguajes de programación estadísticos, por lo que poco a poco comprobaremos cómo se pueden ir concatenando instrucciones una con otra para alcanzar lo que buscamos en cada caso.

Esquema básico de una instrucción en *R*:

instrucción (*objeto/conjunto datos*, ***comando***=”***variable***”)

Instrucciones previas al trabajo con archivos y datos.

Se detallan a continuación una serie de instrucciones básicas de *R* y *Openair* que son de utilidad para el manejo del lenguaje y de ciertos aspectos básicos del mismo. Evidentemente, muchas de estas instrucciones ya han sido simplificadas si utilizamos consolas para el manejo de *R* como *Rstudio*.

- Llamar a R para que utilice el paquete Openair con nuestros datos: ***require***

  Antes de empezar a manejar conjuntos de datos y a iniciar su tratamiento, es preciso indicarle a *R* la librería con la que vamos a trabajar, al objeto de que entienda las instrucciones que posteriormente vayamos a cargar. De no llamar a las librerías correspondientes mediante este comando, es seguro que las instrucciones de *Openair* no funcionarán y tan sólo podremos utilizar las genéricas de *R*.

  Ej: **require(openair)**, se debe utilizar al comienzo de cualquier sesión de trabajo nueva que se inicie con R, pues de lo contrario no se utilizará el paquete Openair y gran parte de las instrucciones no funcionarán. Si trabajamos con algún tipo de interface como Rstudio, la instrucción require se puede sustituir por otra instrucción propia, en este caso: ***library(openair)***. En cualquier caso, la instrucción require siempre funcionará en R.

- Conocer y cambiar la carpeta de trabajo de *Openair*: ***getwd*** o ***setwd***

  R guarda todos sus archivos en una carpeta predeterminada. Si queremos saber la carpeta de la que se trata bastará con utilizar el comando *getwd*. Si lo que queremos es establecer nosotros mismos una carpeta, el comando será *setwd*.

  Ej. **getwd()**, presentará en la pantalla el directorio de trabajo de R donde se guardan todos los archivos y espacios de trabajo que queramos.
  Ej. **setwd(“C:/documentosR”)**, establecerá como directorio de trabajo para el guardado de archivos el indicado entre comillas.

- Listar el contenido de la carpeta de trabajo de *Openair* o *R*: ***list.files***

  Ej. **list.files(pattern=”.txt”)**, presentará un listado de todos los archivos .txt que se presentan en el directorio de trabajo de R. Si queremos un listado completo, bastará con dejar la instrucción con los paréntesis vacíos: **list.files()**.

Instrucciones básicas para el manejo de los conjuntos de datos.

Antes de empezar a utilizar las conjuntos de datos que ya hemos cargado con éxito y configurado como auténticos profesionales, debemos aprender a utilizar una serie de instrucciones que nos serán con posterioridad enormemente útiles a la hora de manejarlos y de sacarles el máximo partido.

En primer lugar es necesario conocer las instrucciones básicas utilizadas para realizar una primera aproximación a los datos que contiene nuestra ventana de datos, de forma que podamos ver encabezados, ver los campos o acceder a una variable concreta de nuestro grupo de datos:

- Conocer las variables de las que dispone el conjunto de datos en su encabezamiento: ***names***

  Ej: **names(datoscont)**, devuelve el encabezado de todas las variables de que dispone el archivo.

  Con la instrucción *names* podemos también cambiar el nombre de las variables para amoldarlo a lo que precisemos.

  Ej: **names(datoscont)[3] <- "o3ref"**, lo que hace es indicarle a la instrucción names que el tercer campo [3] de nuestro conjunto de datos (datoscont) se pasará a denominar "o3ref".

- Conocer las variables y ver los primeros datos de una ventana de datos: ***head***

  Ej: **head(datoscont)**, devuelve el encabezado del archivo con los seis primeros datos de que dispone el archivo de datos. Similar al anterior names, pero mostrando los primeros datos del archivo.

- Conocer las variables y ver los últimos datos de un conjunto: ***tail***

  Ej: **tail(datoscont)**, devuelve el encabezado del archivo con los seis últimos datos de que dispone el mismo. Similar al anterior head, pero mostrando los últimos datos del archivo. Resulta de especial utilidad si se precisa ver dónde acab una ventana de datos y listar todas las filas es imposible o muy costoso.

- Acceder a una sola variable de un conjunto de datos: ***$***

  Ej: **summary(datoscont$no2)**, proporcina el resumen de datos para el parámetro no2 del archivo de datos de datoscont, sin necesidad de que se muestren los datos del resto de las variables.

- Proporcionar la estructura de una ventana de datos: ***str***

  Ej. **str(datoscont)**, devuelve en pantalla en líneas cada uno de los campos de un archivo de datos, incluyendo la tipología de cada uno de los campos. Similar a la instrucción summary, pero permite de una manera más ágil conocer la estructura y características de cada campo. Muy utilizado para comprobar, por ejemplo, la caracterización disponible para el campo fecha.

- Salvar, cargar o exportar datos previamente importados a *R*: ***save, load, write***

  Ej. **save(datoscont, file="datoscont.RData")**, que lo que hace es salvar los datos cargados previamente en R, según lo expuesto en el apartado2, guardándolos en formato R.

  Ej. **load("datoscont.RData")**, que lo que hace es volverlos a cargar desde el fichero de datos guardado.

  Ej. **write.txt("datoscont.txt", row.names=TRUE)**, lo que hace es generar un archivo de texto con el conjunto de datos cargado anteriormente.

- Eliminar objetos de un proyecto que no sean de utilidad: ***rm, remove***

  Ej: **rm(datosconta)**, elimina el objeto de nuestro proyecto denominado "datosconta". Esta función, que también podemos utilizar como remove, se utiliza para eliminar objetos que ya no sean de utilidad para nuestro proyecto y permite realizar limpieza en nuestro proyecto de R evitando que se acumulen los conjuntos de datos u objetos que ya no son de utilidad.

Ya hemos visto cómo podemos realizar un manejo en general de los conjuntos o ventanas de datos. No obstante, antes de comenzar a utilizar instrucciones de mayor complejidad para el manejo y gestión de las ventanas de datos y los propios datos en sí mismos, hemos de aclarar ciertos aspectos de la programación en *R* y *Openair* útiles para complementar y entender determinadas instrucciones y comandos que utilizaremos más adelante.

Uno de estos aspectos viene dado por la necesidad de indicarle a un comando la variable por la que se puede regir , que dependerá tanto de las características de diseño del comando como de la propia variable a la que nos queramos referir. De esta forma, las variables a disponer que utilizaremos a lo largo del presente manual serán del siguiente tipo:

- Variables para comandos lógicos: ***TRUE*** *o* ***FALSE***

  Muchos comandos en *R* y *Openair* requieren tan sólo ser activados o desactivados, utilizando para ello una variable lógica del tipo si/no, que se escribirá frente al comando utilizando los términos *TRUE* o *FALSE*, según corresponda. Las variables lógicas indicadas deberán escribirse siempre en mayúsculas.

- Variables numéricas simples:

  Otros muchos comandos de *R* y *Openair* requerirán que se les suministre una variable numérica del tipo que corresponda en función de las unidades a las que se refiera el comando. En estos casos la variable se expresará conforme al número que corresponda junto al comando. Por ejemplo, el comando para definir el ángulo *angle*, requerirá que le introduzcamos un número entre 0 y 360 para indicarle el ángulo que deseamos, de la forma ***angle=45***. Es necesario recordar que los decimales se separan de la fracción entera del número utilizando siempre el punto, ya que las comas se reservan en *R* y *Openair* para la separación de los comandos.

- Variables de texto o variables *string*: ***" "***

  Algunos comandos de *R* y *Openair*, como aquellos que se refieran a determinados parámetros del conjunto de datos (en el caso de *Openair*) o al etiquetado y denominación de elementos, pueden requerir de la introducción de una variable en formato de texto. En estos casos el texto se ubicará siempre frente al comando e incluido siempre entre comillas.

- Variables vectoriales o multivariable: ***c***

  En ocasiones la variable que requiere un comando es un vector, o lo que es lo mismo, una variable compuesta por dos o más variables, pudiendo ser estas numéricas o de texto. Así, cuando un comando de R precise de la utilización de varias variables para una instrucción concreta, sin, se debe utilizar la instrucción con un ***c(i,ii,iii,...)***, siendo c el complemento a la instrucción que indicará a la misma que tenemos que utilizar las variables que se incluyen entre los paréntesis.

- Rango de variables a utilizar: ***:***

  En ocasiones es preciso indicar a *Openair* o *R* que utilice una serie de variables incluidas dentro de un rango de datos conocido. Para ello, el comando deberá incluir el inicio y el final del rango separados por dos punto: ***x:y***, donde *x* será el mínimo del rango, e *y* el máximo. Esta opción suele ser muy útil para seleccionar, por ejemplo, determinadas variables relacionadas con fechas o parámetros del conjunto de datos.

- Variables dentro de una secuencia determinada: ***seq***

  En ocasiones puede ser preciso la utilización de distintas variables para una misma instrucción o comando, y que estas variables no se correspondan con número aleatorios (como en la instrucción c) sino con una secuencia predeterminada, para lo cual se utilizará el comando o instrucción con un ***seq(x,y,z)***., donde la *x* será el mínimo de la secuencia, la *y* el máximo, y la *z* el intervalo a aplicar a la secuencia. Esta opción puede ser muy útil, por ejemplo, para establecer escalas o rangos de datos personalizados.

Después de precisar las instrucciones para el manejo de los conjuntos o ventanas de datos, y ver cómo se complementan en diversos aspectos las instrucciones y comandos a utilizar, podemos comenzar a profundizar en el estudio de la aplicación de instrucciones encaminadas al manejo y gestión de los conjuntos o ventanas de datos.

En primer lugar se debe estudiar cómo proceder a la modificación de los conjuntos o ventanas de datos, ya sea para la segregación del conjunto de datos, para la combinación de ventanas de datos entre sí, generando nuevos conjuntos de datos con los que poder trabajar, u otras muchas necesidades de modificación de los conjuntos, tal y como se puede observar a continuación.

En estas ocasiones las instrucciones se empiezan a volver ya algo más complejas y requieren de la inclusión de comandos y variables propios, o incluso de la ejecución de otras instrucciones, tal y como veremos progresivamente.

- ***Transponer filas por columnas: t***

  Es posible que los datos que se nos presenten en muchas ocasiones vengan en una ventana de datos que no se corresponda con lo que nosotros precisamos para una posterior gestión de los datos porque la selección original de filas y columnas no coincida con lo esperado o con lo precisado por el usuario.

  Ej. **estaciones ← t(estaciones),** donde se cogerá el fichero de estaciones de control organizado de forma que en los encabezados de columna aparecen las estaciones y en las filas los datos de caracterización de la estación de control, tales como coordenadas, denominación y códigos de clasificación, y "se le dará la vuelta", poniendo como encabezados de las columnas los distintos datos de caracterización y como filas las estaciones. Esto puede ser muy útil, por ejemplo, para mezclar archivos de datos de varias estaciones con los datos de dichas estaciones (el lector podrá comprobar la utilidad de esta herramienta más adelante en este mismo manual).

  Sobre la base conceptual de esta herramienta, se debe mencionar que existe un paquete en R que se denomina ***reshape*** y que resulta de enorme utilidad para flexibilizar, moldear y agregar datos dentro de las propias ventanas de datos, presentando funciones que van mucho más allá de la propia transposición de filas por columnas que acabamos de ver.

  Con el paquete de *reshape* se pueden, por ejemplo, fusionar varios campos o columnas de una ventana de datos de forma pasen a formar parte de un identificador único que presente en un único nivel de agregación, con la función ***melt***, agregar los datos que hayamos obtenido tras la fusión con melt en la forma que considere el usuario, con la función ***cast***, condensar una ventana de datos en función del vector de variables que considere el usuario, con la función ***condense.df***, o incluso combinar varias funciones en una sola, con la función ***funstofun***.

  En todo caso, el desarrollo de estas funciones quedará suscrito al interés del lector por el tema, pues el manual debe quedar reducido por imperativo práctico a lo estrictamente necesario para el manejo básico de los datos de calidad del aire.

- ***Seleccionar una serie de datos de un conjunto de datos: subset***

  Es posible que queramos seleccionar determinados datos de un conjunto de datos que ya tenemos, ya sea porque, por ejemplo, queremos ver tan sólo determinados datos a altas concentraciones de ciertos parámetros, o porque queremos seleccionar los datos de un archivo para determinadas fechas. La instrucción subset nos permite hacerlo, pero utilizando para ello diversas opciones:

  La primera opción es indicarle a subset una condición básica de selección de datos en función de un criterio matemático cualquiera, tal y como se indica en el ejemplo:

  Ej. **datoscont2 ← subset(datoscont, o3>100),** donde se seleccionan las filas de datos cuyo ozono es superior a 100 microgramos para guardar el conjunto de datos como datoscont2.

  Con la instrucción subset es posible seleccionar los datos hasta por fechas, tal y como se puede ver en el siguiente ejemplo adjunto, sin embargo, el problema es que *R* y *Openair* son muy especiales con las fechas, y si bien es posible utilizar esta opción, utilizando la opción *%in%* para agregar datos definiendo previamente el formato, requiere de la definición de variables y puede complicarnos mucho la vida para programar una operación que, por otro lado, *Openair* ya nos deja debidamente personalizada a través de la instrucción *selectByDate*, que veremos en breve.

Ej. **datoscont2 ← subset(datoscont, format (date,"%Y") %in% 2011),** instrucción en la que le decimos a R que haga un subconjunto de datos del conjunto datoscont, utilizando la fecha en formato año, y seleccionando tan sólo aquellas que tengan como año el 2011. Todo este subconjunto se guarda como un conjunto de datos denominado datoscont2.

Además de estas opciones, a lo mejor no nos interesa utilizar todos los parámetros del archivo primigenio, por lo que podemos indicar a subset que seleccione sólo algunos para el nuevo archivo con la opción *select*.

Ej. **datoscont2 ← subset(datoscont, o3>100, select=c(date, o3, no2, no, sr)),** seleccionará los datos indicados en select para aquellos niveles de ozono que sean superiores a 100 microgramos. También utilizamos aquí la expresión c para la selección de varias variables.

♦ ***Combinar dos conjuntos de datos en uno: merge***

Hemos aprendido a dividir un conjunto de datos en varios conjuntos mediante la instrucción *subset*, pero igual de útil puede resultar combinar varios conjuntos de datos en uno sólo, lo cual puede resultar de mucha utilidad cuando queremos combinar datos complementarios (como por ejemplo niveles de calidad del aire con meteorología) o incluir en un único conjunto de datos diversas ubicaciones, etc.

R permite combinar dos conjuntos de datos en uno sólo mediante la función *merge*, que lo que hace es juntar en un único conjunto de datos la línea del primer grupo de datos con la línea correspondiente del segundo grupo de datos en función de un nexo común existente entre los dos conjuntos.

La instrucción *merge* se podrá precisar en sus distintos aspectos a través de los siguientes comandos:

1-. Se deben establecer los conjuntos de datos a combinar:

Como ya hemos podido comprobar sobradamente, el primer parámetro que aparece en cualquier instrucción de *R* y *Openair* es el conjunto de datos sobre el que se quiere aplicar la instrucción. En el caso de *merge* no podría ser menos, así que lo primero que se debe indicar a esta función son los dos conjuntos de datos que se quieren combinar, *x* e *y*, indicando el nombre de cada uno de ellos. La función *merge* sólo puede combinar los conjuntos de datos de dos en dos, si quisiésemos opciones de combinación de conjuntos más elaboradas y flexibles deberíamos acudir a librerías *R* con funciones más avanzadas como las que veremos de *Openair* o como las que disponen paquetes como *reshape2*.

2-. Indicar las columnas de datos que sirven como nexo de unión: ***by***

A la instrucción *merge* se le pueden indicar las columnas que sirven de nexo de unión para combinar conjuntos de datos. Si no se le especifica nada, la función intentará combinar los conjuntos en función de la columna utilizada como índice, que será la fecha *"date"*.

Pero se le puede indicar explícitamente el nexo de unión entre conjuntos de datos, para lo cual se utiliza el comando by. Si el campo es común para ambos conjuntos, como por ejemplo el campo de fecha *"date"*, bastará con indicarle al comando by el nombre del campo que comparten ambos conjuntos ***by="date"***. Esta opción puede ser muy útil cuando uno de los conjuntos, por ejemplo, disponed e datos de distintas ubicaciones y su campo de fechas *"date"* tiene por lo tanto fechas repetidas, obligando a la instrucción *merge* a realizar un paralelismos total entre fechas.

También podría ocurrir que los campos tuviesen distintas denominaciones, por lo que sería necesario indicar a la instrucción *merge* el nombre de cada campo.

Ej. **datoscontmet ← merge(datoscont, met, by.datoscont="date", by.met="fecha"),** mediante esta instrucción estaríamos combinando el conjunto datoscont, con datos de calidad del aire, con el conjunto met, de datos meteorológicos de una estación próxima, combinándolos en función de sus campos de fecha, que tienen distintas denominaciones.

3-. Indicar a la función *merge* qué se hace con las líneas extra: ***all***

Por lo general, dos conjuntos de datos que se combinan entre sí no suelen tener correspondencia exacta entre sus líneas. De hecho, es muy común, y más en grandes conjuntos de datos, que determinadas líneas de un conjunto no tengan su correspondiente línea en el otro conjunto. En estos casos, por defecto, lo que hace la instrucción *merge* es eliminar esta línea sin correspondencia y no contemplarla en su conjunto de datos final. Así, el nuevo conjunto de datos contendrá sólo aquellas líneas que aparezcan en ambos conjuntos, lo que conlleva inevitablemente la pérdida de datos.

Si lo que queremos es conservar, aunque no tenga correspondencia, las líneas de nuestro conjunto de datos, deberemos usar el comando all, al que le indicaremos el nombre del conjunto del que queremos conservar todas las líneas, para posteriormente activarlo con TRUE. Si utilizamos el comando *all* sin establecer ningún nombre de conjunto y lo activásemos con TRUE, conservaríamos las líneas de ambos conjuntos, rellenándose los huecos de datos con N.A. para ambos conjuntos.

Ej. **datoscontmet ← merge(datoscont, met, by.datoscont="date", by.met="fecha", all.datoscont=TRUE)**, este ejemplo combina nuestro conjunto datoscont (de calidad del aire), con un conjunto de datos de meteorología de la zona (met). Con el comando all aplicado a nuestro conjunto de datos, lo que hacemos es garantizar que no perdemos en la combinación ninguna línea de datos de nuestro conjunto, aunque no tenga correspondencia con la meteorología. No usamos el comando all también con el conjunto de datos meteorológicos, porque este es más extenso temporalmente que el nuestro y añadiría datos "N.A." que perjudicarían a nuestro rendimientos y cálculos estadísticos.

4-. Utilizar sufijos para identificar las columnas de cada conjunto: ***suffixes***

En ocasiones puede ocurrir que tengamos que combinar dos conjuntos de datos con la misma denominación de campo, como por ejemplo en el caso de estar realizando una intercomparación de datos, en cuyo caso nos podemos encontrar con un problema de identificación de campos en el resultado final de *merge*.

Para ello *merge* dispone del comando *suffixes*, que permite añadir sufijos a los campos de cada conjunto de datos, pudiendo así diferenciarlos en el conjunto final.

Ej. **datoscontref ← merge(datoscont, ref, by="date", all.datoscont=TRUE, suffixes=c("-est", "-ref"))**, donde estaremos combinando los datos de nuestra estación con un conjunto de datos de una unidad móvil de referencia utilizada frente a nuestra estación, obteniendo como resultado final nuestro propios datos, por ejemplo "no2-est", frente a los mismos datos de la estación de referencia "no2-ref".

♦ ***Combinar conjuntos de datos por filas: rbind.fill***

La función *merge* es una versión muy mejorada de otra función básica en *R cbind*, que permite combinar los archivos de datos por columnas, siempre y cuando compartan el mismo número de filas, razón por la que este manual no profundiza en dicha instrucción de *R*. Sin embargo, *Openair* no tiene previstas funciones o herramientas para combinar conjuntos de datos por filas.

Esta opción de combinación, donde se quieren juntar dos conjuntos de datos agregando filas en función de las variables disponibles (columnas) es, sin embargo, extraordinariamente útil y necesaria en calidad del aire para, por ejemplo, agregar conjuntos de datos previamente desagregados de la misma estación, o unir los datos obtenidos de ciertos parámetros para diversas ubicaciones con el objeto de realizar un estudio espacial de la distribución de la contaminación (ver uso de funciones como *GoogleMapsPlot* que veremos más adelante).

R prevé la utilización de este modelo de combinación mediante el comando *rbind (row bind)*, que funciona de igual manera que *cbind (column bind)*, y que podemos utilizar para combinar conjuntos de datos por filas. Sin embargo esta combinación requiere que los conjuntos tengan el mismo número de columnas (mismo número de variables a combinar) y que se presenten en el mismo orden.

El Paquete de *R plyr* amplía esta herramienta mediante la función *rbind.fill*, que mejora la anterior función proporcionándole mayor velocidad y versatilidad ya que:

- Tiene en cuenta en la combinación el nombre de las columnas de cada uno de los archivos de datos, de forma que garantiza que las filas se combinan correctamente aun cuando las variables de los distintos archivos no se encuentren en el mismo orden.
- Rellena en las filas aquellos campos de datos que no disponen de variable a la hora de realizar la combinación de conjuntos, utilizando para ello la denominación *"NA"*.
- Permite una ejecución rápida, sencilla y segura de la combinación que se pretende realizar.

Ej. **datoscont← rbind.fill(datoscont1, datoscont2)** se trataría en este caso de la combinación más sencilla a utilizar con esta función, donde estaremos combinando dos conjuntos de datos "datoscont1" y "datoscont2", dando lugar al siguiente resultado:

Supongamos que el primer conjunto de datos dispone de 380 obs (380 filas de datos) y 4 variables (4 columnas: date, pm10, so2, o3), mientras que el segundo conjunto de datos dispone de 430 obs (430 filas de datos) y 6 variables (6 columnas: date, pm10, so2, nh3,03,no2). El conjunto final "datoscont" se construirá sobre la base del primer conjunto de datos dispuesto, rellenandolo en función del segundo, por lo que el resultado final será que "datoscont" dispondrá de 810 obs(810 filas de datos) y 6 variables (6 columnas: date, pm10, so2,o3,nh3,no2).

Ej. **datoscont← rbind.fill(datoscont1 [c("date","o3"], datoscont2 [c("date","o3"])** se trataría en este caso de una combinación algo más compleja a utilizar con esta función, donde estaremos combinando dos conjuntos de datos "datoscont1" y "datoscont2", pero tan sólo seleccionando de ambos dos conjuntos las columnas correspondientes a la fecha y los datos del ozono, todo ello dentro de la propia función.

♦ ***Clasificar los datos:*** ***cutData***

Esta instrucción, propia de *Openair*, permite clasificar los datos de un conjunto de datos agregando un nuevo campo clasificatorio que permitirá *"cortar"* los datos en función de la clasificación seleccionada, de forma que sea posible caracterizar y clasificar los datos en distintas particiones.

Si bien es una función que normalmente se integra en las propias funciones y gráficas de *Openair*, mediante el comando ***type*** que veremos más adelante, puede ser muy interesante conocerla de manera particular para personalizar y precisar, si así fuese conveniente, su manejo.

Por otro lado, normalmente se suele ejecutar esta función sobre el propio conjunto de datos, pues lo que suele hacer la misma es añadir un nuevo campo de datos (salvo que se seleccione como criterio un campo del propio conjunto de datos). Pero al igual que en otras funciones de *R* y *Openair*, es posible indicarle al programa que la ejecute para la generación de un nuevo conjunto de datos, indicando tan sólo el nuevo nombre que queramos darle: ***datoscont_nuevo <- cutData(...)***

La instrucción *cutData* se complementa con distintos comandos que nos permiten caracterizar la tipología de clasificación que queremos, entre los que encontraremos como más útiles los siguientes:

1-. El tipo de clasificación que queremos para el conjunto de datos: ***type***

Mediante este comando complementario podremos decirle a *R* si queremos que la función *cutData* clasifique los datos en base a las estaciones del año "*season",* los días de la semana *"weekday"*, los años *"year",* las horas "*hour",* los fines de semana *"weekend",* las horas de luz *"daylight",* los meses *"month",* etc.

Ej. **datoscont ← cutData(datoscont, type="season"),** al indicar al programa que el tipo de clasificación que se requiere es por estación, type="season", lo que se hará es añadir al paquete de datos datoscont un campo que se denomina "season" y que, en función de la fecha, establecerá cuatro clasificaciones de estación para cada línea de datos: primavera, verano, otoño e invierno.

La clasificación también puede llevarse a cabo sobre un campo de datos existente del conjunto de datos, como por ejemplo el ozono, indicando en el tipo de clasificación el nombre del campo de datos, lo que modificará el campo de datos sustituyendo los datos existentes en función de una clasificación por defecto de los mismos en cuatro cuartiles en función del rango en el que se muevan los datos.

Ej. **datoscont ← cutData(datoscont, type="o3")**, al indicar al programa que el tipo de clasificación que se requiere es en función del campo de ozono, lo que hace Openair es dividir en cuatro cuartiles los datos en función del rango de datos y del percentil que corresponda.

2-. El hemisferio en el que nos encontramos: ***hemisphere***

Dado que determinadas clasificaciones de los datos pueden venir condicionadas por el hemisferio en el que nos encontremos, como por ejemplo las estaciones. *Openair* prevé la utilización por defecto del hemisferio norte, y si queremos utilizar el sur habrá que indicárselo con la instrucción añadida: **hemisphere="southern"**.

3-. El número de particiones que se desea hacer de los datos: ***n.levels***

Es posible indicar a la función *cutData* el número de niveles que deseamos utilizar para una clasificación determinada de datos, especialmente útil cuando se trata de un campo de datos del propio conjunto de datos y deseamos que los cuantiles sean distintos a los predeterminados cuatro cuartiles.

Ej. **datoscont ← cutData(datoscont, type="o3", n.levels=10)**, al indicar al programa que el tipo de clasificación que se requiere es en función del campo de ozono, y que los niveles a utilizar será de 10 cuantiles, por lo que Openair dividirá en diez cuantiles los datos de ozono en función de los percentiles

Esta instrucción, propia de *Openair*, permite clasificar los datos de un conjunto de datos agregando un nuevo campo clasificatorio que permitirá "cortar" los datos en función de la clasificación seleccionada, de forma que sea posible caracterizar y clasificar los datos en distintas particiones.

Si bien es una función que normalmente se integra en las propias funciones y gráficas de *Openair*, mediante comandos como ***type***, puede ser muy interesante conocerla de forma individual para el manejo de los datos, sobretodo si se quiere hacer de una forma más personalizada y/o precisa que lo previsto de forma individual en las distintas funciones de *Openair*.

♦ *Seleccionar datos en función de la fecha: selectByDate*

Openair permite, mediante la instrucción *selectByDate*, escoger una serie de datos de un conjunto determinado existente indicando a la instrucción uno o varios criterios de selección relacionados con el campo de fecha, permitiendo multitud de combinaciones distintas.

Esto convierte a esta instrucción en una de las más necesarias y versátiles para su utilización, tanto de forma individual, como en combinación con otras instrucciones gráficas o de cálculo, ya que puede utilizarse para seleccionar uno o varios años, años y meses, días, horas, o incluso en rango de fechas concreto, todo ello combinando uno o varios de los siguientes comandos:

1-. Seleccionar datos en un rango determinado: ***start*** y/o ***end***

Los comandos *start* y *end* permiten indicar a la instrucción las fechas de inicio y final del rango de datos que se desea seleccionar. Es importante tener en cuenta el formato en el que se encuentra la fecha del conjunto de datos que vamos a seleccionar.

Ej. **datoscont10 ← selectByDate(datoscont, start="15/01/2010", end="30/03/2010")**, mediante esta instrucción estamos creando un nuevo conjunto de datos procedente del conjunto previamente existente datoscont, que contendrá tan sólo aquellos datos que vayan del día 15 de enero de 2010 al 30 de enero de 2010.

2-. Seleccionar datos para un año en concreto: ***year***

El comando *year* permite indicarle a la instrucción el año o años que queremos seleccionar del conjunto de datos.

Ej. **datoscont10 ← selectByDate(datoscont, year=2010)**, mediante esta instrucción estamos creando un nuevo conjunto de datos procedente del conjunto previamente existente datoscont, que contendrá tan sólo aquellos datos que sean del año 2010.

3-. Seleccionar datos para un mes en concreto: ***month***

El comando *month* permite indicarle a la instrucción el mes o meses que queremos seleccionar del conjunto de datos, pudiendo para ello utilizar el nombre del mes, su abreviatura o el número de orden del mismo. En caso de utilizar el nombre del mes, este se introducirá teniendo en cuenta el paquete de idioma que se haya seleccionado para instalar R o la consola que estemos utilizando.

Ej. **datoscont10 ← selectByDate(datoscont, year=2010, month=c(3,5,7))**, mediante esta instrucción estamos creando un nuevo conjunto de datos prodecente del conjunto datoscont, que contendrá tan sólo aquellos datos que sean del año 2010 y de los meses de marzo, mayo y julio.

4-. Seleccionar datos para determinados días: ***day***

También se pueden seleccionar los días para los que se quieren extraer los datos indicando a la instrucción el día de la semana que se desea (Lunes, Martes, etc) para lo cual habrá que tener también en cuenta el paquete de idioma que se haya seleccionado para instalar *R* o la consola de trabajo, o solicitándole que seleccione los días laborales de la semana *"weekday"* (de lunes a viernes) o los días del fin de semana *"weekend"* (sábado y domingo), lo que resulta muy útil por ejemplo para estudiar escenarios de afección por tráfico rodado.

Ej. **datoscont10 ← selectByDate(datoscont, year=2010, day="weekend")**, mediante esta instrucción estamos creando un nuevo conjunto de datos procedente del conjunto datoscont, que contendrá tan sólo aquellos datos que sean del año 2010 y que se correspondan con fines de semana, lo que permitirá conocer los niveles de contaminación básicos existentes en dicha estación de inmisión sin la influencia del tráfico rodado propio de los días laborales.

5-. Seleccionar datos para determinados horas del día: ***hour***

Por último, la instrucción *selectByDate* permite seleccionar una o varias horas mediante el comando *hour*, al cual le indicaremos la hora o las horas en concreto a seleccionar, entre 0 y 23, o lo que es más útil todavía, un rango de horas determinado, tal y como se muestra en el ejemplo.

Ej. **datoscont10 ← selectByDate(datoscont, year=2010, day="weekday", hour=6:11)**, mediante esta instrucción estamos creando un nuevo conjunto de datos procedente del conjunto datoscont, que contendrá tan sólo aquellos datos que sean del año 2010, que se produzcan los días laborales, y dentro de las 06:00 am y las 11:00 am, momento en el que el tráfico rodado es previsiblemente mayor.

♦ *Clasificar los datos en función de la fecha: splitByDate*

De similar manera a como funciona *cutData*, y con el espíritu de la instrucción *selectByDate*, surge la función *splitByDate*, que divide un conjunto de datos en función del criterio definido por el usuario pero, en este caso, sólo sobre el campo principal de la fecha "date" y adoptando esta como una variable más para crear un nuevo campo clasificatorio definido por el usuario.

De esta forma, con la instrucción *splitByDate* se agregará un nuevo campo clasificatorio al conjunto de datos de que disponemos, que permitirá caracterizar y clasificar los datos en función del criterio temporal y la definición dada por el usuario. Desde el punto de vista práctico se puede decir que la función *splitByDate* nos evita tener que estar realizando subconjuntos de datos, con la función selectByDate, para obtener escenarios temporales conocidos, permitiendo mantener la integridad del conjunto de datos durante su estudio, lo cual puede resultar muy útil para la posterior aplicación de instrucciones gráficas de *Openair* como *Taylor-Diagram.*

Para ejecutar la instrucción *splitByDate* se dispone de los siguientes comandos:

1-. Establecer las fechas en las que se produce la partición: ***dates***

Es preciso indicar al programa la fecha o fechas en las que se debe realizar la partición de los datos para su caracterización. Es importante recordar que una sola fecha dispuesta generará dos subconjuntos de datos que habrá que definir con posterioridad, dos fechas generarán a su vez tres subconjuntos, y así sucesivamente.

2-. Establecer las etiquetas que definirán los subconjuntos: ***labels***

Al establecer fechas como puntos de corte iremos estableciendo también subconjuntos de datos que deberemos definirle a la función para que los etiquete dentro del nuevo campo que se creará en el conjunto de datos original. Esto se consigue mediante el comando *labels*, al que se le indicarán las etiquetas a utilizar en el nuevo campo, como por ejemplo: ***labels=c("antes","despues")***.

3-. Personalizar el nombre del nuevo campo: ***name***

Ya hemos comentado que la instrucción *splitByDate* no divide realmente el conjunto de datos, sino que caracteriza el mismo al añadir un nuevo campo que subdivide los datos en subconjuntos en función de las etiquetas designadas. En este sentido, es necesario indicarle a la instrucción el nombre que deseamos asignar a ese nuevo campo, de lo contrario la función dispondrá el nombre por defecto de "*split.by*".

A modo de ejemplo, imaginemos que para un histórico de datos disponible de una estación de control dada, queremos comprobar cómo ha afectado a la calidad del aire de la zona la construcción de una circunvalación próxima a dicha estación. Sabemos que la circunvalación , por ejemplo, estuvo en obras entre el 15 de junio de 2005 y el 23 de octubre de 2007, momento en el que fue inaugurada y puesta en funcionamiento.

Una opción posible sería utilizar la instrucción *selectByDate* para dividir el conjunto de datos y seleccionar tres subconjuntos nuevos de datos que deberíamos estudiar de manera individual, lo que triplicaría el trabajo de evaluación de los datos.

La opción más viable sería utilizar la instrucción *splitByDate*, según se muestra en el siguiente ejemplo, de forma que tendremos siempre el mismo conjunto de datos pero con un nuevo campo que nos servirá con posterioridad para evaluar la calidad del aire con una única función pero generando varias gráficas, utilizando el comando *type*, que veremos más adelante.

El ejemplo que acabamos de comentar adoptaría la siguiente línea de comandos:

Ej. **datoscontsplit <- splitByDate (datoscont, dates=c("15/06/2005","23/10/2007"), labels=c("antes","durante","después"), name="carretera")**, generará conjunto de datos nuevo con un nuevo campo, en el que se dispondrán tres variables de texto, la variable "antes", para todas aquellas fechas que sean anteriores al 15/06/2005 y que establecerán los niveles de base de calidad del aire existentes con anterioridad a la construcción de la circunvalación; la variable "durante", para aquellos datos entre el 15 de junio de 2005 y el 23 de octubre de 2007, fechas en las que se estuvo construyendo la circunvalación, por lo que los niveles de inmisión pudieron estar afectados por estos trabajos de construcción y movimiento de tierras; y la variable "después" , para todas aquellas fechas posteriores al 23 de octubre de 2007 y que definen el nuevo perfil de calidad del aire para la estación de control, seguramente ahora incluyendo una nueva fuente de origen de la contaminación, caracterizada por el incremento en el tráfico rodado fruto de la nueva circunvalación.

♦ *Seleccionar datos en función del valor límite de un contaminante: selectRunning*

Parecida a la instrucción subset que hemos visto anteriormente, la instrucción *selectRunning* selecciona rangos de datos de un conjunto de datos existente en función de la superación de un valor de concentración establecido para un contaminante dato y durante un tiempo determinado. Visto de otra forma, esta instrucción proporciona la herramienta precisa para extraer escenarios de calidad del aire cuando se supera el valor límite establecido para un contaminante específico.

Así, esta instrucción de *Openair* se convierte en una herramienta muy útil y potente para el análisis de escenarios concretos de contaminación atmosférica, tales como la superación de umbrales de información o alerta, especialmente cuando esta información se combina con herramientas gráficas de análisis de datos como *polarPlot*.

La instrucción *selectRunning* requiere de los siguientes comandos básicos:

1-. Seleccionar el contaminante que regirá el criterio de selección: ***pollutant***

Se debe indicar a la instrucción el contaminante que se regirá el criterio de selección y sobre el que se impondrá un valor límite para seleccionar el conjunto de datos.

2-. Seleccionar el valor límite a aplicar al contaminante: ***threshold***

Se deberá indicar a la función el valor límite a aplicar para seleccionar el conjunto de datos que queremos, de forma que se seleccionen aquellos datos del conjunto cuya concentración del contaminante indicado esté por encima del valor seleccionado.

3-. Número de datos contiguos por encima del valor límite: ***run.len***

De forma opcional se puede indicar a la instrucción el número de datos que deberán superar de forma continuada el valor límite indicado. Así, la función buscará aquellos periodos de la longitud indicada en *run.len* en los que todos sus datos superan el valor límite establecido en *threshold.*

Ej. **episodiocont ← selectRunning(datoscont, pollutant="o3", threshold=120, run.len=5)**, mediante esta instrucción estamos creando un subconjunto de datos denominado episodiocont en el que se incluirán todos aquellos datos de datoscont que durante cinco horas consecutivas superen el valor de 120 µg/m3, lo que supone una situación de altos niveles de ozono, normalmente asimilada a una posterior superación de los umbrales de información o alerta.

Instrucciones básicas para el manejo de los datos.

Hasta ahora hemos hablado sólo del procesado y gestión de ventanas y conjuntos de datos, pero se debe destacar que la gestión de datos en *R* se realiza también a través de funciones. Funciones que se aplican sobre los distintos elementos de *R*, tales como variables, conjuntos de datos, vectores, etc. y que pueden arrojar sus resultados sobre la propia consola de *R*, o formar parte de una instrucción más compleja con la que incluso se puede terminar generando un nuevo archivo o modificando uno existente. En este punto, el potencial de *R* se muestra en la capacidad del usuario para combinar y resolver sus necesidades en función de las herramientas dispuestas.

R dispone de multitud de funciones previstas con las que es posible proceder al tratamiento, manejo y procesado de los datos, ya sea para hacer cálculos matemáticos básicos, modificar o recalcular los datos, o para hacer estudios estadísticos de los mismos más complejos. Evidentemente el objeto de este manual no es tratar todas estas funciones, aunque sí que se realizará un repaso por aquellas que se considera de mayor utilidad para el técnico especialista en calidad del aire, desde las más genéricas hasta las funciones propias de *Openair* para la gestión específica de datos de calidad del aire.

Comenzaremos el estudio en este sentido realizando un recorrido rápido pero completo, yendo desde los operadores más primarios hasta las funciones más complejas, pasando por las instrucciones básicas de *R* para el control y manejo de datos que nos proporcionarán la base fundamental para entender y desarrollar con posterioridad nuestro trabajo, para finalizar con las funciones propias personalizadas por *Openair* para el manejo de datos de calidad del aire.

- ***Operadores de R para el manejo de datos:***

Antes de hablar incluso de funciones de *R* para la gestión de datos, debemos hablar de operadores. Los operadores son símbolos que utiliza el sistema para trabajar con las variables y se corresponden con los normalmente utilizados en otros ámbitos de programación. *R* dispone de operadores lógicos y aritméticos, según se expone en la siguiente tabla:

Operadores Aritméticos en R

+	Suma
-	Resta
*	Multiplicación
/	División
^ o **	Exponente

Operadores Lógicos en R

<	Menor que	>	Mayor que
<=	Menor o igual que	>=	Mayor o igual que
==	Exactamente igual a	!=	Distinto a
X|Y	X o Y	X&Y	X e Y

- ***Estructuras de control de datos:***

Como cualquier lenguaje de programación, *R* dispone también de estructuras de control de datos que permiten modificar el flujo de ejecución que tengan las instrucciones que le queramos dar, entre estas estructuras podemos encontrar funciones condicionales de gran utilidad para el manejo de los datos de calidad del aire como:

1-. Estructura condicional SI: ***ifelse***

Si *(IF)* ocurre la condición, sucede algo, y si no ocurre la condición *(ELSE)* sucede otra cosa, siguiendo la siguiente estructura de programación: ***ifelse (*** *condición de entrada****,*** *salida si es cierto****,*** *salida si es falso****)***

2-. Estructura condicional MIENTRAS QUE: ***while***

Ejecuta una instrucción Mientras que *(while)* se produzca una condición, siguiendo la siguiente estructura de programación: ***while (*** *condición de entrada****)*** *función o expresión de salida*

Además de las expuestas *R* cuenta con otras estructuras para el control del flujo entre las que podemos encontrar las funciones ***for***, que ejecuta una expresión el número de veces que le indique el usuario, las funciones ***if*** e ***if - else***, derivadas de la vista anteriormente, la función ***repeat***, para repetir una función o expresión, o la función ***switch***, que evalúa una expresión dada para escoger entre una serie de alternativas o argumentos dados. Además de las funciones ***break***, que interrumpe un cualquier bucle dado por las funciones *for*, *repeat* o *while*, o la función ***next***, que pasa a la siguiente instrucción.

♦ *Funciones matemáticas básicas de R para la gestión de datos de Calidad del Aire:*

R, tal y como se ha visto en anteriores capítulos, es un lenguaje de programación y gestión de datos estadístico, por lo que entre sus múltiples e innumerables instrucciones cuenta con todo tipo de funciones matemáticas a aplicar a variables numéricas o conjuntos de datos en sí mismos, entre las cuales se cuentan algunas tan básicas y fundamentales como:

1-. La suma de un conjunto de datos: ***sum***

Ej. **sum(datoscont$r, na.rm=TRUE)**, devolverá la suma de nuestros datos de lluvia para el conjunto de datos datoscont, aunque debemos recordar siempre que en la fórmula se debe indicar que se eliminen del cálculo los datos nulos.
Si lo que se quiere es sumar variables numéricas directamente, R contempla, tal y como hemos visto antes, el uso de operadores aritméticos, lo que facilita la programación de fórmulas para la realización de operaciones com-

2-. El cálculo de la raíz cuadrada: ***sqrt***

Ej. **sqrt(568516)**, calculará la raíz cuadrada del dato indicado, que en nuestro caso será 754. La operación se puede llevar a cabo, como siempre sobre una variable o sobre un conjunto de variables, o incluso como parte de un cálculo más complejo como, por ejemplo, el error cuadrático medio.

3-. El cálculo de logaritmos: ***log*** , ***log10***

Ej. **log(300)**, calcula el logaritmo de la cifra dada, en nuestro caso 300.

4-. El cálculo de la función exponencial: ***exp***

Ej. **exp(5.703782)**, calcula la función exponencial del número dado. Cuyo resultado será seguramente 300.

ATENCIÓN: ***R no puede tratar matemática ni estadísticamente aquellos conjuntos de datos que contengan datos nulos en alguno de sus puntos (datos NA), hecho este que se da también para alguna que otra función de Openair.***

Este hecho puede evitarse, en casi todas las ocasiones, indicando a la función que no tenga en cuenta estos datos dentro del cálculo que se le indique, con la instrucción na.rm=TRUE.

No obstante, puede resultar tedioso tener que introducir siempre este comando en cada una de las funciones que le vayamos indicando a R, por lo que en ocasiones es recomendable solicitar a R que elimine del conjunto de datos aquellas datos que se muestre como nulos, con la función na.omit que veremos más adelante, de forma previa a su tratamiento estadístico.

♦ *Funciones estadísticas básicas de R para la gestión de datos:*

Además de las funciones matemáticas vistas hasta el momento, *R* dispone de innumerables funciones estadísticas para el estudio y tratamiento de los datos, desde la media más sencilla, con la función *mean*, hasta el tratamiento estadístico más complejo que nos podamos imaginar. Algunas de estas herramientas estadísticas básicas que nos podrían resultar de utilidad en calidad del aire son:

1-. La media de los datos: ***mean***

Ej. **mean(datoscont$pm10, na.rm=TRUE)**, devolverá la media de del total de los datos de partículas PM10 disponible para nuestro conjunto de datos, obviando los datos nulos NA.

2-. Los máximos y mínimos de los datos: ***max***, ***min***

Ej. **max(datoscont$pm10, na.rm=TRUE)**, devolverá el máximo valor del total de los datos de partículas PM10 disponible para nuestro conjunto de datos, obviando los datos nulos NA.

3-. Los cuantiles de los datos: ***quantile***

Ej. **quantile(datoscont$pm10, na.rm=TRUE)**, devolverá los cuartiles propios del conjunto de datos 0% (mínimo), 25%, 50% (mediana), 75% y 100% (máximo).
Ej. **quantile(datoscont$pm10, c(.30, .45, .62, .995) , na.rm=TRUE)**, devolverá los cuantiles personalizados ahora sí por el usuario para el 30%, el 45%, el 62% y el 99.5%. Como veremos más adelante, Openair dispone de sus herramientas propias para ejecutar esta instrucción tan útil y necesaria en Calidad del Aire.

4-. La mediana de los datos: ***median***

Ej. **median(datoscont$pm10, na.rm=TRUE)** devolverá la mediana de las partículas PM10 registradas en nuestro conjunto de datos.

5-. La desviación estándar encontrada en los datos: ***sd***

Ej. **sd(datoscont$pm10, na.rm=TRUE)** devolverá la desviación estandar de las partículas PM10 registradas en nuestro conjunto de datos.

6-. La varianza: ***var***

Ej. **var(datoscont$o3, na.rm=TRUE)** devolverá la varianza del ozono registrado en nuestro conjunto de datos.

7-. La desviación media absoluta: ***mad***

Ej. **mad(datoscont$o3, na.rm=TRUE)** devolverá la desviación media absoluta del ozono registrado en nuestro conjunto de datos.

Hemos visto alguno de los estadísticos básicos de R, pero este lenguaje dispone también de paquetes de cálculo estadístico de enorme utilidad que nos sirven para calcular varios estadísticos a la vez, tales como:

8-. Calcular el rango en el que se mueven los datos: ***range***

Ej. **range(datoscont$pm10, na.rm=TRUE)**, calcula el rango en el que se mueven los datos de PM10 de nuestro conjunto de datos o, lo que es lo mismo, devuelve el mínimo y el máximo en un mismo cálculo.

9-. Devolver un resumen estadístico de los datos de nuestro conjunto: ***summary***

Ej. **summary(datoscont)**, devolverá un resumen donde aparecen los mínimos, máximos, cuartiles, medianas y datos en blanco o datos NA de cada uno de los parámetros de que dispone la tabla, lo que resulta muy útil para hacer una primera comprobación de los datos que hemos importado y disponer de una primera visión general y rápida de nuestro conjunto de datos.
Esta función no requiere que se le eliminen los datos nulos NA, evidentemente.

♦ *Otras funciones básicas de R de interés:*

Pero es que R es algo más que funciones matemáticas, lógicas o estadísticas, y permite llevar a cabo funciones adicionales de gran utilidad para el manejo de los datos de calidad del aire como son:

1-. Redondear una variable numérica o serie de datos a unos decimales establecidos: ***round***

Ej. **datoscont$o3 <- round (datoscont$o3, digits=1)**, redondeará nuestros datos de ozono a un solo decimal, sustituyendo los datos de ozono actuales del conjunto datoscont por datos de ozono debidamente redondeados. Si no se dispone de dígitos, el redondeo es al

2-. Limitar una variable numérica o serie de datos a un número de dígitos significativos: ***signif***

Ej. **signif (1348.75, digits=3)**, redondea la cifra dada a los tres dígitos significativos datos, por lo que su resultado final será de 1350.

3-. Eliminar las filas con datos nulos o designados como NA: ***na.omit***

Tal y como hemos visto anteriormente, los datos nulos o "*NA*" pueden perjudicar a la realización de cálculos estadísticos y matemáticos, por lo que en ocasiones puede resultar útil omitir estos datos de nuestro conjunto de datos, de forma previa a su tratamiento matemático, mediante la instrucción *na.omit*, que eliminará aquellas filas que contengan esta expresión NA.

No obstante, se debe recordar que los datos nulos son también característicos de los conjuntos de datos de calidad del aire, entrando a formar parte incluso de su caracterización estadística, pues se precisan también para el cálculo de los rendimientos de los equipos de medición, por lo que es preciso recordar al usuario que su eliminación del conjunto de datos se debería hacer siempre con cautela y tomando las debidas precauciones, por lo que se recomienda que cuando se lleve a cabo se haga generando un nuevo conjunto de datos, tal y como se propone en el siguiente ejemplo.

Ej. **datoscontsin <- na.omit(datoscont)**, elimina las filas con datos NA del conjunto de datos, guardando los resultados en un nuevo conjunto de datos denominado datoscontsin.

4-. Agregar los datos realizando un cálculo estadístico determinado: ***aggregate***

Con R es posible agregar los datos de un conjunto usando variables que funcionen como listas y que nos permitan agruparlas, con el comando ***by***, así como indicando a la función *aggregate* la función matemática o estadística que queremos aplicar para agregar el dato, con el comando ***FUN***.
Si bien esta función puede resultar de enorme utilidad para la gestión de los datos, se debe recordar al lector que Openair dispone de herramientas de sencilla ejecución que podrían solventar mucho mejor las necesidades de agregación de datos de calidad del aire, tales como ***timeAverage***, que veremos más adelante.

Ej. **datoscontsup <- aggregate(datoscont, by=list(datoscont$superacion), FUN=mean, na.rm=TRUE)**, habiendo creado antes un campo en nuestro conjunto de datos denominado "superación" compuesto por las variables "Superación" y "No superación" que caracterizan el hecho de que la concentración de PM10 sea mayor o menor de 50 µg/m3, lo que conseguimos a continuación es calcular la media de todos los parámetros de nuestro conjunto para dos situaciones concretas, que se superen o que no se superen las partículas, lo cual puede ser de gran utilidad.

Podemos comprobar que con la combinación de las instrucciones y funciones vistas hasta el momento, se pueden llevar a cabo multitud de operaciones para la gestión de los datos, tal y como se expone a continuación a modo de ejemplo para el tratamiento de conjuntos de datos de calidad del aire:

1-. Crear nuevas variables a partir de otras existentes aplicando funciones:

La combinación de las técnicas de cálculo matemático, gestión de datos y creación de variables, permite forzar la creación de nuevas variables en nuestro conjunto de datos

Ej. **datoscont$nox <- round((datoscont$no2 + (datoscont$no * 1.5333)), digits=1),** , genera un nuevo campo en nuestro conjunto de datos que denomina nox, donde se dispondrán los datos calculados de NOx (como NO2) a partir de los datos empíricos disponibles para NO2 y NO de nuestra estación de control, , produciendo además al redondeo del dato obtenido, lo cual puede resultar de enorme utilidad para el estudio posterior de los datos de óxidos de nitrógeno.

2-. Crear nuevas variables para clasificar los datos de un conjunto:

De manera similar a como se generaban los subconjuntos con la instrucción *splitByDate* de Openair que hemos visto en el apartado anterior, en la que se utilizaba la fecha para seccionar los datos de un conjunto y crear subconjuntos, el usuario puede crear sus propios subconjuntos seleccionando los datos que quiera, tan sólo disponiendo una serie de instrucciones de decisión del tipo *ifelse* en la línea de código, tal y como se muestra a continuación:

Ej. **datoscont$ICAo3 <- ifelse (datoscont$o3 >120, c("Calidad Mala"), ifelse (datoscont$o3 > 80, c("Calidad Normal"), c("Calidad Buena"))),** , genera un nuevo campo en nuestro conjunto de datos que denomina ICAo3, en el cual establece una triple clasificación utilizando un blucle ifelse. De esta forma, los datos horarios de ozono se clasifican como un índice de calidad del aire que establece la Calidad como Mala, Normal o Buena.

Después de esta pequeña introducción sobre cómo se puede operar con los datos en *R*, que el lector deberá perdonar por su brevedad, se debe indicar que *Openair* tienen previstas funciones específicas encaminadas al procesado y tratamiento específico de los datos y campos de datos que nos pueden resultar de enorme utilidad para el manejo y gestión de datos de calidad del aire, muchas ellas orientadas a los cálculos específicos para la demostración del cumplimiento de valores límite, umbrales y objetivos legales, y entre las que podemos encontrar las siguientes:

- ***Cálculo de la media móvil de un parámetro:*** ***rollingMean***

La media móvil de un parámetro contaminante es útil por cuanto que su cálculo ayuda a suavizar la representación gráfica del mismo, aproxima los datos de concentración obtenidos a los límites de exposición establecidos científicamente, y además, en el caso de la normativa sobre calidad del aire, porque es requerida como estadístico para el estudio de algunos contaminantes como el ozono o las partículas.

La instrucción *rollingMean* aplicada a un conjunto de datos determinado no modifica ningún campo, sino que genera una variable nueva en el conjunto de datos a la que se le asigna la media móvil calculada para el parámetro designado.

Esta media móvil es un parámetro muy utilizado por la normativa aplicable para fijar valores límite y objetivo de parámetros como el ozono o el CO, por lo que será de enorme utilidad a la hora de tratar grandes series de datos para el cálculo de este estadístico.

La instrucción *rollingMean* requiere para su ejecución de la introducción de los siguientes comandos:

1-. Seleccionar el contaminante sobre el que se calculará la media móvil: ***pollutant***

Será necesario indicar a la instrucción qué contaminante es sobre el que se requiere hacer la media móvil, indicándoselo mediante el comando pollutant.

2-. Indicar el periodo que contemplará dicha media: ***hours***

La media móvil es un estadístico que calcula hora por hora la media para un número de horas anteriores a la hora a la que se asigna el valor, por lo que requiere para su cálculo que se defina en horas el periodo que contemplará la media.

3-. Establecer el nombre que se dará al nuevo campo creado para la media móvil: ***new.name***

Tal y como se ha comentado, la instrucción calculará la media móvil para el periodo indicado y la dispondrá dentro de un nuevo campo en el conjunto de datos de que ya disponemos, no siendo necesario crear un nuevo conjunto de datos, como ocurre con otras instrucciones. Sin embargo, es imprescindible indicar a la instrucción el nombre del mismo.

4-. Número mínimo de datos válidos: ***data.thresh***

Los cálculos estadísticos que realicemos para cualquier parámetro requerirán siempre de un número mínimo de datos válidos, expresado como porcentaje, que garanticen siempre la representatividad del dato estadístico finalmente obtenido. Esta información es la que proporciona el comando data.thresh, al que se deberá asignar un número del 0 al 100.

Ej. **rollingMean(datoscont, pollutant="o3", hours=8, new.name="octoh", data.thresh=75)**, esta será la instrucción que nos permitirá calcular la media móvil octohoraria del ozono, con un porcentaje mínimo de datos del 75% (6 horas), incluyendo en nuestro conjunto de datos un nuevo campo que se denominará "octoh".

♦ *Agregar datos en función de distintos intervalos de tiempo: timeAverage*

Agregar datos en función de distintos intervalos de tiempo es una de las funciones más relevantes y de mayor utilidad que permite ejecutar Openair por cuanto que es base fundamental del trabajo que normalmente se realiza con los datos de calidad del aire.

Agregar datos en función de distintos intervalos de tiempo es una función que se utiliza siempre para aspectos como:

a. El cálculo de distintos estadítiscos para la comprobación del cumplimiento legal en materia de calidad del aire, como por ejemplo el valor límite diario de PM10 o el máximo octohorario diario, en combinación con la herramienta que hemos podido ver en el punto anterior, para el valor objetivo de protección de la salud humana del ozono.
b. Acondicionar conjuntos de datos de manera previa a su combinación en uno sólo, cuantos estos tienen periodos de integración distintos, como por ejemplo, cuando se desean combinar datos horarios del control automático con datos diarios procedentes de muestreos manuales.
c. Facilitar el estudio de grandes series temporales de datos, sobretodo cuando dicho estudio se fundamenta en la realización de gráficas con base temporal, evitando así que un exceso de resolución en el dato oculte aspectos relevantes. En cualquier caso, y puesto que Openair es un software adaptado al tratamiento de grandes series de datos, determinadas funciones gráficas que requerirán de esta herramienta de agregación ya la contemplan dentro de sus comandos.

No obstante, agregar datos en función de distintos intervalos de tiempo es, en sí mismo, una herramienta estadística de tratamiento de datos de enorme utilidad que requerirá de la definición de los siguientes comandos:

1-. Indicar el periodo de tiempo en el que se desean agregar los datos: ***avg.time***

Se le debe indicar a la instrucción el periodo de tiempo en el que se desean agregar los datos, pudiendo este ser: segundos *"sec",* minutos *"min",* horas *"hour",* días *"day",* semanas *"week",* meses *"month",* estaciones del año *"season",* cuatrimestres *"quarter",* o años *"year".* Además, y al objeto de hacer más flexible aún este comando, es posible incluir un número delante del periodo seleccionado para indicar el número de periodos que se desean para la partición del tiempo.

El periodo seleccionado puede ser mayor que el original, y entonces los datos se agregarán, para lo cual se deberá aplicar un estadístico para calcular el dato final, o inferior al original (siempre y cuando sea múltiplo exacto del mismo), en cuyo caso lo que se hace es expandir el valor al nuevo intervalo de tiempo, desagregando el dato disponible y repitiéndolo sucesivamente hasta rellenar el periodo original con el seleccionado.

Ej. **datoscontc <- timeAverage(datoscont, avg.time="15 min")**, mediante esta instrucción lo que conseguimos es crear un nuevo conjunto de datos quinceminutales, a partir de datos horarios, desagregando el dato para repetirlo cuatro veces hasta rellenar la hora a la que estaba referido el conjunto de datos original.

2-. Porcentaje de datos válidos para agregar los datos: ***data.thresh***

Cuando agregamos datos a un cierto periodo de tiempo puede ser necesario indicar a la instrucción *timeAverage* cuál es el porcentaje mínimo de datos requerido para considerar dicha agregación representativa y llevara a cabo el estadístico que se le indique.

De esta forma, si el porcentaje de datos establecido por este comando es de 75, la función requerirá para el periodo seleccionado que existan un mínimo del 75% de datos válidos (por ejemplo, en un día deberán existir un mínimo de 18 datos horarios disponibles para el parámetro en el archivo original). Si el valor es menor al porcentaje indicado la herramienta no calculará el dato y dispondrá en su lugar de un *"N.A."*. Si el número de datos horarios es igual o mayor, calculará el estadístico que se le haya indicado.

3-. Estadístico a utilizar para agregar los datos: ***statistic***

Por defecto, la instrucción *timeAverage* utiliza como estadístico a aplicar la media *"mean"*, por ser este el de uso más frecuente cuando se agregan los datos. Sin embargo, el comando *statistic* nos permite también para esta función calcular el máximo *"max"*, el mínimo *"min"*, la mediana *"median"*, el sumatorio *"sum"*, la frecuencia de aparición de datos *"frecuency"*, la desviación estándar *"sd"*, o incluso el percentil *"percentile"*.

Si seleccionamos como estadístico a calcular el percentil, la herramienta *timeAverage* calculará por defecto el percentil 95. Sin embargo, es posible indicarle un valor distinto si incluimos el comando ***percentile*** en la instrucción con el porcentaje que deseamos aplicar: ***percentile=98.*** No obstante, para el cálculo de percentiles de un conjunto de datos, podremos comprobar más adelante que *Openair* dispone de sus propias herramientas, algo más potentes que esta.

4-. Indicar la fecha de inicio del cálculo: ***start.date***

Se puede forzar a la herramienta a iniciar el cálculo en una fecha específica del conjunto de datos original, ya sea porque sólo se precisan determinados datos, por que es preciso por el periodo de agregación que se ha solicitado, o porque se desea forzar una secuencia determinada en función del periodo seleccionado. En cualquier caso, el comando *start.date* permite indicarle a la función la fecha de inicio completa que precisa el usuario.

5-. Calcular la media vectorial de la velocidad del viento: ***vector.ws***

En ocasiones puede ser útil realizar un cálculo vectorial de la media de la velocidad del viento. No obstante, este hecho no suele tener una utilidad inmediata habitual, por lo que este comando está por defecto desactivado. ***vector.ws=FALSE***.

Ej. **datoscontday <- timeAverage(datoscont, avg.time="day", statistic="mean", datathresh=75)**, calculará la media diaria del conjunto de datos datoscont, siempre que exista un mínimo del 75% de datos diarios (18 horas de datos validos al día).

♦ ***Cálculo de percentiles:*** *calcPercentile*

El cálculo de percentiles, siendo un estadístico sencillo en cuanto a su concepto, suele ser uno de los que más trabajo da al técnico de explotación de datos de calidad del aire, puesto que además muchos software de cálculo no proporcionan herramientas adecuadas para el mismo.

En el caso de *Openair*, se dispone de una instrucción específica que permite el cálculo de percentiles para un parámetro dado, pudiendo incluso seleccionar el periodo de agregación, lo que proporciona al usuario una herramienta sencilla, cómoda y muy versátil para el cálculo de este estadístico.

Si bien, tal y como seguramente el lector ya habrá pensado, la función *timeAverage* vista anteriormente ya contemplaba el cálculo de percentiles entre sus estadísticos, debe decirse que la función *calcPercentile* está destinada a proporcionar al técnico una herramienta más dinámica y versátil que *timeAverage* para el cálculo de uno de los parámetros estadísticos más utilizados en calidad del aire, los percentiles, mostrando los resultados directamente en la consola de R.

La función dispondrá de los siguientes comandos básicos de configuración:

1-. Contaminante sobre el que se quiere realizar el cálculo: ***pollutant***

Se deberá identificar mediante este comando el parámetro o contaminante sobre el que se quiere llevar a cabo el cálculo de los percentiles. *Openair* tan sólo permite actualmente la asignación de un único parámetro por función.

2-. Periodo de integración de datos requerido: ***avg.time***

Se deberá indicar a la función el periodo en el que se quiere llevar a cabo el cálculo de los percentiles, pudiendo ser cualquiera de los contemplados en la función *timeAverage*, entre los que se encuentran los días *"day"*, las semanas *"week"*, los meses *"month"*, o incluso los años *"year"*, entre otros.

3-. El percentil o percentiles a calcular: ***percentile***

Debemos indicar a la función qué percentil o percentiles queremos calcular para el parámetro seleccionado. En esta ocasión será posible indicarle a la función que proceda al cálculo de uno o varios percentiles a la vez.

4-. Establecer el porcentaje mínimo de datos válidos para el cálculo: ***data.thresh***

Cuando realizamos el cálculo para un periodo determinado, es muy posible que sea necesario indicar a la instrucción cuál es el porcentaje mínimo de datos requerido para considerar el cálculo del percentil como representativo para dicho periodo. De esta forma, si el porcentaje de datos es superior al indicado al comando *data.thresh*, se calcularán los percentiles indicados, mientras que en caso contrario se omitirá este cálculo y sen su lugar se dispondrá un *"NA"*.

5-. Indicar la fecha de inicio de los cálculos: ***start***

La instrucción permite que, mediante el comando *start,* se disponga de una fecha específica con la que iniciar el cálculo sobre el conjunto de datos original, evitando así que el cálculo se realice sobre el total de datos disponibles en el mismo, especificando el periodo necesario.

Ej. **percent <- calcPercentile (datoscont, pollutant="o3", avg.time="year", percentiles=C(75,90,95,99.5,99.9), data.thresh=75)**, calculará los percentiles 75, 90, 95, 99.5 y 99.9 del parámetro ozono para cada año de nuestro conjunto de datos, aunque en vez de devolver los datos a nuestra consola de R, puesto que nos interesa conservarlos, grabará los cálculos en un conjunto de datos denominado "percent". Cabe hacer notar en esta ocasión, que los percentiles establecen sus decimales mediante la utilización de puntos, y no de comas, aspecto este muy importante para tener en cuenta.

♦ *Cálculo de estadísticos principales de la calidad del aire: aqStats*

Un de la últimas herramientas incluidas en *Openair* y que mayor utilidad representa para el técnico, es la función *aqStats*, que permite el cálculo de los principales y más comunes datos estadísticos anuales para los distintos contaminantes que se precisen. La herramienta si bien es de sencillo manejo no deja de ser una de las más útiles dispuestas por el paquete *Openair*.

El funcionamiento de la herramienta es bien sencillo, al ejecutarla sobre un conjunto de datos y para una serie de parámetros determinados, lo que hará la función *aqStats* es devolver los siguientes estadísticos para cada uno de los años de datos disponibles en el conjunto de datos y de los contaminantes indicados:

- El porcentaje total de datos válidos disponibles para realizar el cálculo.
- La media anual.
- El máximo valor horario del año.
- El mínimo valor horario del año.
- La mediana.
- El máximo valor diario alcanzado en el año.
- El máximo de la media móvil octohoraria.
- El máximo de la media móvil de 24 horas.
- El percentil 95, aunque en este caso, se debe adelantar que es posible indicarle a la función que calcule más de un percentil, tal y como veremos en los comandos posteriores.

Además de los resultados estadísticos mostrados hasta el momento, la función *aqStats* es capaz de reconocer el o los contaminantes establecidos en la función y calcular estadísticos específicos de cada uno de ellos.

- Si detecta que el contaminante es el Ozono:
  - Número de días en los que el máximo de las medias móviles octohorarias es superior a 100 µg/m3.
  - La AOT 40
- Si detecta que el contaminante es el NO2:
  - El número de horas en las que la concentración de NO2 es superior a 200 µg/m3.
- Si detecta que el contaminante es PM10:
  - El número de días en los que la concentración media diaria es mayor de 50 µg/m3.

Destacar que la herramienta asume en cualquier caso que los datos se encuentran en todos los parámetros en µg/m3, a excepción de CO que se considera que está expresado en mg/m3. Así mismo, la función puede asumir conjuntos de datos que dispongan de datos de diversas ubicaciones, realizando ella misma la agregación por sitios.

Para la ejecución de la función *aqStats* se utilizarán los siguientes parámetros.

1-. Contaminante sobre el que se quiere realizar el cálculo: ***pollutant***

Se deberá identificar mediante este comando el parámetro o contaminante sobre el que se quiere llevar a cabo el cálculo de los percentiles. *Openair* tan permite actualmente la asignación de varios parámetros por función, calculando los estadísticos que correspondan para cada uno de ellos.

2-. Establecer el porcentaje mínimo de datos válidos para el cálculo: ***data.thresh***

Cuando realizamos el cálculo de los distintos estadísticos, es muy posible que sea necesario indicar a la instrucción cuál es el porcentaje mínimo de datos requerido para considerar el cálculo como representativo para los distintos periodos de cálculo establecidos. De esta forma, si el porcentaje de datos es superior al indicado al comando *data.thresh*, se calcularán los percentiles indicados, mientras que en caso contrario se omitirá este cálculo y en su lugar se dispondrá un *"NA".*

3-. Establecer los percentiles a calcular: ***percentile***

Se deberá identificar mediante este comando el parámetro o contaminante sobre el que se quiere llevar a cabo el cálculo de los percentiles. *Openair* tan permite actualmente la asignación de varios parámetros por función, calculando los estadísticos que correspondan para cada uno de ellos.

4-. Transponer los resultados obtenidos: ***transpose***

Por defecto, cuando ejecutamos la función *aqStats*, el sistema devuelve a la consola de *R* un listado de estadísticos organizados por columnas. De esta forma, para cada ubicación y año, tendremos los datos estadísticos por filas, donde cada estadístico se corresponderá con una columna. Sin embargo, este formato de presentación de los datos puede no ser práctico, por lo que con el comando lógico *transpose* es posible indicarle a la función que organice todos los datos estadísticos por filas y los agrupe en función del año, de forma que devuelve a la consola tres columnas, el año, la variable calculada y el dato correspondiente.

Ej. **aqStats(datoscont, pollutant="o3", data.thresh=75, percentiles=c(98,99,99.5,99), transpose=TRUE)**, calculará los datos estadísticos para el ozono de nuestro conjunto de datos, disponiéndose los datos en tres columnas, con el comando transpose, según se puede ver en resultado expuesto a continuación.

year	variable	o3
2009	data.capture	99.00000
2009	mean	50.24088
2009	minimum	2.00000
2009	maximum	133.00000
2009	median	49.12500
2009	max.daily	100.33333
2009	max.rolling.8	124.87500
2009	max.rolling.24	101.04167
2009	percentile.98	112.00000
2009	percentile.99.x	117.00000
2009	percentile.99.5	120.00000
2009	percentile.99.y	117.00000
2009	roll.8.o3.gt.100	58.00000
2009	AOT40	12945.87500
2010	data.capture	98.30000
2010	mean	58.18198
2010	minimum	2.00000
2010	maximum	146.00000
2010	median	61.00000
2010	max.daily	102.12500
2010	max.rolling.8	129.75000
2010	max.rolling.24	108.16667
2010	percentile.98	114.80000
2010	percentile.99.x	120.00000
2010	percentile.99.5	124.00000
2010	percentile.99.y	120.00000
2010	roll.8.o3.gt.100	62.00000
2010	AOT40	15594.00000

INSTRUCCIONES GRÁFICAS EN R Y OPENAIR :

Ya manejamos los datos con maestría y conseguimos lo que buscamos con las cargas y gestiones que hacemos con ellos, pero donde *R* y *Openair* presentan una ventaja muy importante frente a otros software de gestión de datos, y donde se muestra todo su potencial es en el apartado de las instrucciones gráficas, especialmente en el caso de *Openair*.

El lenguaje *R* presenta la ventaja de poder realizar gráficas complejas de grandes series históricas de datos, con complicados cálculos estadísticos y matemáticos, sin que se presenten problemas en el movimiento y gestión de los datos, y permitiendo además una rápida configuración de las gráficas y selección de datos y variables.

Por ejemplo, generar graficas temporales de grandes series históricas en distintos periodos de integración de los datos con otros software es prácticamente imposible, o conlleva el tratamiento previo de los datos durante horas, En el caso de R y *Openair* se pueden graficar datos en series temporales con distintos periodos de integración (semanales, mensuales, etc.), sin tratamiento previo y con una posibilidad de generación gráfica que conlleva como mucho uno o dos minutos.

Hasta aquí la ventaja potencial de graficar con *R* y *Openair*. Sin embargo, se debe añadir que además *Openair* tiene la ventaja sobre *R* de que está preparado para el manejo de datos relativos a la calidad del aire, por lo que:

a) Reconoce directamente muchas de las variables de nuestro archivo de datos, siempre y cuando mantengamos la codificación en ingles de los encabezamientos del conjunto de datos, como por ejemplo: fecha *(date),* velocidad del viento *(ws),* dirección del viento *(wd),* etc.

a) Dispone de gráficas específicamente diseñadas y desarrolladas para el tratamiento de datos sobre calidad del aire.

Lo importante de una gráfica en *R* y *Openair* no es tanto el resultado gráfico final, muy útil para el estudio y análisis de los datos, como el objeto de cálculo estadístico que se genera y que se puede guardar y utilizar, total o parcialmente, para la programación de otras funciones.

Conviene destacar, antes de comenzar a ver cada una de las gráficas que podemos elaborar, que *R* y *Openair* son lenguajes de programación que utilizan funciones para el manejo de conjuntos de datos y sus datos, lo cual se mantiene para las funciones gráficas.

Las gráficas, tanto en *R* como en *Openair* son funciones, idénticas a las vistas hasta el momento, que de hecho podríamos haber incluido en el apartado anterior de *"Instrucciones para el manejo de datos",* pues es lo que realmente hacen.

Cuando ejecutamos una instrucción gráfica, esta calcula los estadísticos pertinentes para el conjunto de datos y las variables indicadas, y posteriormente los representa en una gráfica, vinculada a dichos cálculos que podremos exportar en formato de imagen, si lo que necesitamos es esta vertiente de la gráfica.

De hecho, un prueba de lo expuesto es que con *R* siempre podremos guardar los resultados de la aplicación de dichas funciones asignándose un nombre como objeto de R, al igual que hacíamos con instrucciones vistas anteriormente como subset para la generación de subconjuntos de datos.

Ej. **histograma <- hist (datoscont$o3)**, genera un histograma con la frecuencia de aparición de los distintos niveles de ozono en nuestro archivo de datos y lo guarda como un objeto de R denominado "histograma" que posteriormente podremos editar y manejar a nuestro antojo.

Una función gráfica guardada como objeto *R* contendrá en dicho objeto todos los cálculos realizados sobre el conjunto de datos, es decir todos los cálculos realizados para poder elaborar el gráfico solicitado, distribuidos en distintos apartados. Si queremos comprobar los distintos cálculos realizados para una gráfica en concreto, así como lo guardado de dicha gráfica como objeto de *R*, bastará poner el nombre del objeto en el *promt* de *R*.

Ej. **histograma ,** muestra en la consola el contenido del objeto que antes hemos guardado al generar la gráfica de histograma sobre nuestros datos de ozono del conjunto de datos datoscont.

Al mostrarlo como texto aparecerán cada uno de los campos que componen el objeto gráfico de R (similares a los campos de un conjunto de datos cualquiera) con los datos que forman dichos campos. En el caso que nos ocupa en este ejemplo se mostrarán los puntos de corte generados ($breaks), el conteo de datos realizado para cada rango (*$counts),* las intensidades calculadas del histograma *($intensities),* las densidades *($density)* y los puntos medios de cada rango *($mids),* entre otros atributos de la gráfica.

Una función gráfica guardada como objeto *R* podrá volver a ejecutarse como función gráfica llamándola directamente con la instrucción ***plot***, a la que tan sólo habrá que indicarle el nombre del Objeto de *R* a contemplar dentro de la misma. Al ejecutar *plot* sobre el objeto gráfico correspondiente la instrucción de *R* interpretará el objeto y reproducirá la gráfica tal y como se hubiese programado originalmente.

Ej. **plot (histograma) ,** reproduce la gráfica histograma tal y como fue guardada originalmente, igual que si estuviésemos metiendo el código original **hist (datoscont$o3).**

Al igual que con el comando ***plot***, con el comando ***print*** podemos ejecutar una gráfica de *Openair* guardada como objeto haciendo que, además de reproducir la gráfica original, nos proporcione en la consola principal de *R* una descripción de los datos de salida del objeto.

Ej. **print (histograma) ,** reproduce la gráfica histograma tal y como fue guardada originalmente, y nos proporciona en la consola R los datos de salida de la función.

También podemos extraer los resultados usados para elaborar una gráfica, sin necesidad de reproducir dicha gráfica, con el comando ***results***. Si los datos elaborados por la instrucción gráfica producen varios conjuntos de datos, la instrucción los extraerá como varios conjuntos de datos, mostrándolos de esa forma.

Por lo tanto, tal y como podemos ver, al guardar una función gráfica como objeto de *R*, también podemos aprovecharnos de la información contenida dentro de dicho objeto para guardarla como un conjunto de datos, exportarla, o incluso hacer otro tipo de gráficas.

Ej. **plot(histograma$density) ,** elaborará una gráfica de densidades de los datos obtenidos a partir de la función guardada como histograma, mostrando dichas densidades en una gráfica de dispersión de puntos xy.

Por otro lado, y a pesar de que resulta muy interesante conocer las instrucciones gráficas como instrucciones de tratamiento estadístico y/o matemático de los conjuntos de datos, las funciones gráficas de *R* y *Openair* son, en su resultado final, una gráfica, y como tal se presentan inicialmente como un "dibujo gráfico" mas o menos elaborado, que podremos exportar para su utilización en otros ámbitos de nuestro trabajo, tales como informes, estudios de datos, análisis de la contaminación, etc.

La exportación de las gráficas se puede llevar a cabo utilizando formatos compatibles con los principales editores gráficos del mercado (*pdf, png, bmp, tiff, jpg*). La exportación se lleva a cabo mediante la utilización del menú superior de R cuanto estamos en la ventaja gráfica: ***Archivo -> Guardar como…*** y seleccionando la ubicación correspondiente, o mediante la selección directa del botón ***"Export"*** de la ventana gráfica de *RStudio*.

En todos los casos la imagen se puede guardar en nuestro equipo o, si queremos utilizarla directamente, copiarla a nuestro ***"Clipboard"*** para usarla en el mismo momento.

Instrucciones gráficas básicas en R.

A pesar de que R no es un lenguaje pensado estrictamente para el tratamiento explícito de los datos de calidad del aire, se debe recordar que es el lenguaje base sobre el que se desarrolla *Openair*, por lo que proporciona el entorno de desarrollo gráfico y matemático, y aporta muchas de las instrucciones básicas de configuración de la gráfica (*main, xlab, etc*).

Además, su estudio es importante puesto que ayuda a comprender el funcionamiento en su base de diseño de las funciones gráficas de *Openair* que posteriormente podremos ver, aportando también una serie de instrumentos que se podrían utilizar perfectamente para el estudio de los datos, tal y como se muestra a continuación. En cualquier caso, en este apartado trataremos las principales cuestiones relativas al graficado en *R* y aprenderemos como se ejecutan las funciones gráficas de *R* con mayor interés para la calidad del aire, dejando el resto para la descripción que hagamos con posterioridad del paquete Openair y sus gráficas.

En primer lugar, se debe especificar, al igual que haremos con las gráficas de *Openair* en posteriores apartados, que existen una serie de elementos comunes a todas las gráficas *R* que aparecen en todas o en casi todas las gráficas utilizadas por este lenguaje. De hecho, muchos de estos elementos comunes de las gráficas *R* se suelen encontrar también en la configuración de las gráficas de *Openair* y, en muchas ocasiones, se configuran de idéntica forma, como por ejemplo los títulos de las gráficas, que tanto en el caso de *R* como en el de *Openair* se asignan con el mismo comando.

- Incluir un título en la gráfica: ***main***

  Ej. **plot(datoscont$no2, main="Evolución del NO2")**, lo que nos permite incluir en la gráfica para el NO2 de datoscont un título.

- Incluir un subtítulo en la parte inferior de la gráfica: ***sub***

  Ej. **plot(datoscont$no2, main="Evolución del NO2", sub="Evolución graficada para el mes de julio de 2008")**, lo que nos permite incluir en la gráfica para el NO2 de datoscont, además del título del apartado anterior, un sub-título explicativo de la gráfica.

- Incluir un título de los ejes: ***xlab*** o ***ylab***

  Ej. **plot(datoscont$no2, xlab="horas", ylab="concentración alcanzada")**, nos permite definir el eje x como las horas de la gráfica, mientras que el eje y será la concentración alcanzada para el NO2.

- Establecer los límites de los ejes: ***xlim*** o ***ylim***

  Ej. **plot(datoscont$no2, xlab="horas", ylab="concentración", ylim=c(0,400))**, nos permite definir los límites en los que se moverá el eje de las "y", que es donde se representa las concentraciones del no2. Se puede ver cómo el comando "c()" nos permite introducir a la variable ylim los dos valores que la misma precisa para su ejecución.

- Cambiar los colores de una gráfica: ***col***

  R permite cambiar los colores de una gráfica mediante el comando *col*, seguido por el elemento del que se quiera cambiar el color y asignándole, entre comillas, el código de color estándar utilizado por *R*. Si no se asigna después del comando *col* el elemento concreto, *R* interpreta que lo que se quiere cambiar es el color de la representación gráfica realizada (puntos, columnas, barras, etc).

  Ej. **plot(datoscont$no2, xlab="horas", ylab="concentración", ylim=c(0,400), main="Evolución del NO2", col="blue", col.main="green", col.lab="red")**, utiliza la gráfica de asignación de límites anterior y le asigna un título y los correspondiente de la gráfica (azul "blue") , del título (verde "green" y de las etiquetas de los ejes (rojo "red"). El gusto en la asignación de los colores ya es algo propio de cada usuario.

  Además de este comando, posteriormente podremos comprobar cómo cada gráfica en R dispone de comandos propios o combinaciones de comandos con los que podremos personalizar los colores de los distintos apartados de una gráfica, personalizando así su visualización.

Conocemos ya los principales parámetros de configuración básica de las gráficas de *R* y entendemos cómo se configuran, por lo que ya estamos capacitados para comenzar a programar gráficas básicas en *R* y empezar a sentar las bases en el manejo de las funciones gráficas de *R* y *Openair*.

» Gráfica básica de representación de datos en dos ejes: *plot*

La instrucción *plot* es la forma genérica de llamar a una gráfica. Si la utilizamos en combinación con alguna gráfica previamente grabada como objeto de *R*, tal y como hemos visto anteriormente, lo que *R* hace es representar dicha gráfica en función de los datos guardados. Sin embargo, si la utilizamos directamente sobre un conjunto de datos, realiza la representación lineal típica en dos ejes.

Utilizada de esta última forma, la instrucción requiere que le indiquemos las variables *X* e *Y* a representar, no obstante, si le damos sólo una, será esta la que represente linealmente en función del orden en el que se presenten los datos. Si la serie de datos es muy larga, en ocasiones es preciso indicar a *R* que haga antes algún tipo de tratamiento estadístico de los datos, similar a los vistos en anteriores apartados, para conseguir que la representación sea más clara.

Como función genérica de *R*, *plot* utiliza una gran parte de argumentos de configuración y diseño que con posterioridad nos serán útiles para la configuración de diversos aspectos gráficos en otras funciones gráficas más avanzadas, tal y como podremos ver en cada momento. Así, comandos como *pch, lty, lwd, bg, log,* etc., serán luego de enorme utilidad para configurar y mejorar el diseño gráfico de otras funciones más adelante, razón que explica por qué se trata con semejante profusión y detalle la función *plot* en este manual.

Ej. **plot(datoscont$date, datoscont$no2),** representa en una gráfica lineal los datos de no2 de este paquete de datos frente a los datos correspondientes a la fecha. Si escogiésemos cualquier otra variable la representaría frente a los datos de no2 también.

Variaciones sobre la función genérica *plot*:

1-. Modificar el tipo de gráfica a utilizar para representar los puntos: ***type***

Se le puede indicar a *R* que la gráfica se presente en distintos formatos (por defecto en puntos), tales como: puntos *"p"*, líneas *"l"*, barras *"h"*, puntos unidos por líneas *"o"*, escalones *"s"*, etc.

Ej. **plot(datoscont$no2, type="s"),** dará lugar a un gráfico de escalones de los datos de no2 correspondientes a nuestro archivo de datos.

2-. Modificar la escala del eje de lineal a logarítmica: ***log***

Por defecto las escalas de los ejes se presentan de forma lineal, pero es posible indicarle a *R* que utilice una escala logarítmica para representar el eje que queramos, siendo estos ejes las variables del comando *log*. "x" o "y". Este comando es de gran utilidad cuando, como veremos más adelante, se representan varios parámetros para un solo eje.

Ej. **plot(datoscont$no2, log="y"),** genera un gráfico de puntos de los datos de no2 correspondientes a nuestro archivo de datos, estableciendo la escala del Eje Y como escala logarítmica .

3-. Establecer el tipo de símbolo a utilizar el los puntos de la gráfica: ***pch***

Con el comando *pch* le podemos decir a la gráfica que represente la distribución de puntos utilizando el símbolo que le indiquemos por cada punto. La variable que define al comando *pch* es una variable numérica e identifica a un símbolo predeterminado del tipo compatible con *S* (si el número es entre 1 y 18), adicional de *R* (número entre 19 y 25) o caracteres *ASCII* (si el número está entre 32 y 127).

Ej. **plot(datoscont$no2, pch=16),** genera un gráfico de puntos de los datos de no2 correspondientes a nuestro archivo de datos, disponiendo los puntos de la gráfica como puntos negros rellenos.

4-. Modificar la tipología de la línea de dibujo: ***lty***

En el caso de que como dibujo se utilice una línea, se le puede indicar a *R* el tipo de línea que se quiere disponer identificándola con un número entero, según sea: 0 para líneas en blanco, *1* para líneas sólidas (por defecto), *2* para líneas discontinuas, *3* para líneas de puntos, *4* para líneas de puntos y rallas, etc.

Ej. **plot(datoscont$no2, type="s", lty=2),** dará lugar a un gráfico de escalones de los datos de no2 correspondientes a nuestro archivo de datos donde las líneas serán discontinuas.

5-. Modificar el ancho de la línea de dibujo: ***lwd***

En el caso de que como dibujo se utilice una línea, se le puede indicar a *R* que la línea de dibujo de la gráfica se presente en mayores grosores, puesto que por defecto es *1*. De forma que la gráfica presente una mayor visibilidad.

Ej. **plot(datoscont$no2, type="s", lwd=2)**, dará lugar a un gráfico de escalones de los datos de no2 correspondientes a nuestro archivo de datos con una línea mucho más marcada y visual.

6-. Cambiar el color de fondo de la gráfica: ***bg***

Aunque sólo se trata de un efecto estético, en ocasiones puede ser útil cambiar el color de fondo de una gráfica. Esto se consigue indicando al comando *bg* el tipo de color que queremos utilizar como fondo. Habrá que recordar siempre que el color del fondo se debe escribir en inglés.

7-. Encuadrar la gráfica: ***frame.plot***

El comando *frame.plot* es un comando lógico, es decir, utiliza las variables *TRUE* o *FALSE*, para activar o desactivar el encuadre que rodea toda la gráfica como si fuese una caja. Por defecto viene activado en la función *plot* de R.

8-. Modificar el aspecto de representación de los datos: ***asp***

El comando *asp* de *R* requiere un factor para establecer una relación del aspecto que presentarán las unidades a utilizar en los Ejes *X* e *Y*. Así, si el valor dado al comando es positivo (*asp=a, donde a>0*), entonces la relación será del tipo *x=ay*, aplicando el factor a la representación de las unidades en el eje Y. Esto puede ser de enorme utilidad, por ejemplo, para redimensionar la representación de planos con latitud / longitud.

9-. Representar los ejes de la gráfica: ***axes***

Es posible indicar a la función que omita la representación de los ejes, desactivando el comando lógico *axes* con la expresión que ya conocemos ***axes=FALSE***, que al ejecutar presentará la gráfica sin ejes, sólo con el área de representación.

Si utilizamos los comandos *xaxt* o *yaxt* con la variable *"n"* podemos indicar a la gráfica que elimine tan sólo uno de los ejes, el Eje X si utilizamos ***xaxt="n"***, o el Eje *Y* si utilizamos ***yaxt="n"***, lo cual aporta mayor flexibilidad que el propio comando *axes*, no obstante, la utilización de estos comandos queda especificada en posteriores puntos con mayor detalle.

10-. Modificar el tipo de caja a representar: ***bty***

Otra opción de configuración gráfica interesante es la posibilidad de representar la caja de la gráfica como una caja completamente cerrada en sus cuatro lados, la opción por defecto, que se correspondería al comando ***bty="o"***, o proceder a representarla abierta por uno o varios de sus lados con las variables string siguientes: *"l", "c", "7","u" o "]".*

11-. Incrementar o disminuir el tamaño de representación de diversos elementos en la gráfica: ***cex***

El comando *cex* permite también definir el factor para incrementar o disminuir el tamaño de diversos elementos de nuestra gráfica, añadiendo al propio comando el elemento a modificar y el factor a utilizar, este último como variable numérica, tal y como ocurría con el comando col visto al principio de este capítulo, y tal y como se muestra en el siguiente ejemplo.

Con el comando *cex* se puede modificar el tamaño de las etiquetas de los ejes, con *cex.lab*, el tamaño de las escalas de los propios ejes, con *cex.axis*, el del título o el subtítulo de la gráfica, con *cex.main* o cex.sub, etc. Utilizado sin ningún añadido, lo que hace es cambiar el tamaño de la gráfica, en nuestro caso en concreto el tamaño de los puntos representados.

Ej. **plot(datoscont$no2, type="s", cex.lab=1.5, cex.axis=0.5)**, dará lugar a un gráfico de escalones de los datos de no2 correspondientes a nuestro archivo de datos, tal y como hemos visto en nuestro ejemplo anterior, salvo que en este ocasión se incrementará el tamaño de las etiquetas de los ejes, y se diminuirá el tamaño de las unidades representadas en los mismos.

12-. Designar el color para el primer plano del cuadro de la gráfica: ***fg***

Con el comando *fg* (*foreground*) es posible designar un color específico para el primer plano del cuadro de la gráfica, o lo que es lo mismo, lo que sería los márgenes laterales del área de representación gráfica. El comando deberá en todo caso ir acompañado del color, utilizando su denominación en inglés.

13-. Establecer la fuente del texto de distintos elementos de la gráfica: ***font***

El comando *font* permite definir el tipo de fuente de texto a utilizar en diversos elementos de nuestra gráfica, añadiendo al propio comando el elemento a modificar y el factor numérico a utilizar. Este último se corresponderá con las siguientes codificaciones: *1* - texto plano, *2* - negrita, *3* - itálica, *4* - negrita itálica, etc.

Con el comando *font* se puede modificar el tipo de texto de las etiquetas de los ejes, con *font.lab*, el de las escalas de los propios ejes, con *font.axis*, el formato del título, con *font.main*, o el del subtítulo de la gráfica, con *font.sub.*

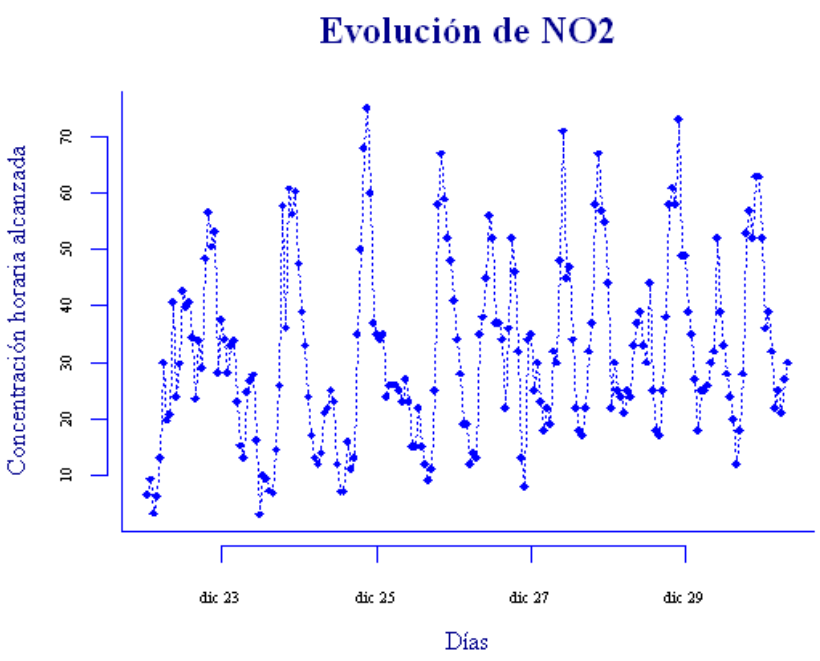
También se puede utilizar en la gráfica el comando *family,* que permite configurar el tipo de fuente de letra a utilizar en la gráfica generada, incluyéndolo como variable de texto con el comando. Por defecto, en R se encuentran los tipos *"serif", "sans"* y *"mono",* aunque los distintos paquetes pueden modificar estos tipos ampliándolos o eliminándolos.

14-. Cambiar la orientación de las etiquetas que forman la escala de los ejes: ***las***

Por defecto *R* presentará las etiquetas que forman las escalas de los ejes *X* e *Y* de forma perpendicular al propio eje, por ser esta la forma que permite una mayor economía espacial. Sin embargo, el usuario podrá cambiar la orientación de las etiquetas asignando al comando las los siguientes números, atendiendo a la siguiente codificación: *0* - siempre paralelas al eje, *1*- siempre horizontales, *2*- siempre perpendiculares al eje, o *3* - siempre verticales.

15-. Modificar la distancia existente entre los distintos elementos de la gráfica y el área de dibujo: ***mgp***

Con el comando *mgp* es posible variar la distancia o margen que separa a los distintos elementos de la gráfica del área de dibujo. Este comando necesita que se le adjunte un vector con tres variables que significarán lo siguiente: La primera la distancia existente entre los títulos de los ejes y la gráfica, la segunda la distancia entre la numeración de los ejes y la gráfica y, la tercera la distancia entre la propia escala y los ejes. Por defecto *R* presentará el comando con la configuración ***mgp=c(3,1,0)***.




```
>plot(datoscont$date, datoscont$no2, type="o", pch=18, lty=3, cex=0.8,
col="blue", lwd=1.5, fg="blue", bty="l", cex.axis=0.6, main="Evolución de
NO2", cex.main=1.5, col.main="darkblue", col.lab="darkblue", xlab="Días",
ylab="Concentración horaria alcanzada", family="serif", mgp=c(3,1,0.5)
```

Ya hemos construido nuestra primera gráfica en *R* que, aunque sencilla en su presentación y conceptos de desarrollo, es fundamental para comprender el concepto de diseño gráfico seguido por *R* y, por lo tanto, también por *Openair* y otros paquetes de representación gráfica. Pero esto no acaba aquí….

La generación de una gráfica en *R* no muere en el momento en el que dicha gráfica se genera en una ventana aparte, sino que comienza en ese mismo momento, ya que *R* permite añadir y modificar los elementos de dicha gráfica, lo que es extraordinariamente útil para, por ejemplo, generar varias gráficas en una, permitiendo así la intercomparación de datos, o añadir elementos gráficos que faciliten la interpretación de la gráfica, entre otros.

La inclusión de nuevos elementos en nuestra gráfica podrá realizarse, por ejemplo, a través de las siguientes instrucciones gráficas básicas, sobre las que podremos actuar también para su diseño con los comandos propios vistos para la función *plot* :

Instrucciones gráficas básicas que se pueden añadir a la función:

1-. Añadir una línea recta a la gráfica: ***abline***

Con la instrucción ***abline*** podemos añadir una línea recta a la gráfica, que nos podrá ser de utilidad para marcar límites, referencias, etc. Para incluir una línea recta horizontal, bastaría con indicar a la instrucción el valor del comando ***h***, que establece el valor del Eje Y, si lo que queremos es incluir una línea vertical, el comando será **v**, y el valor a contemplar se corresponderá con el Eje X. Si la línea es una pendiente, habrá que indicar a la instrucción los valores del punto de intercepción, mediante el comando ***a***, y de la pendiente, mediante el comando ***b***.

2-. Añadir una flecha a nuestra gráfica: ***arrows***

Con la instrucción arrows se pueden añadir flechas a nuestra gráfica, para lo cual indicaremos los siguientes parámetros de configuración básicos, marcando con los comandos ***x0*** e ***yo*** las coordenadas x/y de inicio de la flecha, y con los comandos ***x1*** e ***y1*** las coordenadas x/y de final de la flecha, entre otros comandos de configuración.

3-. Añadir una cuadrícula a una gráfica: ***grid***

Disponer de una cuadrícula en la gráfica puede ser de enorme utilidad para interpretar correctamente los datos dispuestos. Si utilizamos la instrucción ***grid()***, sin ningún comando, podemos ajustar rápida y directamente una cuadrícula a los ejes de nuestra gráfica. No obstante, podemos utilizar algunos comandos como ***nx***, que indica el número de celdas que queremos que tenga la cuadrícula para el Eje X, ***ny***, que indica lo mismo pero para el Eje Y, o ***equilogs***, que es un comando lógico que sirve para ajustar la cuadrícula cuando el eje se representa en escala logarítmica.

4-. Añadir una línea a la gráfica: ***lines***

Con este comando se pueden añadir líneas adicionales a las ya representadas en la propia gráfica. Lo único que se precisa es que se le indiquen los parámetros o datos a representar, y el tipo de línea a utilizar, con el comando ***type*** visto anteriormente en la gráfica *plot.*

5-. Añadir puntos a una gráfica: ***points***

Con este comando se pueden añadir puntos adicionales a los ya representados en la propia gráfica. Lo único que se precisa es que se le indiquen las coordenadas de ubicación de dichos puntos y el tipo de representación gráfica a utilizar, con el comando ***pch*** visto anteriormente en la gráfica *plot.*

6-. Añadir una línea de marcas: ***rug***

Se puede incluir una línea de marcas interiores a la gráfica para representar una serie de valores en la escala del eje que se indique, en lo que en inglés se denomina *rug* (o alfombra de datos). Esta herramienta puede resultar de utilidad para representar en una dimensión la distribución de los datos de un parámetro, complementando así gráficas como los histogramas, las gráficas de densidad, o los gráficos de datos agregados.

Para configurar esta instrucción se precisa como mínimo indicar el conjunto de datos a utilizar, aunque también se pueden utilizar otros comandos como ***ticksize***, que establece la longitud de las marcas representadas, o ***side***, que establece si estas marcas se presentan en la parte inferior de la gráfica, junto al eje *X*, con el valor *1*, junto al eje *Y*, con el valor *2*, en la parte superior del área de la gráfica, con el valor *3*, o en la parte derecha, con el valor *4.*

7-. Añadir texto a la gráfica: ***text***

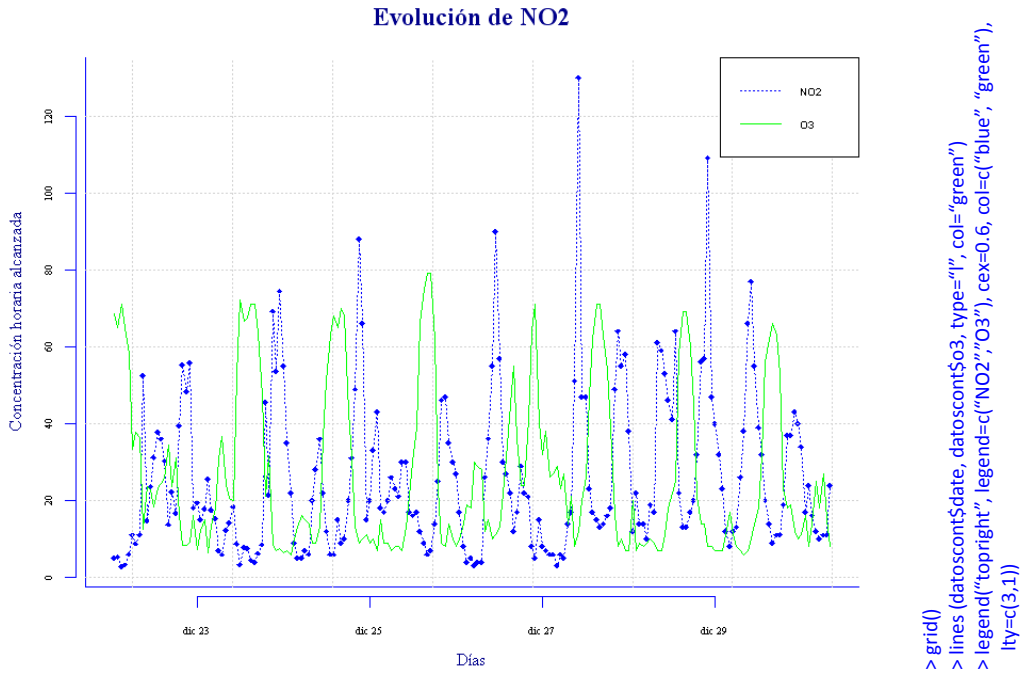
Se puede añadir también texto a la gráfica, aspecto este de especial interés para incluir, por ejemplo, las etiquetas a determinados elementos de la gráfica. Para configurar esta instrucción es necesario indicar a la misma en primer lugar las coordenadas x e y a utilizar, que pueden ser variables numéricas que se correspondan con la escala de los ejes, o vectores (si queremos introducir más de un texto o un grupo de etiquetas). El resto de comandos nos permiten: Introducir las etiquetas que queremos que aparezcan como texto en la gráfica, mediante el comando ***labels***, indicar la posición del texto o etiqueta respecto a las coordenadas, mediante el comando ***pos***, al que se le puede dar el valor de *1* (inferior), 2 (a la izquierda), 3, (superior), 4 (a la derecha), o el comando ***cex***, como factor para modificar el tamaño de la etiqueta, tal y como ya vimos en *plot*, entre otros comandos.

8-. Añadir una leyenda a la gráfica: ***legend***

Se puede incluir también una leyenda a la gráfica, usada normalmente para la descripción de escalas o de los elementos de un área gráfica. En nuestro caso *R* prevé la inclusión de una leyenda en la gráfica atendiendo a multitud de parámetros de configuración entre los que podemos encontrar como obligatorios las coordenadas de ubicación *x/y* de la leyenda, aunque también podríamos utilizar la instrucción *locator**, o las abreviaturas previstas por *R "topright", "top", "bottom", "bottomleft", "center",* etc. Además de las coordenadas, también se debe indicar a la instrucción *legend,* mediante texto o un vector de textos, las denominaciones a incluir en la leyenda.

Junto con estas variables obligatorias, la instrucción *legend* contempla comandos de configuración como: ***fill***, para establecer cajas que aparezcan al lado de los textos de la leyenda con los colores que se especifiquen, ***lty***, para establecer el tipo de líneas a representar al lado de las cajas, ***col***, para establecer los colores de los elementos de al lado de los textos, entre otros muchos comandos que se podrán consultar en la ayuda de *R*.

De esta forma, si ampliamos la gráfica anterior e incluimos diversas instrucciones adicionales para su mejora y ampliación, tal y como se observa en el siguiente ejemplo, podemos obtener infinidad de posibilidades a la hora de graficar datos, entre las que se muestra la siguiente:



* Tal y como se ha comentado antes, una gráfica en R es un elemento vivo con el que se puede interactuar. Entre otras opciones *//* presenta la instrucción *locator()*, que nos permite obtener las coordenadas de un elemento gráfico para hacer uso de las mismas . La instrucción *locator* se puede utilizar por ejemplo en combinación con otras instrucciones gráficas que requieran de unas coordenadas definidas por el usuario. Para conocer más sobre este tema se puede consultar la ayuda del paquete de R "Graphics".

» Gráfica básica de distribución de datos en un histograma: *hist*

Otra gráfica interesante a utilizar en el análisis de datos de calidad del aire son los histogramas. Los histogramas son gráficas de conteo, en las que es posible representar la frecuencia de aparición de datos de ciertos contaminantes o parámetros de calidad del aire, lo que permite por ejemplo comprobar la distribución de los niveles en inmisión de un determinado parámetro.

Para construir un histograma lo que hace la función es dividir el Eje X en intervalos, también denominados clases, para luego proceder al conteo del número de datos que se corresponde con cada intervalo.

Los histogramas pueden ser ampliamente modificados en su ejecución, dando muchísimo juego en su representación. Todo buen programa estadístico que se precie nos permitirá siempre cambiar las opciones de diseño del gráfico, y en este caso *R* no podía ser menos, tal y como veremos a continuación.

Variaciones sobre la función:

1-. Modificar el número de intervalos a graficar: ***breaks***

Con el comando breaks se le puede indicar a la gráfica que modifique el número de clases o intervalos en los que dividir el Eje X. Si bien el histograma elaborado por *R* ya calcula el número de intervalos idóneo para el conjunto de datos, podría ser interesante que el usuario estableciese los suyos propios.

En este sentido, muchos expertos recomiendan siempre que los intervalos sean iguales, para evitar la distorsión que pudiera darse en la observación de la gráfica si se utilizan celdas que no sean equidistantes. Para hacer esto bastará con indicarle a R el número aproximado de cortes que se requieren, tal y como se muestra en el ejemplo:

Ej. **hist(datoscont$o3, breaks=16),** realizará un histograma un número aproximado de 16 puntos de corte entre clases, de forma aproximada y siempre en función del rango de datos.

Por otro laso, otros autores consideran que la utilización de rangos distintos para los intervalos puede mejorar la ilustración de la gráfica, motivo por el que la función *hist* permite que el usuario indique al comando *breaks* los puntos concretos de corte en el Eje *X*, definiendo así con precisión los intervalos deseados:

Ej. **hist(datoscont$o3, breaks=c(2,50,120,180,210)),** realizará un histograma para los datos de ozono de nuestro conjunto de datos teniendo en cuenta los puntos de corte marcados a breaks, por lo que elaborará cuatro intervalos o clases. Es importante tener en cuenta que, de utilizar esta fórmula debemos iniciar y finalizar el vector de números con el mínimo de los datos y el máximo, respectivamente. De no hacerlo así, algún dato quedaría fuera de la representación y la gráfica no se ejecutaría.

2-. Utilizar la frecuencia para la representación de las barras: ***freq***

El histograma puede representar las frecuencias de aparición de un parámetro en un rango determinado, con el conteo clásico mencionado de datos, tal y como hace por defecto si los rangos establecidos son equidistantes, con ***freq=TRUE***, o representar la frecuencia relativa o densidad de probabilidades encontrada para cada intervalo, con ***freq=FALSE***, tal y como hace por defecto si los rangos establecidos por la gráfica son de distintas dimensiones.

En cualquier caso, las opciones predeterminadas pueden ser modificadas y el usuario podrá dar el valor que considere pertinente al comando lógico *freq*.

3-. Hacer los intervalos abiertos o cerrados: ***right***

Con el comando lógico *right* podemos indicarle a la función si los intervalos a adoptar están cerrados por la derecha o valor superior, y por lo tanto abiertos por la izquierda, ***right=TRUE***, o al contrario, con ***right=FALSE***.

De esta forma una función con *right=TRUE*, presentará un intervalo del tipo *(a,b]* y por lo tanto incluirá en dicho intervalo todos los valores que sean mayores que *a* y menores o iguales que *b*.

En caos de que especifiquemos a la gráfica el tipo de intervalo a utilizar con el comando right, podremos indicarle a su vez qué es lo que tiene que hacer con el intervalo abierto en la primera barra del histograma (si *right=TRUE*) o en la última barra del mismo (si *right=FALSE*), con el comando lógico ***include.lowest***, que vendrá activado por defecto.

4-. Dibujar líneas de sombreado en la gráfica: ***density***

Entre sus opciones de configuración, *hist* tiene prevista la posibilidad de dibujar las áreas de las barras con líneas de sombreado sobre las que el usuario podrá establecer su densidad en el dibujo, con el comando *density*, al que asignará el número de líneas por área de dibujo.

También podrá definirse por parte del usuario el ángulo con el que se dibujará la línea, con el comando *angle*, al que se le podrá asignar cualquier valor angular en grados.

Se deberá tener en cuenta que si asignamos líneas para el relleno del histograma, a la hora de asignar color a dicho relleno, con el comando *col*, realmente le estaremos asignando el color a las líneas dibujadas, y no a las áreas.

5-. Establecer las etiquetas de las barras: ***labels***

Con el comando *labels* se puede indicar a la función que disponga de etiquetas para cada una de las columnas dispuestas en la gráfica, identificando así numéricamente la frecuencia o densidad de que disponga cada intervalo. Para hacerlo basta indicar que ***labels=TRUE***.

No obstante, el comando *labels* también puede usarse como un comando vectorial de texto e introducir en él los nombres o etiquetas que deseamos asignar a cada una de las columnas, si así quisiese el usuario.

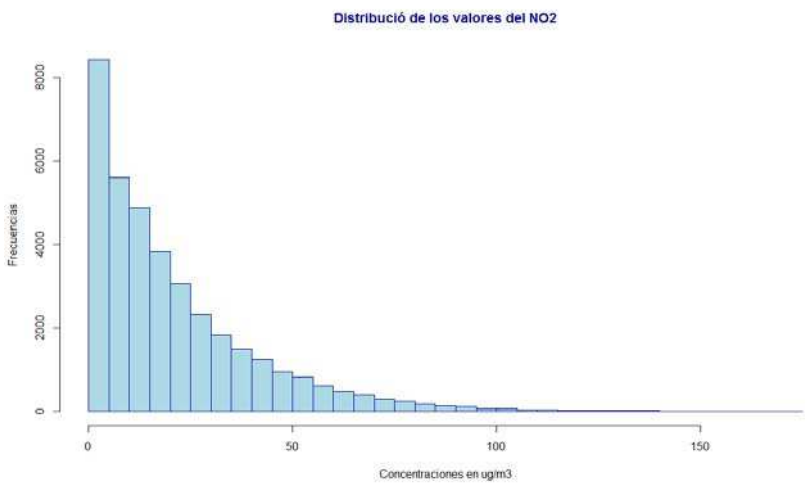
Ej. **hist(datoscont$o3, breaks=c(2,50,120,180,210), labels=c("1 rango", "2 rango", "3 rango", "4 rango"))**, realizará un histograma para los datos de ozono de nuestro conjunto de datos teniendo en cuenta los puntos de corte marcados a breaks, tal y como hemos visto en el ejemplo anterior, y luego asigna las etiquetas "1 rango" al rango de 2 a 50, "2 rango" al rango de 50 a 120, y así sucesivamente.

6-. Asignar un color al borde que dibuja las barras del histograma: ***border***

La gráfica de histogramas *hist* dispone de las opciones de configuración de color normales que ya hemos visto en R, con el comando col y su posible aplicación a los distintos puntos o parámetros de la gráfica, tales como col.main. En el caso de los histogramas, además podemos dotar de color al borde del histograma, con el comando border, al que deberemos asignar, como siempre, el nombre de un color en inglés.

7-. Dibujar los Ejes X e Y de la gráfica: ***axes***

Por defecto la gráfica viene acompañada siempre por los correspondientes ejes de representación debidamente escalados en función de las unidades del parámetro (Eje *X*) y de las frecuencias o densidades encontradas (Eje *Y*). Sin embargo, con el comando lógico *axes*, es posible indicar a la gráfica que no represente los ejes, ***axes=FALSE***.




```
>hist (datoscont$no2, breaks=40, right=TRUE, bor-
der="darkblue", col.main="darkblue",
main="Distribución de los valores de NO2",
col="lightblue", xlab="Concentraciones en ug/m3",
ylab="Frecuencias").
```

Si bien los histogramas presentan una enorme utilidad para comprobar la distribución que presentan los datos de un parámetro determinado por rango de concentración, tal y como hemos podido ver, presentan el problema de que las frecuencias se presentan por tramos y la selección de los mismos puede dificultar o enmascarar la visualización correcta de la distribución, ya que esta dependerá de los puntos de origen y la amplitud de los intervalos.

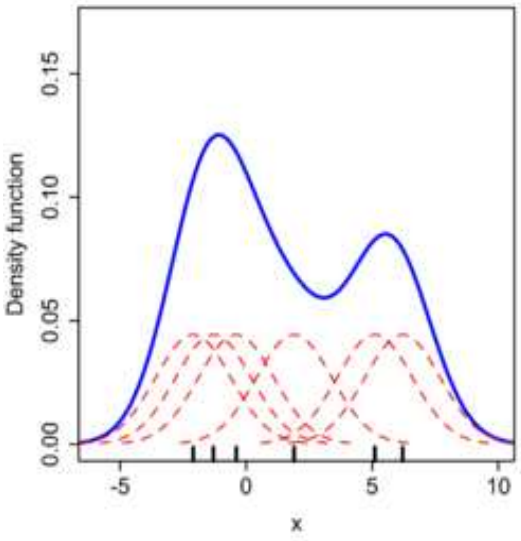
En este sentido, existe funciones matemáticas como la estimación de la densidad que solventan este problema, eliminando la discontinuidad que dan los histogramas, y evitando la distorsión al realizar la estimación centrándose en cada valor. De hecho, la propia función *hist* de *R* presenta como resultado de su ejecución un apartado denominado densidad *$density*, que proporciona los puntos medios obtenidos para cada uno de los rangos, permitiendo así la representación de una curva suavizada ajustada al histograma.

No obstante, y puesto que resulta interesante en cuanto a su aplicación para calidad del aire y como desarrollo al actual apartado, se procederá a describir a continuación las principales ventajas, propiedades y aplicaciones del cálculo de la densidad.

Una función muy utilizada en este sentido, y estrechamente relacionada con los histogramas, es la estimación de la densidad *kernel* de un conjunto de datos, que es un método no paramétrico muy utilizado para calcular la función de la densidad de probabilidades de una variable de la que se desconoce su distribución. Básicamente, lo que hace dicha función es adaptar una línea suavizada y continua a la distribución obtenida de densidades de un conjunto de datos. Para ello utiliza los siguientes elementos:

a) La función de ponderación K *(kernel)* o función de núcleo, que se utiliza como base para la estimación de la función de densidad del conjunto de datos. Debiendo satisfacer ciertas condiciones de simetría y regularidad.

   Se utilizan comúnmente un rango de funciones *kernel* básicas entre las que se encuentran la función uniforme o función rectangular, la función triangular, la función *Epanechnikov,* la función *gausiana*, *cuártico o biponderado, triponderado, tricubo* o *arco-coseno.*



Estimación de tipo núcleo mostrando los núcleos individuales.

b) El ancho de banda - *h*, similar al rango en los histogramas, es un parámetro de suavizado que establece el ancho de la ventana para el ajuste de la función de núcleo. Este ancho de banda es un parámetro de designación libre y tiene una fuerte influencia sobre el resultado final.

   La selección de un ancho de banda adecuado es fundamental para conseguir una buena estimación de la densidad. Un ancho de banda grande dará lugar a estimaciones de la densidad sobresuavizadas con poca varianza y mucho sesgo, y sin embargo un ancho de banda pequeño dará lugar a todo lo contrario.

La estimación de la densidad *kernel* está implementada directamente en *R* a través de la función ***density***, que presenta varios comandos básicos adicionales para permitir su personalización y configuración, tal y como se ve a continuación:

Principales variaciones sobre la función:

1-. Establecer el ancho de banda de la función: ***bw***

El ancho de banda en la función de densidad se puede seleccionar bien asignando al comando *bw* una variable numérica directamente dada por el usuario, bien seleccionando mediante una variable de texto predeterminada los métodos de cálculo del ancho de banda de que dispone R para las funciones de densidad kernel, entre los que encontramos *"nrd0"*, basado en la ecuaciones de Silverman (1986) y que la función utiliza históricamente por defecto aunque no por ello signifique que sea el más adecuado, *"nrd"* basado en la variación más común dada por Scott (1992), *"ucv"* o *"bcv"* para los métodos de validación cruzada y, por último *"sj"*, quizá el más recomendado, basado en los métodos de *Sheather & Jones (1991)*, aunque de costosa aplicación en caso de realizar minería de datos sobre grandes bloques.

Se puede añadir a la determinación del ancho de banda un factor multiplicador mediante el comando ***adjust***, que lo que hace es multiplicar el ancho de banda resultante por la variable numérica que se le indique a este comando.

2-. Establecer la función de ponderación K: ***kernel***

R permite la selección de la función de ponderación K (kernel), o función de núcleo, asignando al comando kernel una variable de texto que se correspondería con las distintas funciones disponibles, entre las que se cuentan: *"gaussian"*, que es la que está establecida por defecto, *"rectangular", triangular", "epanechnikov", "biweight", "cosine"* u *"optocosine"*. Se puede llamar también a estas funciones sin necesidad de escribir todo el nombre, sino acudiendo tan sólo su primera letra.

3-. Eliminar los datos nulos o inexistentes del cálculo de la densidad: ***na.rm***

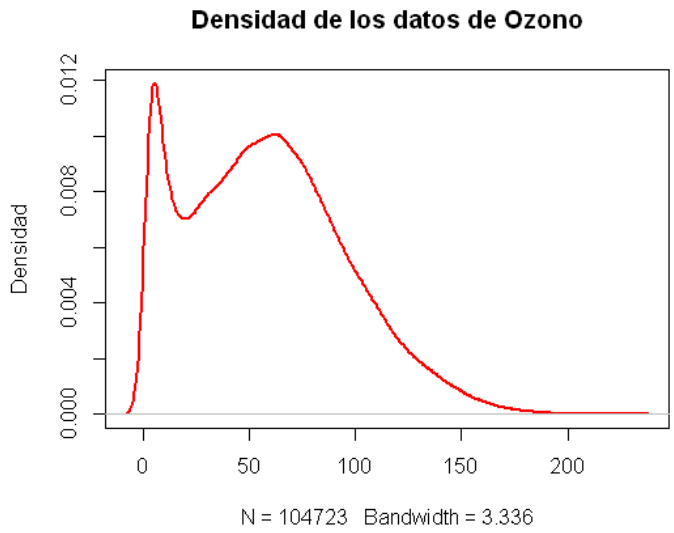
Si nuestro conjunto de datos no está formado exclusivamente por variables numéricas, sino que incluye algunos datos *"NA"* correspondientes a huecos de datos o datos nulos, algo que es extraordinariamente común en calidad del aire, la densidad del conjunto no podrá ser calculada y la función nos devolverá un error. Para estos casos el comando *na.rm* nos permite indicar a la función que elimine este tipo de variables no numéricas del cálculo, procediendo a calcular la densidad exclusivamente con las variables numéricas de nuestro conjunto de datos.

4-. Establecer el rango de valores para el que se desea calcular la densidad: ***from***, ***to***

En muchas ocasiones puede ocurrir que nos interese calcular la densidad para todo el rango de valores, ya sea porque nos interesa conocer la distribución en un rango específico, o porque queremos evitar la distorsión generada en las colas de los datos. Esta limitación en el rango de cálculo de la densidad se consigue con los comandos ***from***, que establece el límite inferior del rango con una variable numérica, y ***to***, que establece el límite superior.

Graficar la densidad calculada es luego tan sencillo como guardar los datos obtenidos como un objeto de *R* y representarlos con un gráfica genérica *plot*, o directamente mezclar ambas funciones en una sola , y graficar con *plot* la función *density*, tal como muestra el siguiente ejemplo:

Ej. **plot(density (datoscont$o3, na.rm=TRUE), main"Densidad de los datos de Ozono", ylim="Densidad", col="red", lwd=2)** , obtiene la función de densidad de kernel para el conjunto de datos de ozono seleccionado, omitiendo los datos nulos directamente, para grafica directamente la función de densidad obtenida mediante la función *plot,* incluyendo en la etiqueta de x los datos correspondientes al número de datos válidos y el ancho de banda utilizado por la función de densidad.

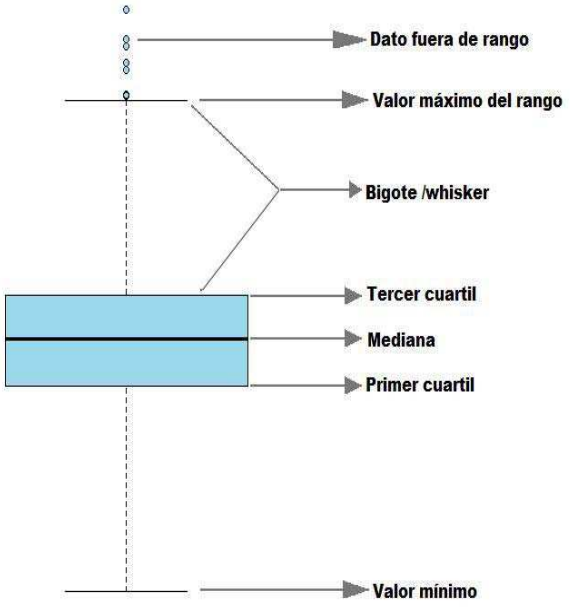


Por último, conviene destacar que *R* tiene paquetes específicos dedicados al cálculo de funciones de suavizado, y entre ellos paquetes que disponen herramientas muy completas en relación al cálculo y graficado de la densidad de un conjunto de datos, o de diversos conjuntos de datos entre sí, tales como el paquete de datos *sm*. *Openair* también proporciona funciones de suavizado avanzadas específicamente preparadas para los datos de calidad del aire, tal y como veremos más adelante.

» Gráfico de Cajas o Caja de Tukey: *boxplot*

Las Gráficas de Cajas o Cajas de Tukey, así llamadas por su ideólogo, son gráficas muy sencillas, a la par que de gran potencia, para la representación de series de datos de forma que en un solo gráfico podamos tener información completa sobre la simetría, la dispersión y el sesgo en nuestros datos.

Toda esta información se representa en una función gráfica *boxplot* al incluir dentro de la misma, tal y como se puede observar en la gráfica, la representación de la mediana (línea que corta la caja), el primer cuartil de datos (extremo inferior de la caja), el tercer cuartil de los datos (extremos superior de la caja), y los valores máximos y mínimos (bigotes superior e inferior de la gráfica, también llamados *whisker*), con excepción de los valores atípicos o considerados fuera de rango, que se representan como círculos individuales fuera del bigote.



Por otro lado, una de las mayores ventajas de este gráfico radica en que, más allá de representar las distribuciones dadas para un solo conjunto de datos, nos permite graficar tantos conjuntos de datos como queramos de forma simultánea, lo que en el caso de la calidad del aire nos permitirá, por ejemplo, comparar distintas ubicaciones o distintos periodos temporales.

Estadísticamente hablando, en cuanto al cálculo de datos en *R*, la función *boxplot* no tiene ningún valor especial más allá de la estimación de los bigotes o *whisker*, ya que el resto de datos estadísticos los calculan directamente funciones como *summary*, que hemos podido ver en apartados anteriores. De hecho, el cálculo de datos estadísticos propios de las cajas de Tukey se puede llevar a cabo a través de la función de *R* ***fivenum***, para cada conjunto de datos, tal y como se muestra a continuación, en cuyo caso *R* devuelve a la consola los datos estadísticos correspondientes al mínimo de los datos, el primer cuartil, la mediana, el tercer cuartil y el valor máximo de los datos, respectivamente.

Variaciones sobre la función:

1-. Incluir datos extremos o fuera de los rangos dentro de los *whisker*: ***range***

En ocasiones el número de datos apartados, extremos o fuera de rango considerados por la función *boxplot* como valores atípicos por defecto, puede ser excesivo o alejado de la realidad del parámetro a representar, descartándose así concentraciones que pueden ser descriptivas de determinadas situaciones episódicas o puntuales representativas igualmente de la calidad del aire en la zona.

Mediante el comando *range* el usuario podrá controlar el número de datos atípicos que desea incorporar al bigote o *whisker* de la función gráfica, quedando el resto de puntos fuera del mismo. Así, el comando *range* deberá acompañarse de una variable numérica que marcará el rango intercuartílico que queremos aplicar, de forma que, a mayor rango, menos número de puntos individuales o valores atípicos quedarán fuera del bigote.

En cualquier caso, si disponemos que *range=0*, la función interpretará que el número de datos atípicos debe ser nulo, por lo que englobará todos los puntos individuales dentro del bigote, alargando el mismo hasta el valor máximo y/o mínimo de nuestro conjunto de datos.

2-. Establecer el ancho de las cajas: ***width***

Con el comando *width* se establece el ancho de la caja o cajas de que dispongamos, personalizándolo. Si tenemos varias cajas y queremos personalizar su ancho, no valdrá con incluir una sola variable numérica, sino que el usuario deberá incluir uno a uno los distintos anchos establecidos para cada conjunto de datos, utilizando para ello el correspondiente vector.

Ej. **boxplot (datoscont07$pm10, datoscont08$pm10, range=5, width=c(10,15)),** previamente hemos dividido nuestro conjunto de datos original datoscont en los subconjuntos correspondientes a los años 2007 y 2008 para elaborar la gráfica comparando datos anuales (para ello hemos utilizado la función selectByDate). Con la función boxplot lo que hacemos es seleccionar los subconjuntos para ambos años, con indicación del parámetros que queremos calcular (pm10), representando las dos Cajas de Tukey para dichos años con un rango intercuartílico de 5, y con un ancho de caja de 10 para el año 2007, y de 15 para el año 2008.

3-. Dibujar las cajas con anchos proporcionales: ***varwidth***

Puede ser que nos interese que los anchos de las cajas, más allá de representar un aspecto puramente estético de la gráfica, nos sirva también para poder observar la representatividad de los distintos conjuntos de datos. En estos casos bastará con indicar a la función que ***varwith=TRUE*** para que el ancho de la caja se establezca de forma proporcional al número de datos disponible en el conjunto.

4-. Añadir intervalos de confianza a la mediana: ***notch***

Una variación muy común de la gráfica de cajas es aquella que incluye una serie de cuñas a los lados de la mediana para representar dicho estadístico con intervalos de confianza incluidos. Dicha variante se contempla en *R* a través del comando lógico *notch*. La utilidad de la representación de dichos intervalos de confianza en la gráfica se basa en que si las cuñas de las cajas no se solapan adecuadamente, se puede concluir que existe una evidencia clara de que ambas medianas son significativamente diferentes.

5-. Representar en la gráfica los valores de fuera de rango: ***outline***

Las gráficas de cajas se pueden representar con inclusión de los datos que quedan fuera de rango, tal y como se hace por defecto, con ***outline=TRUE***, y por lo tanto la escala del Eje *Y* se adaptará a toda la escala de valores del parámetro representado, o también se pueden elaborar eliminando de la gráfica dichos datos puntuales de fuera de rango, por lo que la gráfica se limitará a los valores de la caja y los *whisker*, con ***outline=FALSE***.

6-. Incluir etiquetas en el Eje X: ***names***

Cuando representamos varios conjuntos de datos se generan varias gráficas de cajas, una al lado de la otra, compartiendo la escala presentada en el Eje Y. En estos casos, seguramente resultará muy útil, para facilitar la interpretación de la gráfica, disponer de los nombres correspondientes para cada conjunto de datos (para cada caja de la gráfica), utilizando para ello el Eje X, lo cual se consigue asignando una etiqueta descriptiva a cada conjunto con el comando *names*.

7-. Escalar los distintos elementos de la gráfica: ***boxwex*** , ***staplewex***

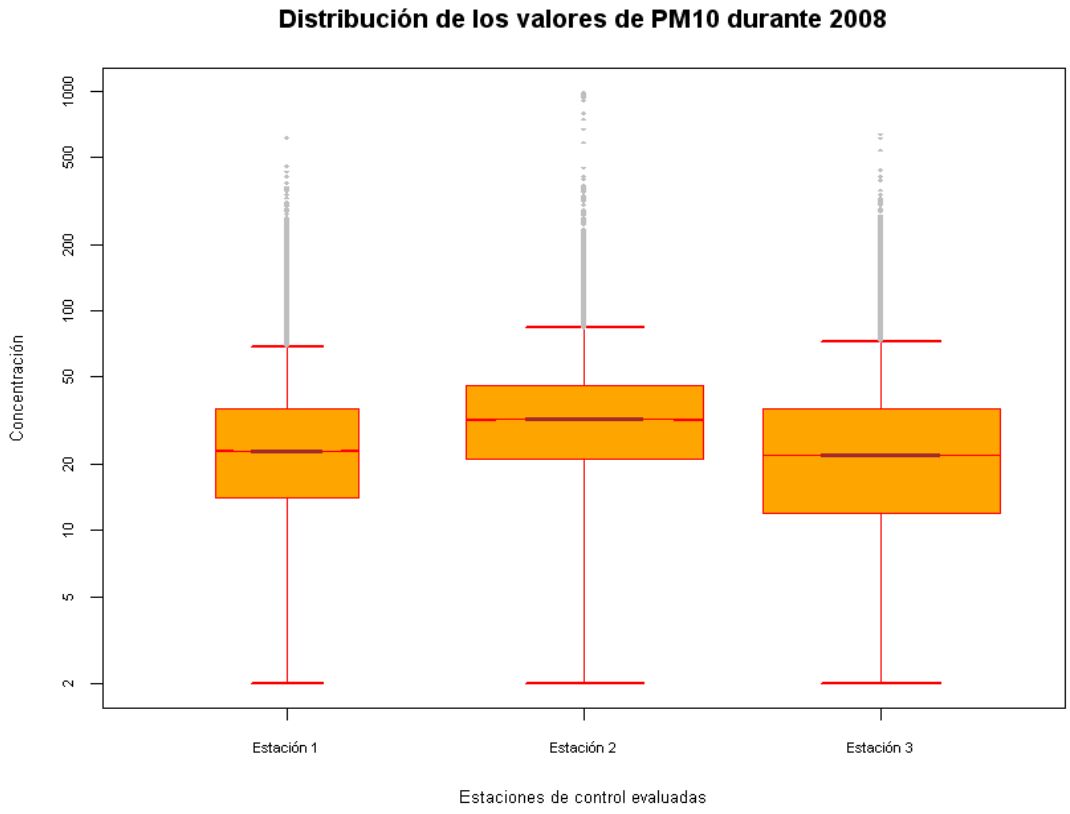
La existencia de un mayor o menor número de cajas en nuestra gráfica puede dificultar la interpretación de la misma, ya sea porque se solapan las cajas o porque quedan muy separadas. Con el comando *boxwex* se puede aplicar a la gráfica una variable numérica de escala que se aplicará como un factor para hacer que todas las canas sena más o menos anchas, en función de lo indicado por el usuario, respetando siempre la proporción de las dimensiones marcadas por otros comandos como *varwidth*. Por defecto este comando presenta un valor de ***boxwex=0.8***.

A su vez, el comando *staplewex* hace lo mismo con la línea superior e inferior que marca el final del bigote o *whisker*, estableciendo el tamaño proporcionalmente al de las cajas. Por defecto, el factor que presenta este comando es de ***staplewex=0.5***, o lo que es lo mismo, la mitad del tamaño de la caja.

8-. Dar formato a los distintos elementos de la gráfica: ***box*** , ***med***, ***whisk***, ***staple***, ***out*** + ***lty***, ***lwd*** ,***cex*** ,***pch***, ***bg***

Para la función *boxplot* es posible configurar el aspecto estético de todos y cada uno de sus elementos gráficos. Para ello se utilizará la combinación del nombre del elemento gráfico: ***box*** (para la caja), ***med*** (para la mediana), ***whisk*** (para el bigote), ***staple*** (para la línea límite del bigote) y ***out*** (para los valores de fuera de rango); y de las distintas opciones de configuración gráfica, según corresponda: ***lty*** (para el tipo de línea), ***lwd*** (para el grosor de la línea), ***cex*** (para incrementar o disminuir el tamaño del elemento), ***pch*** (para el tipo de punto), ***col*** (para el color), y ***bg*** (para el color de fondo, aunque en algunas ocasiones se mantienen la antigua denominación *fill*)

Mediante la función y los comandos expuestos hasta el momento es posible generar gráficas de cajas que presentan una completa comparativa de la distribución de sus datos y de sus principales estadísticos, ya sea por parámetros o por estaciones de control, tal como muestra el siguiente ejemplo:




```
> boxplot(datoscont1$pm10, datoscont2$pm10, datoscont3$pm10, varwidth=TRUE, notch=TRUE, log="y", staplewex=0.5, names=c("Estación 1", "Estación 2", "Estación 3"), main="Distribución de los valores de PM10 durante 2008", xlab="Estaciones de control evaluadas", ylab="Concentración", outcex=0.5, outpch=18,outcol="grey", boxcol="red", boxfill="orange", whisklty=1, whiskcol="red", cex.lab=0.8, cex.axis=0.7, staplelwd=2, staplecol="red", medcol="brown")
```

Instrucciones gráficas en Openair

Ya hemos visto que *R* permite el manejo fácil y graficado de grandes series históricas de datos, aunque también hemos podido comprobar que el diseño gráfico se muestra algo limitado para determinadas funciones relacionadas con la calidad del aire. En este sentido, *Openair* es una librería que nos permite dar una vuelta de tuerca a *R* y adaptarlo mucho más a nuestras necesidades, ampliando con mucho las utilidades de *R*, sobretodo a nivel gráfico.

En los próximos apartados de este manual se van a desarrollar, una a una, cada una de las gráficas de *Openair*, de forma individual, e indicando los principales comandos diseñados para su modificación, personalización y diseño. Se recomienda al lector, sobretodo si es principiante, seguir paso por paso la lectura de las distintas instrucciones gráficas, de forma paralela a su experimentación en *R*.

Para lectores más avanzados y experimentados en *R* ya está previsto el sistema de ayudas que proporciona el lenguaje *R*, con el comando ***help***, o la consulta directa de funciones, mucho más rápida, proporcionada por el comando ***?***. Además, los distintos paquetes disponen de instrucciones que se pueden consultar sobre la marcha a la hora de programar las funciones

En primer lugar, se debe considerar, al igual que ocurría con las gráfica de *R*, que existen una serie de comandos básicos para la configuración de las gráficas de *Openair* que son comunes a la mayoría de ellas, entre los que encontramos los siguientes:

- Cambiar los colores de una gráfica: ***cols***

  Openair dispone de un comando propio para configurar los colores de representación de sus gráficas. De hecho, y en un intento por facilitar al máximo las labores de configuración de las gráficas al usuario, muchas de ellas tienen configurados paquetes de colores que permiten la gradación automática de los mismos en la gráfica, tales como *"greyscale", "heat", "jet",* etc. En todo caso, estos paquetes de colores no son los mismos en todas las gráficas ni están disponibles de igual forma para todas ellas, por lo que en cada una se tratará de forma breve este apartado de la configuración grafica.

  Además de los colores configurados de forma automática por paquetes en *Openair*, es posible que el usuario quiera personalizar al máximo la gráfica estableciendo sus propios colores para la escala de representación de la gráfica o los distintos elementos de la misma. Para ello se utilizará el mismo comando, con la excepción de que, en vez de hacer referencia a un paquete de colores, lo que se hará es referencia a cada uno de los colores a utilizar.

  Ej. **polarPlot(datoscont, pollutant="03", cols=c("green", "yellow", "orange","red","brown"))**, realizará una gráfica polar para representar las concentraciones de ozono utilizando para el escalado los colores que se muestran para el comando cols, de forma que las menores concentraciones se colorearán de verde "green", mientras que las mayores se colorearán de marrón "brown".

  Es posible que en ciertas ocasiones la función gráfica permita la configuración de colores de otros elementos gráficos o aspectos de la gráfica, adaptándose así a las posibles necesidades del usuario. En estos casos el presente manual hará las referencias precisas para desarrollar este tipo de configuraciones en cada apartado.

- Cambiar la disposición espacial de los gráficos múltiples cuando se generan: ***layout***

  Tal y como se verá más adelante *Openair* "complica" el desarrollo gráfico para adaptarlo a las necesidades que puedan surgir en materia de calidad del aire. Dentro de este desarrollo puede ocurrir que una instrucción gráfica genere una gráfica múltiple (varias gráficas concatenadas) frente a lo cual es posible que no nos guste su disposición en la pantalla gráfica. Para ello *Openair* tiene previsto el comando layout, que permite precisar al programa el número de filas y columnas que se quiere que tenga la gráfica combinada.

  Ej. **polarPlot(datoscont, pollutant="03", type="season", layout=c(4,1)**, realizará una gráfica polar para representar las concentraciones de ozono en función de la estación en la que se encuentre el dato, por lo que generará una gráfica múltiple formada por 4 filas y 1 columna (es decir, pondrá las gráficas ordenadas en una columna).

- Formatear de forma automática los textos y leyendas de una gráfica: ***auto.text***

Por defecto *Openair* tiene previsto para sus gráficas la opción de incluir texto de forma automática en la gráfica (encabezados, leyendas, anotaciones, etc), o incluso de formatear el texto que introduzcamos nosotros en función de las notaciones propias de la calidad del aire (subíndices, superíndices, mayúsculas, etc.), configurando fórmulas, parámetros y unidades. Este automatismo, como casi todos en la vida, puede tanto facilitar como complicar la representación gráfica. Dado este hecho, es posible que sea preciso indicar a *Openair* que no queremos esta opción que viene por defecto, lo que le indicaremos con la orden: ***auto.text=FALSE***.

El presente manual o libreto de apuntes pretende realizar un breve repaso sobre las principales gráficas a utilizar para el tratamiento de datos de calidad del aire, tanto en *R* como en *Openair,* así como de las funciones y variables principales que podemos aplicar a cada una de ellas. Desarrollar la información más allá de lo aquí expuesto será parte del trabajo de investigación posterior que haga el lector sobre los diversos manuales.

Las gráficas que se vayan explicando en el manual se han probado previamente, ya que estos apuntes han sido elaborados por un neófito en R que no se hubiese aventurado nunca a poner algo que no funcionase. Prueba de ello serán los distintos ejemplos que irán apareciendo en cada uno de los apartados de gráficas, que incluirán en su parte inferior el código utilizado.

- Personalizar los textos y definiciones de las leyendas de una gráfica: ***key***

Openair precisa para casi todas sus gráficas de una leyenda explicativa para los parámetros que representa, donde se establece la escala en la que se representan, las unidades y la definición del propio parámetro representado. En este sentido, Openair permite, siempre que exista esta leyenda explicativa, personalizar sus contenidos y ubicación a través de los comandos:

1. ***key.header:*** que permite incluir un título o encabezamiento para la leyenda, donde podría ir el contaminante.
2. ***key.footer:*** que permite incluir un pie de leyenda, donde podrían ir las unidades de representación.
3. ***key.possition:*** donde se le indica el lugar de la gráfica donde se quiere incluir la leyenda: abajo *"bottom",* a la izquierda *"left"* o a la derecha *"right"*

Ej. **windRose(datoscont, key.header="velocidad del viento", key.footer="metros por segundo", key. position="left"),** realizará una rosa de vientos en la que la leyenda se mostrará a la izquierda de la misma, con el nombre y las unidades del parámetro representado en castellano.

Determinadas gráficas de *Openair,* como *timePlot* dispondrán de leyendas distintas a las habituales, donde tan sólo se detallan los parámetros representados, sin incluir ningún tipo de escala o unidades. En estos casos, la leyenda se configurará de forma distinta a la especificada en el punto anterior, aunque su denominación continuará siendo igual. La forma de modificar esta leyenda se especificará en cada gráfica.

» Gráfica de resumen de datos: *summaryPlot*

En el tercer apartado del presente manual pudimos comprobar, dentro de las instrucciones generales del lenguaje de programación *R*, que disponíamos de un comando denominado *summary* que permite la presentación de un resumen estadístico básico de los datos.

Openair amplía esta función de *R* y permite añadir gráficas a este resumen de estadísticos mediante el comando ***summaryPlot.***

Así pues, la función *summaryPlot* puede darnos una gráfica completa resumen de los datos de una determinada estación de control que nos puede ser de enorme utilidad, sobretodo cuando se precisa comprobar los datos con anterioridad a la realización de otras gráficas y operaciones sobre el conjunto de datos.

Ej. **summaryPlot(datoscont)**. Esta función gráfica proporciona un resumen estadístico de datos en la pantalla de datos y una gráfica resumen de cada uno de los parámetros, consistente en una gráfica lineal de distribución por años y contaminantes, así como un histograma de los datos. Si existen fechas con todos los datos nulos podría ocurrir que esta función diese error de inicio.

Variaciones sobre la función:

1-. Eliminación de datos fuera de rango: ***clip*** o ***percentile***

Se puede indicar a la función que elimine aquellos datos que previsiblemente puedan estar fuera de rango, evitando así que desvirtúen los histogramas de resumen de datos. Para ello se puede utilizar la instrucción clip o la instrucción *percentile*, para indicar el percentil de datos deseado a eliminar.

Ej. **summaryPlot(datoscont, clip=TRUE)**, o **summaryPlot(datoscont, percentile=0.95)**, donde se le indicará a la instrucción que elimine aquellos datos que se encuentre fuera de rango (clip) o que estén por encima del percentil 95, si es que queremos tener más control sobre los datos eliminados.

2-. Mostrar datos nulos del conjunto de datos: ***na.len***

En largas series de datos es muy difícil poder comprobar dónde se producen huecos de datos. Al incluir esta variable en la función, se mostraran aquellos huecos de datos que cumplan con la premisa de número especificada, incluyendo una barra de color en la parte inferior de la gráfica de evolución de los datos.

Ej. **summaryPlot(datoscont, na.len=48)**, mostrará aquellos huecos de datos en los que exista un número superior a los 48 datos nulos de forma continuada.

3-. Cambiar los colores de los distintos elementos de la gráfica: ***col***

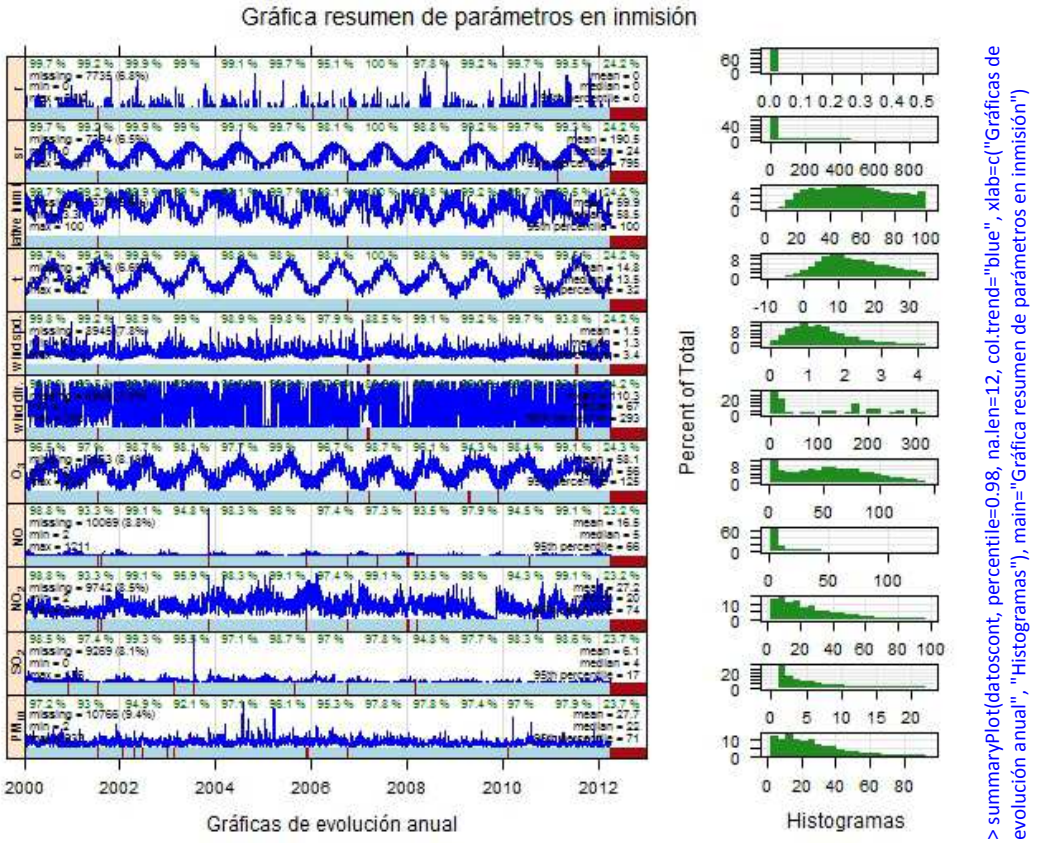
Se pueden cambiar los colores de ciertas partes de la gráfica utilizando el comando col que ya vimos en las gráficas R, seguido por el elemento de la gráfica *summaryPlot* que queremos cambiar de color. Es decir:

col.trend, para la línea de la gráfica lineal.
col.data, para la barra horizontal que muestra la existencia de datos
col.mis, para definir el color de la barra cuando no hay datos,
col.hist, para definir el color del histograma.

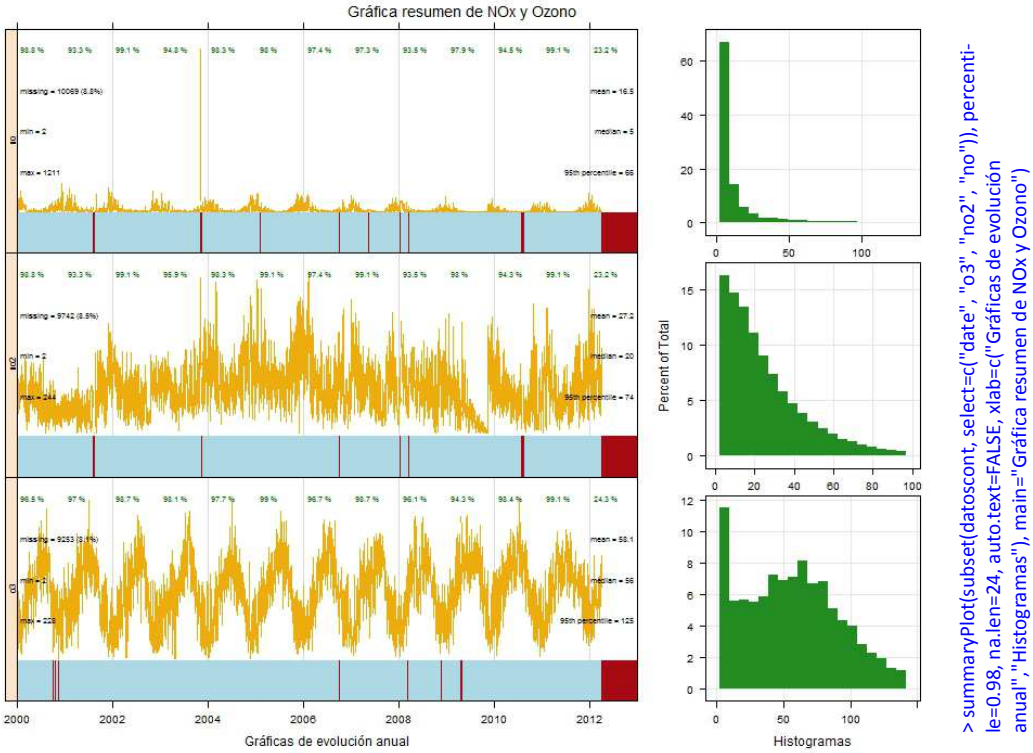
Ej. **summaryPlot(datoscont, na.len=48, col.mis="red")**, mostrará aquellos huecos de datos en los que exista un número superior a los 48 datos nulos de forma continuada coloreandolos de rojo.

4-. Establecer el periodo de cálculo de la gráfica: ***period***

Las gráficas y estadísticas de la función *summaryPlot* se calculan por defecto para periodos anuales, aunque es posible cambiarlo para decirle a *Openair* que lo haga de forma mensual, mediante la inclusión del comando: ***period="month"***.



Sin embargo, *R* y *Openair* son lenguajes de programación, por lo que no terminan aquí. Permiten la utilización cruzada de funciones y expresiones, por lo que se puede combinar la gráfica con otras operaciones para simplificar o manejar mejor los datos. Por ejemplo, podría ocurrir que el exceso de carga en el número de parámetros dificulte la comprensión y estudio de la gráfica, como de hecho así ocurre, o que quisiésemos comprobar el resumen de determinados parámetros. Es aquí donde podemos empezar a aplicar los conocimientos adquiridos en el apartado 3 y utilizar, por ejemplo, la función ***subset,*** tal y como se muestra en el siguiente ejemplo gráfico.



» La Rosa de Vientos: *windRose*

Muchas aplicaciones no disponen de la posibilidad de representar una rosa de vientos, y aquellas que están específicamente diseñadas para calidad del aire permiten la generación de gráficas polares para la representación de datos relativos al a velocidad y dirección del viento, pero pocas presentan la versatilidad y posibilidades de interacción que ofrece *Openair* en su Rosa de Vientos.

Variaciones sobre la función:

1-. La designación de los campos de velocidad y dirección del viento: ***ws*** o ***wd***

Se puede indicar a la función que seleccione los campos de velocidad *(ws)* y dirección del viento *(wd)* cuando estos campos no están denominados mediante los acrónimos previstos por *openair*.

Ej. **windRose(datoscont, ws="VV", wd="DD")**, realizará una rosa de vientos para el conjunto de datos de datoscont, adoptando como campo de velocidad de viento el denominado con el encabezado VV, y como dirección de viento el encabezado DD.

2-. Intervalos en los que se representa la velocidad del viento: ***ws.int***

Por defecto la función *windRose* representará la velocidad en intervalos de 2 m/sg. Sin embargo, es posible indicar a la función, si estos intervalos no se adaptasen a nuestro caso en concreto, que adopte otros intervalos distintos estableciendo las velocidades a las que se dispondrán.

Ej. **windRose(datoscont, ws.int=1.5)**, realizará una rosa de vientos para el conjunto de datos de datoscont, adoptando un intervalo de 1,5 m/sg para representar la velocidad en los distintos ejes de la rosa de vientos.

3-. Intervalos en los que se representa el ángulo: ***angle***

Por defecto la función *windRose* adopta para representar un ángulo de 30º, es decir, cada 30º representa una línea en la que aparecerá la frecuencia media ocurrida en ese intervalo y los rangos de velocidades que se detectan en el mismo. Si queremos una mayor o menor precisión en la rosa de vientos, será cuestión de indicarle a la función que el ángulo sea menor o mayor, respectivamente.

Ej. **windRose(datoscont, ws.int=1.5, angle=15)**, realizará una rosa de vientos de datoscont, adoptando un intervalo de 1,5 m/sg para representar la velocidad en los distintos ejes de la rosa de vientos y representando los datos en ejes cada 15º (un total de 24).

4-. Intervalos en los que se representa la frecuencia o porcentaje de datos de la gráfica: ***grid.line***

Por defecto la función *windRose* adoptará los intervalos que mejor se adapten al rango de frecuencias de aparición de las distintas direcciones de viento, adaptando los círculos concéntricos de la gráfica a lo que mejor se adapte. Sin embargo, es posible personalizarlo indicando a la función el porcentaje aplicado a cada intervalo.

Ej. **windRose(datoscont, grid.line=10)**, realizará una rosa de vientos para el conjunto de datos de datoscont, adoptando círculos concéntricos en intervalos del porcentaje de velocidad que vayan de 10% en 10%.

5-. Desagregar los datos a representar para elaborar distintas rosas de vientos: ***type***

Ya habíamos visto en apartados anteriores que existe una función *cutData* que nos permite clasificar los datos y cortarlos agregando a los mismos un campo clasificatorio. También comentábamos en este apartado que muchas funciones sustituyen esta instrucción por el comando *type*, que de forma básica permite hacer lo mismo, y eso es precisamente lo que ocurre con la Rosa de Vientos.

Con el comando *type* podemos decirle a la función de rosa de vientos que divida los datos y los represente en distintas rosas de vientos en función de las estaciones del año *"season"*, los años *"year"*, los días de la semana *"weekday"*, etc.

También, al igual que ocurría con la instrucción *cutData*, podemos indicarle a la función que divida la representación de las rosas de vientos en función de una de las variables del conjunto de datos, de forma que represente cuatro rosas de vientos en función de los cuatro percentiles que encuentre para el rango de datos disponible.

Ej. **windRose(datoscont, type="pm10")**, realizará cuatro rosas de vientos para el conjunto de datos de datoscont, cada una para un intervalo distinto de partículas, en función del rango de concentración de partículas que dispongamos y los percentiles aplicados, lo que podría resultar extraordinariamente útil si queremos ver en qué direcciones y velocidades predominantes se nos producen las mayores concentraciones de partículas, por ejemplo.

Incluso, si quisiéramos rizar el rizo, podríamos indicarle al comando *type* que seleccionase varios criterios para la representación de las rosas de vientos, utilizando la expresión c que veíamos en anteriores apartados.

Ej. **windRose(datoscont, type=c("season" , "pm10"))**, realizará dieciséis rosas de vientos para el conjunto de datos de datoscont, cuatro filas de rosas de vientos en función de las estaciones, y para cada una de ellas otras cuatro en función de los rangos de partículas.

6-. Cambiar los colores predeterminados de la función gráfica: ***cols***

Ya se ve al principio de este apartado la posibilidad de utilizar el comando *cols* para cambiar los colores de la gráfica. Esta gráfica es, en este caso, una de las que mayor potencial tiene para el uso de este tipo de utilidad, junto con otras como *polarPlot, polarFreq,* etc. El cambio se puede hacer por un formato de color predeterminado de *Openair*, como los especificados en la introducción, o por una serie de colores fijados por el usuario.

7-. Regular la anchura con la que se presentan los rangos de velocidad en la gráfica: ***width***

En las rosas de vientos previstas por *Openair* las velocidad de viento representada, además de cambiar su color, va incrementando su anchura progresivamente, para que se puedan visualizar mejor. Mediante este comando es posible indicar a la gráfica que el diferencial aplicado a esta anchura sea mayor o menor, pudiendo así ver los ejes con una mayor o menor anchura.

Ej. **windRose(datoscont, width=1.5)**, realizará una rosa de vientos en las que las velocidades de viento se representarán ligeramente más anchas de lo normal, para que se vean mejor, siempre y cuando el número de ejes no sea excesivo.

8-. Establecer el número de intervalos o puntos de ruptura de los datos de velocidad: ***breaks***

En las rosas de vientos previstas por *Openair* podemos determinar con mucha exactitud cómo podemos representar los datos de velocidad. El comando breaks establece no el tamaño de los intervalos de representación (tal y como ocurría con el comando *ws.int*) sino el número de intervalos en los que se representa.

Ej. **windRose(datoscont, breaks=6)**, realizará una rosa de vientos en las que las velocidades de viento se representarán utilizando para ello 6 intervalos en función de las velocidades registradas para el conjunto de datos, adaptando así la distribución de velocidades hasta alcanzar dichos intervalos.

Es posible también indicarle al comando breaks los puntos de corte en lugar del número de intervalos. Esto se consigue si le damos varias variables, consiguiendo así que el comando breaks interprete que los números aportados con los límites a adoptar para los intervalos.

Ej. **windRose(datoscont, breaks=c(1,2,5,7))**, realizará una rosa de vientos en la que se representará la velocidad en función de los siguienter intervamos: 0-1, 1-2, 2-5, 5-7, >7.

9-. Estilo de los ejes de la gráfica: ***paddle***

En los ejes la velocidad se representa por defecto en estilo *"cuña"* (rectángulos que incrementan su lateral en función del incremento de los intervalos, pues este estilo se encuentra activado *TRUE*. Si lo desactivásemos la gráfica adoptaría el estilo "remo" (triángulos anexos distribuidos en función de cada eje).

10-. Tamaño del agujero central de la gráfica: ***offset***

Todas las rosas de viento disponen de un agujero central de representación que puede ser más o menos grande en función de este comando, que permite disponer el tamaño del mismo. Por defecto el *offset* de la gráfica se representa con un tamaño de 10.

11-. Número de dígitos representativos utilizados para representar las cifras: ***dig.lab***

Se puede personalizar también el número de dígitos representativos con los que aparecerán las cifras de velocidad de viento en la gráfica. Por defecto la función de *openair* asigna 5 cifras significativas que, por lo general son suficientes.

12-. Presentar una pequeña anotación relativa a los estadísticos surgidos en la gráfica: ***annotate***

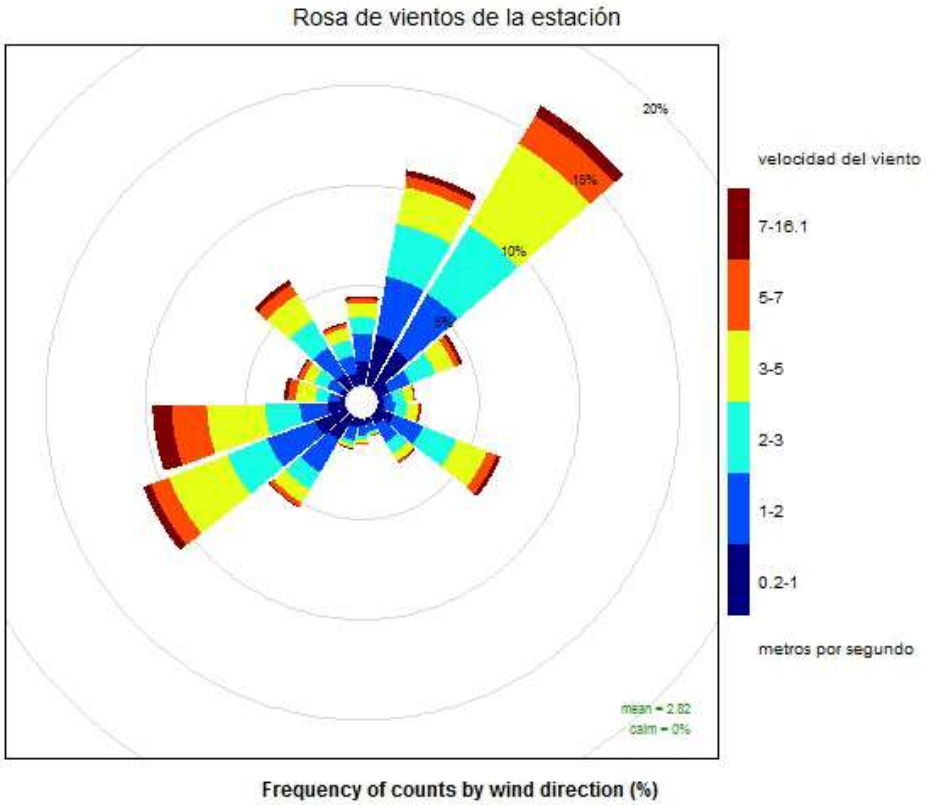
Se puede indicar a la instrucción que incluya un pequeña notación con los principales estadísticos de la rosa de viento, media y porcentaje de calmas, indicando que el comando *annotate* sea *TRUE*, opción que se presenta por defecto.

13-. Modificar el método estadístico utilizado para la representación de los ejes: ***statistic***

La rosa de vientos utiliza por defecto la frecuencia con la que aparecen las distintas direcciones para cada uno de los datos del conjunto analizado de forma acumulativa, por lo que las unidades obtenidas son en porcentaje sobre el total de datos. El comando para este estadístico por defecto es el del conteo proporcional ***statistic="prop.count"***.

Sin embargo, es posible indicarle a la instrucción *windRose* que el método estadístico se base en la contribución a la media de la velocidad, ***statistic="prop.mean"***, o al conteo absoluto de datos para cada uno de los ejes, sin establecer la frecuencia de aparición sobre el total, ***statistic="abs.count"***.

Ej. **windRose(datoscont, statistic="abs.count")**, realizará una rosa de vientos en la que los círculos concéntricos representarán no el porcentaje, sino el número total de datos para cada una de las direcciones previstas, representándose los distintos rangos de velocidades




```
> windRose(datoscont, breaks=c(1,2,3,5,7), angle=20, cols="jet", paddle=FALSE, offset=5, key.header= "velocidad del vien-
to",  key.footer="metros por segundo",  key.position="right", main="Rosa de vientos de la estación")
```

» La Rosa de Contaminantes: *pollutionRose*

La Rosa de Contaminantes es una variante de la rosa de viento, un desarrollo de esta última función que básicamente sustituye la velocidad del viento por la concentración de un contaminante determinado de que se disponga en el conjunto de datos, por lo que resulta interesante en su uso más básico para conocer el tiempo que un determinado contaminante, y sus distintas concentraciones, se encuentran en una dirección de viento determinada.

Variaciones sobre la función:

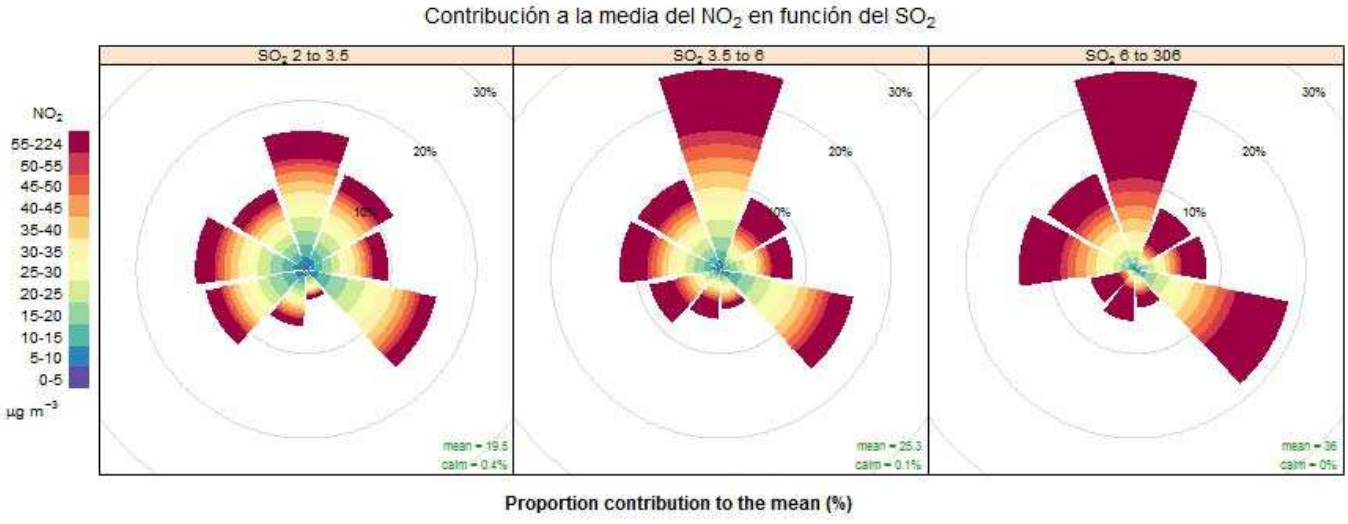
La función *pollutionRose* utiliza prácticamente los mismos comandos que la función *windRose*, menos para los comandos directamente relacionados con la velocidad del viento. Así:

- El comando ***ws***, que establece el campo de la velocidad del viento en la rosa de vientos, queda sustituido por el comando ***pollutat***, que establece el contaminante a representar en la misma.
- El comando ***ws.int***, que establece el tamaño de los intervalos de la velocidad del viento a representar, se elimina de la función pollutionRose, que regulará los intervalos a partir de otros comandos como ***breaks***.

El resto de comandos previstos por la función *windRose* se mantiene exactamente igual para la función *pollutionRose*. Alguno de estos comandos, de hecho, presentan mayor utilidad para la función *pollutionRose* que para la función *windRose*. Estos comandos permitirán desarrollar la función *pollutionRose* en función de múltiples criterios, estadísticos y de análisis, como por ejemplo:

Ej. **pollutionRose(datoscont, pollutant="no2", statistic="prop.mean")**, realizará una gráfica polar en la que la concentración del no2 se representará en función del porcentaje de contribución proporcional a la media del contaminante, permitiendo así comprobar qué dirección del viento contribuye más a la concentración global del mismo, a la par que proporciona información sobre los distintos niveles de concentración alcanzados.

Ej. **pollutionRose(datoscont, pollutant="no2", type="so2")**, realizará una rosa de vientos en la que la se representa la frecuencia con la que se producen las distintas concentraciones del contaminante no2 en cada una de las direcciones del viento, y teniendo en cuenta los distintas




```
> pollutionRose (datoscont, pollutant="no2", angle=40, grid.line=10, offset=0, breaks=10, key.header="NO2", key.footer="ug/m3", key.position="left", main="Contribución a la media del NO2 en función del SO2", statistic="prop.mean", type="so2")
```

» La Rosa de Percentiles: *percentileRose*

Otra forma de representar la concentración de los contaminantes frente a la dirección del viento, distinta a la representación de las concentraciones que propone la gráfica *pollutionRose* y como paso intermedio entre esta y las gráficas polares que veremos más adelante, es la que propone la instrucción *percentileRose*, que lo que hace es representar los distintos percentiles de un contaminante seleccionado frente a la dirección del viento.

La Rosa de Percentiles modelizar los niveles de percentil de los datos del contaminante seleccionado por cada dirección del viento, consiguiendo así una mayor claridad a la hora de mostrar la distribución de las concentraciones y su entidad real, pudiendo ayudar a detectar aquellas fuentes de origen de la contaminación que afecten a los percentiles más altos, lo que resulta de gran utilidad en combinación con otras funciones gráficas.

La función *percentileRose* utilizará los siguientes comandos básicos:

Variaciones sobre la función:

1-. Introducir el contaminante a graficar: ***pollutant***

Se debe indicar a la gráfica el contaminante que se desea graficar. También es posible incluso indicarle a la función que represente varios contaminantes, en cuyo caso la gráfica se dividirá en dos, elaborando una rosa de percentiles para cada contaminante, pero con una escala común, lo cual es importante a la hora de seleccionar los parámetros, que deberán compartir dicha escala: ***pollutant=c("o3","o3ref").***

2-. Desagregar los datos de un contaminante para elaborar varias gráficas: ***type***

Mediante el comando *type*, similar a la instrucción *cutData* vista en anteriores apartados, es posible desagregar los datos de un contaminante para elaborar varias gráficas en las que se contemplarán los datos en función de la estación del año *"season",* los años *"year",* los días de la semana *"weekday",* etc.

También es posible dividir la gráfica de percentiles de un contaminante en función de otro parámetro del conjunto de datos, indicando el mismo al comando *type,* en cuyo caso se elaborarán cuatro gráficas distintas en función de los rangos alcanzados por el parámetro elegido.

3-. El valor del percentil o percentiles que se desea graficar: ***percentile***

La instrucción representa, por defecto, cinco percentiles distintos en la misma gráfica: los percentiles 25, 50, 75, 90, 95. No obstante, tendremos la posibilidad de modificar esta opción indicándole a la gráfica mediante el comando *percentile* que represente el que queramos, ya sea un único percentil o varios.

4-. Los colores a utilizar en la gráfica: ***cols***

Al igual que en gráficas anteriores, es posible indicarle a la instrucción *percentileRose* que utilice otros colores en la gráfica distintos a los que se muestran por defecto, ya sea según los predeterminados por *Openair*: *"increment", "heat" o "yet",* o en función de lo deseado por el usuario: ***cols=c("green", "yellow", "orange", "red")***

5-. Rellenar las áreas entre percentiles: ***fill***

Realmente la función *percentileRose* se trata de una gráfica de representación de los percentiles mediante líneas modelizadas en función de los ejes de dirección del viento, salvo que, por defecto, la función rellena las áreas que tiene entre líneas, creando rangos a los que asigna el color del percentil superior, de forma que sea más visual la interpretación de los resultados que podamos obtener. ***fill=TRUE***

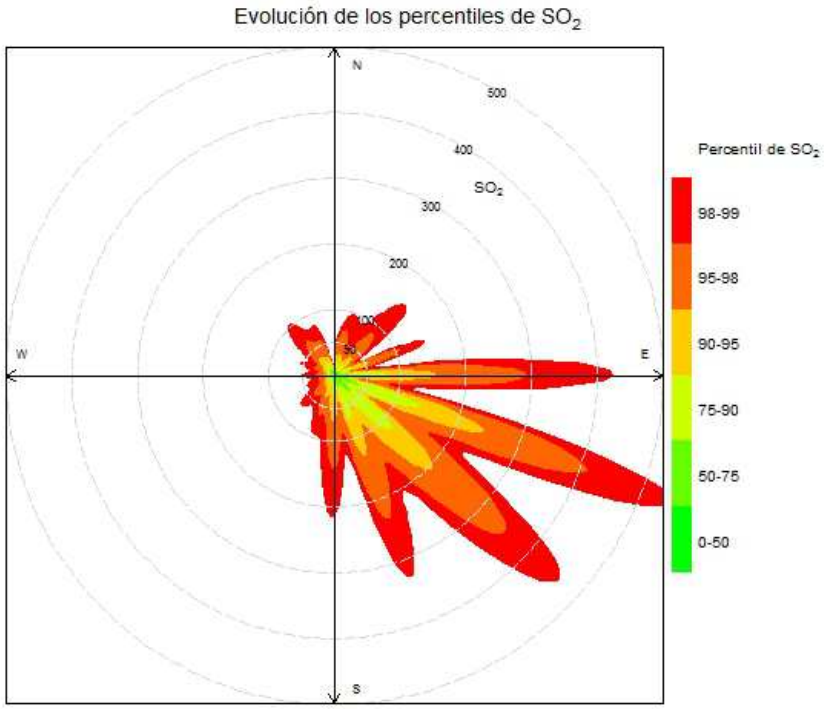
Si lo que queremos es evitar este efecto gráfico, predeterminado por la función para facilitar la interpretación visual de la gráfica, y lo que queremos es que se muestren únicamente las líneas correspondientes a los percentiles, bastaría con desactivar la función mediante un ***fill=FALSE***

6-. Establecer manualmente los intervalos del eje: ***intervals***

Las direcciones del viento se dividen automáticamente en la escala que la instrucción gráfica considera más adecuada para la representación de los percentiles obtenidos, no obstante, mediante este comando es posible modificar los intervalos dispuestos para personalizarlos en función de lo que el usuario considere más conveniente para su representación.

7-. Establecer el ángulo en el que se representa la escala: ***angle.scale***

Por defecto, la función gráfica representará y numerará los intervalos de la escala en el ángulo de 45º de la gráfica. Sin embargo, esta opción podrá personalizarse mediante este comando, indicándole un ángulo distinto al establecido.




```
> percentileRose (datoscont, pollutant="so2", percentile=c(50,75,90,95,98,99), cols=c("green", "yellow", "red"), intervals=c (50,100,200,300,400,500), angle.scale=20, main="Evolución de los percentiles de so2", key.header="Percentil de so2", key.footer="", key.position="right")
```

» La Gráfica Polar de Frecuencias: *polarFreq*

Parecida a la Rosa de Vientos, la función polar de frecuencia incluye una serie de innovaciones para mostrar cómo evolucionan los parámetros que seleccionemos con respecto a la distribución de la velocidad y dirección del viento, permitiendo a su vez tratar los datos de los parámetros utilizados a través de múltiples comandos estadísticos.

De esta forma, la función polar de frecuencias escala la dirección del viento en función de la velocidad del mismo, disponiendo de pequeños tramos en los que representará el resultado estadístico que corresponda al conjunto de datos incluido dentro de esa combinación de velocidad y dirección del viento.

Usada directamente sobre un conjunto de datos meteorológicos, sin seleccionar ningún parámetro adicional, lo que muestra es una representación de las frecuencias de aparición de esa combinación en código colorimétrico (el número de horas que se produce esa combinación de velocidad y dirección del viento), representando en la leyenda una escala de colores cuya unidad será el número de horas.

Si por el contrario le indicamos a la gráfica que haga uso de cualquier parámetro contaminante que queramos, lo que hará será representar el estadístico que le indiquemos de dicho contaminante para cada una de las combinaciones de velocidad y dirección del viento. Por ejemplo, la media de los datos del contaminante que se encuentren en esa combinación de velocidad y dirección del viento utilizando para su representación un código colorimétrico.

Variaciones sobre la función:

1-. La designación del parámetro a utilizar en la gráfica: ***pollutant***

Se puede indicar a la función gráfica *polarFreq* el parámetro contaminante que deseamos representar con respecto a la distribución de la velocidad y dirección del viento. Tal y como ya se ha visto, si no disponemos de ningún parámetro, Openair interpreta que queremos representar el tiempo (el número de horas o frecuencia con que se repite la combinación de dirección y velocidad del viento) y genera una rosa de vientos. Si le introducimos un parámetro, las "casillas" de dirección y velocidad del viento se rellenarán con el estadístico que le indiquemos (ver siguiente punto).

2-. Establecer el cálculo estadístico a realizar: ***statistic***

La instrucción gráfica *polarFreq* requiere, por definición, de la utilización de un cálculo estadístico concreto para el tratamiento del conjunto de los datos, de forma que se puedan representar en la gráfica. Por defecto, la instrucción gráfica utilizará la frecuencias, si no se ha incluido el contaminante, o la media, si se ha incluido el contaminante pero no el estadístico a utilizar.

La función *polarFreq* dispone de las siguientes estadísticas: la media "mean", la mediana "median", el máximo "max", la desviación estandar "stdev", y la media ponderada "weighted.mean". La media ponderada se calcula en función de la frecuencia de aparición (se multiplica la media de la celda por el número de datos y se divide por el total de datos disponibles) lo que proporciona una información muy útil sobre las medias dominantes según las condiciones meteorológicas.

Ej. **polarFreq(datoscont, pollutant="o3", statistic="max")**, proporcionará una función polar en la que para cada cuadrante de velocidad/dirección de viento de la gráfica, se representará la máxima concentración de ozono detectada, asignando un código de color.

3-. Incrementar o disminuir la escala de velocidad: ***ws.int***

La instrucción permite mediante este comando incrementar o disminuir el intervalo de velocidades de viento en que se escalan las direcciones, incrementando o disminuyendo así la "resolución" de la gráfica.

Ej. **polarFreq(datoscont, pollutant="o3", statistic="mean", ws.int=0.5)**, presentará la gráfica anterior aumentando la resolución al disminuir el intervalo de velociades de viento a 0.5 m/sg.

4-. Modificar el espaciado existente entre las líneas radiales: ***grid.line***

Aunque es sólo un comando para la configuración visual de la gráfica, como se ha visto en anteriores gráficas polares, es posible a través del mismo forzar a la función para que establezca intervalos de representación de las líneas concéntricas de la gráfica distintos a los establecidos por defecto.

5-. Modificar la escala de representación de los valores: ***breaks***

El usuario puede modificar la escala con la que se representan los valores del parámetro seleccionado con respecto al viento (dirección y velocidad). Para ello, al igual que con otras gráficas ya vistas, se puede utilizar el comando breaks, que divide la escala de representación de dichos datos.

Ej. **polarFreq(datoscont, pollutant="o3", statistic="max", breaks=seq(0,210,10)),** elaborará una gráfica polar de frecuencias donde se representarán los máximos de ozono en función de la velocidad y dirección del viento, siguiendo una escala de colores que irán de 0 µg/m3 a 210 µg/m3 en intervalos de 10 µg/m3.

6-. Aplicar una transformación no lineal a la escala: ***trans***

En ocasiones determinadas representaciones gráficas, como la que nos ocupa, así como determinadas funciones estadísticas, como la de máximos, pueden hacer que los valores más elevados predominen con el mismo color en la gráfica, lo que no nos permite diferenciar bien su distribución y resta utilidad a la función.

El comando trans, por defecto activado en la instrucción polarFreq (trans=TRUE), lo que hace es compensar la escala en función de la concentración, haciendo sus intervalos más grandes a bajas concentraciones y más pequeños a las altas, lo que permite una mejor diferenciación de la escala a altos valores.

Si no se desea aplicar este comando basta con indicar a polarFreq que trans=FALSE, o utilizar el comando breaks para personalizar directamente la escala.

7-. Cambiar los colores predeterminados de la gráfica: ***cols***

Como se ha visto en anteriores gráficas, Openair permite personalizar los colores por otros ya programados en el paquete de R, o personalizarlos al gusto del usuario. El comando cols se mantiene también en la gráfica polarFreq, donde presenta el mismo funcionamiento.

8-. Desagregar los datos para hacer varias gráficas: ***type***

El comando type, similar a la instrucción cutData, ya se ha tratado en anteriores gráficas, y forma parte fundamental del estudio gráfico de los datos. Mediante este comando se pueden dividir las gráficas en función de diversas categorías como días y noche "daylight", estaciones del año "season", o días de la semana "weekday", etc.

También se puede forzar al comando a que proceda a la división de los datos en función de otro parámetro del conjunto de datos, o incluso se puede forzar la división en función de varios parámetros múltiples, como se ha visto en otras gráficas.

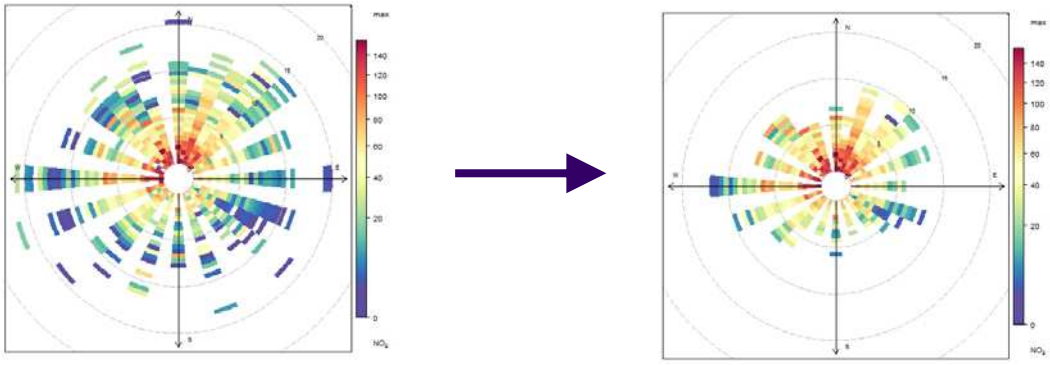
9-. Cambiar el número mínimo de datos por combinación: ***min.bin***

Este comando es preferible no usarlo si no se quiere afectar al conjunto de los datos y perder determinados datos. Mediante el comando min.bin lo que se hace es indicarle a la instrucción el número mínimo de datos permitido para cada conjunto o casilla de dirección / velocidad de viento, es decir, el número mínimo de datos que se considera representativo que existan para hacer el cálculo estadístico que le indiquemos. Por defecto el valor adoptado por la gráfica es 1, por lo que se usan todos los datos del conjunto de datos.

No obstante, el usuario puede indicar a la gráfica un número superior, en cuyo caso, lo que hace la instrucción es contar el número de datos para cada celda o rango de velocidad / dirección de viento. Si el número existente de datos por celda no alcanza el número mínimo indicado por el usuario, los datos disponibles son eliminados del conjunto de datos asignándoles la denominación "n.a." para posteriormente realizar el cálculo estadístico y graficar los datos.

En todo caso, el comando min.bin, utilizado con precaución sobre copias del conjunto de datos que estemos manejando, puede ser de enorme utilidad al filtras los datos y representan solo aquellas velocidades y direcciones del viento que son más representativas, tal y como se puede ver en el siguiente ejemplo, al que se acompaña no sólo la gráfica que "depura" los datos mediante el comando min.bin, sino la que originalmente resultaba sin la mencionada depuración.

Ej. **polarFreq(datoscontcopia, pollutant="no2", statistic="max", ws.int=0.5, offset=5, min.bin=5),** proporcionará una gráfica en la que se eliminarán aquellas celdas con velocidad y dirección del viento en las que no se encuentre más de cinco datos de contaminantes, lo que en la práctica viene a eliminar aquellas combinaciones de velocidad y dirección que resultan menos representativas. En la gráfica adjunta se puede ver que el comando min.bin "depura" en gran medida los datos, y deja tan sólo aquellos más representativos.



10-. Marcar un máximo para la representación de la velocidad: ***ws.upper***

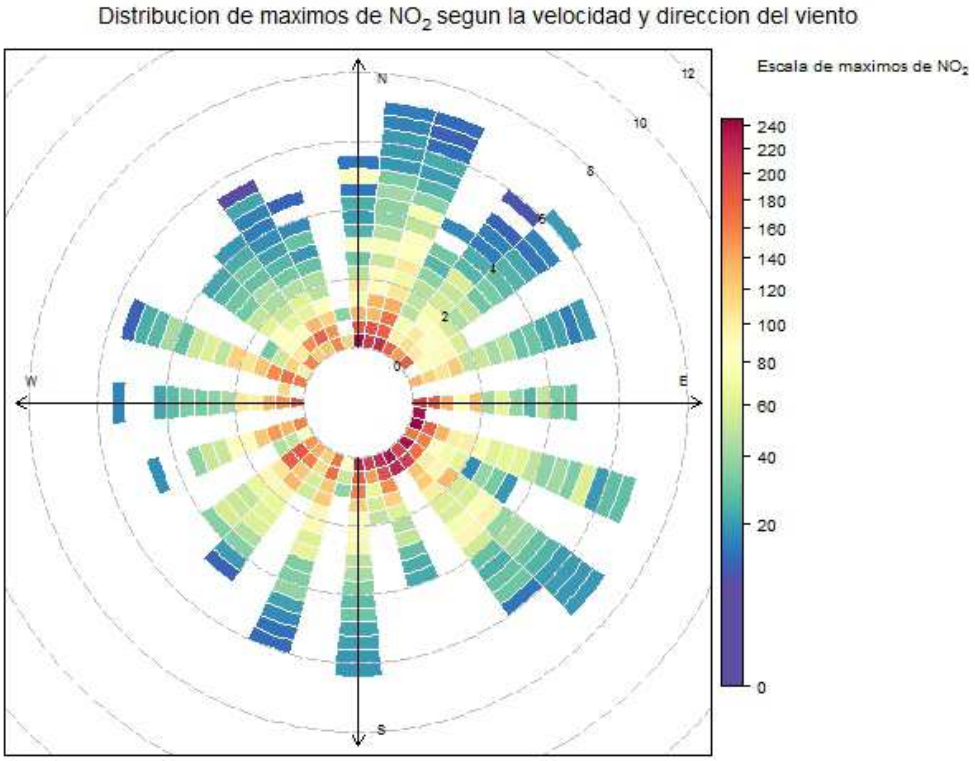
Si bien la instrucción polarFreq diseña una gráfica adaptada al rango de velocidades de que dispone el conjunto de datos, es posible que el usuario le indique a la misma un máximo de velocidades a usar distintos, ya sea mayor (para alejar la gráfica si el rango es muy pequeño) o menor (si lo que se quiere es ampliar la gráfica).

11-. Cambiar el tamaño del agujero central: ***offset***

Tal y como ya se ha explicado para otras gráficas polares, es posible hacer que el agujero central de la gráfica cambie de tamaño mediante el comando offset.

12-. Asignar un color al borde de la celda representada en la gráfica: ***border.col***

Una utilidad muy interesante de configuración de esta gráfica es la de dotar de color al borde de las celdas de velocidad y dirección del viento. Por defecto los bordes son transparentes y las celdas se juntan unas con otras, lo que puede hacer que se pierda resolución en la visualización de ciertos cambios cromáticos a pequeña escala. Dotando de color al borde de las celdas, sin que estos tengan que ser muy oscuros, éstas se separan y la gráfica adquiere una presencia muy distinta.




```
> polarFreq (datoscontcopy, pollutant="no2", statistic="max", ws.int0.4, ws.upper=8,
min.bin=5, grid.line=2, border.col="white", main="Distribucion de maximos de NO2
según la velocidad y direccion del viento", key.header="Escala de maximos de NO2",
key.footer="")
```

» La Gráfica Polar: *polarPlot*

La gráfica polar polarPlot es muy parecida en cuanto a su concepto a la gráfica polar de frecuencias polarFreq vista anteriormente, por cuanto que representa un parámetro seleccionado en función de las distintas combinaciones de dirección y velocidad del viento, salvo que la representación en este caso responde a un modelizado matemático que suaviza la cuadrícula y termina por proporcionar una superficie continua.

La utilidad de este tipo de gráficas es evidente puesto que son capaces de identificar con bastante detalle fuentes potenciales de origen de la contaminación y su influencia respecto a los niveles globales de contaminación detectados por una estación.

La utilidad se incrementa también si se utiliza como parte del estudio por puntos de una red de estaciones próximas , o si el estudio se realiza en combinación con un análisis exhaustivo del entorno micro y marco escalar de la estación. Además, la herramienta gráfica polarPlot presenta numerosos comandos de enorme utilidad para el análisis combinado de los niveles de contaminación, tal y como se verá a continuación.

Variaciones sobre la función:

1-. Parámetro contaminante a representar ***pollutant***

El comando pollutant es preciso para indicarle a la gráfica el parámetro que debe representar. En caso de no introducirlo, la instrucción polarPlot acude automáticamente a buscar el parámetro “nox”.

También tenemos la oportunidad de indicar a la instrucción que represente más de un parámetro contaminante, por ejemplo ***pollutant=c(“pm10”,”pm25”)***, que presentan concentraciones similares. Mediante esta opción polar Plot representará dos gráficas polares, una por cada parámetro, compartiendo una misma escala. Esta opción es muy útil, por ejemplo, para las intercomparaciones de equipos o técnicas.

2-. Parámetro a representar en los ejes de dirección: ***x***

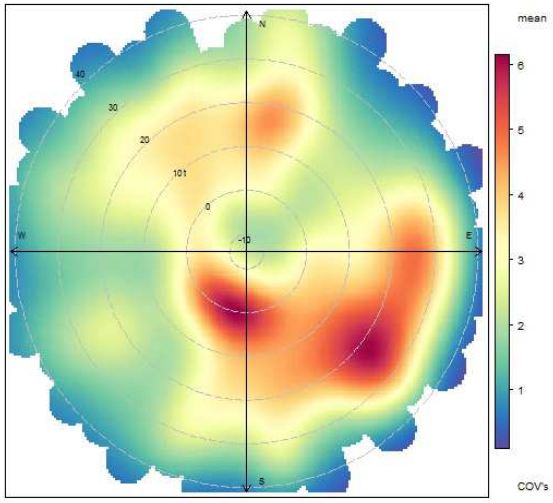
Por defecto la función polarPlot utiliza la velocidad del viento para combinarla con la dirección, al igual que sucedía con la instrucción polarFreq. De hecho, si la velocidad del viento no tiene la denominación estándar, es posible asignarla mediante el comando ***ws*** vista en gráficas anteriores.

Sin embargo, una de las novedades más interesantes que aporta la función polarPlot, es que ha sido diseñada con la capacidad de sustituir la velocidad del viento en la gráfica, pudiendo disponer en su lugar de cualquier otro parámetro.

Este hecho aporta valor añadido a la gráfica, ya que se puede ver la evolución de dos parámetros en paralelo a la dirección del viento. La cuestión fundamental en este caso es que la variable a representar frente a la dirección del viento debe ser selectiva de alguna manera frente al parámetro que se desea estudiar.

Por ejemplo, la utilización de una variable como la temperatura, en sustitución de la velocidad del viento, puede ser muy útil para estudiar la generación de emisiones difusas de compuestos orgánicos volátiles con el incremento de la temperatura, lo que unido a su representación frente a la dirección del viento, proporciona a su vez información sobre su procedencia, tal y como se observa en la siguiente gráfica.

Ej. **polarPlot (datoscont, pollutant=“cov”, x=“t”),** proporcionará la gráfica que se muestra a continuación donde se representan las concentraciones medias de compuestos orgánicos volátiles en relación al incremento de la temperatura y la dirección del viento, pudiendo observarse cómo existe una dirección predominante.



3-. Dividir los datos para generar varias gráficas en función de los criterios dispuestos: ***type***

El comando type, similar a la instrucción cutData, ya se ha tratado en anteriores gráficas, y forma parte fundamental del estudio gráfico de los datos. Mediante este comando se pueden dividir las gráficas en función de diversas categorías como días y noche "daylight", estaciones del año "season", o días de la semana "weekday", etc.

También se puede forzar al comando a que proceda a la división de los datos en función de otro parámetro del conjunto de datos, o incluso se puede forzar la división en función de varios parámetros múltiples, como se ha visto en otras gráficas.

4-. Utilizar diversos parámetros estadísticos: ***statistic***

La Gráfica polarPlot puede utilizar las siguientes funciones estadísticas para la representación de los datos: la media "mean", que se aplica por defecto, el máximo "max", la mediana "median", la desviación estándar "stdev" y la media ponderada "weighted.mean", al igual que en la función polar de frecuencias vista anteriormente.

Sin embargo, se debe tener en cuenta que esta instrucción gráfica aplica a posteriori un sistema de modelizado matemático a los resultados obtenidos, por lo que la utilización de determinados comandos estadísticos, como la media ponderada en función de la frecuencia, pueden no corresponderse con la realidad de los datos y desvirtuar la gráfica.

5-. Aumentar la resolución de la gráfica: ***resolution***

Es posible indicar a la gráfica que opte por dos tipos de resolución del renderizado utilizado. La resolución normal "normal", que se encuentra por defecto, y que suele ser suficiente en la mayoría de los casos, y la resolución de más detalle "fine", que requiere de mayores recursos para el cálculo, pero que proporciona gráficas más detalladas.

6-. Establecer los valores límite para la representación: ***limits***

Por defecto la instrucción polarPlot escoge sus valores límite de forma automática. No obstante, el usuario puede utilizar otros valores límite distintos, debiendo indicar a la gráfica el mínimo y el máximo que tiene que aplicar el comando. Este comando es especialmente útil si queremos centralizar los resultados obtenidos para determinadas concentraciones, evitando así interferencias.

Ej. **polarPlot (datoscont, pollutant="cov", x="t", limits=c(2,7)),** conseguirá que la gráfica vista en l página anterior no se vea tan dispersa a lo largo de todo el rango de temperaturas, representando aquellas medias de compuestos orgánicos volátiles que se encuentran entre 2 y 7, lo que eliminará gran parte del contorno exterior de la gráfica.

7-. Limitar la modelización de los datos: ***exclude.missing***

La instrucción polarPlot modeliza los datos a lo largo de todo el rango de velocidades del viento (o del parámetro que se haya designado como x), interpolando los datos existentes a aquellas zonas donde no los hay. Sin embargo, la representación gráfica global de este modelizado está desactivada por defecto en polar Plot, ***exclude.missing=TRUE***, de forma que el modelizado se aplique sólo a los datos existentes, evitando modelizar aquellos que no existan, lo que permite evaluar la coherencia de los datos reales y comprobar si existe discrepancias en los mismos.

Si se quiere desactivar este comando, de forma que la gráfica polatPlot modelice las concentraciones en toda su extensión, basta con indicar que exclude.missing=FALSE, y la gráfica se rellenará en toda su extensión.

8-. Mostrar la incertidumbre de la predicción: ***uncertainty***

Es posible ver la incertidumbre con la que se calcula la superficie de la gráfica, indicando al comando que se active: ***uncertainty=TRUE***. Al activar el comando se generan tres gráficas que comparten la escala, donde se muestra la predicción o modelización realizada junto a la incertidumbre aplicada, tanto inferior como superior, lo que se muestra muy útil a la hora de comprobar si la gráfica se corresponde o no a la realidad.

9-. Cambiar los colores predeterminados de la gráfica: ***cols***

Como se ha visto en anteriores gráficas, Openair permite personalizar los colores por otros ya programados en el paquete de R, o personalizarlos al gusto del usuario. El comando cols se mantiene también en la gráfica polarPlot, donde presenta el mismo funcionamiento.

10-. Cambiar el número mínimo de datos por combinación: ***min.bin***

Este comando es preferible no usarlo si no se quiere afectar al conjunto de los datos y perder determinados datos. Mediante el comando min.bin, tal y como se ha visto en la gráfica polarFreq, lo que se consigue es indicarle a la instrucción el número mínimo de datos permitido para cada conjunto o casilla de dirección / velocidad de viento, es decir, el número mínimo de datos que se considera representativo que existan para hacer el cálculo estadístico que le indiquemos. Por defecto el valor adoptado por la gráfica polarPlot es 1, por lo que se usan todos los datos del conjunto de datos.

No obstante, el usuario puede indicar a la gráfica un número superior, en cuyo caso, lo que hace la instrucción es contar el número de datos para cada celda o rango de velocidad / dirección de viento. Si el número existente de datos por celda no alcanza el número mínimo indicado por el usuario, los datos disponibles son eliminados del conjunto de datos asignándoles la denominación "n.a." para posteriormente realizar el cálculo estadístico y graficar los datos.

El comando es en sí muy útil, tal y como ya hemos visto, pero puede resultar muy peligroso para la integridad del conjunto de datos que tengamos, por lo que es recomendable ejecutarlo sobre ventanas de datos que sean copia de las que ya disponemos.

11-. Indicar el valor máximo de la velocidad del viento: ***upper***

El usuario puede cambiar el límite superior a aplicar a la velocidad del viento que se representa en la grafica, lo que es especialmente útil cuando existen unos pocos datos a latas velocidades que desvirtúan la escala e impiden comprobar con comodidad lo que ocurre a rangos de velocidad inferiores.

12-. Cambiar el ángulo en el que se representa la escala: ***angle.scale***

Por defecto, la escala de velocidades del viento o de las unidades del parámetro seleccionado por el usuario se representan en el ángulo 315 de la gráfica (cuadrante superior izquierdo). Sin embargo, pudiera ocurrir que la utilización de este ángulo impidiese una correcta visualización de algún tipo de fenómeno de interés con respecto a las distribuciones, por lo que es posible cambiar el mismo personalizando el ángulo mediante este comando.

13-. Establecer las unidades de la escala: ***units***

Es posible establecer las unidades en las que se representa la escala en la gráfica mediante el comando units, al que se debe añadir entre comillas las unidades que se presentan en los ejes.

14-. Forzar los valores negativos de la modelización a positivos: ***force.positive***

La modelización realizada por la instrucción polarPlot podría arrojar valores negativos para determinados puntos de la gráfica, lo que evidentemente sería del todo imposible, si se tratase de un parámetro contaminante (que es el normalmente utilizado en estas gráficas).

Por defecto, la instrucción polarPlot tiene activada la opción de forzar los datos positivos, ***force.positive=TRUE***, pues en la mayoría de los casos los valores negativos no serían correctos. Sin embargo, es posible cambiar esta opción si el usuario considera que el parámetro a representar puede tener valores negativos.

15-. Cambiar el parámetro de modelizado K: ***k***

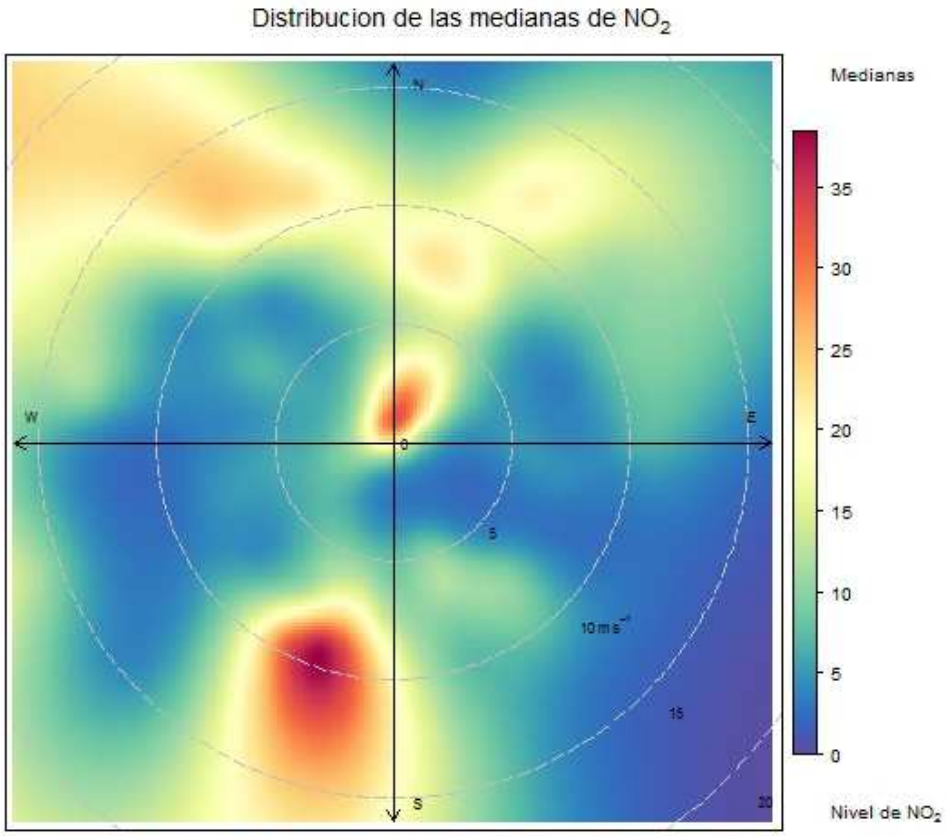
K es el parámetro utilizado por la ecuación de modelización que utiliza polarPlot para el suavizado de la cuadrícula de datos que forma la superficie de la gráfica. La amplia experiencia de los programadores de Openair en la explotación de este tipo de gráficas ha terminado por recomendar que el valor por defecto de esta variable sea de ***k=100***, pues es el que se ha demostrado más eficiente, aunque se ha previsto la posibilidad de que el usuario de la instrucción introduzca un valor distinto.

En cualquier caso, conviene saber que valores de K superiores a 100 generan gráficas algo más definidas en su malla pero incrementando enormemente el tiempo de computación, sin que realmente se afecte de forma relevante a las predicciones que arroja el modelo. Valores de K inferiores a 100 incrementan la suavidad con la que se presenta la gráfica pero pueden generar fallos en aquellos puntos en los que los datos sean mínimos, lo que aún es menos recomendable.

16-. Normalizar los datos en función del promedio: ***normalise***

En puntos anteriores hemos visto que podemos solicitar a polarPlot que represente varias gráficas polares para distintos contaminantes, indicando al comando ***pollutant*** que adopte varias variables, aunque también hemos precisado a este respecto que, al compartir escala, es recomendable que las concentraciones de ambos contaminantes se muevan en escalas muy similares.

Con este nuevo comando, la gráfica ofrece la posibilidad de normalizar los resultados una vez realizada la modelización, dividiendo los resultados obtenidos por la media de concentración del parámetro seleccionado, lo que normaliza a una única escala las concentraciones encontradas para los distintos contaminantes. Este comando resulta así de enorme utilidad si lo que se quiere es comparar concentraciones de parámetros que se encuentran a distintas escalas, eliminando la limitación que teníamos anteriormente.




```
> polarPlot (datoscont, pollutant="no2", statistic="median", exclude.missing=FALSE, angle.scale=135, ws.int=0.2, resolution="fine", units="m/s", main="Distribuicon de las medianas de NO2", key.header="Medianas", key.footer="Nivel de NO2")
```

» La Gráfica Temporal Polar o Función Anular: *polarAnnulus*

La función polar temporal o función anular, dada su forma de anillo, es una función que está todavía en fase de desarrollo e investigación, pero que ya apunta buenas maneras en cuanto a su potencial y practicidad, razón por la que se incluye en este manual.

Dicha función representa la concentración alcanzada del contaminante que se seleccione en función de los aspectos temporales que se precisen, actuando como si quisiésemos representar las opciones temporales del comando type o cutData en los ejes de dirección del viento. Dicho de otra forma, la función polarAnnulus viene a representar en la dirección del viento las concentraciones alcanzadas del parámetro en función de las divisiones temporales que indique el usuario.

De esta forma, se puede ver, por ejemplo, a qué horas del día se producen las mayores concentraciones de un contaminante dado (la escala de las horas la darán los ejes de la gráfica) y en qué dirección del viento se producen de forma mayoritaria, lo que aporta un potencial adicional a la hora de identificar las fuentes de origen de la contaminación.

Si bien es cierto que este mismo efecto se podría lograr con el comando type y la gráfica polarPlot vista en el anterior apartado, sin perder por ello la definición de la velocidad del viento, la instrucción polarAnnulus nos permitiría ver la evolución un una sola gráfica y de manera más visual.

Se debe prestar especial atención al hecho de que el conjunto de datos sea lo suficientemente grande y que exista un correcto registro de la dirección del viento para que el programa pueda ejecutar correctamente el modelizado de los datos sin presentar ningún error en la predicción matemática que realiza para la representación gráfica, o huecos en blanco en la superficie de la gráfica.

Del mismo modo, al ser una función gráfica que representa periodos temporales en los ejes de dirección, habrá que tener en cuenta que, en el caso de que dichos periodos se integren a escala horaria, podrían quedar afectados por el horario local del área de estudio, lo que habrá que tener en cuenta, llegando incluso a ser necesaria la transformación de datos de GMT a horario local (ver punto 3 del apartado de variaciones sobre la función).

Variaciones sobre la función:

1-. El contaminante o parámetro a representar: ***pollutant***

Se deberá indicar a la instrucción gráfica el contaminante a representar mediante este comando. También se le puede indicar a la gráfica, al igual que se ha visto en otras funciones polares, que represente varios parámetros que estén dentro de la misma escala, para realizar la comparativa gráfica correspondiente, generando así varias gráficas de anillo con la misma escala.

2-. Modificar la resolución a aplicar a la gráfica: ***resolution***

Al igual que ocurría con la gráfica polarPlot, es posible indicarle a la instrucción polarAnnulus el tipo de resolución que queremos aplicar a nuestra gráfica: “normal” o “fine”. En esta ocasión, sin embargo, la opción “fine” es la que encontraremos por defecto, lo que habrá que tener en cuenta si lo que queremos es agilizar la generación gráfica para determinados paquetes de datos.

3-. Corregir a horario local los datos: ***local.time***

Es posible indicarle a esta gráfica que corrija los datos del conjunto a horario local mediante este comando que, no obstante, conviene recordar que está activado por defecto TRUE, por lo que siempre realizará la mencionada corrección, salvo que se le indique lo contrario.

La opción de corregir a horario local es muy interesante para esta función puesto que los niveles en inmisión se detectan en realidad en horario local, y en conjunción con el horario que siguen las actividades generadoras de emisiones a la atmósfera (que también se rigen por el horario local). Además, al mencionado horario local le afectar el cambio horario que se produce en determinados países de verano a invierno, al igual que a las actividades emisoras, lo que desvirtuaría aún más las gráficas polares anulares si utilizásemos los datos directamente en GMT.

No obstante, conviene recordar que el paquete Openair es de origen británico, por lo que la corrección se realiza en función de la franja horaria británica (BTU), que no tiene por qué coincidir con la franja temporal local aplicable a la estación de inmisión sobre la que queramos realizar el estudio de datos. De darte este hecho, el comando local.time no sería práctico en nuestra función, y deberíamos desactivarlo para evitar que desvirtúe la representación, debiendo realizar el cambio a horario local directamente sobre los datos.

4-. Establecer el periodo a representar en los ejes: ***period***

Ya hemos comentado, al principio de esta instrucción gráfica, que la principal novedad de la función gráfica polarAnnulus es la utilización de los periodos temporales como base para dividir los ejes que representan la dirección del viento. En este sentido, mediante el comando *period*, la instrucción *polarAnnulus* permite representar las horas del día *"hour"*, las estaciones del año *"season"*, los días de la semana *"weekday"*, o incluso la propia evolución de los datos a lo largo de todo el periodo contemplado por el conjunto de datos *"trend"*.

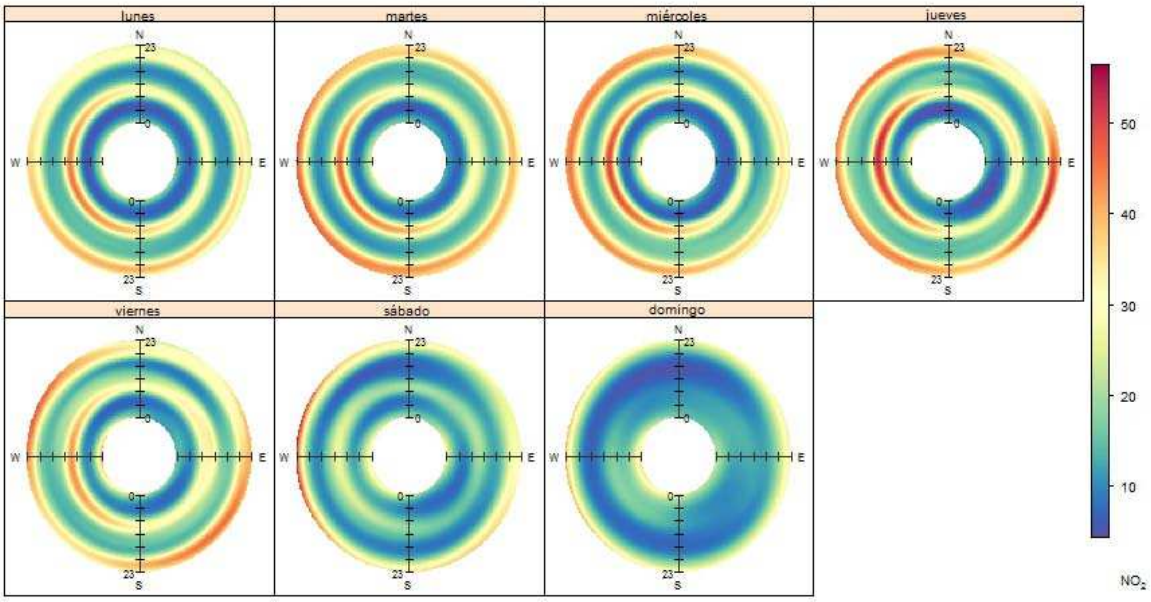
Ej. **polarAnnulus(datoscont, pollutant="no2", period="hour"),** representará las concentraciones de NO2 para cada una de las direcciones del viento, dividiendo dichas direcciones en cada una de las horas del día, lo que nos permite ver a qué horas se producen las concentraciones medias más altas, y en qué direcciones del viento, permitiendo así una mejor caracterización de las fuentes de origen.

5-. Dividir la gráfica desagregando los datos: ***type***

Este comando, visto en anteriores instrucciones gráficas, consigue en el caso de la función *polarAnnulus*, dar una vuelta de tuerca más al concepto de desagregación de datos al permitir una "doble clasificación" de los mismos, en conjunción con el comando *period* visto anteriormente.

Si se utiliza de forma adecuada, permite añadir a la función un doble criterio de clasificación de los datos, en primer lugar el criterio temporal establecido por *period* y representado en los ejes de dirección del viento, y en segundo lugar el criterio temporal o de clasificación en el que dividirá la gráfica el comando *type*.

Ej. **polarAnnulus(datoscont, pollutant="no2", period="hour", type="weekday"),** dividirá la gráfica vista en el punto anterior en siete gráficas distintas, una por cada día de la semana, y establecerá para cada una de esas gráficas polares las concentraciones medias detectadas de NO2 para cada hora y dirección del viento, lo que aporta una valiosa información sobre las periodicidades con las que se dan las concentraciones más altas, y las direcciones en las que preponderan las mismas. Un ejemplo de esta función gráfica es el que se muestra a continuación, según los criterios de desagregación aquí expuestos, aunque en el ejemplo de final de apartado también se muestran los datos desagregados mediante el comando type.



6-. Límites en los que se moverá la escala de colores: ***limits***

La instrucción polarAnnulus escoge automáticamente una escala de representación del contaminante seleccionado en función del rango en el que se muevan las concentraciones del mismo, optimizando la escala para representar todos los valores. Sin embargo, en algunas ocasiones puede ser necesario indicar al programa que utilice un rango de datos distinto al originalmente indicado, para lo cual se utiliza el comando *limits*, al que hace falta indicarle los valores mínimo y máximo en los que se moverá la representación gráfica a realizar, como por ejemplo ***limits=c(10,80)***

7-. Cambiar la combinación de colores a utilizar en la gráfica: ***cols***

Como se ha visto en anteriores gráficas, Openair permite personalizar los colores por otros ya programados en el paquete de R, o personalizarlos al gusto del usuario. El comando cols se mantiene también en la gráfica polarAnnulus, donde presenta el mismo funcionamiento.

8-. Cambiar el ancho aplicado al anillo: ***width***

Este comando afecta exclusivamente a la representación gráfica y no a las unidades o a la escala en que se presenten los datos, permitiendo cambiar el ancho a aplicar a la representación anular atendiendo a tres posibles variantes: *"thin"*, para un anillo fino, *"normal"* para un grosor estandar del anillo, aplicado por defecto, o *"fat"* cuando la representación requiere de un grosor superior.

9-. Establecer el número mínimo de datos representativos: ***min.bin***

Este comando es preferible no usarlo si no se quiere afectar al conjunto de los datos y perder determinados datos. Mediante el comando min.bin, tal y como se ha visto en las gráficas polares anteriores, lo que se consigue es indicarle a la instrucción el número mínimo de datos permitido para cada conjunto o casilla de dirección / velocidad de viento, es decir, el número mínimo de datos que se considera representativo que existan para calcular el promedio. Por defecto el valor adoptado por la gráfica polarAnnulus es 1, por lo que se usan todos los datos del conjunto de datos.

No obstante, el usuario puede indicar a la gráfica un número superior, en cuyo caso, lo que hace la instrucción es contar el número de datos para cada celda o rango de velocidad / dirección de viento. Si el número existente de datos por celda no alcanza el número mínimo indicado por el usuario, los datos disponibles son eliminados del conjunto de datos asignándoles la denominación "n.a." para posteriormente realizar el cálculo estadístico y graficar los datos.

El comando es, en sí mismo, muy útil, tal y como ya hemos visto en anteriores apartados, pero puede resultar muy peligroso para la integridad del conjunto de datos que tengamos, por lo que es recomendable ejecutarlo sobre ventanas de datos que sean copia de las que ya disponemos.

10-. Eliminar aquellas previsiones demasiado lejos del dato real: ***exclude.missing***

La instrucción polarAnnulus modeliza los datos a lo largo de todo el anillo, en función del rango temporal asignado por el comando period, interpolando los datos existentes a aquellas zonas donde no los hay. Sin embargo, la representación gráfica global de este modelizado está desactivada por defecto en polar Plot, ***exclude.missing=TRUE***, de forma que el modelizado se aplique sólo a los datos existentes, evitando modelizar aquellos que no existan, lo que permite evaluar la coherencia de los datos reales y comprobar si existe discrepancias en los mismos.

Si se quiere desactivar este comando, de forma que la gráfica polatPlot modelice las concentraciones en toda su extensión, basta con indicar que exclude.missing=FALSE, y la gráfica se rellenará en toda su extensión.

11-. Rellenar la gráfica para cubrir las fechas no disponibles: ***date.pad***

Si utilizamos el periodo *"trend"*, que viene por defecto en la gráfica polarAnnulus, y por lo tanto representamos las concentraciones a lo largo de todo el periodo de datos disponible, la gráfica extenderá sus anillo desde la fecha de inicio del paquete de datos hasta la fecha de final del mismo.

Sin embargo, mediante el comando *date.pad* es posible indicar a la gráfica que rellene el anillo, utilizando la modelización de datos, hasta alcanzar como límites de representación los comienzos y finales de los años, en función del rango de datos disponible, extendiendo así sus predicciones a los periodos anuales completos, lo que facilita la comprensión de las gráficas. La activación del comando para permitir este modelizado se realizará mediante la expresión ***date.pad=TRUE***.

12-. Forzar resultados positivos en la predicción del modelo: ***force.possitive***

Tal y como ya se vio en la gráfica polarPlot, en ocasiones el modelizado de la superficie de la gráfica a través del modelo matemáticos previsto puede terminar por dar datos de concentración negativos. Evidentemente, al hablar de concentraciones de contaminantes, este hecho es imposible por definición, por lo que por defecto la instrucción polarAnnulus tiene activado el comando *force.possitive=TRUE*, lo que hace que los datos negativos del modelo se fuercen a positivos.

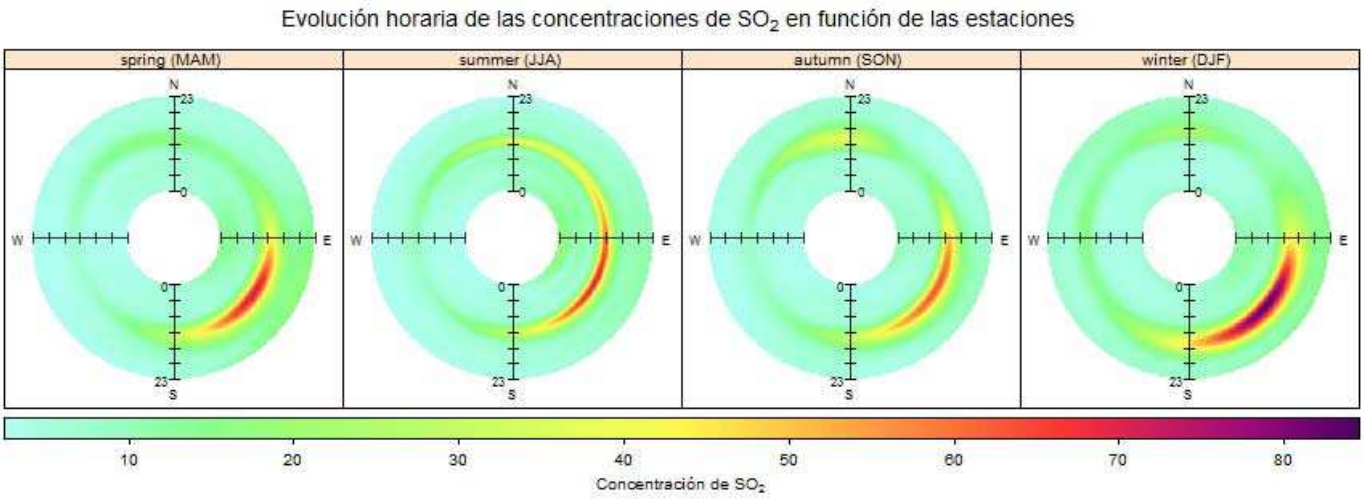
13-. Cambiar el parámetro de modelizado K: ***k***

K es el parámetro utilizado por la ecuación de modelización que utiliza polarAnnulus, al igual que en el caso de la función polarPlot, para el suavizado de la cuadrícula de datos que forma la superficie de la gráfica. La amplia experiencia de los programadores de Openair en la explotación de este tipo de gráficas ha terminado por recomendar que el valor por defecto de esta variable sea de ***k=100***, pues es el que se ha demostrado más eficiente, aunque se ha previsto la posibilidad de que el usuario de la instrucción introduzca un valor distinto: superior, si lo que se quiere es una gráfica algo más definida, o inferior a 100 si se quiere incrementar la suavidad y reducir el tiempo de computación.

14-. Normalizar los datos en función del promedio: ***normalise***

En puntos anteriores hemos visto que podemos solicitar a polarAnnulus que represente varias gráficas polares para distintos contaminantes, indicando al comando ***pollutant*** que adopte varias variables, aunque también hemos precisado a este respecto que, al compartir escala, es recomendable que las concentraciones de ambos contaminantes se muevan en escalas muy similares.

Con este nuevo comando, la gráfica ofrece la posibilidad de normalizar los resultados una vez realizada la modelización, dividiendo los resultados obtenidos por la media de concentración del parámetro seleccionado, lo que normaliza a una única escala las concentraciones encontradas para los distintos contaminantes. Este comando resulta así de enorme utilidad si lo que se quiere es comparar concentraciones de parámetros que se encuentran a distintas escalas, eliminando la limitación que teníamos anteriormente.




```
> polarAnnulus (datoscont, pollutant="so2",  period="hour", type="season", width="fat", exclude.missing=FALSE, cols="increment", layout=c(4,1), key.header="Concentración de SO2", key.footer="", key.possition="bottom", main="Evolución horaria de las concentraciones de SO2 por estaciones")
```

» La Gráfica Polar de Particiones: *polarCluster*

La Gráfica Polar con clusters o particiones es una gráfica complementaria a las gráficas polares vistas hasta el momento, y específicamente a la gráfica *polarPlot*. Su uso está destinado en exclusividad al técnico ambiental encargado del análisis de los datos de calidad del aire, por cuanto que su misión es hacer particiones de la superficie polar y representarla como subconjuntos de datos con características similares en una gráfica polar, lo que permite identificar así áreas de estudio para un procesado posterior de los datos con el resto de gráficas y funciones de *Openair*.

El fundamento de uso de esta gráfica consiste en eliminar el error o sesgo que pudiese existir en el estudio directo de gráficas como *polarPlot,* al aplicarse por parte del técnico criterios que podrían resultar arbitrarios, ya sea por existir condicionantes previos en el estudio de los datos, o por factores de diseño que pudiesen afectar a su interpretación (como por ejemplo el uso de determinadas escalas de colores).

La función polarCluster, al ejecutarse, asume para su cálculo una contribución homogénea de los componentes vectoriales del viento y de la concentración del contaminante seleccionado por el usuario, para luego estandarizar los datos disponibles y posteriormente aplicar una técnica de partición de la superficie de la gráfica polar basada en un algoritmo *PAM (Partition Around Medoids)*, uno de los más usados en el análisis de clusters por partición basada en centroides *(k-means)* . De esta forma, a nivel práctico lo que conseguimos es disponer de áreas en las que los datos disponen de características muy parecidas y sobre las que se podrá realizar un análisis con fundamentos científicos sólidos.

A pesar de que podría resultar accesoria, la gráfica *polarCluster* es esencial a la hora de realizar una prospección correcta de grandes series de datos de calidad del aire (también conocida como minería de datos) y evitar un tratamiento arbitrario de los mismos.

La función *polarCluster* se ejecutará bajo los siguientes comandos básicos:

Variaciones sobre la función:

1-. Parámetro contaminante a representar ***pollutant***

El comando *pollutant* es preciso para indicarle a la gráfica el parámetro que debe representar. En caso de no introducirlo, la instrucción *polarCluster* acude, al igual que *polarPlot*, automáticamente a buscar el parámetro *"nox"*. Sin embargo, y al contrario que en la gráfica *polarPlot*, la función *polarCluster* no admitirá para la representación más de un parámetro.

2-. Parámetro a representar en los ejes de dirección: ***x***

Por defecto la función *polarCluster*, al igual que *polarPlot*, utiliza la velocidad del viento para combinarla con la dirección del viento representada en los ejes. Sin embargo, es posible sustituir la velocidad del viento en la gráfica, pudiendo disponer en su lugar de cualquier otro parámetro, tal y como ya se contemplaba con *polarPlot*. Este hecho permite ampliar el estudio por áreas a los mismos niveles que en la función *polarPlot*.

3-. Nombre del campo que representa la dirección del viento: ***wd***

En ocasiones es posible que el campo de dirección del viento no aparezca en nuestro conjunto de datos como *"wd"* (su abreviatura en inglés), por lo que con el comando *wd* deberemos indicar a la función *polarCluster* el nombre del campo de nuestro conjunto de datos que dispone de los datos sobre dirección de viento para que sea capaz de representarlos en una gráfica polar.

4-. Establecer el número de clusters a representar: ***n.clusters***

Uno de los parámetros a configurar en la función *polarCluster* de mayor importancia es el que establece el número de clusters o particiones a realizar en la gráfica, ya que un número bajo de particiones diluirá las áreas de interés al presentar pocas superficies, por lo que será imposible interpretar el interés surgido en las distintas áreas de estudio, y un número muy elevado puede hacer impracticable la interpretación y posterior estudio de los datos al existir demasiadas divisiones.

En ocasiones puede ser interesante indicar a la función que utilice un rango de variables, del tipo *n.clusters=a:b* , para utilizar varios clusters distintos a la vez. De esta forma la gráfica se dividirá para representar tanta gráficas polares como clusters se le hayan indicado en el rango, y el usuario podrá comprobar qué número de clúster se adapta mejor a la descripción de las zonas de estudio más representativas.

5-. Cambiar el ángulo en el que se representa la escala: ***angle.scale***

Por defecto, la escala de velocidades del viento o de las unidades del parámetro seleccionado por el usuario se representan en el ángulo 315 de la gráfica (cuadrante superior izquierdo). Sin embargo, pudiera ocurrir que la utilización de este ángulo impidiese una correcta visualización de algún tipo de fenómeno de interés con respecto a las distribuciones, por lo que es posible cambiar el mismo personalizando el ángulo mediante este comando.

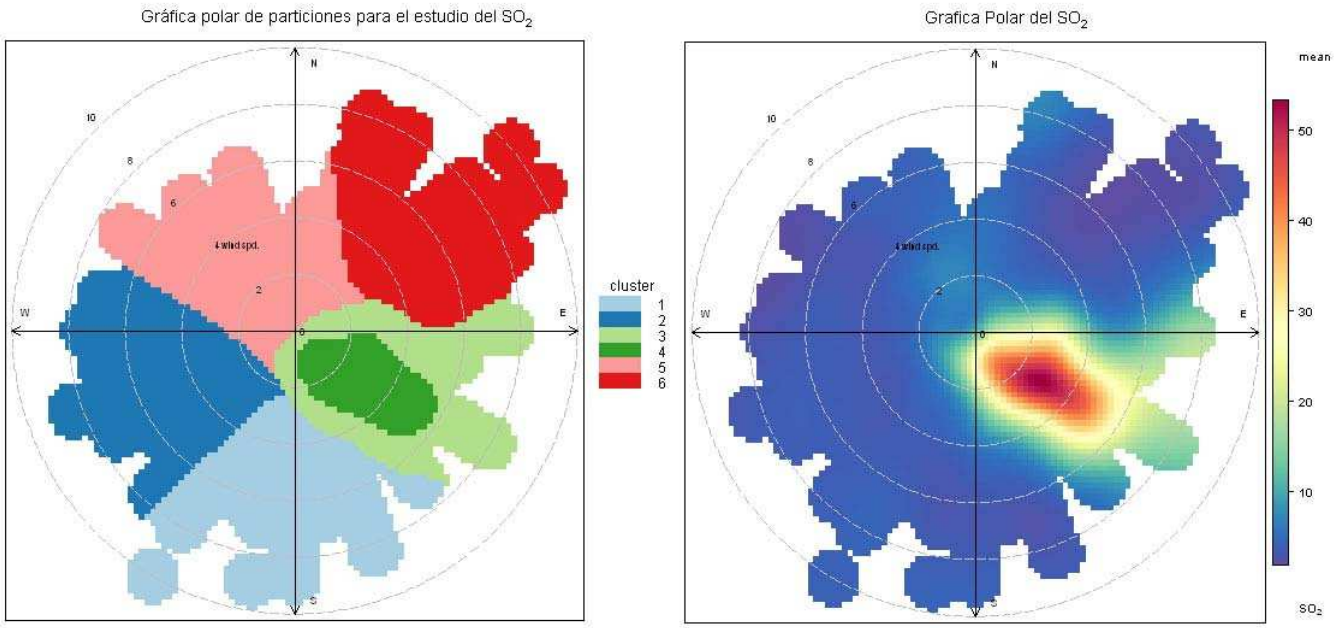
6-. Cambiar los colores predeterminados de la gráfica: ***cols***

Como se ha visto en anteriores gráficas, Openair permite personalizar los colores por otros ya programados en el paquete de R, tales como “Accent”, “Dark2”, “Paired”, etc. o personalizarlos al gusto del usuario. El comando cols se mantiene también en la gráfica polarCluster, donde presenta el funcionamiento habitual.

7-. Establecer las unidades de la escala: ***units***

Es posible establecer las unidades en las que se representa la escala en la gráfica mediante el comando units, al que se debe añadir entre comillas las unidades que se presentan en los ejes.

Se muestra a continuación la ejecución de una Gráfica polar de clusters sobre el mismo conjunto de datos y el mismo parámetro contaminante, en este caso el SO_2, que una gráfica polarPlot. Se puede observar como las particiones calculadas muestran fielmente las áreas de estudio que presentan los datos, sirviendo así como fundamento matemático para la realización de estudios posteriores centrados en las áreas de mayor interés.




```
> polarCluster ( datoscont,  pollutant="so2",  main="Gráfica polar de particiones para el estudio del SO2",  n.cluster=6).
> polarPlot ( datoscont,  pollutant="so2",  main="Grafica Polar del SO2")
```

» La Función Gráfica de Tiempo: *timePlot*

La función gráfica de tiempo se corresponde con la función gráfica clásica, similar al Plot que anteriormente hemos visto que utilizaba R, en la que se representa la evolución de las concentraciones de un determinado parámetro, representado en el eje Y, frente al tiempo, representado en el eje X.

La función gráfica de tiempo es en sí misma una gráfica típica, ampliamente utilizada en calidad del aire, y disponible a través de muchas de las actuales herramientas de cálculo del mercado. Sin embargo, Openair dota a esta función gráfica de unas peculiaridades y una versatilidad que la acaban convirtiendo en algo único, muy útil y práctico para el estudio de la evolución temporal de la contaminación.

Así, la función timePlot diseñada en Openair nos permitirá graficar el número de parámetros que deseemos, indicarle el periodo de tiempo que queramos, para graficar con mayor o menor detalle, en una sola gráfica o en varias, entre otras muchas opciones de configuración que se estudiarán a través de los siguientes comandos.

Variaciones sobre la función:

1-. Los contaminantes a representar en la gráfica: ***pollutant***

A la gráfica timePlot se le puede indicar que represente uno o varios contaminantes mediante el comando pollutant, ya visto en anteriores gráficas, al que podremos indicar distintas variables: ***pollutant=c("no2", "no")***, que podrán compartir o no la misma escala, tal y como veremos en posteriores comandos.

2-. Agrupar los contaminantes en una sola gráfica: ***group***

Por defecto, la gráfica timePlot, según le vayamos pidiendo parámetros a representar los irá agregando en gráfica nuevas, uno debajo del otro, sin agrupar y con su propia escala de representación. Esto es debido a que la opción de agrupamiento de datos está, de inicio, desactivada: ***group=FALSE***. Si activamos esta opción, todos los contaminantes que le indiquemos se representarán en la misma gráfica compartiendo escala.

3-. Elaborar gráficas por años: ***stack***

Al elaborar gráficas de un conjunto de datos muy extenso en el tiempo, podemos encontrarnos con dificultades a la hora de visualizar los datos, por lo que una de las herramientas más útiles podría ser la de dividir el conjunto de datos en años, activando el comando ***stack=TRUE***.

Si seleccionamos varios contaminantes para su representación gráfica, e indicamos a la instrucción timePlot que divida la gráfica en años, con el comando stack, nos encontraremos con que la función agrupará dichos contaminantes en una sola gráfica, para representarlos por años, aunque el comando group establezca lo contrario.

4-. Normalizar los datos graficados: ***normalise***

La gráfica timePlot dispone de la opción de normalizar los datos, que por defecto estará desactivada, pero que puede ser muy útil para realizar comparativas entre distintos datos y periodos temporales. La normalización de los datos se lleva a cabo mediante el comando normalise, que podrá ejecutarse de dos maneras muy distintas:

a) p.e. ***normalise="mean"***, en cuyo caso procederá a dividir todos los datos de los contaminantes seleccionados por la media obtenida para el mismo, en lo que sería la normalización habitualmente más usada para el contraste de los datos.

b) p.e. ***normalise="01/08/2011"***, utilizando la fecha que mejor considere el usuario en el formato dd/mm/AAAA, en cuyo caso la concentración obtenida para dicha fecha (en nuestro caso el primero de agosto de 2011), se fija en el valor 100, y el resto de datos se escala en función de dicha normalización. Este tipo de normalización resulta muy útil para poder ver mejor la evolución de un parámetro en el tiempo y referida a una fecha en concreto.

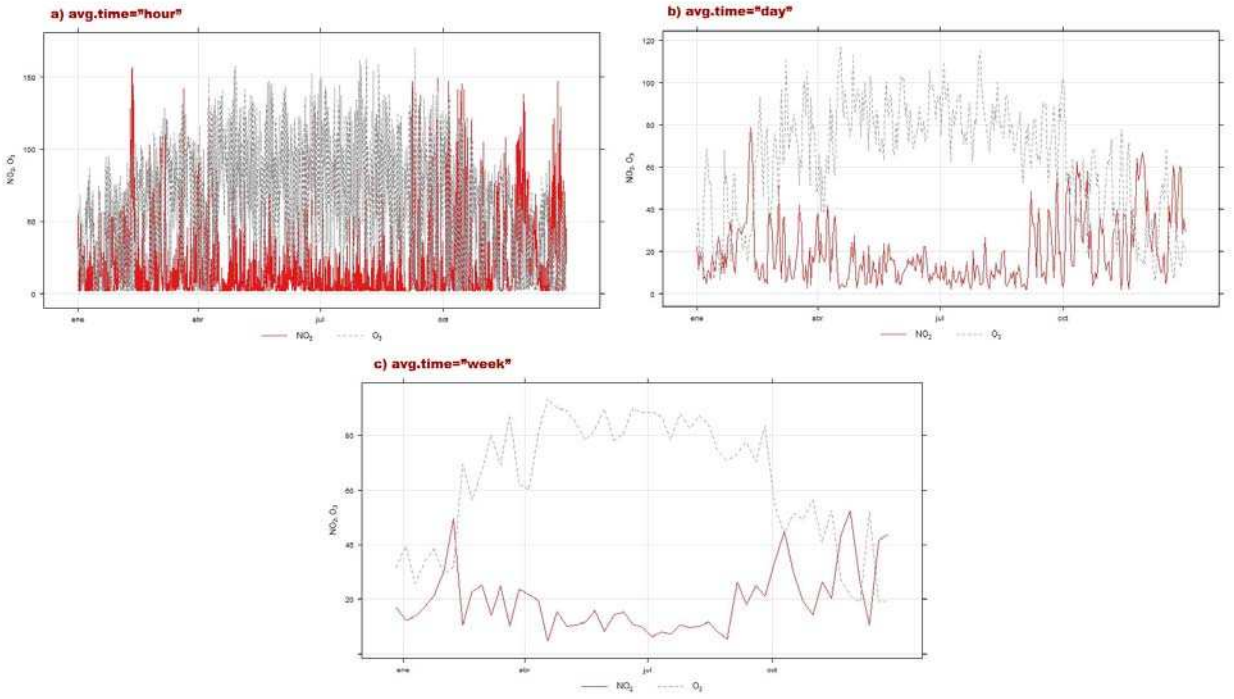
5-. Cambiar el tiempo de agregación de los datos: ***avg.time***

Probablemente, aún con periodos de tiempo relativamente cortos, la representación gráfica de los datos horarios no termine de verse claramente, más aún cuando se busca representar más de un parámetro, o incluso el usuario prefiera la representación de los datos en otro periodo de integración que no sea el utilizado por el conjunto de los datos.

Para estos casos la función timePlot, al igual que otras gráficas temporales que veremos más adelante, integra en sus comandos a la instrucción ***timeAverage***, vista en otros capítulos de este manual, y permite con el comando avg.time seleccionar distintos periodos de tiempo para la agregación o desagregación de los datos por segundos "sec", minutos "min", horas "hour", días "day", semanas "week", meses "month", estaciones del año "season", cuatrimestres "quarter", o incluso por años "year".

Además, cabe recordar que la variable seleccionada para el comando avg.time podrá venir precedida de un número que modificará el comando seleccionado, lo que aporta una mayor versatilidad todavía al mismo.

Ej. **a) timePlot (datoscont, pollutant=c("no2", "o3"), group=TRUE),** que graficará los datos de nuestro conjunto de datos en el periodo horario en el que viene de orígen. // **b) timePlot (datoscont, pollutant=c("no2", "o3"), group=TRUE, avg.time="day"),** que graficará los datos de nuestro conjunto de datos utilizando las medias diarias resultantes, lo que clarifica mucho más la relación existente entre el O3 y el NO2 a lo largo del año. // **c) timePlot (datoscont, pollutant=c("no2", "o3"), group=TRUE, avg.time="week"),** que graficará los datos de nuestro conjunto de datos utilizando las medias semanales.



6-. Utilizar diversos estadísticos para el periodo de agregación: ***statistics***

Cuando se integran los datos de que disponemos en un periodo de integración mayor al original, el programa utiliza por defecto como cálculo estadístico la media de los datos en el periodo seleccionado "mean", tal y como se puede observar en las gráficas anteriores.

No obstante, el estadístico a utilizar puede ser este o el máximo "max", el mínimo "min", la mediana "median", al frecuencia "frequency", la desviación estandar "sd" o el percentil "percentile", siempre que se indique mediante el comando statistic.

Si utilizamos el estadístico "percentile" será preciso que le indiquemos a la gráfica el percentil que queremos utilizar , en porcentaje, mediante el comando percentile. También será posible indicar a la gráfica la posibilidad de calcular más de un percentil, por ejemplo ***percentile=c(99,95)***, en cuyo caso la gráfica se realizará para cada uno de los percentiles indicados.

7-. Establecer el número mínimo de datos necesarios: ***data.thresh***

Dado que la gráfica se basa en la realización de cálculos estadísticos, podría resultar preciso contemplar un cierto grado de representatividad en los datos graficados, tal y como ocurría con la instrucción timeAverage vista anteriormente. Mediante el comando data.thresh es posible indicar a la función el porcentaje mínimo de datos necesarios para considerar un periodo como representativo en el cálculo del estadístico.

8-. Extraer los huecos de datos de la gráfica: ***date.pad***

Cuando un conjunto de datos está compuesto por periodos de datos inconexos temporalmente, con huecos de datos entre sí, pueden presentarse problemas en la representación gráfica que realice la función. Activar este comando de la función , ***date.pad=TRUE***, permite tratar adecuadamente los huecos de datos evitando la generación de líneas de conexión entre datos.

9-. Desagregar los datos a representar en la gráfica: ***type***

Mediante el comando type , visto en gráficas anteriores, se le puede indicar a la instrucción que desagregue los datos a representar en función de variables ya vistas como las estaciones del año “season”, los años “year”, o incluso cualquier otro parámetro del grupo de datos, entre otros. El comando type terminará dividiendo la gráfica para representar la línea temporal en función de lo indicado en el mismo.

10-. Cambiar los colores de la representación: ***cols***

El comando cols permite cambiar los colores de las líneas que representan los datos, ya sea asignando los paquetes de colores predeterminados de que dispone Openair, como “brewer1”, “heat” o “increment”, o disponiendo por parte del propio usuario de los colores que se consideren más oportunos para la representación.

11-. Modificar el tipo de símbolo a utilizar en la gráfica: ***plot.type***

Otro comando de utilidad para personalizar la representación gráfica es el comando plot.type, equivalente al comando type que pudimos ver al prinicipio en las gráficas plot genéricas de R. Este comando permite asignar distintos tipos de representación de datos, entre los que encontramos líneas que unen puntos “l” (establecido por defecto), puntos sin líneas “p”, líneas verticales “h”, puntos unidos por líneas “o”, o escalones “s”. También se le puede indicar que no represente los datos, mediante la variable “n”, ocultándolos para, por ejemplo, permitir la mejor visualización de las líneas suavizadas que veremos más adelante.

12-. Modificar la leyenda de la gráfica: ***key***

En el caso de las gráficas timePlot la leyenda no se configura ni gestiona con en las gráficas vistas hasta el momento, y según se establece al principio de este apartado sobre instrucciones gráficas, puesto que dicha leyenda no sirve para representar la escala de datos, sino que simplemente ilustra los parámetros representados en la gráfica.

En el caso de la función gráfica timePlot el comando key es un comando lógico que permite la visualización o no de la leyenda. Así, ***key=FALSE*** eliminará la leyenda de la gráfica.

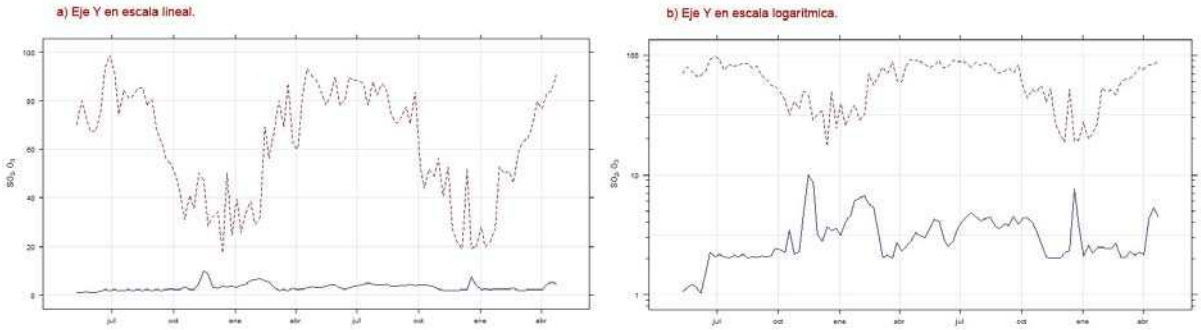
Además, la función contempla la posibilidad de configurar la leyenda en la gráfica, permitiendo indicar a la misma el número de columnas a utilizar cuando se dispone de más de un contaminante a representar. El comando previsto en este sentido es ***key.columns***, al que le asignaremos el número de columnas que deberán componer la leyenda.

13-. Dividir el Eje Y en escala logarítmica: ***log***

Cuando representamos más de un contaminante en la misma gráfica, con ***group=TRUE***, y estos presentan valores que se encuentran en distintas escalas numéricas, como habitualmente ocurre por ejemplo con el SO2 y el O3, es muy probable que la escala lineal habitualmente utilizada en la gráfica haga que uno de los parámetros representados (el de menor escala) pierda resolución y, en algunas ocasiones, quede prácticamente plano sin posibilidad de observar su evolución.

Para estos casos está previsto el comando log, que una vez activado con ***log=TRUE***, sustituye la escala lineal que divide el eje Y por una escala logarítmica que permite un mejor contraste de los datos, tal y como se puede ver a continuación para el ejemplo propuesto.

Ej. **a) timePlot (datoscont, pollutant=c("no2", "o3"), group=TRUE, avg.time="week"),** que graficará los datos de nuestro conjunto de datos utilizando las medias semanales y en una escala lineal // **b) timePlot (datoscont, pollutant=c("so2", "o3"), group=TRUE, avg.time="week", log=TRUE),** que graficará los datos de nuestro conjunto de datos utilizando las medias semanales y en una escala logarítmica que se adaptará mucho mejor a las escalas de ambos contaminantes.



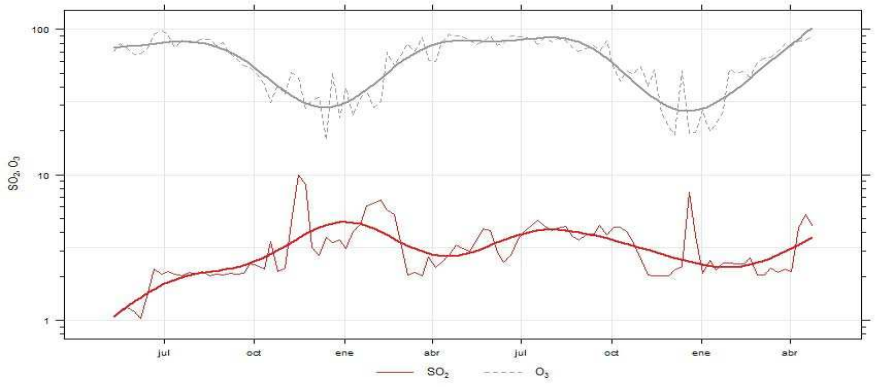
14-. Graficado de datos con líneas suavizadas y marcadores: ***smooth***

También es posible aplicar un algoritmo a los datos representados en la gráfica para suavizar su representación, modificando la posición de alguno de sus puntos para eliminar las perturbaciones que existan y representar las tendencias básicas de la curva.

El ajuste de la curva para su suavizado está desactivado por defecto, pero es posible activarlo mediante el comando ***smooth=TRUE***.

Por otro lado, y dado que la gráfica suavizada no se ajusta a los datos reales, el algoritmo calcula a su vez la dispersión de los datos reales respecto a las líneas suavizadas, y la representa en esta mediante un coloreado traslucido que el usuario puede optar por mantener o por eliminar de la misma mediante el comando ***ci=FALSE***.

Ej. **timePlot (datoscont, pollutant=c("no2", "o3"), group=TRUE, avg.time="week", log=TRUE, smooth=TRUE, ci=FALSE),** esta gráfica es igual que la gráfica del punto anterior a excepción de que en esta ocasión hemos pedido a timePlot que represente la línea suavizada de la gráfica pero eliminando de la representación la dispersión, lo que dará lugar a la gráfica que podemos ver a continuación.



15-. Añadir líneas de referencia a la gráfica: ***ref.y / ref.x***

El usuario puede añadir a la gráfica líneas de referencia adicionales a la cuadrícula sombreada que por defecto presenta siempre la gráfica. Estas líneas de referencia se presentarán como líneas marcadas de rayas, trazadas en aquellos puntos que indique el usuario, como por ejemplo ***ref.y=c(20,50)***.

16-. Cambiar el nombre de los parámetros utilizados: ***name.pol***

El ocasiones puede ser preciso cambiar los nombres de los parámetros que aparecen en la leyenda de la gráfica, que por defecto provienen del encabezamiento del conjunto de datos utilizado. Esto se consigue mediante el comando name.pol, al que habrá que asignar, siempre entre comillas, el nombre o nombres que deseamos utilizar para los parámetros, como por ejemplo: ***name.pol=c("SO2—equipo nuevo", "O3 Quimiluminiscencia")***.

17-. Cambiar la configuración de fechas del Eje X: ***date***

El eje de las X se configura automáticamente en función del formato y de los tramos que más fácilmente se adaptan al periodo de representación seleccionado por el usuario, pero la función *timePlot* permite personalizar también estos aspectos de la gráfica.

Mediante el comando *date.breaks*, se puede incrementar o disminuir el número de particiones realizadas ene l eje de las *X* por la instrucción gráfica *timePlot*, indicando al comando el número de particiones deseadas: ***date.breaks=5***.

Mediante el comando *date.format*, se puede cambiar el formato de la fecha representada en el eje *X*, indicando a la función el formato deseado, según los formatos previstos por *Openair* (ver los formatos que establece *strptime* y que se tratan al principio de este manual), como por ejemplo: ***date.format="%m—%Y"***.

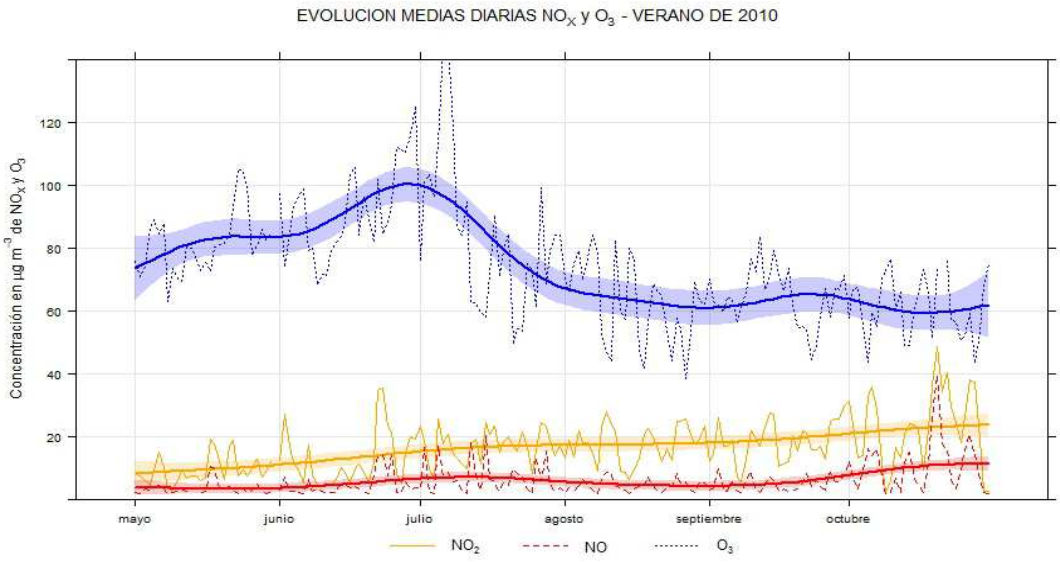
La función *timePlot* es una gráfica basada en la instrucción *plot* de *R*, con representación de datos en dos ejes, por lo que además de poder cambiar la representación de los puntos, tal y como se ha visto con el comando *plot.type*, también podemos añadir los títulos a los ejes o establecer sus límites de representación, tal y como se observará en el ejemplo siguiente.

Llegados a este punto, la función *timePlot* y el ejemplo que normalmente se incluye en estos casos, son una muy buena oportunidad para recordad que *R* y *Openair* son lenguajes de programación y que se utilizan para realizar cálculos estadísticos y funciones de grandes series de datos. Decir esto en este punto del manual puede parecer obvio, pero es importante si queremos entender que su potencial aumenta exponencialmente si aprendemos a combinar las funciones e instrucciones entre si de una manera provechosa.

Por ejemplo, consideremos que el conjunto de datos *datoscont* es enorme, y lo que realmente nos interesa es saber qué pasó con el ozono y sus precursores en el periodo de verano de 2010, comprobando así su evolución en esas fechas. La gráfica a usar es *timePlot* pero, ¿con qué instrucción podemos combinarla?, ¿Cómo?.

La instrucción a utilizar sería *selectByDate*, pero no sería necesario establecer distintas líneas de comando, como seguramente se pueda pensar en un primer momento, generando nuevos conjuntos de datos y concatenando instrucciones, sino que bastaría con introducir *selectByDate* en el preciso lugar donde iría el nombre del conjunto de datos a utilizar con *timePlot*, tal y como se observa en el siguiente ejemplo, y como ya hemos visto en otras ocasiones:

Ej. **timePlot (selectByDate(datoscont, year=2010, month=c(5,6,7,8,9,10)), pollutant=c("no2", "no", "o3"), group=TRUE, avg.time="day", cols=c ("orange", "red", "blue"), smooth=TRUE, date.breaks=6, date.format="%B", ylab="Concentración en ug/m3 de NOx y O3", main="EVOLUCIOÓN MEDIAS DIARIAS NOX Y O3—VERANO DE 2010", ylim=c(0,140)).**



» La Gráfica de Calendario: *calendarPlot*

En ocasiones la utilización de gráficas lineales y polares puede que no resulten asequibles al público en general, o que se precise de un formato más visual para comprender la evolución de los parámetros.

Una herramienta muy útil en este sentido es la función calendarPlot, que lo que hace básicamente es representar la evolución en el tiempo de un parámetro, tal y como hacía timePlot, pero en vez de usar como base una gráfica lineal de puntos x-y, utiliza un calendario para representar el parámetro en escala de colores.

Dicho de otra forma, lo que hace calendarPlot es calcular las medias diarias de un año determinado y para un parámetro, establecer una escala de colores para dichas medias, y pintar con el color resultante cada uno de los días de un calendario de ese año.

Esto permite, en un vistazo rápido, comprobar la evolución de las concentraciones, su estacionalidad y su distribución según los días de la semana.

La gráfica esta, a fecha de elaboración de este manual, en fase de desarrollo y experimentación, no disponiendo de la posibilidad de desagregar datos (type) o de establecer el cálculo estadístico a realizar (statistic), que será únicamente la media diaria, pero dispone de otras muchas opciones de configuración interesantes:

Variaciones sobre la función:

1-. Introducir el contaminante a graficar: ***pollutant***

La gráfica permite indicar el parámetro a representar mediante el comando pollutant, como en ocasiones anteriores. Sin embargo, en esta ocasión, sólo estará permitido representar un contaminante, y no varios a la vez.

2-. Año en el que se quiere la gráfica: ***year***

La función gráfica requiere siempre que se introduzca un año, sobre el cual se dispondrá el calendario y se graficarán los datos.

3-. Variable a representar en las celdas del calendario: ***annotate***

Por defecto, el parámetro que se utiliza en las celdas del calendario es la fecha del conjunto de datos “date”, incluyendo en las celdas el número del día correspondiente, como en cualquier calendario clásico.

Sin embargo, calendarPlot permite también utilizar para las celdas la dirección del viento “wd”, en cuyo caso lo que representa es la media vectorial de la dirección cada día (mediante una flecha), o la velocidad del viento “ws”, en cuyo caso se representa la misma media vectorial de la dirección, pero escalada en función de la velocidad registrada.

La última versión evaluada de Openair en este manual permite además anotar en las celdas del calendario el valor promedio diario alcanzado por el parámetro designado por el usuario para representar, para lo cual habrá que indicar al comando que ***annotate=“value”***. Al seleccionar esta opción el usuario podrá además configurar las cifras representadas por la gráfica, tal y como se verá con posterioridad.

4-. Establecer los colores de la gráfica: ***cols***

El comando cols permite cambiar los colores de las líneas que representan los datos, ya sea asignando los paquetes de colores predeterminados de que dispone Openair, como “brewer1”, “heat” o “increment”, o disponiendo por parte del propio usuario de los colores que se consideren más oportunos para la representación.

5-. Indicar a la gráfica los límites de representación: ***limits***

Por defecto la gráfica busca el intervalo de representación del parámetro por colores que mejor se adapte a los datos de que disponemos, pero puede ser preciso que dichos límites los establezca el propio usuario, por ejemplo, si lo que se desea es mostrar la evolución respecto a determinados valores límite o valores objetivo de carácter legal.

Para estas ocasiones la función calendarPlot dispone del comando limits, al que se le deberán indicar los valores mínimo y máximo a tener en cuenta en la gráfica: ***limits=c(0,180)***.

6-. Diferenciar en la gráfica la superación de los valores límite diarios que se indiquen: ***lim***

La última versión evaluada de *Openair* incluye un comando de gran utilidad para representar de forma distinta aquellos días que superen el valor límite diario que se le indique mediante el comando *lim*, siempre y cuando el comando *annotate* tenga como valor asignado *"value"*.

Al establecer un límite con el comando *lim*, como por ejemplo ***lim=50*** para el caso de parámetros como PM10, nos será posible diferenciar los valores representados que estén por encima o por debajo de dicho límite. De hecho, la propia función calendarPlot por defecto aplica una configuración determinada a la gráfica que permite destacar los días que superan dicho valor límite.

No obstante, la función permite personalizar la configuración de las cifras cuando estén por encima o por debajo del límite marcado por el usuario mediante los siguientes comandos adicionales, a los que se les deberá indicar un vector compuesto por dos variables, teniendo en cuenta que la primera se referirá al valor de los promedios por debajo del límite, y la segunda al valor de los promedios por encima del mencionado límite:

* ***col.lim***, que establecerá el color de las cifras numéricas representadas en cada cuadro, en función de que se encuentren por debajo o por encima del valor límite.
* ***font.lim***, que establecerá el tipo de fuente aplicable a las variables representadas en el cuadro, siendo *1* si la fuente es normal, o 2 si la fuente es en negrita. Por defecto la variable por encima del valor límite se presenta en negrita, es decir, con ***font.lim=c(1,2)***.
* ***cex.lim***, que establece la relación de tamaño aplicable a las variables, como factor al tamaño normal de la fuente que representa el valor en el gráfico, según se encuentren estas variables por debajo o por encima del valor límite.

7-. Establecer el número mínimo de datos necesarios: ***data.thresh***

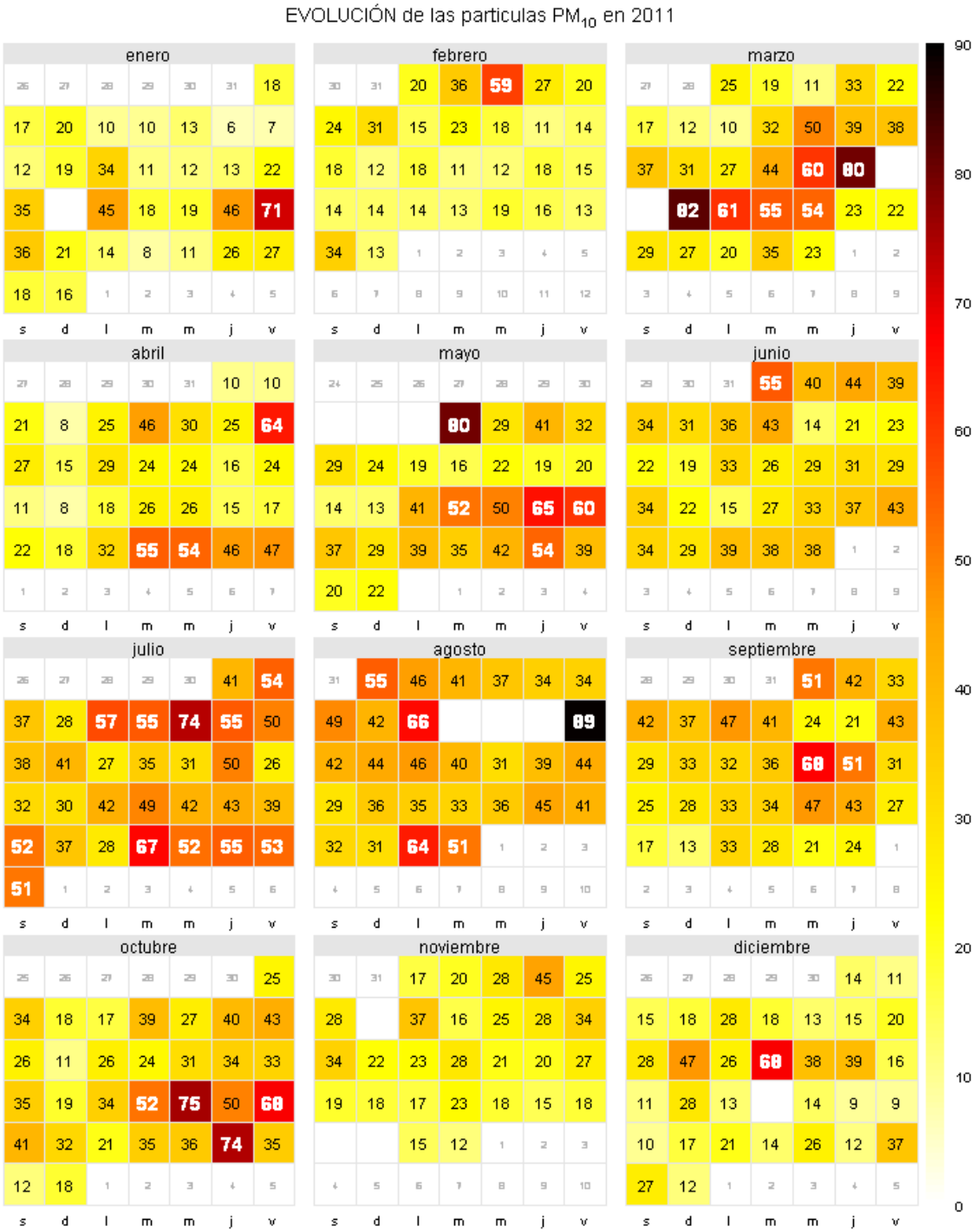
Puesto que la gráfica de calendario realmente incluye de forma implícita una agregación de datos en periodos de integración diarios, la última versión evaluada de *Openair* incluía ya la posibilidad de contemplar la representatividad del dato a graficar, tal y como ocurría con la instrucción *timeAverage*. Esto se consigue mediante el comando *data.thresh*, con el que es posible indicar a la función el porcentaje mínimo de datos necesarios para considerar un periodo como representativo en el cálculo del de la media diaria realizado.

En aquellos casos en los que el número mínimo de datos disponibles para el parámetro a representar en el día no supere el porcentaje indicado, la función *calendarPlot* eliminará el dato y no se representará en la gráfica, dejando vacía completamente la celda de la misma.

8-. Número de decimales a utilizar en la cifra que se representa en el calendario ***digits***

Al realizar el promedio diario del parámetro indicado para representarlo en el calendario mediante *annotate="value"*, es posible que las cifras obtenidas presenten un mayor número de decimales del que sería deseable para representar la cantidad en el calendario, dificultando su dimensionamiento o perjudicando su visibilidad. Mediante el comando *digits* se le puede indicar a la gráfica el número de decimales que queremos que acompañen a la gráfica para representar las cantidades medias diarias del parámetro.

La función gráfica *calendarPlot* aporta al técnico una herramienta muy práctica a la hora de elaborar gráficas que resulten más asequibles al público en general, y que sirvan para detectar más detalladamente la evolución de las medidas diarias de un contaminante, que quedan ahora recogidas de una forma más comprensible, tal y como se puede observar en la siguiente gráfica para la evolución del Ozono en nuestro conjunto de datos *datoscont* durante el año 2011.




```
> calendarPlot ( datoscont, pollutant="pm10", year=2011 , annotate="value", cols=c("white", "yellow", "orange", "red", "black"), limits=c(0,90), lim=50, col.lim= c("black", "white"), font.lim= c(1,2), digits=0, cex.lim= c(0.9,1.2), main="EVOLUCIÓN de las partículas PM10 en 2011" )
```

» La Gráfica de densidad de Kernel para las superaciones de las medias diarias: *kernelExceed*

Probablemente recordemos aún el concepto de densidad de kernel para el estudio de la distribución de los datos que se contemplaba en el apartado de histogramas de las gráficas en *R*. En este caso, *Openair* aprovecha la gran capacidad para el suavizado y la estimación de la probabilidad de aparición de la función de densidad de kernel para elaborar gráficas que permiten caracterizar determinados días en los que se supera el valor límite de un contaminante.

La función *kernelExceed* utiliza la densidad de Kernel para estudiar las condiciones que llevan a la superación de los valores límite diarios en inmisión de los contaminantes que se precisen. Así, con la gráfica de densidad de kernel lo que se hace es destacar las condiciones que se dan durante los días en los que se supera el valor límite para que sean más visibles, estableciendo la densidad que se da del parámetro contaminante por condición en estudio.

Actualmente la gráfica está preparada para el estudio de la superación de las medias diarias, por lo que es especialmente útil para el caso de la superación de los niveles medios diarios de partículas PM10, que son los que disponen de límite diario legalmente establecido, pero puede ser aplicada a otros parámetros contaminantes, siendo el usuario el que establece el valor límite diario que considere oportuno.

Variaciones sobre la función:

1-. Introducir el primer parámetro o condicionante a estudiar: ***x***

A la gráfica se le debe indicar un primer parámetro o condicionante a estudiar en relación a la superación del valor límite diario del parámetro contaminante que se requiera. Normalmente se asigna a este parámetro cualquier condicionante climatológico, y especialmente frecuente es que se le asigne la dirección del viento.

2-. Introducir el segundo parámetro o condicionante a estudiar: ***y***

A la gráfica se le debe indicar un segundo parámetro o condicionante a estudiar en relación a la superación del valor límite diario del parámetro contaminante que se requiera, y que sería complementario con el primero. Normalmente se asigna a este parámetro cualquier condicionante climatológico, y especialmente frecuente es que se le asigne la velocidad del viento.

3-. Introducir el contaminante del que se desean analizar las superaciones de los valores límite diarios: ***pollutant***

También resulta obligatorio indicarle a la función kernelExceed el parámetro contaminante del que se desea representar la densidad de datos para los condicionantes establecidos. Tal y como se ha comentado anteriormente, la gráfica está especialmente diseñada para estudiar las superaciones del valor límite diario de partículas PM10, pero puesto que es el usuario el que establece el umbral a superar, podrá incluirse cualquier otro contaminante.

4-. Desagregar los datos para estudiar las superaciones por tramos: ***type***

Mediante el comando type , visto en gráficas anteriores, se le puede indicar a la instrucción que desagregue los datos a representar en función de variables ya vistas como las horas del día “hour”, los días de la semana “weekday”, los meses del año “month”, los años “year”, o incluso cualquier otro parámetro del grupo de datos. El comando type terminará dividiendo la gráfica para representar una gráfica de densidad por cada división establecida por type.

5-. Concentración media diaria que se considera como valor límite: ***limit***

La gráfica necesitará que se establezca la concentración media diaria que se considera como valor límite diario, a partir de la cual se seleccionarán los días para realizar el estudio de densidad de kernel para las condiciones establecidas.

En este sentido, conviene destacar la utilización del comando lógico ***more.than***, que permite indicar a la función si tiene que seleccionar los días que están por encima del valor límite, opción establecida por defecto con ***more.than=TRUE***, o seleccionar aquellos días que estén por debajo, de forma que el análisis se realizará para aquellos días que no se supere, con ***more.than=FALSE***.

6-. Establecer cómo se deben seleccionar los datos: ***by***

Para aquellos días en los que se supera el valor límite diario establecido por el usuario, la selección de los datos puede realizarse de varias maneras, y quedará definida por el comando *by*. En primer lugar, si establecemos que ***by="day"***, se seleccionarán todos los datos del contaminante en aquellos días que se supere el valor límite establecido, superen o no ellos de forma individual dicho valor límite. Sin embargo, si indicamos que ***by="dayhour"***, entonces se seleccionarán tan sólo aquellos datos que hora tras hora superen también el valor límite indicado por el usuario, descartando aquellos que no lo superen.

7-. Establecer el número mínimo de datos necesarios: ***data.thresh***

Dado que la gráfica se basa en la realización de cálculos estadísticos diarios para comprobar si se supera o no el valor límite diario establecido, podría resultar preciso contemplar un cierto grado de fiabilidad en los datos utilizados para dicho cálculo, asegurando así siempre que se utilizan días lo suficientemente representativos. Mediante el comando data.thresh es posible indicar a la función el porcentaje mínimo de datos necesarios para considerar un periodo como representativo en el cálculo de la media diaria.

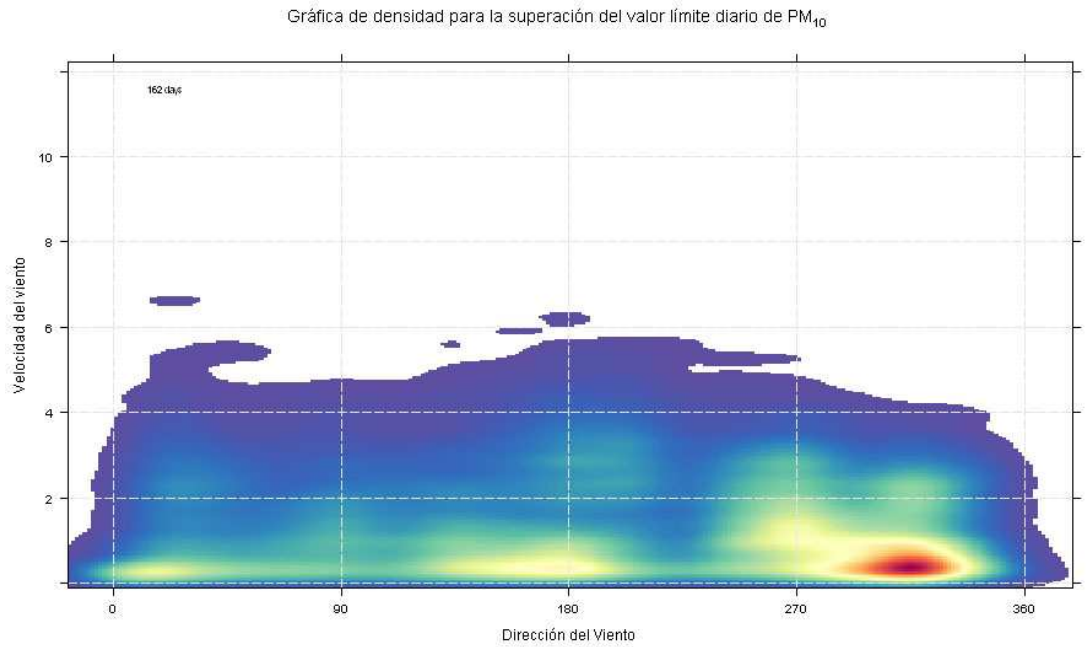
8-. Número de celdas usadas para la estimación de la densidad: ***nbin***

Se puede modificar el número de celdas utilizadas a lo largo de los ejes X e Y para representar la densidad, modificando así el ancho de banda de la función de densidad de Kernel utilizada, lo que influirá sustancialmente en la resolución que finalmente disponga la gráfica. Por defecto, la grafica adopta la definición que considere más adecuada para el número de datos disponible. Sin embargo, el usuario podrá asignar cualquier variable numérica al comando *nbin* para cambiar esta selección automática.

9-. Cambiar los colores de la representación: ***cols***

El comando cols permite cambiar los colores de las líneas que representan los datos, ya sea asignando los paquetes de colores predeterminados de que dispone Openair, como "defaul", "heat" o "increment", "hue" o "brewer1", o disponiendo por parte del propio usuario de los colores que se consideren más oportunos para la representación.

Haciendo una utilización clásica de la función *kernelExceed*, podemos obtener gráficas como la que se adjunta como ejemplo final, en la que se analizan las condiciones de dirección y velocidad del viento que predominan en aquellos días en los que se produce una superación del valor límite diario de partículas.



» La Función de Regresión Lineal: *TheilSen*

La función TheilSen grafica los datos de un parámetro dado en función del tiempo y calcula su regresión lineal o tendencia temporal, estimando su pendiente, lo que convierte a la función TheilSen en una función de enorme interés y utilidad para el análisis de la calidad del aire con Openair, por cuanto que nos permitirá, entre otros aspectos, tener una idea general de cómo cambian las concentraciones de un contaminante con el tiempo, más allá de comportamientos cíclicos y estacionales propios, o conocer cómo de significante es una tendencia observada, atendiendo a un modelo matemático preciso.

La función TheilSen, utilizada en las últimas versiones de Openair (desde la versíon 0.5-11), sustituye a la función MannKendall, frente a la que presenta alguna diferencia leve, aunque continúa calculando los parámetros propios de una regresión lineal, tales como la pendiente, la incertidumbre de la misma o la correlación.

Matemáticamente se puede decir que el método Theil-Sen, utilizado por esta función de Openair, es un método estadístico de estimación de líneas de regresión robustas muy eficiente, insensible a los valores fuera de rango *"outliers"*, y significativamente más exacto que los métodos de cálculo simples, permitiendo además desestacionalizar los datos para calcular la regresión, por lo que se convierte en el mejor método para el estudio de tendencias globales en datos ambientales.

El funcionamiento de TheilSen a grandes rasgos consiste en, dado un grupo de pares de datos, con n pares, calcular todas las pendientes para todos los pares de datos, y luego calcular la media de dichas pendientes para dar con la recta de regresión final.

Este método, si bien confiere a la función sus características de eficiencia y exactitud vistas con anterioridad, requiere de bastante potencia de computación para su cálculo, por lo que la función programada en Openair calcula la media mensual del parámetro a partir de valores con resoluciones iguales o superiores, y luego aplica el método estadístico descrito.

La función TheilSen dispondrá de las siguientes opciones de programación.

Variaciones sobre la función:

1-. Introducir el contaminante a graficar: ***pollutant***

La gráfica requiere que le sea indicado el parámetro sobre el que se desea realizar el cálculo estadístico.

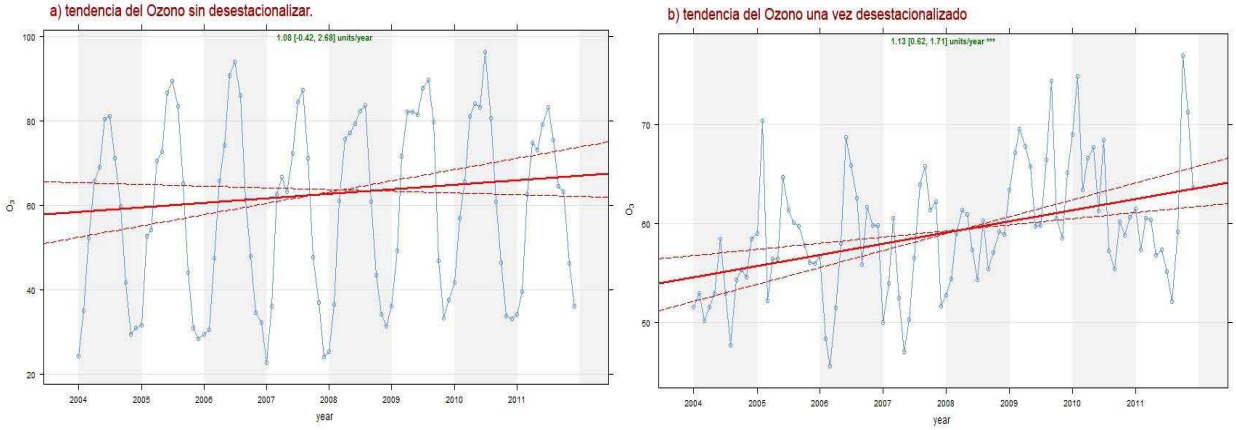
2-. Des-estacionalizar los datos del parámetro: ***deseason***

Algunos parámetros presentan un carácter claramente estacional, modificando su concentración media en función de la época del año en la que se encuentren, como por ejemplo el ozono.

Por otro lado, dado que el promedio utilizado por la función es mensual, este factor estacional podría afectar considerablemente al cálculo de la recta de regresión. En este sentido, la función ***timeVariation*** que veremos más adelante puede ayudarnos a calcular si existe un comportamiento cíclico estacional o no para un determinado parámetro que debería corregirse antes de calcular la recta de regresión.

Por defecto la des-estacionalización está desactivada, pero si la activamos, ***deseason=TRUE***, la función TheilSen aplica otra función de forma previa, la función STL (descomposición de tendencias estacionales, en inglés) que elimina dichas tendencias estacionales con anterioridad a la realización de los cálculos estadísticos. En este sentido, habrá que tener en cuenta que la función STL requiere interpolar linealmente los datos nulos para poder manejarlos, lo que podría afectar al resultado final de TheilSen.

Ej. **a) TheilSen (datoscont, pollutant="o3"),** calcula la pendiente para el ozono de que disponemos en nuestro conjunto de datos, incluyendo en dicho cálculo los datos brutos con todas sus fluctuaciones cíclicas. El resultado obtenido muestra una pendiente baja con un intervalo de confianza muy amplio que aporta demasiada incertidumbre a la función, por lo que no se puede obtener ningún tipo de conclusión fiable.. // **b) TheilSen (datoscont, pollutant="o3", deseason=TRUE),** se trata de la misma función gráfica y los mismos datos que la anterior, salvo que ahora desestacionaliza los datos antes de calcular la pendiente, consiguiendo que estos se muestren de una manera más coherente. La pendiente aparece ahora más marcada y, lo que es más importante, el intervalo de confianza es ahora mucho más estrecho y se mueve siempre en rangos positivos, lo que permite concluir, ahora sí, que existe una tendencia al incremento progresivo de los niveles de ozono en la zona.



3-. Desagregar los datos para representar varias gráficas: ***type***

Al igual que ocurre con otras muchas funciones gráficas de Openair, con TheilSen también tenemos la posibilidad de desgregar los datos en función de la fecha, por ejemplo por estaciones "season" o años "year", lo cual puede ser muy útil para analizar la correspondencia de unos periodos con otros respecto al cálculo de cada tendencia, o incluso en función de otro parámetro del conjunto de datos, lo cual también puede resultar muy útil si queremos analizar la existencia de interacciones adicionales a la propia estacionalidad del parámetro.

4-. Disponer el periodo de agregación de la función: ***avg.time***

Por defecto, tal y como ya se ha comentado con anterioridad, la función utiliza la media mensual para el establecimiento de los pares de datos, pero se le podría exigir que utilizase periodos de integración de datos mayores, siempre y cuando el conjunto de datos de que disponemos nos lo permitiese, tales como periodos estacionales "season" o anuales "year".

5-. Cambiar el estadístico a calcular para la función: ***statistic***

La función TheilSen utiliza por defecto la media mensual de los datos del parámetro que le hayamos indicado. Sin embargo, esta función nos permite cambiar esta media por el cálculo de los percentiles, lo que puede resultar de enorme utilidad, por ejemplo, a la hora de estudiar la evolución experimentada de un parámetro en sus máximas concentraciones, teniendo en cuenta que son estas las que finalmente provocan la superación del valor límite, objetivo o umbral legalmente establecido.

Si utilizamos como estadístico el percentil, ***statistic="percentile"***, será necesario indicar a la función el percentil deseado, disponiendo para ello del comando ***percentile***, al que le indicaremos de 0 a 100 el porcentaje del percentil que se precisa.

6-. Establecer el porcentaje de datos mínimo: ***data.thresh***

Al utilizar un cálculo estadístico para establecer los pares de datos que se usan en la función TheilSen, puede ser preciso indicarle a la función que tenga en cuenta un porcentaje mínimo de datos para considerar el cálculo realizado como representativo, lo que se consigue mediante el comando data.thresh.

Si alguno de los periodos indicados para el cálculo de los puntos no llegase al porcentaje mínimo de datos establecido por el usuario, el cálculo no se realizaría y el dato se registraría como NA.

7-. Establecer el valor de alfa de la función TheilSen: ***alpha***

La probabilidad de que la función acierte el dato real con la tendencia calculada es lo que en estadística se llama nivel de confianza, y se representa matemáticamente como 1-α, donde α(alfa) es lo que se llama error aleatorio o nivel de significancia, o lo que sería lo mismo, la posibilidad de fallar en la estimación.

El intervalo de confianza calculado y mostrado por la función gráfica que nos ocupa, tiene realación directa con este nivel de confianza ya que varían conjuntamente, calculándose el uno a partir del otro. Es decir, a mayor intervalo de confianza (mayor margen en el que se pueda encontrar el dato), mayor será el nivel de confianza (más probabilidades de acierto existirán), aunque la estimación realizada será menos precisa.

De forma práctica, a la hora de programar la función gráfica TheilSen, se puede decir que a mayor alfa (α) que dispongamos en la función, más pequeño será el nivel de confianza (1-α), así como el intervalo de confianza calculado y, por lo tanto, pese a ganar en precisión, perderemos en exactitud.

Por defecto la función TheilSen establece un ***alpha=0.05***, más que válido para la mayoría de las ocasiones en que precisemos de esta función, sin embargo, es posible modificarlo para incrementarlo o disminuirlo en función de lo que se precise en cada caso, poniendo cualquier valor que vaya del 0 al 1.

8-. Aplicar la autocorrelación al cálculo de la incertidumbre: ***autocor***

La autorcorreación se da comúnmente en series temporales y consiste en la existencia de una correlación serial de las perturbaciones o, lo que es lo mismo, surge cuando los términos de error del modelo calculado no son independientes entre sí.

Este efecto de autocorrelación puede darse bien porque exista un comportamiento estacional o cíclico del parámetro no contemplado en la función, bien porque existan otros factores que esten correlacionados a través del tiempo y que no se tienen en cuenta en la regresión calculada.

En definitiva, lo que hacemos al tener en cuenta la autocorrelación en el cálculo de nuestra pendiente es incluir en la incerdibumre asociada a la misma la interferencia estadística por la que se ven afectados los datos de nuestra serie temporal, incrementando así los márgenes del intervalo de confianza mostrado, pero asegurando por otro lado la fiabilidad de los resultados obtenidos.

Por defecto Openair descarta esta correción en los cálculos, pero puede ser tenida en cuenta si se activa el comando ***autocor=TRUE***. En todo caso, habrá que tener en cuenta que esta corrección puede afectar gravemente al intervalo de confianza mostrado.

9-. Dar formato a la pendiente resultante:

Al aplicar la función TheilSen no sólo se elabora una gráfica y se representa la recta de regresión en la misma, sino que se ofrecen al usuario los datos precisos sobre la pendiente obtenida y los intervalos de confianza aplicados, disponiéndolos también en la parte superior central de la gráfica. No obstante, existen diversos comandos que nos permitirán la personalización del formato de dicha pendiente y del intervalo para su presentación en la gráfica, según se expone a continuación:

- ***dec.place:*** Comando que establece el número de decimales con que se muestra la pendiente y su intervalo de confianza en la gráfica, por defecto 2.
- l***ab.frac:*** Comando que permite ubicar la pendiente dentro de la vertical del área de la gráfica, en tanto por uno. Por defecto el valor es 0.99, lo que sitúa el resultando en la parte superior de la gráfica.
- ***lab.cex:*** Comando que permite cambiar el tamaño de la fuente que se utiliza para disponer de la pendiente en la gráfica. Por defecto en 0.8.
- ***text.col:*** Comando que sirve para establecer el color de los números y letras que representan la pendiente en la gráfica.
- ***slope.percent:*** Por defecto la pendiente se muestra en la gráfica como un valor numérico que se expresa como unidades del parámetro por unidad de tiempo. sin embargo, activando el comando slope.percent=TRUE, se permite mostrar la pendiente y el intervalo en porcentajes.

10-. Dar color a los otros elementos de la gráfica: ***cols***

Al igual que en otras gráficas de *Openair*, es posible dar color a los diferentes elementos de la gráfica, más allá del formato de la pendiente que hemos visto en el punto anterior. No obstante, en la actualidad el único formato de colores predefinido activado para esta función es el de *"greyscale"*, lo que no concede mucho juego al usuario.

Si queremos dar color a los elementos de la gráfica y personalizarlos, podemos utilizar los siguientes comandos:

- ***line.col:*** que permite personalizar el color de la línea de tendencia graficada.
- ***data.col:*** que permite personalizar el color de los datos que se grafican como base a la línea de tendencia.

11-. Número de intervalos a usar en el Eje X: ***date.breaks***

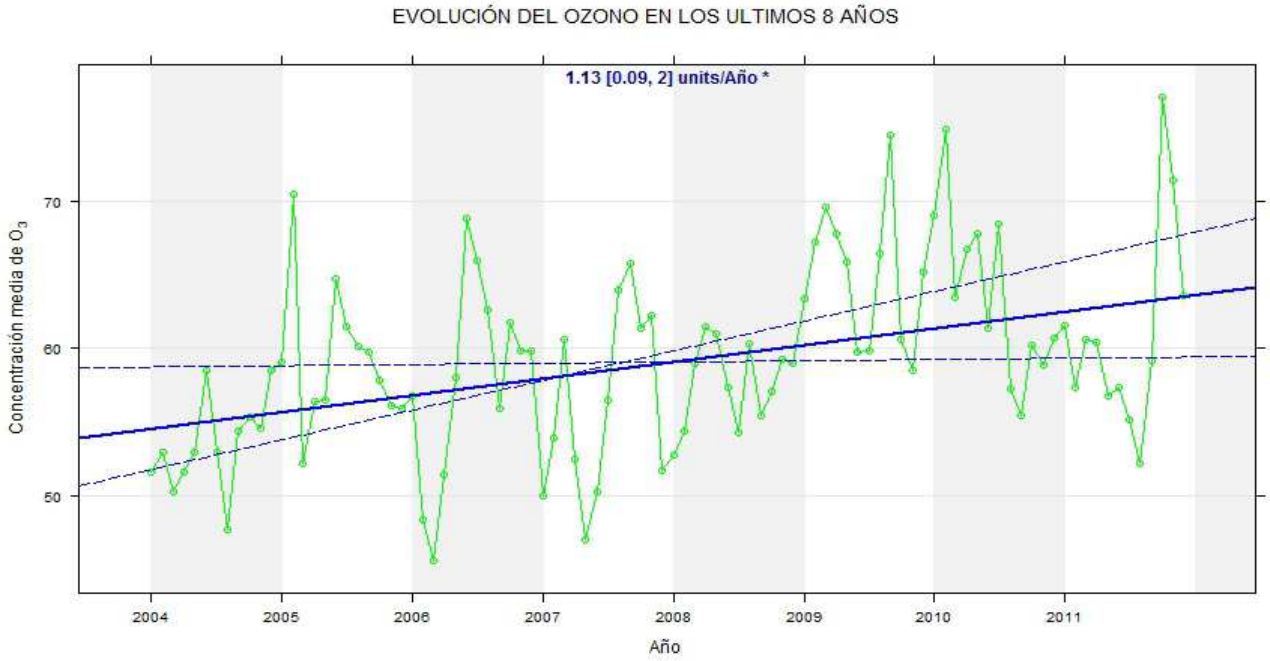
De forma automática la función *TheilSen* divide el Eje X en los intervalos que mejor se adaptan al rango de fechas de que disponga el conjunto de datos. No obstante, el usuario podría personalizar el número de intervalos al alza o a la baja mediante la asignación del número que corresponda al comando *date.breaks*.

12-. Modificar la relación entre ejes de distintas gráficas: ***x.relation*** , ***y.relation***

En algunas ocasiones podemos necesitar dividir una gráfica en varias para comprobar la tendencia presentada en distintas situaciones, lo que podemos lograr con el comando *type*. En estas ocasiones las gráficas generadas comparte la escala original tanto para el eje *x* como para el eje *y*, lo que puede perjudicar la representación.

Podemos evitar esto utilizando indistintamente los comandos x.relation o y.relation, a los que podemos indicarles que la escala utilizada por las gráficas sea siempre la misma, con la variable ***"same"*** (por ejemplo, x.relation="same") que es lo que la función hace por defecto, o que se utilicen escalas distintas adaptadas a cada una de las nuevas gráficas generadas con la variable ***"free"*** (por ejemplo: y.relation="free").

Cabe recordar que la función *TheilSen* continua siendo en esencia una derivación de la instrucción *plot* de *R*, con la representación de los datos en dos ejes. En este sentido, es posible utilizar ciertos comandos de *R* como *ylim*, *ylab* o *xlab*. En el caso del comando *xlab*, conviene tener en cuenta que la pendiente calculada refiere sus unidades a la denominación que hagamos en la gráfica del Eje *X*.



> TheilSen (datoscont, pollutant="o3", deseason=TRUE, autocor=TRUE, lab.cex=1, text.col="darkblue", data.col="green", line.col="blue", main="EVOLUCIÓN DEL OZONO EN LOS ÚLTIMOS 8 AÑOS", ylab="Concentración media de O3", xlab="Año").

» La Función de Tendencias Suavizadas: *smoothTrend*

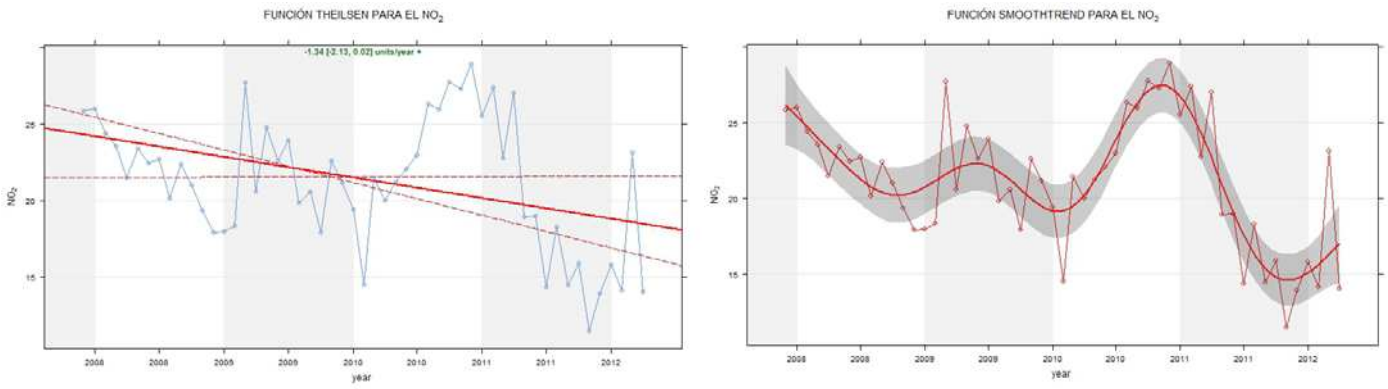
Similar a la función TheilSen de regresión lineal vista en el apartado anterior, la función de tendencias suavizadas smoothTrend va más allá, y sustituye los parámetros de regresión lineal utilizados para calcular la tendencia por funciones de suavizado que van adaptando la línea de tendencia a la evolución del parámetro.

Desde el punto de vista práctico, y en su configuración más básica, la función smoothTrend lo que hace es generar una gráfica de concentraciones medias mensuales de un parámetro seleccionado, dando lugar al diagrama de dispersión de pares de datos sobre el que construye una línea de tendencia suavizada, y finalmente muestra el intervalo de confianza al 95%.

Para hacer esto la función smoothTrend utiliza un modelo de regresión adicitivo denominado GAM (de sus siglas en inglés General Additive Model) proporcionado por otra librería de R especializada en el modelizado aditivo de datos que se llama ***mgcv***. El modelizado GAM sustituye los coeficientes o parámetros de cálculo de la regresión lineal habitual por funciones de suavizado (de funcionamiento similar a la media móvil) que van adaptando la pendiente a la respuesta de la variable.

Este modelo matemático es muy útil, como ya se puede deducir, cuando la relación entre dos variables va más allá del modelo establecido por una regresión lineal como TheilSen, y la pretensión es calcular una forma más compleja que se adapte mejor a los datos.

En la siguiente gráfica adjunta se puede comprobar claramente la diferencia práctica que existe en una función de regresión lineal como TheilSen, y una función de tendencia suavizada como smoothTrend, pues ambas gráficas han sido elaboradas con los mismos datos del parámetro seleccionado.



La función smoothTrend hereda muchas de las opciones de personalización de la fución TheilSen, aplicables a la misma, pero también incorpora otras de gran utilidad como las simulaciones bootstrap, que veremos más adelante. Los comandos de desarrollo de la función smoothTrend son:

Variaciones sobre la función:

1-. Introducir el contaminante a graficar: ***pollutant***

La función gráfica smoothTrend requiere que se asigne un parámetro de un conjunto de datos con fechas para realizar la estimación de la pendiente suavizada. En este caso, a diferencia de TheilSen, es posible además la selección de más de un parámetro, lo que da un valor añadido a esta función para comparar parámetros.

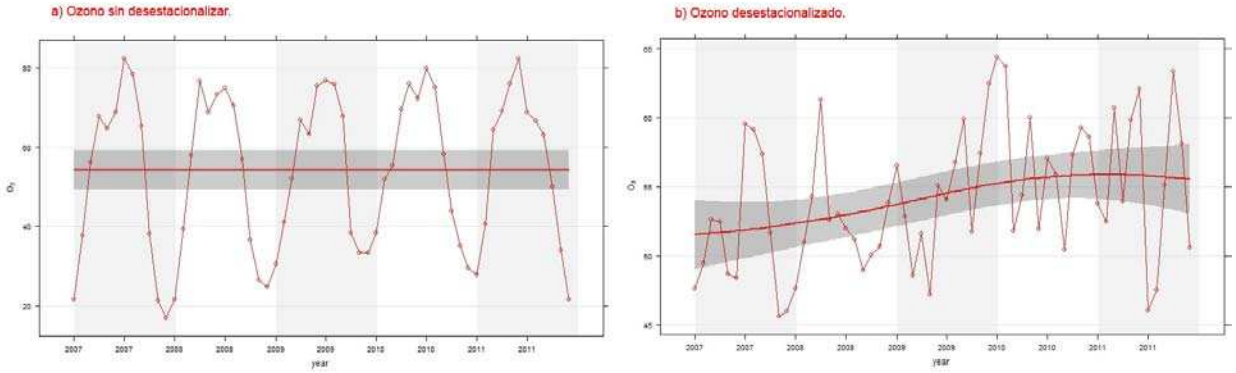
2-. Des-estacionalizar los datos del parámetro: ***deseason***

Al igual que ya pudimos comprobar con la función TheilSen, algunos parámetros contaminantes presentan un claro comportamiento estacional que podría afectar a la estimación de la pendiente.

Este efecto de desvirtuación de la tendencia calculada es aún más relevante en el caso de la función smoothTrend por cuanto que la estacionalización de un parámetro afecta a las medias mensuales (estadístico de base utilizado por el diagrama de dispersión), y la línea de tendencia de la función smoothTrend depende de dichas medias, lo que podría originar problemas como el expuesto en las siguientes gráficas de ejemplo para el ozono.

Por ello es importante que, antes de proceder a ejecutar la función smoothTrend tengamos claro si el parámetro o parámetros utilizados tienen algún tipo de comportamiento estacional, y en caso de que así sea, activemos el comando ***deseason=TRUE*** para permitir a la función gráfica aplicar la instrucción de desestacionalización de los parámetros.

Ej. **a) smoothTrend (datoscont, pollutant="o3"),** calcula la tendencia suavizada del ozono sin desestacionalziar. El problema es que este contaminante presenta un comportamiento estacional tan marcado y evidente que la función smoothTrend termina por generar siempre una línea recta, no teniendo posibilidad de adaptar la tendencia a las fluctuaciones. // **b) smoothTrend (datoscont, pollutant="o3", deseason=TRUE),** desestacionaliza antes el parámetro para calcular luego la tendencia. El efecto que finalmente se consigue es que se obtiene una línea de tendencia clara para los datos.



3-. Desagregar los datos para representar varias gráficas: ***type***

La función smoothTrend también tiene la posibilidad de desagregar los datos en función de la fecha , por ejemplo por estaciones "season" o años "year", o también en función de otros parámetros del conjunto de datos, lo cual puede resultar de enorme interés si se quiere analizar la existencia de interacciones adicionales.

4-. Disponer el periodo de agregación de la función: ***avg.time***

Por defecto, y al igual que ocurría con la fguncón TheilSen, la función smoothTrend utiliza la media mensual para el establecimiento de los partes de datos, pero se le podría exigir que utilizase periodos de integración mayores, como "season" o "year", siempre que nuestro conjunto de datos sea lo suficientemente extenso en su periodo temporal.

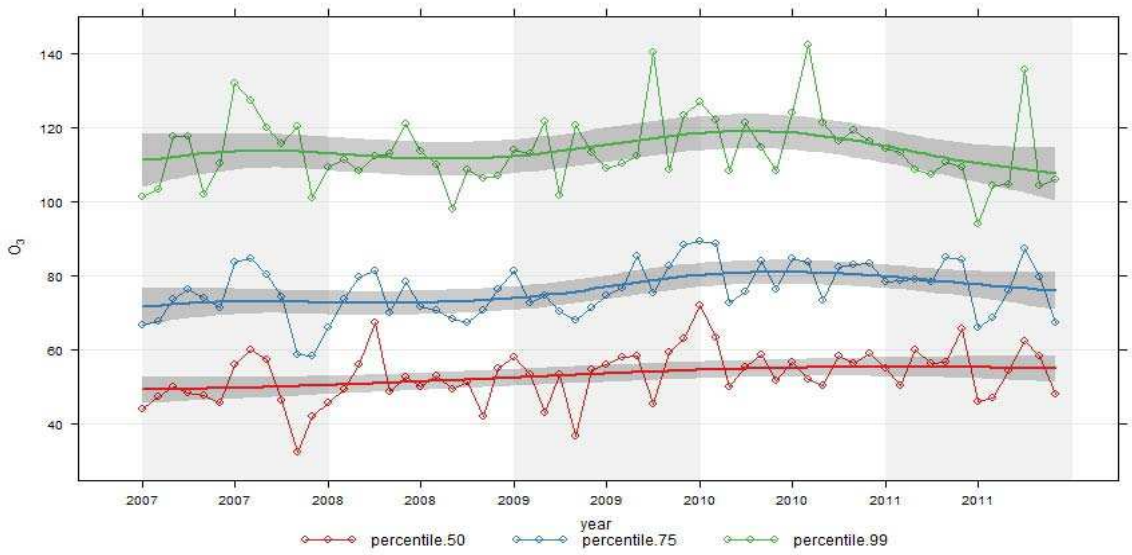
5-. Cambiar el estadístico a calcular para la función: ***statistic***

La estimación de la función smoothTrend se realiza, por defecto, sobre un diagrama de dispersión que responde a las medias mensuales. No obstante, el comando statistic nos permitirá cambiar este estadístico por el cálculo de percentiles, ***statistic="percentile"***.

Como en anteriores ocasiones, si finalmente optamos por utilizar los percentiles como base de cálculo para nuestra gráfica de tendencias suavizadas, deberemos indicar a la función el porcentaje que queremos que se calcule mediante el comando ***percentile***.

A este respecto, la diferencia fundamental de la función smoothTrend sobre la función TheilSen es que ahora sí que estaría permitido el cálculo de la tendencia suavizada para distintos percentiles, lo cual puede ser de enorme utilidad para observar la evolución del parámetro a distintas concentraciones. Lo que no resulta posible hacer con la función es el cálculo de distintos percentiles cuando previamente se han seleccionado varios contaminantes a graficar.

Ej. **smoothTrend (datoscont, pollutant="o3", deseason=TRUE, statistic="percentile", percentiles=c(50,75,99)),** calcula hasta tres percentiles distintos para nuestro conjunto de datos para el ozono, una vez desestacionalizado, y representa las distintas líneas de tendencia suavizadas de forma que se puede comprobar que para las medianas (percentil 50), existe una tendencia alcista en los datos, mientras que para los percentiles de datos más elevados (percentiles 75 y 99) y sobretodo para los valores máximos de este parámetro (percentil 99) existe una marcada tendencia a la baja desde el año 2010.



6-. Establecer el porcentaje de datos mínimo: ***data.thresh***

Como en todas las funciones de Openair en las que se pueden establecer distintos periodos de integración de datos, puede ser recomendable que para realizar el cálculo del estadístico indiquemos a la función el porcentaje mínimo de datos que consideramos representativo. Esto se consigue indicando al comando data.thresh, con un número del 1 al 100, el porcentaje de datos válidos que se requerirán para cada periodo.

7-. Realizar simulaciones bootstrap sobre la función: ***simulate***

La función smoothTrend permite realizar sobre la gráfica simulaciones bootstrap o simulaciones de remuestreo. Dichas simulaciones se realizan sobre los propios datos y están orientadas a obtener una estimación empírica de la distribución muestral, lo que permitirá valorar el sesgo y el error muestral, realizar una prueba de hipótesis respecto a nuestra función basada en la simulación y, lo que es más importante y práctico para nosotros, proporcionar un método alternativo para recalcular un intervalo de confianza adaptado a la tendencia calculada.

De hecho, es esto último lo que ocurre cuando activamos el comando ***simulate=TRUE***, ya que entonces las simulaciones bootstrap se asumen a la propia gráfica recalculando un nuevo intervalo de confianza (zona sombreada de la gráfica).

Los márgenes del intervalo de confianza irán variando en función del número de simulaciones que le digamos a la función que tiene que ejecutar sobre la tendencia, fijadas mediante el número que se le asigne al comando ***n***, como por ejemplo ***n=100***.

Dado que el comando simulate afecta a la estimación de la incertidumbre, recalculando los intervalos de confianza, es preciso tener en cuenta este hecho a la hora de aplicar otros comandos como el de autocorrelación , ya que se podrían ver muy afectados los márgenes de confianza finalmente calculados con su combinación.

8-. Aplicar la autocorrelación al cálculo de la incertidumbre: ***autocor***

Tal y como ya se explicaba para la función TheilSen anterior, las series temporales de datos como las que se utilizan en calidad del aire, son susceptibles de presentar una correlación serial de las perturbaciones, también denominada autocorrelación.

Sin embargo, es posible tener en cuenta esta autocorrelación en el cálculo de nuestra pendiente incluyendo la incertidumbre asociada a la función la interferencia que se daría por la autocorrelación, lo cual incrementaría los márgenes del intervalo de confianza, pero incrementaría también la fiabilidad de los resultados obtenidos.

El comando lógico, ***autocor=TRUE***, permite la activación de esta corrección en los datos del intervalo de confianza, aunque habrá que tener en cuenta que esta corrección puede afectar gravemente al intervalo mostrado, mas aún si este se ha calculado mediante herramientas alternativas como la simulación bootstrap.

9-. Mostrar el intervalo de confianza en la gráfica: ***ci***

Por defecto, la gráfica muestra el intervalo de confianza como una zona sombreada que acompaña a la línea de tendencia . No obstante, es posible que no nos interese que aparezca dicho intervalo en la gráfica, en cuyo caso bastaría con indicar a la función que ***ci=FALSE***.

10-. Personalizar los colores de la gráfica: ***cols***

Como en gráfica anteriores, la función smoothTrend permite personalizar los colores utilizados en las pendientes y en el diagrama de dispersión, ya sea utilizando una de las combinaciones predefinidas de Openair, o personalizando la gráfica con los colores que el usuario considere más adecuados.

11-. Cambiar el número de columnas de la leyenda: ***key.columns***

Si para ejecutar la función smoothTrend utilizamos como estadístico los percentiles, o seleccionamos más de un contaminante para graficar, la función forzará una leyenda en la parte interior de la gráfica donde se dispondrá la tipología del estadístico utilizado.

Si por ejemplo, indicamos a la función smoothTrend que realice el cálculo para varios estadísticos, veremos que la leyenda va ampliándose a los estadísticos programados, disponiéndolos en línea (tantas columnas como estadísticos se le indiquen). En este caso sería posible indicarle a la función el número de columnas en las que queremos que se dispongan los percentiles en la leyenda, asignando el número que corresponda al comando key.columns.

12-. Modificar la transparencia de los intervalos de confianza: ***alpha***

El intervalo de confianza, además de poder ocultarse o mostrarse a criterio del usuario mediante el comando lógico ci, que hemos visto anterioremente, puede ser modificado en cuanto a su transparencia mediante el comando alpha, según el cual se podrá conseguir que el intervalo sea totalmente transparente, ***alpha=0***, o totalmente opaco, ***alpha=1***.

13-. Personalizar las divisiones del Eje X: ***date.breaks***

La función smoothTrend ya establece por defecto el número de divisiones del eje X que mejor se adapta al intervalo temporal que se desea representar. En todo caso se podrá indicar a la función que utilice un número de divisiones distintas mediante el comando date.breaks.

14-. Modifica la relación del Eje Y entre distintas gráficas: ***y.relation***

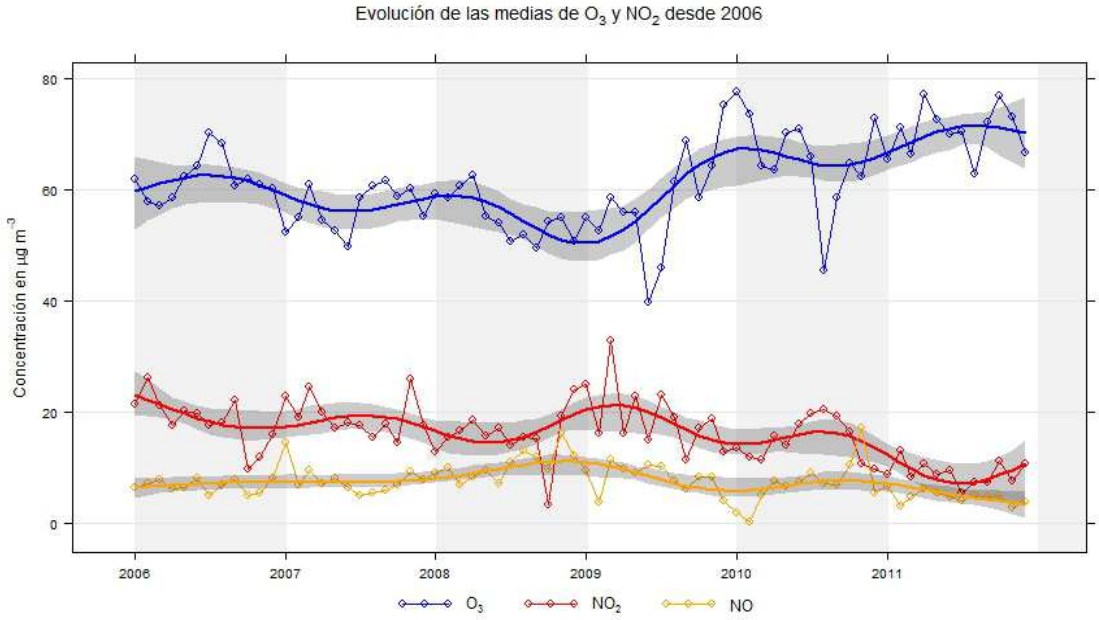
Tal y como ya pudimos comprobar en la función TheilSen, cuando representamos varias gráficas dividiendo la original, al utilizar comandos como type, podemos comprobar que las gráficas generadas comparten la misma escala para el eje Y, lo cual en ocasiones puede dificultar la interpretación de la gráfica.

Mediante el comando *y.relation* podemos hacer que las gráficas compartan la misma escala, ***y.relation="same"***, situaciones que se presenta por defecto, o que se presenten en escalas distintas adaptadas a cada una de las gráficas, con ***y.relation="free"***, lo que puede resultar muy útil cuando alguna de las gráficas presenta una distribución dispar.

Cabe recordar que la función smoothTrend, al igual que ocurría con la función TheilSen, continua siendo en esencia una derivación de la instrucción plot de R, con la representación de los datos en dos ejes. En este sentido, es posible utilizar ciertos comandos de R como ylim, ylab o xlab para personalizar aún más determinados detalles de la gráfica.

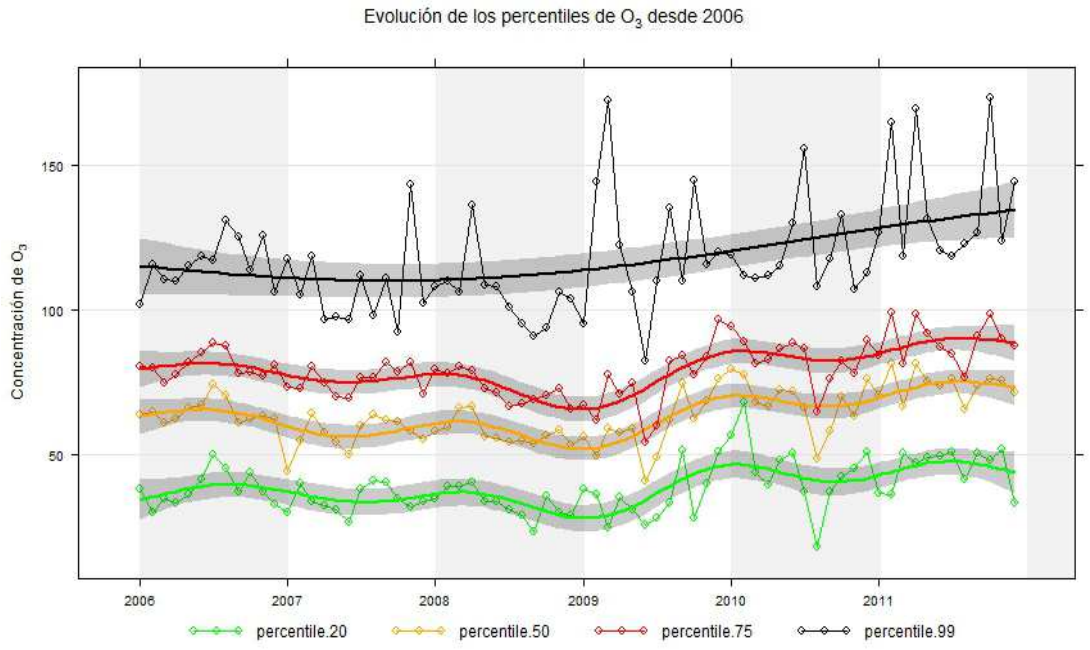
La función smoothTrend presenta una amplia variedad de aplicaciones para el análisis de la evolución temporal de la contaminación, siendo de hecho la base sobre la que se diseñan otras funciones gráficas de Openair como la que podremos ver en el siguiente apartado, dedicado a la función timeVariation.

La función gráfica smoothtrend puede servir, como se presenta en el primer ejemplo gráfico de final de apartado, para comprobar la evolución experimentada por dos parámetros como el Ozono y los Óxidos de Nitrógeno en nuestro conjunto de datos a lo largo de los años, una vez desestacionalizados los datos.




```
> smoothTrend (datoscont, pollutant=c("o3", "no2", "no"), deseason=TRUE, simulate=TRUE, n=100, cols=c("blue", "red", "orange"), main="Evolución de las medias de O3 y NO2 desde 2006", ylab="Concentración en ug/m3", xlab="")
```

O también puede servir para descomponer los datos de un mismo parámetro, como se hace en la segunda gráfica de ejemplo con el Ozono, comprobando así como evolucionan los distintos percentiles del mismo contaminante, lo que nos permitirá, por ejemplo, hacernos una idea de cuales son las concentraciones del contaminante (definidas por los percentiles) que están subiendo o bajando, y en qué medida podría afectar la subida generalizada de un parámetro a la aparición de un mayor número de superaciones de los valores límite legales.




```
> smoothTrend (datoscont, pollutant="o3", deseason=TRUE, statistic="percentile", percentiles=c(20,50,75,99), cols=c("green", "orange", "red", "black"), autocor=TRUE, main="Evolución de los percentiles de O3 desde 2006", ylab="Concentración de O3", xlab="")
```

» La Función de Matrices de Correlación: *corPlot*

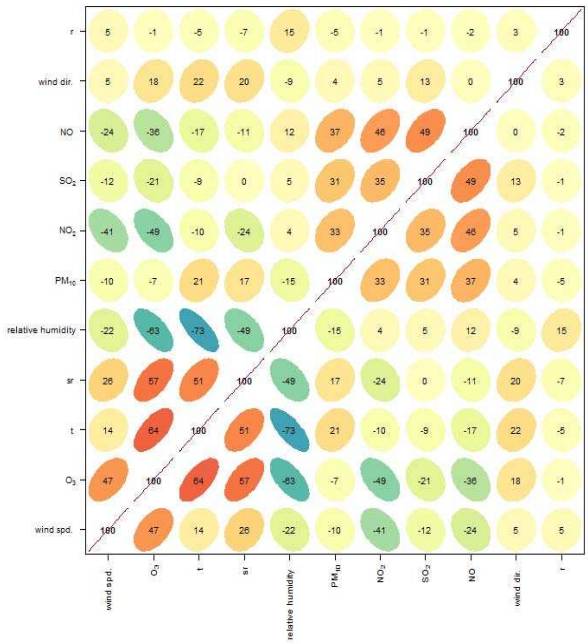
En calidad del aire los distintos contaminantes encontrados suelen presentar relaciones más o menos fuertes entre sí, y entre ellos y los parámetros meteorológicos de la zona, en muchas ocasiones registrados en paralelo por la misma estación de control. Para un técnico ambiental dedicado a la calidad del aire resulta muy importante conocer y entender en su máxima extensión estas relaciones. Sin embargo, puede resultar difícil teniendo en cuenta las sinergias, relaciones cruzadas, e interferencias que suelen existir, además de la complejidad de enfrentarnos al tratamiento de largas series de datos.

Una de las técnicas más usadas para ayudar al estudio de las relaciones entre parámetros es la de graficar en una matriz la correlación existente entre todos los pares de parámetros disponibles, contrastándolos de uno en uno entre el Eje X y el Eje Y de la matriz.

En elc aso de Openair esta técnica se puede llevar a cabo mediante la función corPlot , que representa una matriz cruzada con los parámetros de nuestro conjunto de datos que deseemos, estableciendo su correlación por pares, pero mejorando las opciones de representación normales de forma que la interpretación de la gráfica sea mucho más sencilla e intuitiva. Para ello utiliza tres elementos:

a) La Elipse, que se asemeja al dibujo de la recta de regresión, ya que será más o menos achatada en función de más o menos fuerte que sea la correlación. De hecho, en la diagonal, donde se cruza el mismo dato (y por tanto la correlación es perfecta) la elipse es realmente una línea recta. La orientación de la elipse también marcará si la relación es directamente proporcional "/ " o indirectamente proporcional " \ ".

b) El color, que se utiliza para rellenar las elipses, y que se escala para representar también la fortaleza y el sentido de la correlación. Si bien el color de la representación se puede personalizar, por defecto, la función corPlot marca unos colores que tienden al rojo cada vez más intenso para las relaciones directas, mientras que utiliza el azul para aquellas que se acercan a la relación inversa.



b) El número, que se establece como el coeficiente de correlación de Pearson en escala centesimal, de forma que una relación directamente proporcional perfecta equivaldrá a 100, tal y como ocurre en la diagonal de la matriz, mientras que una relación inversamente proporcional perfecta equivaldría a –100.

La función puede ejecutarse directamente sólo con el nombre del conjunto de datos, en cuyo caso cruzará todos los parámetros existentes y calculará la matriz completa, tal y como se muestra en el anterior ejemplo. Sin embargo, dispone de los siguientes comandos para ayudar a su configuración más básica.

Variaciones sobre la función:

1-. Introducir los contaminantes que se desea contrastar: ***pollutant***

Por defecto, si no se le indica nada al respecto, la función corPlot representa en los Ejes X e Y todos los campos / parámetros de los que disponga nuestro conjunto de datos. Sin embargo, con el comando pollutant es posible limitar esta matriz a los parámetros que el usuario considere más relevantes.

2-. Desagregar la correlación dividiendo así la gráfica: ***type***

Como en muchas gráficas de Openair, el comando type permite que desagreguemos el conjunto de datos para representar diversas matrices de correlación. Dicha división puede llevarse a cabo para diversos periodos de tiempo como estaciones "season", años "year", o días de la semana "weekday", entre otros, o puede llevarse a cabo para otros campos de nuestro conjunto de datos.

3-. Ordenar los ejes en función de la correlación entre datos: ***cluster***

La función corPlot es capaz de organizar los parámetros dispuestos en los Ejes X e Y para aproximar en la matriz los pares de datos que presenten una correlación más fuerte, tanto positiva como negativa. Por defecto, esta opción está activada, por lo que será el usuario el que tendrá que indicar a la función si desea desactivarla con el comando lógico ***cluster=FALSE***.

4-. Cambiar los colores de representación de la matriz: ***cols***

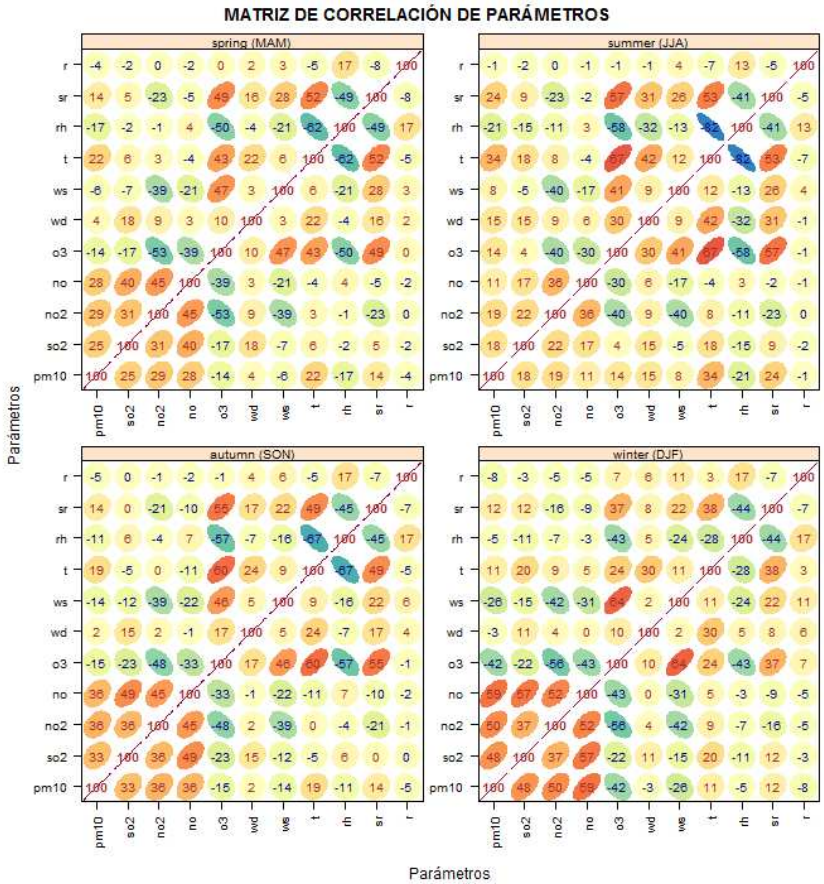
Los óvalos de la matriz vienen coloreados con el pack por defecto de Openair "default". Sin embargo, es posible indicar a la gráfica otros colores predeterminados como "hue", "increment", "heat", "brewer1", o definirlos de forma personalizada de forma que gradualmente se pase por todos los colores indicados con el comando, de menor a mayor.

5-. Destacar en negrita los números de la matriz: ***r.thresh***

Con el comando *r.thresh* es posible indicarle a la función a partir de qué coeficiente se considera por parte del usuario que conviene destacar las correlaciones remarcando en negrita su número. Esto permite que visualmente sean más evidentes aquellas correlaciones que superan el coeficiente indicado por el usuario. Destacar, sin embargo, que el comando tan sólo permite configurar la relaciones positivas o directamente proporcionales.

6-. Cambiar el color de los números en función de la correlación: ***text.col***

También es posible indicar a la función el color que deben adoptar los números en la representación de la matriz en función de que la correlación observada sea positiva o negativa. Con el comando text.col podemos indicar a la función los colores a utilizar, debiendo tener siempre en cuenta que el color negativo va primero y después el positivo, tal y como se puede observar en el ejemplo final.



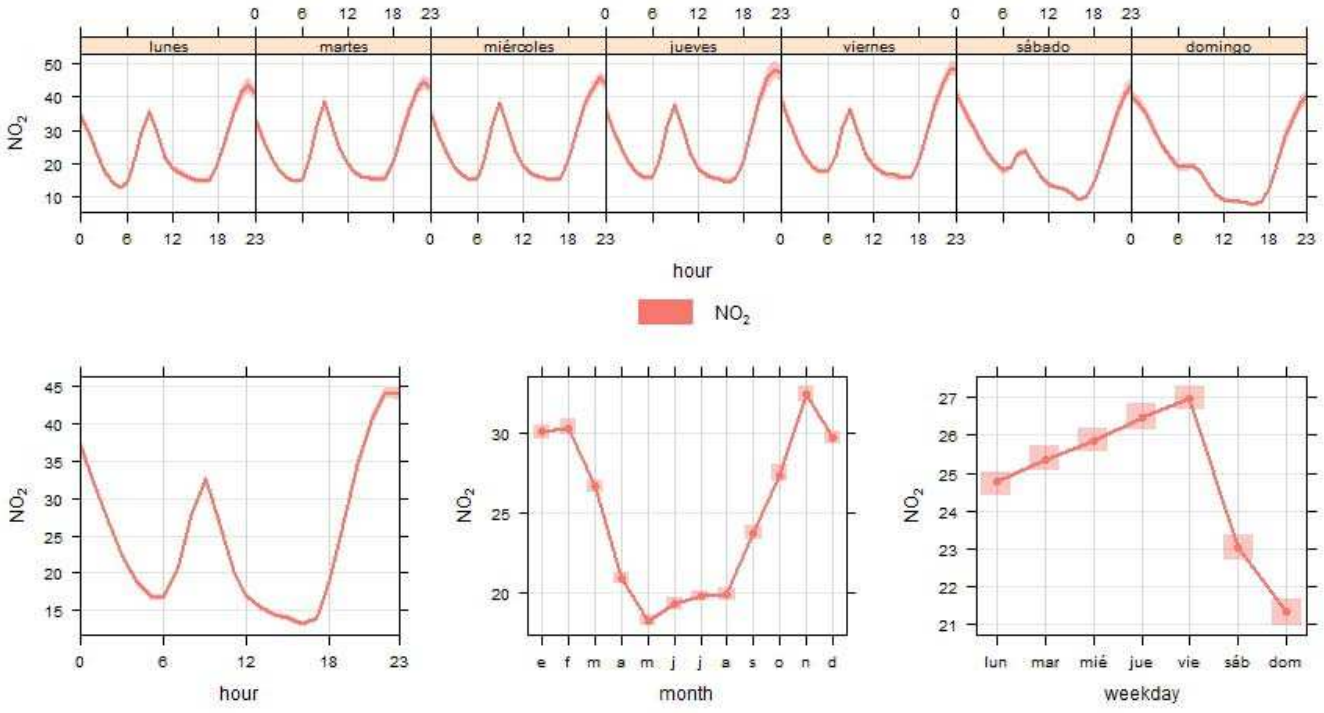

```
> corPlot (datoscont, type="season", layout=c(2,2), cluster=FALSE, main="MATRIZ DE
CORELACIÓN DE PARÁMETROS", xlab="Parámetros", ylab="Parámetros", au-
to.text=FALSE, text.col=c("darkblue", "brown"))
```

» La Función de Variación Temporal: *timeVariation*

Poder analizar la variación que presentan uno o más parámetros con respecto a distintos periodos temporales de carácter cíclico (días, semanas, años, etc), estableciendo así patrones de evolución en el comportamiento de dichos parámetros en la zona de estudio es, evidentemente, una herramienta de enorme utilidad para el análisis de la calidad del aire.

Esta herramienta la proporciona Openair a través de la función timeVariation, basada en la función smoothTrend, y que en su ejecución lo que presenta al usuario es un conjunto de gráfica, como el que se muestra a continuación como modelo básico explicativo, entre las que encontramos:

a) Una gráfica compuesta, con la evolución semanal del parámetro o parámetros seleccionados, dividida por días de la semana y, a su vez en horas del día. Esta gráfica nos permitirá ver los días de la semana en los que se producen las principales concentraciones contaminantes y la distribución en horas del día de las mismas, dibujando así un patrón semanal detallado de la evolución horaria de las concentraciones.
b) Una gráfica resumen donde se establece la evolución del parámetro en las horas del día, por lo que podremos ver en qué horas predominan las mayores concentraciones, y si el perfil obtenido se repite de la misma forma en la gráfica anterior para todos los días de la semana.
c) Otra gráfica resumen donde se establece la evolución del parámetro en función de los meses del año, con la que se podrá ver la evolución a lo largo del año y comprobar, por ejemplo, si existe algún tipo de estacionalidad en el dato.
d) Una última gráfica resumen de evolución de las concentraciones por días de la semana, que vendrá a resumir las concentraciones por cada día, pudiendo observarse cómo evolucionan a lo largo de la semana.



Descrita la función timeVariation, y vistos sus resultados en la anterior gráfica, es evidente que esta instrucción resulta ser una interesante y potente herramienta para, entre otras funciones:

a) Fijar escenarios de calidad del aire en las distintas zonas de estudio y para un amplio rango de parámetros. A través de dichos escenarios, se descubre también como una potente herramienta para el estudio comparativo de zonas o la evolución ene l tiempo de la calidad del aire en una misma zona.
b) Completar a otras funciones de Openair, tales como polarPlot o smoothTrend, siendo una herramienta adicional muy útil por cuanto que permite caracterizar la evolución de las concentraciones y, por lo tanto, focalizar con posterioridad la ejecución de otras funciones, o proceder a tratar los datos con anterioridad a su graficado (como por ejemplo desestacionalizándolos).
c) Proporcionar información de utilidad sobre las posibles fuentes de origen de la contaminación en una zona determinada y de su contribución a los niveles de calidad del aire de la zona en estudio.

La instrucción timeVariation descrita hasta el momento podrá ejecutarse bajo los siguientes comandos y opciones de configuración y desarrollo:

Variaciones sobre la función:

1-. Introducir el contaminante a graficar: ***pollutant***

Como en todas las funciones gráficas de Openair, lo primero que se le debe indicar a la instrucción timeVariation, una vez señalado el conjunto de datos a usar, es el contaminante o contaminantes que se desea representar en la gráfica. En este caso la instrucción timeVariation permitirá la introducción de uno o más parámetros para su representación.

2-. Cambiar los datos a horario local: ***local.time***

Teniendo en cuenta que uno de los periodos clásicos a representar con timeVariation es el de horas al día, distribuido incluso por días de la semana, puede ser muy recomendable hacer que los datos a utilizar sean en horario local, antes que en GMT.

Este aspecto ya se trataba en otras funciones de similar problemática, tales como polarAnnulus, y se comentaba que la utilización de datos en horario local para el estudio de distribuciones temporales es especialmente interesante si queremos poder comparar dicha evolución con ciertas actividades de orígen antropogénico, que podrían presentar cierta distribución según el horario local, que es el que rige el horario oficial (como por ejemplo el tráfico rodado en días laborales).

Sin embargo, también hemos de recordar que el comando lógico local.time está preparado por ingleses, por lo que de forzar la conversión, esta se produciría como si estuviésemos en la franja horaria inglesa (BTU) lo que no se correspondería seguramente con nuestro conjunto de datos. Será pues recomendable realizar el cambio a horario local utilizando otras herramientas de manera previa a la ejecución de esta función, o en su caso, contemplar el posible error de trabajar en horario GMT.

3-. Normalizar las variables de la gráfica: ***normalise***

Tal y como hemos podido ver en otras funciones, en ocasiones puede ser necesario representar sobre las gráficas diversos parámetros de calidad del aire que difieran considerablemente en cuanto al rango en el que se mueven sus datos, lo que dificulta su interpretación una vez graficados.

Una técnica que puede utilizarse para representar ambos parámetros sin perder la entidad de sus fluctuaciones es la normalización del dato o, lo que es lo mismo, dividir cada valor del parámetro por el promedio resultante para el mismo.

Esta técnica de normalización, cuya activación se lleva a cabo mediante el comando ***normalise=TRUE***, permite intercomparar los parámetros de interés y su evolución, aunque se pierde la representación con respecto a las unidades reales.

4-. Incluir los títulos de los Ejes X: ***xlab***

Al igual que se hace en las gráficas R y alguna de las gráficas de Openair, se puede cambiar el título por defecto dispuesto para el eje X y personalizarlo. En el caso concreto de la función timeVariation, también se puede hacer, aunque habrá que tener en cuenta que en esta ocasión se cuenta con cuatro gráficas y, por lo tanto, con cuatro ejes que habrá que personalizar, tal y como se puede ver en el siguiente ejemplo.

Ej. **timeVariation(datoscont, pollutant="o3", xlab=c("Distribución horaria por días de la semana", "Distribución Horaria", "Distribución mensual", "Distribución por días de la semana"),** dispone los títulos de las cuatro gráficas que forman timeVariation.

5-. Establecer los nombres de los contaminantes: ***name.pol***

Por defecto la función timeVariation proporciona, entre las gráficas elaboradas, una leyenda de los parámetros representados. Con el comando name.pol se puede personalizar dicha leyenda cambiando las denominaciones dadas por defecto a los parámetros por aquellos que se consideren más adecuados por el usuario.

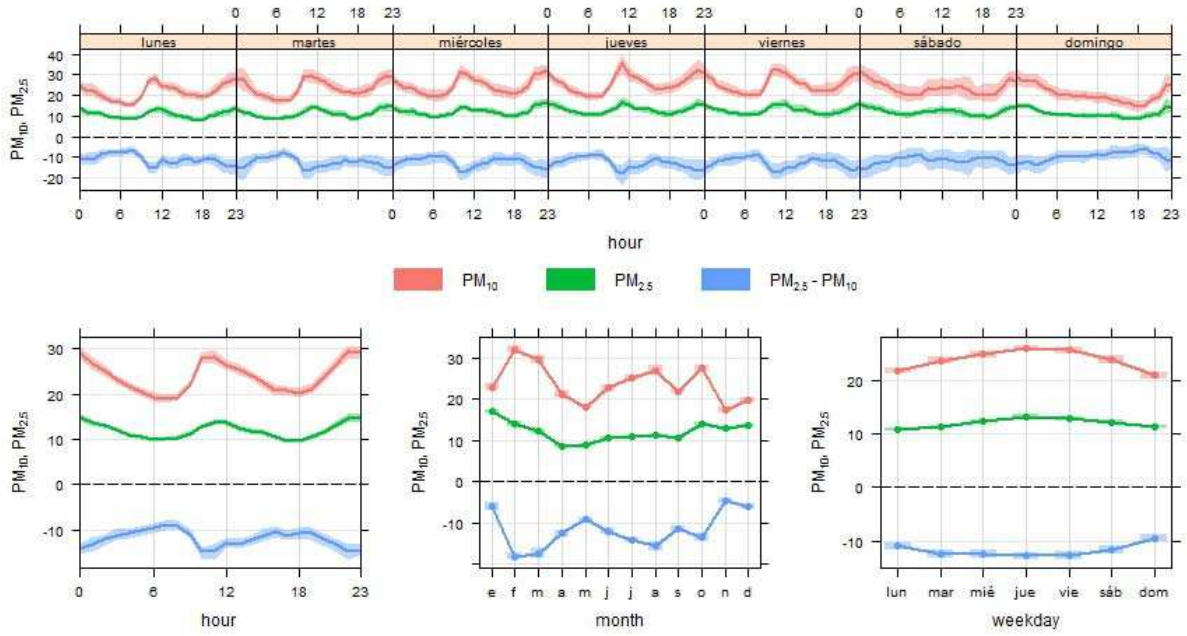
6-. Desagregar los datos de la gráfica: ***type***

Como en gráficas anteriores, es posible desagregar los datos para dividir la representación gráfica en distintos periodos temporales, tales como estaciones *"season"*, años *"year"*, etc., o en función de los intervalos de un determinado parámetro. Se debe tener en cuenta que en esta ocasión, ejecutar el comando type producirá la subdivisión de cada una de las cuatro gráficas, lo que podría complicar la representación e interpretación de los datos.

7-. Graficar la diferencia entre parámetros: ***difference***

En ocasiones puede ser útil aprovechar la gráfica timeVariation para graficar dos parámetros relacionados entre sí y representar la diferencia, como por ejemplo, cuando se quiere realizar una intercomparación de equipos midiendo en paralelo o, cuando se quiere comprobar la relación existente entre dos parámetros vinculados (PM_{10} y $PM_{2,5}$, por ejemplo). Activando el comando lógico ***difference=TRUE***, la función gráfica timeVariation calcula una tercera línea de tendencia que se corresponderá con la diferencia entre el segundo y el primer parámetro dispuestos en pollutant.

Ej. **timeVariation(datoscont, pollutant=c("pm10","pm25"), difference=TRUE),** activa, tal y como se puede observar en la gráfica siguiente, el cálculo de una tercera línea de tendencia destinada a establecer la evolución en la diferencia entre $PM_{2,5}$ y PM_{10}.



8-. Agrupar los datos de un parámetro dentro de un conjunto de datos con división: ***group***

Recordaremos que entre las instrucciones disponibles en Openair para el manejo de datos existían algunas como splitByDate que generaban un nuevo campo en el conjunto de datos y dividían el mismo en función de un criterio establecido, proporcionando dos o más escenarios de calidad del aire para el mismo conjunto de datos.

Pues bien, dado que la instrucción timeVariation es una buena herramienta para el análisis de escenarios, dispone de un comando que nos permite comparar distintos escenarios para una misma estación agrupándolos dentro de la misma gráfica. Esto se consigue mediante el comando group, al que se le deberá asignar el campo que se utiliza en el conjunto de datos para definir el escenario, por ejemplo *"split.by"* (denominación por defecto utilizada en splitByDate).

Ej. **timeVariation(datoscontsplit, pollutant="pm10", group="carretera"),** si recordamos el ejemplo que utilizábamos en la explicación del comando splitByDate (si no lo recuerda, este es el mejor momento para realizar un repaso), dividíamos un conjunto de datos dado en tres escenarios: "antes", "durante" y "después". Mediante el comando group="carretera", lo que hace la función timeVariation es representar en la misma gráfica los tres escenarios producidos para las partículas, generando tres líneas suavizadas para cada uno de ellos, pudiendo atsi comparar en cada una de las gráficas la evolución de las concentraciones de PM_{10}, antes, durante y después de la construcción de la circunvalación.

La opción *group* también se puede utilizar señalando a un campo del conjunto de datos que contenga una variable numérica cualquiera, y no sólo una variable de texto que divida el conjunto, tal y como hemos visto hasta ahora. En este último caso, el comando *group* buscará los escenarios a representar en las gráficas de forma conjunta, dividiendo los datos disponibles en cuantiles, de igual manera a como hace el comando *type* cuando apunta también a una variable numérica del conjunto de datos.

9-. Modificar el número de simulaciones bootstrap: ***B***

La función *timeVariation* se basa en el diseño de líneas suavizadas a partir del estudio estadístico de los datos en función de los distintos periodos temporales requeridos en la misma, por lo que su ejecución es similar a la de otras funciones como *smoothTrend*, presentando una línea de tendencia y un intervalo de confianza, calculándose este último en función de simulaciones *bootstrap* realizadas sobre la propia tendencia.

Es posible que, por razones de economía en los tiempos de ejecución, al querer representar varios parámetros a la vez, o por las propias necesidades de cálculo, precisemos personalizar el número de simulaciones *bootstrap* que terminarán dando valor al intervalo de confianza. Con el comando B, podremos indicar de forma numérica el número de simulaciones que se considera más conveniente aplicar.

10-. Mostrar el intervalo de confianza: ***ci***

Por defecto la gráfica *timeVariation* muestra el intervalo de confianza como un sombreado que acompaña a la línea de tendencia suavizada. No obstante, en ocasiones puede ser útil eliminar esta representación gráfica, dejando solamente la línea de tendencia, lo cual se puede llevar a cabo mediante el comando lógico ***ci=TRUE***.

11-. Personalizar los colores de la gráfica: ***cols***

Como en la mayoría de las gráficas de *Openair*, el comando cols dentro de la gráfica *timeVariation* permite la personalización de los colores de la representación en función de una serie de colores estandarizados de *Openair*, o de los colores que el propio usuario considere más convenientes.

12-. Añadir una leyenda a cada una de las gráficas: ***key***

Por defecto, y al considerar que la utilidad de la función *timeVariation* se basa en la representación de las cuatro gráficas elaboradas por la instrucción gráfica, la misma establece una sólo leyenda compartida para las cuatro que se ubica en el centro de la representación.

No obstante, y por si el usuario quisiese con posterioridad desagregar las gráficas para su uso individualizado, la función permite añadir la misma leyenda a cada una de las cuatro gráfica, para lo cual será preciso indicar que ***key=TRUE***.

13-. Cambiar el número de columnas de la leyenda: ***key.columns***

En aquellos casos en los que se representen varios parámetros, puede ser útil indicar a la función que los distribuya en la leyenda por medio de distintas columnas, que podremos personalizar con el comando *key.columns*. De no disponer nada en contrario la función establecerá todos los parámetros en una fila (tantas columnas como parámetros existan).

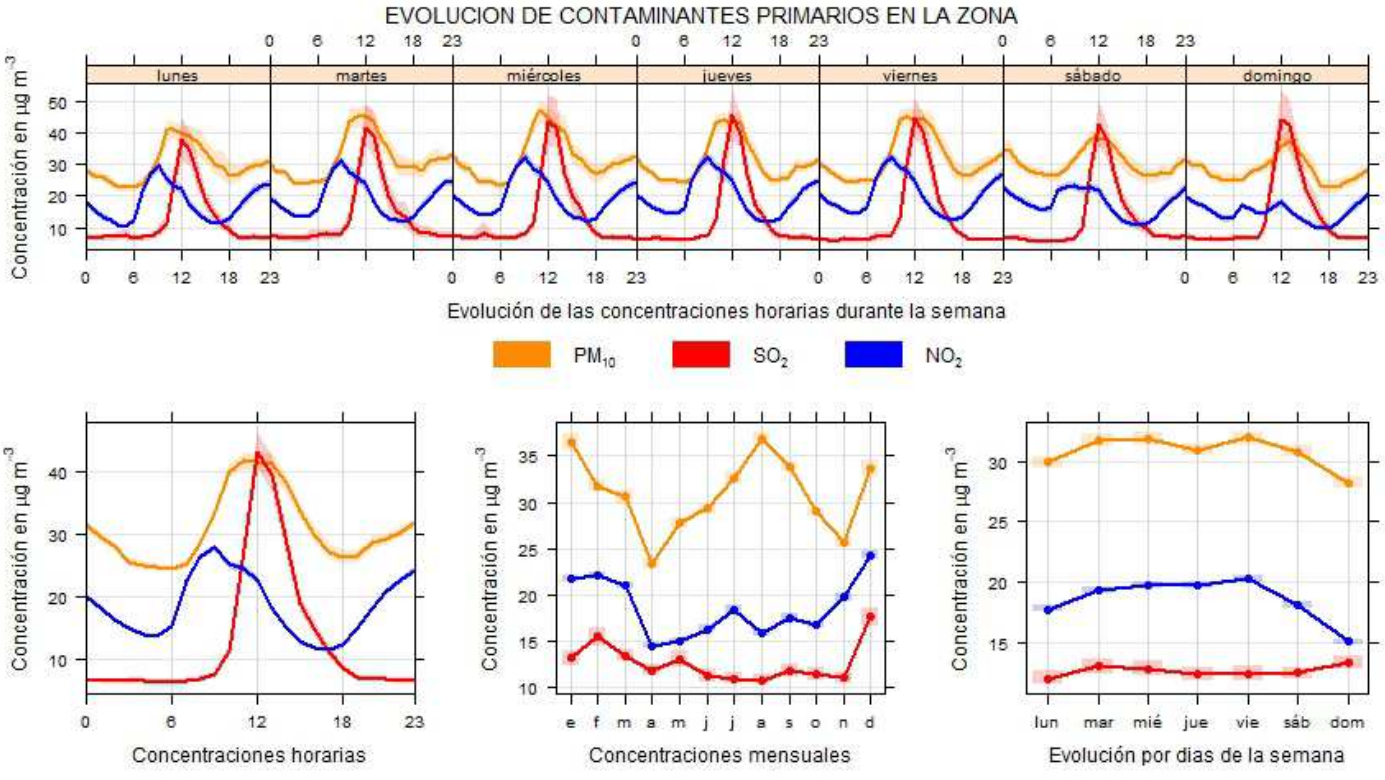
14-. Modificar la transparencia de los intervalos de confianza: ***alpha***

El intervalo de confianza, además de poder mostrarse u ocultarse a criterio del usuario mediante el comando ci, pueden modificarse en cuanto a su transparencia mediante el comando alpha, que se establece como ***alpha=1***, para un intervalo totalmente opaco que se confundirá con la línea de tendencia, o ***alpha=0***, totalmente transparente, que equivaldrá al comando ci.

Por defecto, el intervalo de confianza en la gráfica está establecido en un alpha=0.4, aunque en ocasiones puede ser recomendable incrementar su transparencia para, por ejemplo, permitir la visualización de otras líneas de tendencia graficadas en paralelo para varios parámetros.

Dado que la función *timeVariation* presenta su mayor ventaja y desarrollo en la descripción de escenarios de contaminación, como ejemplo final se mostrará el escenario resultante para los niveles en inmisión de los contaminantes primarios presentes en nuestro archivo de datos *datoscont*. Las conclusiones que se pueden adoptar en este caso son varias, según se expone a continuación:

a. Las concentraciones medias diarias de SO_2 se mantienen constantes a lo largo de toda la semana, y los picos diarios se producen a la misma hora aproximadamente durante todos los días, por lo que se puede deducir que existe detrás de estos niveles la influencia de algún tipo de foco de emisión industrial de funcionamiento constante.
b. Las emisiones de NO_2 parecen sin embargo vinculadas al tráfico rodado, presentando picos a primera hora de la mañana 8 a 10 y última de la tarde 20 a 23. Esta conclusión queda además apoyada por el hecho de que las concentraciones medias diarias descienden considerablemente los fines de semana, adquiriéndose incluso un perfil distinto de distribución los sábados y domingos.
c. Las emisiones de partículas van vinculadas siempre a la evolución de NO_2, por lo que se deduce una fuente común.
d. Por otro lado, la distinta evolución de las concentraciones medias mensuales de partículas con respecto a los NOx hacen prever la contribución de otra fuente de origen a los niveles medios globales. El hecho de que el pico sea en los meses de julio, agosto y septiembre, hace previsible la contribución de una fuente de origen natural.



> timeVariation (datoscont, pollutant=c("pm10", "so2", "no2"), cols=c("darkorange", "red", "blue"), alpha=0.2, main="EVOLUCION DE LOS CONTAMINANTES PRIMARIOS EN LA ZONA", ylab="Concentración en ug/m3", xlab=c("Evolución de las concentraciones horarias durante la semana","Concentraciones horarias", "Concentraciones mensuales", "Evolución por días de la semana"))

» El Diagrama de Dispersión: *scatterPlot*

Los diagramas de dispersión son ampliamente utilizados en estadística para comprobar cómo se relacionan dos variables entre sí, y en materia de calidad del aire son también una herramienta muy útil para la intercomparación de datos de dos parámetros relacionados, así como la correlación de equipos y técnicas de determinación de contaminantes, o la realización de ejercicios de intercomparación.

Sin embargo, con R y Openair los diagramas de dispersión adquieren nuevas posibilidades y una dimensión superior ya que entre las habituales funciones, también permiten:

a. Incluir una tercera variable "z" al diagrama de dispersión de pares de puntos mediante la inclusión de una escala de color aplicada a los pares.
b. Utilizar técnicas adicionales de modelizado de los datos, muy útiles cuando existe un exceso de pares de datos en la gráfica (*over-plotting*) que impide comprobar de forma realista la relación existente entre las dos variables, algo que suele ser muy habitual en largas series de datos.
c. Añadir al diagrama de dispersión tanto una línea de regresión como una línea de tendencia suavizada, muy útil cuando no se presente una relación lineal estricta entre dos variables, pero se observe una relación entre ellas en el sentido estricto.
d. Utilizar distintos periodos temporales de agregación de los datos, facilitando así el estudio de las mismas sin necesidad de llevar a cabo el pretratamiento de los datos de forma manual.
e. Sustituir la variable utilizada en el eje X por la fecha, lo que proporciona al usuario algo muy parecido a la función *timePlot* pero con mejoras añadidas.

La función *scatterPlot*, en combinación con otras funciones de *Openair* como *timePlot* o *timeVariation*, permite una buena caracterización de la calidad del aire en relación a parámetros previamente definidos. Para conseguir una adecuada configuración de esta función gráfica, *Openair* tiene previstos los siguientes comandos:

Variaciones sobre la función:

1-. Parámetro a incluir en el eje X: ***x***

Se debe indicar mediante este comando el parámetro a representar en el eje X, como parte del par de datos, pudiendo en este eje sustituir el parámetro que originalmente debería ir por la variable temporal, para obtener una función similar a *timePlot*.

2-. Parámetro a incluir en el eje Y: ***y***

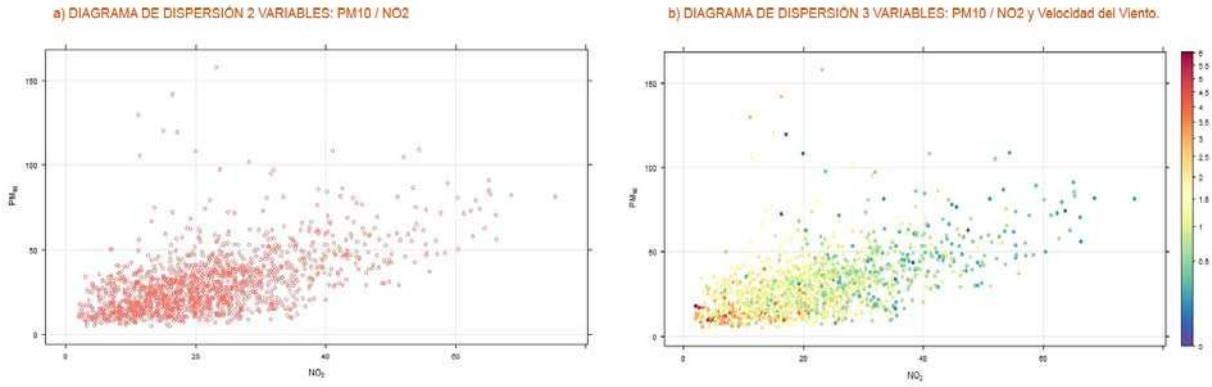
Las gráficas de dispersión son , por definición, representaciones de pares de datos de dos variables relacionadas entre sí, por lo que será necesario incluir mediante este comando la segunda variable a representar.

3-. Incluir una tercera variable en la representación: ***z***

Si bien las dos anteriores variables eran obligatorias, tal y como se ha comentado en la introducción, es e posible incluir en la representación una tercera variable, asignada por el comando z, que se integrará en la función mediante la asignación de colores a los pares de datos atendiendo a la escala que se establezca para la tercera variable.

Dicho de otra manera, cada uno de los pares de datos obtenidos por x e y quedará coloreado por la escala proporcionada por z. Esta novedad gráfica proporciona al usuario una herramienta de inestimable utilidad para el análisis de los partes de datos y/o diversas variables, tal y como se puede ver en el siguiente ejemplo gráfico.

Ej. **a) scatterPlot (datoscont, x="no2", y="pm10")** , muestra una gráfica de dispersión para los datos de NO2 y PM10, donde se puede ver que el grueso de los datos de estos contaminantes primarios presentan una cierta correspondencia (a excepción de ciertos datos de PM10 elevados que seguramente se correspondan con episodios naturales propios de la zona), lo que permite concluir que comparten las mismas fuentes de origen. **b) scatterPlot (datoscont, x="no2", y="pm10", z="ws")** , muestra la gráfica de dispersión de datos anterior, asociada a un tercer parámetro, que en este caso es la velocidad del viento, lo que proporciona una información valiosa al observar que los parámetros PM10 y NO2 adquieren sus valores más elevados a velocidades de viento bajas y viceversa, lo que confirma la procedencia común, y hace suponer que se trata de una fuente cercana a la estación de control.



4-. Método de representación de la dispersión: ***method***

La representación de pares de datos mediante puntos en una gráfica de dispersión tipo x-y puede generar, en función del número de puntos disponible, un exceso de puntos que impida la correcta interpretación de la gráfica, o lo que se conoce como *"over-plotting"*, razón por la que la función *scatterPlot* contempla diversos modos de representación de la dispersión.

Cuando representamos tres variables en el diagrama de dispersión, de similar manera a como veíamos en la gráfica anterior, podemos hacerlo mediante puntos, ***method="scatter"***, establecido por defecto en la función, o mediante una superficie escalonada dividida por niveles medios, ***method="levels"***, de forma que se conserve siempre la representación de esa tercera variable y su evolución.

Sin embargo, si las variables disponibles son sólo dos, la representación en superficie puede contemplar el conteo de datos, y por lo tanto podemos proceder a la representación por puntos *"scatter"*, establecida por defecto, o forzar al programa a que represente en superficie la distribución de los puntos, ya sea mediante el ***method="hexbin"***, que consiste básicamente en cortar los pares de datos que caen en cada celda hexagonal y colorear su contenido en función del número (para lo cual se utiliza el paquete *hexbin* de R), o mediante el comando ***method="density"***, que utiliza una herramienta de estimación de la densidad kernel mostrando una superficie continua donde se presenta la distribución en la densidad de los puntos en escala de colores.

5-. Agrupar la representación de datos por escenarios: ***group***

El comando *group* de la función *scatterPlot* permite la representación de varias series de datos (una por cada escenario representado) dentro de una misma gráfica, señalando el campo del conjunto de datos que dispone de los escenarios a representar en la gráfica.

De esta forma, tal y como ya se describía para la función *timeVariation*, si el comando apunta a un campo de la base de datos que consiste en una variable de texto (variable descriptiva procedente, por ejemplo, de la aplicación de una instrucción tipo *splitByDate*), la gráfica representará los pares de datos que seleccione el usuario en función de los distintos escenarios dispuestos en dicho campo de texto.

Por el contrario, si el comando *group* apunta a un campo del conjunto de datos con variables numéricas, generará sus propios escenarios dividiendo los datos en cuantiles que luego representará en la gráfica como si fuesen distintos escenarios de contaminación. Cabe destacar que, en este último caso, al seleccionar una variable numérica del propio conjunto de datos para el agrupamiento, no podremos utilizar otros comando importantes para la gráfica como el que nos permite cambiar el periodo de agregación de los datos: *avg.time*.

6-. Disponer de otro periodo de agregación de los datos: ***avg.time***

La gráfica scatterPlot dispone los pares de datos de la dispersión tal y como se presentan en nuestro conjunto de datos, por lo que tal y como se comentaba en el punto cuarto, es muy probable que exista over-plotting (o exceso de puntos graficados) ante la representación de históricos de datos muy prolongados en el tiempo.

Por ello, además de disponer de las herramientas descritas en el comando method, es muy posible que sea recomendable representar los datos en periodos de agregación temporal mayores a los originalmente dispuestos, razón por la que se dispone del comando avg.time.

Mediante este comando, que en *scatterPlot* podemos usar en toda su extensión, podemos promediar los datos en periodos que van desde el segundo *"sec"*, hasta el minuto *"min"*, la hora *"hour"*, el día *"day"*, la semana *"week"*, el mes *"month"*, el cuatrimestre *"quarter"*, o incluso el año *"year"*, pudiendo además agregar un número a la definición del periodo de agregación, lo que le aporta aún más flexibilidad al comando.

7-. Establecer el porcentaje de datos mínimo: ***data.thresh***

Ya hemos visto en funciones gráficas anteriores que, al cambiar el periodo de integración del dato, es muy posible que precisemos indicar al cálculo el número mínimo de datos válidos que se requerirán para considerar el dato obtenido estadísticamente como válido, lo cual lograremos indicando al comando *data.thresh* el porcentaje mínimo previsto.

8-. Estadístico a utilizar para agregar los datos: ***statistic***

Por defecto, la instrucción *scatterPlot* utiliza como estadístico para la agregación de los datos la media *"mean"*, pero podría ser igualmente interesante indicarle que utilice otros cálculos estadísticos como el máximo *"max"*, el mínimo *"min"*, la mediana *"median"*, la frecuencia de aparición (conteo de datos) *"frequency"*, la desviación estándar *"sd"*, o el percentil *"percentile"*.

Si utilizamos este último comando estadísticos, *scatterPlot* calculará por defecto el percentil 95, por lo que deberemos usar el comando ***percentile*** si queremos indicarle a la función un porcentaje distinto al dispuesto por defecto.

9-. Desagregar los datos para hacer varias gráficas: ***type***

Tal y como hemos visto para gráficas anteriores, con el comando type podemos dividir las gráficas en función de diversas categorías como días de la semana *"weekday"*, estaciones *"season"*, años *"year"*, etc. de similar forma a como se hacía con la instrucción *cutData*.

También se puede forzar al comando a que proceda a la división de los datos en función de otro parámetro del conjunto de datos, en cuyo caso dividirá la gráfica en cuatro cuartiles, o incluso puede forzar la división en función de parámetros múltiples.

10-. Incluir en la gráfica distintos modelizados de la tendencia en los datos: ***smooth***, ***spline***, ***linear***

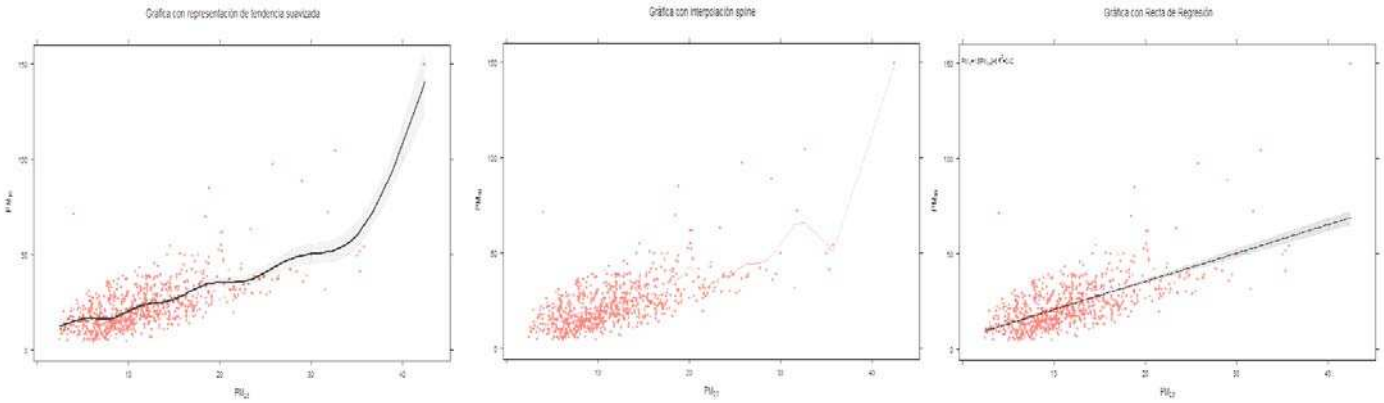
Por defecto la gráfica *scatterPlot* sólo muestra los pares de datos dispuestos por el usuario, pero en ocasiones es preciso conocer de forma matemática cuál es la tendencia básica que siguen dichos datos, bien para comprobar su comportamiento básico, o para prever cuales pueden ser las concentraciones a futuro, o incluso para establecer la existencia o no de correspondencia matemática entre ellos. Para ello *scatterPlot* dispone de tres opciones básicas, según se expone a continuación:

a) *Línea de tendencia suavizada*. Al igual que se utilizaba en gráficas como smoothTrend, pero aplicada a una distribución de pares de datos. Esta línea de tendencia se activa mediante el comando lógico ***smooth=TRUE***, con el que se activará la representación de la línea de tendencia junto a un margen de confianza del 95%, tal y como se puede ver a continuación.

b) *Interpolación de datos con función "spline"*. En aquellas ocasiones en las que el número de pares de datos es muy bajo, o se corresponden a una secuencia dada para la que se requiere de una línea ajustada a los pares, es posible indicarle a la función *scatterPlot* que realice una interpolación por *"splines"*. De esta forma, al activar el comando lógico ***spline=TRUE***, lo que se consigue es generar una curva diferenciable dividida en distintos tramos para los que se aplican distintos polinomios, tal y como se puede ver en la siguiente gráfica.

b) *Recta de regresión*. Por último, en cuanto al modelizado de los datos, es posible ajustar a la gráfica de dispersión una línea de regresión a la que se añadirá un intervalo de confianza del 95%. Tay y como se puede comprobar en la siguiente gráfica, además en esta ocasión se muestran dentro de la propia gráfica la ecuación de regresión lineal y el coeficiente de determinación R2. Esto se consigue activando el comando lógico ***linear=TRUE***.

El usuario deberá tener siempre en cuenta que en la actual versión de *Openair* el modelizado de la tendencia en los datos no es compatible con la utilización de una tercera variable, motivo por el que deberá decantarse en todo caso por una u otra opción.



11-. Mostrar el intervalo de confianza: ***ci***

Tanto el comando *smooth* como el comando linear incorporan por defecto a la gráfica de tendencia un intervalo de confianza al 95% que se muestra sobre la misma con un sombreado. Con el comando ló+gico ***ci=FALSE***, es posible desactivar este sombreado, haciendo que en la gráfica se represente sólo la tendencia.

12-. Agregar líneas de interrelación de variables: ***mod.line***

Dado que los ejes *x* e *y* seguramente se encuentren en rangos distintos (salvo que personalicemos sus límites), y puesto que en ocasiones puede ser útil disponer de algún tipo de herramienta que nos permita comprobar la exactitud y/o precisión con la que una dispersión de pares de datos se aproxima a la correlación perfecta 1:1, la función *scatterPlot* tiene previsto el comando lógico *mod.line*.

Este comando, una vez activado mediante ***mod.line=TRUE***, traza una línea recta que une los puntos en los que *x* es igual que *y* o, dicho de otra manera, la relación entre variables es unitaria 1:1, sirviendo así como línea de referencia. Además de la línea de referencia 1:1 que muestra en la gráfica este comando, se dibujan otras dos líneas secundarias, en formato de línea discontinua, que establecen las relaciones 2:1 y 0.5:1 de las variables *x* e *y*, completando así las referencias.

El comando *mod.line* puede ser muy útil, por ejemplo, para facilitar la interpretación visual de ciertas gráficas elaboradas por la función *scatterPlot* para la intercomparación de equipos o la realización de estudios de correlación.

13-. Establecer los colores de la gráfica: ***cols***

Al igual que otras gráficas en *Openair*, a través del comando *cols* se pueden personalizar los colores de la gráfica, ya sea usando paquetes de colores ya estandarizados, tales como *"increment"*, *"heat"* o *"hue"*, o disponiendo de los colores personalizados por el propio usuario.

14-. Cambiar el tipo de representación de los pares de datos: ***plot.type***

Los pares de datos de la dispersión se presentan por defecto como puntos *"p"*, pero pueden también representarse como líneas *"l"*, o como líneas y puntos *"b"*, según se asigne cada variable entre comillas al comando *plot.type*. Las dos últimas pueden ser especialmente útiles cuando se representa la fecha en el eje de las *x*.

15-. Cambiar la escala de los ejes: ***log.x*** , ***log.y***

Por defecto la escala de los ejes x e y es una escala aritmética, lo que podría ocasionar que la representación de los puntos no fuese del todo adecuada cuando las concentraciones de alguno de los parámetros presenta fluctuaciones muy fuertes. De esta forma, la función *scatterPlot* permite transformar los ejes a escala logarítmica (en logaritmo 10) para el eje *x* o el eje *y*, indistintamente, utilizando los comandos lógicos ***log.x=TRUE*** y/o ***log.y=TRUE***.

Transformar los ejes a escala logarítmica puede ser también una buena idea si queremos comprobar la existencia real de una relación lineal calculada entre dos parámetros, ya que la utilización de este tipo de escala exacerba las discrepancias que pudieran existir en la dispersión, alejándolas de una forma mucho más visual de la relación lineal.

16-. Modificar el tamaño de cuadrícula del método "level": ***x.inc*** , ***y.inc***

Ya hemos visto que cuando presentamos en una gráfica tres variables, podemos utilizar el método de pares de datos clásico *"scatter"* o utilizar un método de niveles *"levels"* en el que se modeliza la distribución en una cuadrícula dada en función de los pares de datos encontrados y la concentración media del tercer parámetro *"z"*.

Mediante los comandos *x.inc* e *y.inc* , a los que se les debe asignar el tamaño en función de las unidades de parámetro representado, podemos personalizar el tamaño de esa cuadrícula en sus dos dimensiones, ajustando la gráfica a una mayor o menor resolución en función de lo que más nos convenga.

17-. Modificar la relación entre ejes de las distintas gráficas: ***x.relation*** , ***y.relation***

Al utilizar el comando *type* y dividir la gráfica original en varias gráficas en función del parámetro indicado, podemos comprobar cómo las gráficas generadas comparten por defecto la misma escala, tanto para el eje *x* como para el eje *y*. Este hecho hace que en aquellos casos en los que la gráfica no presente una distribución uniforme, se pueda presentar un problema en la representación que dificulte su interpretación.

Para solventar esta situación la función scatterPlot dispone de los comandos *x.relation* e *y.relation*, que permiten utilizar la misma escala *"same"* (opción por defecto), o disponer de escalas independientes y adaptadas a cada gráfica para los ejes que el usuario quiera, con la opción *"free"*.

18-. Dibujar encabezados a las gráficas divididas: ***strip***

Cuando dividimos una gráfica *scatterPlot* en varias gráficas, con el comando *type*, la función presenta por defecto una banda superior con la que titula e identifica cada una de las gráficas elaboradas en función del parámetro seleccionado por el comando *type*.

En esta ocasión, la función *scatterPlot* permite al usuario eliminar este encabezado individual para cada gráfica, debiendo para ello desactivarse el comando con ***strip=FALSE***.

19-. Añadir líneas divisorias de referencia: ***ref.x*** , ***ref.y***

La función *smoothTrend* permite añadir a la gráfica, además de la cuadrícula de fondo que presenta por defecto, líneas destacadas que sirvan de referencia en la cuadrículas por las razones que el usuario considere más convenientes. Al utilizar los comandos *ref.x* y/o *ref.y*, el usuario tan sólo tendrá que indicar el punto concreto del eje correspondiente desde el que se desea trazar la línea destacada.

Estas líneas son muy útiles, por ejemplo, para la representación en la gráfica de la distribución de los partes de un determinado parámetro en función de sus valores límite, valores objetivo o umbrales de referencia., tal y como se muestra en el ejemplo dispuesto al final de este apartado.

20-. Utilizar una escala lineal o una escala comprimida para "z": ***trans***

Cuando representamos una tercera variable *z* en la gráfica *scatterPlot*, puede suceder que la escala de color que utilicemos no se adapte correctamente a los valores de la tercera variable, pudiendo predominar los valores muy bajos cuando nos interesaría conocer la evolución a valores más elevados.

Este efecto, ocasionado en muchas ocasiones por utilizar una escala lineal, puede solventarse activando el comando lógico ***trans=TRUE***, en cuyo caso se utilizará una transformación de raíz cuadrada que permitirá distribuir mejor la escala de colores utilizada para incluir mayoritariamente los valores de la parte superior de la escala.

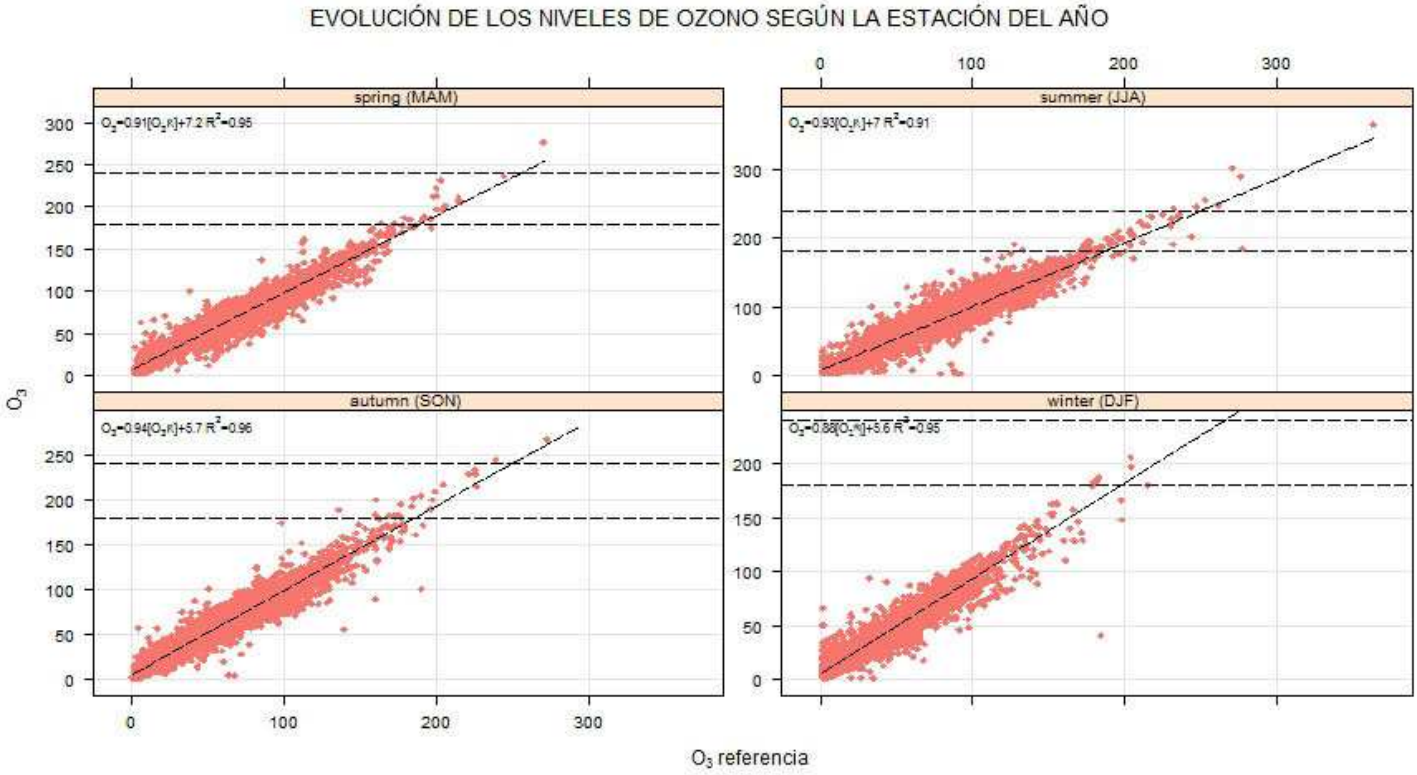
Por último, se debe tener en cuenta que la función *scatterPlot* sigue siendo un desarrollo de la función básica de *R* denominada *plot*, por lo que muchos de sus comandos de configuración son utilizables también en *scatterPlot*, siendo de enorme interés alguno de ellos , tales como:

- El comando *pch*, que modifica el símbolo utilizado para el marcador del par de datos.
- Los comando *ylim* y *xlim*, que permiten modificar los rangos en los que se moverán los ejes de la gráfica.
- Los comandos *ylab* y *xlab*, que permiten añadir títulos a los ejes x e y para describir las variables.

En la siguiente gráfica de ejemplo, se muestra la evolución de las concentraciones horarias de dos equipos de ozono funcionando en paralelo para su intercomparación. El primer equipo, denominado O3 en la gráfica, es el equipo habitual de la estación de control, mientras que el equipo denominado O3 referencia, es un equipo previamente calibrado y supervisado por organismo externo que hemos tenido en intercomparación con el anterior durante un periodo superior al año, y del cual se han adquirido los datos en paralelo para, con la función *merge*, incorporar los datos a nuestro conjunto de datos *datoscont*.

El estudio de intercomparación, dada la estacionalidad del parámetro, finalmente se divide por estaciones del año, siendo el resultado el que se muestra a continuación. En él se puede observar que:

- La evolución de ambos parámetros es prácticamente paralela confirmándose que existe una clara correlación entre ambos equipos.
- La correlación, sin embargo, atendiendo al coeficiente R2, se ve ligeramente afectada en la época de verano, donde parece que pueden darse factores que generan interferencias sobre alguno de los equipos, aunque se encuentran por encima de una R2 de 0,9 en cualquier caso.
- La gráfica presenta las líneas de referencia de 180 µg/m3 y 240 µg/m3 para que el lector pueda comprobar además la evolución de las superaciones de los umbrales horarios de información y alerta para los equipos y su correlación.




```
> scatterPlot ( datoscont, x="O3R", y="O3", linear=TRUE, type="season", pch=18, y.relation="free", ref.y=c(180,240), trans=TRUE, layout=c(2,2), xlab="O3 referencia", ylab="O3", main="EVOLUCIÓN DE LOS NIVELES DE OZONO SEGÚN LA ESTACIÓN DEL AÑO")
```

La función scatterplot es una función ampliamente utilizada en *R* y dispone de paquetes de desarrollo de la misma muy potentes, dato que es interesante que conozca todo aquel usuario que esté interesado en conocer o utilizar con mayor profusión esta herramienta gráfica. Especialmente interesantes para la representación de datos de calidad del aire podrían resultar paquetes de representación como ***scatterplot3d***, disponible como siempre en el recurso CRAN de R.

» La Función de relación lineal: *linearRelation*

La función de relación lineal representa gráficamente cómo evoluciona la relación entre dos parámetros contaminantes. Para ello analiza la variación de la pendiente de su regresión lineal con el tiempo (obviando el término independiente) mediante un modelo lineal que calcula dicha pendiente para el periodo temporal seleccionado con un intervalo de confianza del 95%.

De forma práctica, a la hora de evaluar y estudiar la calidad del aire, podemos decir que según hemos ido avanzando con este manual:

- Primero teníamos la función *scatterPlot* con la que podemos comprobar qué tipo de relación existe entre dos parámetros, incluso adoptando un tercer parámetro de referencia.
- En segundo lugar podíamos comprobar sobre dicha función gráfica cómo de estrecha era esa relación, utilizando el comando *linear* para calcular la recta de regresión y comprobar la correlación entre ambos parámetros.
- Y por último, surge la función *linearRelation*, que va más allá y analiza cómo evoluciona dicha correlación en función del tiempo, e incluso en función de distintos periodos temporales.

Básicamente la gráfica l*inearRelation* supone que entre dos parámetros dados existe una relación lineal que se corresponde con la ecuación básica *"y=ax + b"*, que se ha podido estudiar y comprobar a través de funciones anteriores, para luego comprobar y estudiar como evoluciona la pendiente de esta ecuación *"a"* en función de distintos periodos temporales.

Para llevar a cabo la función *linearRelation*, y estudiar en toda su extensión la evolución de la regresión lineal de dos parámetros con el tiempo, disponemos de los siguientes comandos:

Variaciones sobre la función:

1-. Parámetro a incluir en el eje X: ***x***

Se debe indicar mediante este comando el parámetro de referencia, o contaminante que deseamos que aparezca en el eje x.

2-. Parámetro a incluir en el eje Y: ***y***

Se deberá indicar con este comando el parámetro que deseamos que aparezca en el eje y.

3-. Periodo para el que se realiza el análisis de la pendiente: ***period***

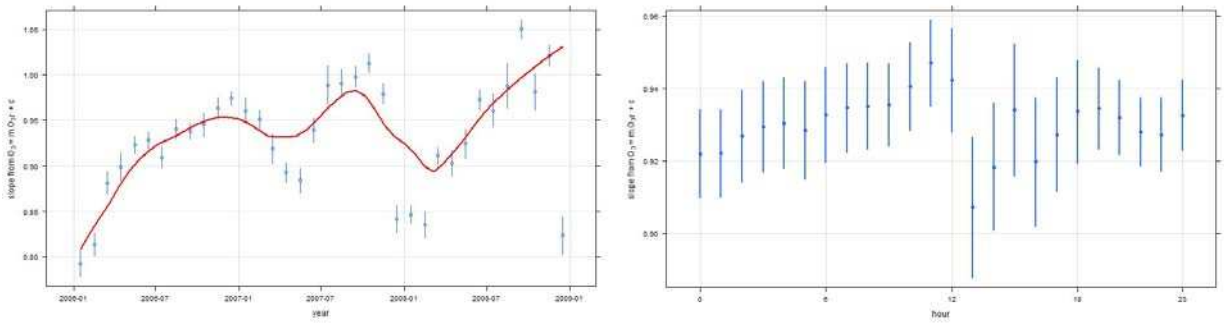
La gráfica de evolución de la pendiente puede realizarse para el análisis de distintos periodos de tiempo, y en distintos formatos, según se expone a continuación:

a) De forma lineal en el tiempo, representando los pares de datos en función de la fecha, de principio a fin, se puede analizar la evolución de la pendiente en promedios mensuales, ***period="monthly"***, o incluso también en promedios semanales, ***period="weekly"***. Para este tipo de gráficas la función *linearRelation* añade además una línea de tendencia suavizada que permite comprobar la evolución de la pendiente de forma más visual.

b) También podemos pedir a la gráfica que grafique ciertos ciclos temporales en los que se considere que pueden existir variaciones en la pendiente, pudiendo realizar un análisis de la evolución promedio durante las horas del día, ***period="hour"***, los días de la semana, ***period="weekday***", o una incluso utilizando una combinación de ambos, ***period="day.hour"***.

En las gráficas expuestas a continuación se trabaja sobre la base de la correlación existente entre el *O3* y el *O3R* visto en el anterior apartado, destinado a la función *scatterPlot*. En este caso el ejemplo tratará de establecer las dos posibilidades de representación por periodos que establece la función *linearPlot*, utilizando los pares de datos del ozono tratados con anterioridad.

Ej. **a) linearRelation (datoscont, x="o3R", y="o3", period="monthly")** , muestra una gráfica de evolución de la pendiente para la correlación existente entre los dos equipos de ozono vistos con anterioridad, durante todo el periodo de estudio contemplado para la intercomparación de los equipos. **b) linearRelation (datoscont, x="o3R", y="o3", period="hour")** , muestra la gráfica de evolución de la pendiente como promedio en las 24 horas de un día, pudiendo observarse como la pendiente se acerca a la unidad durante las horas centrales del día, cuando mayor radiación solar neta se observa.



4-. Dividir la gráfica por años: ***condition***

Cuando solicitamos a la función *linearRelation* que nos elabore una gráfica en la que haga un análisis de la evolución promedio durante los días de la semana, las horas del día o ambos, según se ha visto en el apartado anterior para el análisis de los ciclos temporales, es posible que ante largas series temporales el promedio realizado enmascare la evolución a lo largo de los años.

Dado que la función no dispone del comando *type*, para poder desagregar los datos de la gráfica, como si que ocurre en otras gráficas de *Openair*, dispone del comando lógico *condition*, que una vez activado mediante ***condition=TRUE***, dividirá la gráfica en años, mostrando la evolución deseada para cada año.

5-. Número mínimo de pares de datos para cada periodo: ***n***

En ocasiones puede ocurrir que el número de pares de datos completos *(x-y)* para un determinado periodo, y en especial para aquellos que son más reducidos, sea tan pequeño que pueda restar representatividad al modelo lineal calculado.

Mediante el comando *n*, al que se deberá asignar el número mínimo de pares de datos completos que se desea, se puede actuar sobre la representatividad del modelo lineal. De esta forma, aquellos periodos que no lleguen al número mínimo establecido por el usuario como representativo serán sencillamente ignorados en el cálculo del modelo lineal.

Cabe destacar en este caso, que la variable que requiere el comando *n* es una variable numérica que se refiere al número total de pares de datos que deberán estar disponibles, y no así al porcentaje sobre el total que se requieren como representativos, tal y como hasta el momento hemos visto en comandos como *data.thresh*.

6-. Coeficiente de correlación mínimo exigido: ***rsq.thresh***

En ocasiones la correlación existente para dos parámetros puede que no sea muy buena en determinados periodos de tiempo, ya sea porque no existe una relación efectiva, o porque se ve afectada por condicionantes externos o imprevistos.

Con el comando *rsq.thresh* es posible imponer a la función un límite de aceptación al coeficiente de correlación (*R2*) que se presente, de forma que cualquier recta de regresión cuya correlación sea menor que el número indicado por el usuario, será ignorada y no se representará.

El valor que podrá adquirir el comando *rsq.thresh* podrá ir de 0 a 1, aunque se deberá tener en cuenta que valores muy elevados de este comando podrían ocasionar que el número de rectas de regresión calculadas sea muy bajo y no se pueda obtener una evolución lógica de la tendencia. En cualquier caso, este comando aplicado a los distintos periodos previsto por la función puede ser muy útil para comprobar la evolución temporal y/o cíclica de la correlación, más allá de la propia pendiente.

7-. Establecer los márgenes de representación de la escala del Eje Y: ***ylim***

Por defecto la función ajustará los márgenes de representación del eje *Y* a los valores con que se presentan las pendientes calculadas por el modelo. No obstante, resulta posible modificar este aspecto mediante un comando básico de *R* como *ylim*, con el que podemos personalizar los valores mínimo y máximo a utilizar.

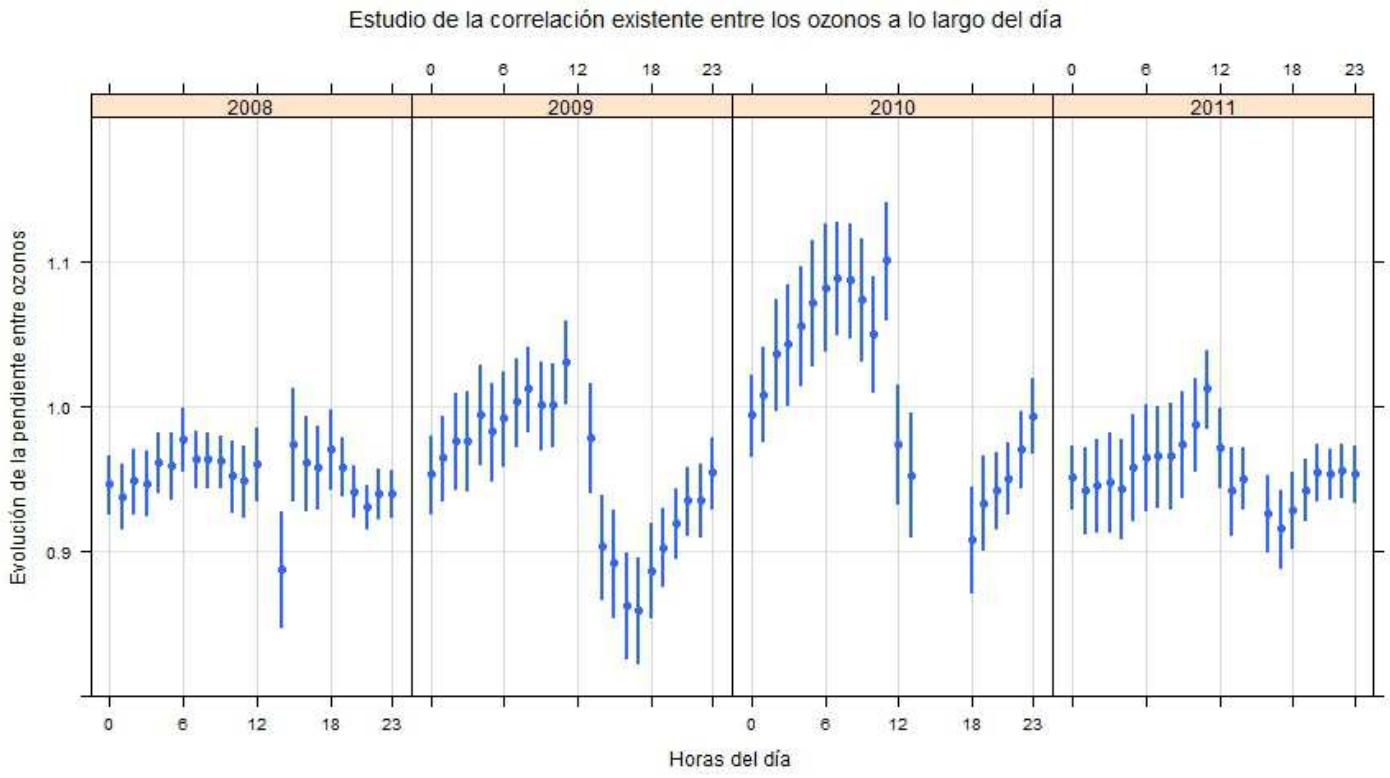
8-. Cambiar el ajuste de la línea de tendencia suavizada: ***span***

La línea de tendencia suavizada que dibuja la función cuando seleccionamos un periodo de representación temporal lineal, de principio a fin según la fecha, puede ajustarse en mayor o menor grado modificando el valor del comando span, de forma que sea una línea de tendencia más o menos ajustada a los puntos representados.

Este comando se hereda de la función de *R scatter.smooth*, y permite fijar el grado de ajuste de una curva de tendencia del tipo LOESS para la dispersión de puntos establecida. De esta forma, valores de ***span=0.2*** generarán un mayor ajuste de la curva de tendencia suavizada que valores de ***span=1*** que prácticamente la acercarán a una recta.

Por último, se debe recordar que para la función *linearRelation* es posible también utilizar los comandos propios de etiquetado de los ejes, como *ylab* o *xlab*, o de titulación de la gráfica, como *main*.

Como ejemplo ilustrativo de esta función gráfica de *Openair* se ofrece a continuación un ejemplo que continúa con los datos de la regresión lineal obtenida en el anterior apartado para la gráfica *scatterPlot* y los dos equipos de ozono de que dispone nuestro conjunto de datos *datoscont*. En esta ocasión se estudia la evolución de la pendiente existente en la correlación de los dos equipos con respecto a las horas del día y los distintos años de datos.



> linearRelation (datoscont, x="o3r", y="o3", period="hour", condition=TRUE, rsq.threash=0.85, n=320, ylab="Evolución de la pendiente entre ozonos", xlab="Horas del día", main"Estudio de la correlación existente entre ozonos a lo largo del día", ylim=c(0.8, 1.2))

» La Función de datos condensados: *trendLevel*

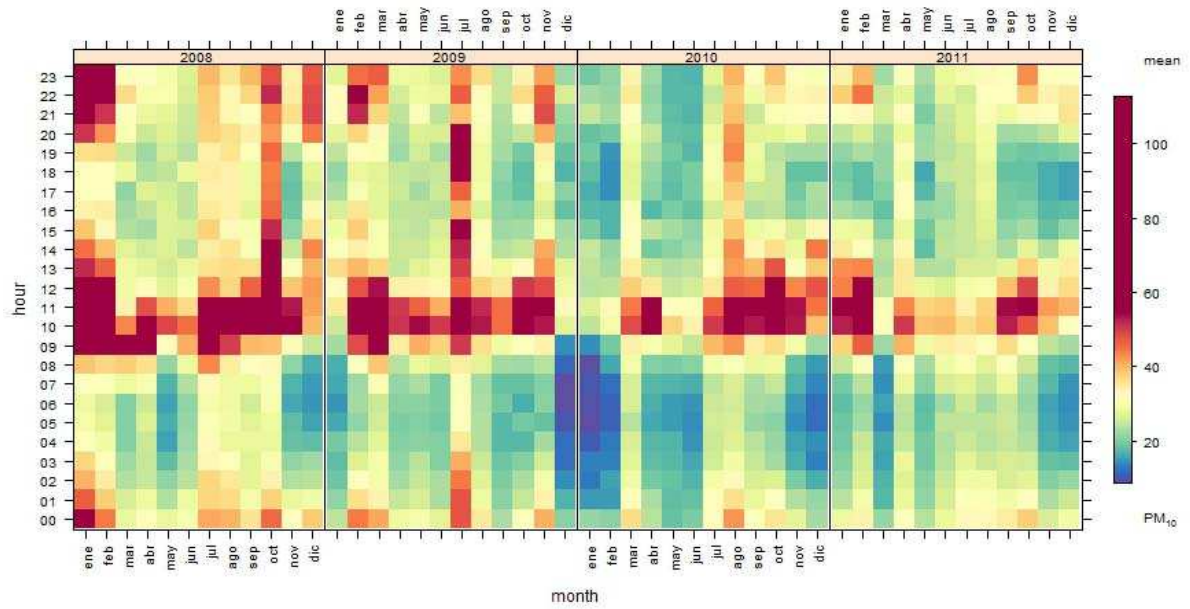
La función gráfica *trendLevel* proporciona una manera muy práctica y visual de condensar grandes bloques de datos de calidad del aire en una sola gráfica en la que además podremos comprobar cómo evolucionan las tendencias globales, diarias e incluso estacionales para los distintos contaminantes.

La propiedad fundamental que tiene *trendLevel*, y que la diferencia del resto de funciones de *Openair*, es precisamente la capacidad de poder condensar la información al ofrecer diversas combinaciones para calcular el estadístico deseado del contaminante o parámetro que se le indique. Esto se consigue al definir los ejes x e y de la gráfica como funciones de desagregación de datos, similares al comando *type* o la instrucción *cutData*, permitiendo a su vez desagregar la gráfica en varias.

De esta forma, la función *trendLevel* se limita a calcular el estadístico que corresponda, para un parámetro dado, en el nivel de desagregación que se le indique, que generalmente es de carácter temporal (horas del día por meses del año, días de la semana por estaciones, etc), para luego representarlo en la gráfica mediante una escala de colores.

En su versión más simple, donde las variables básicas de la función se adoptan por defecto, la función *trendLevel* muestra ya su gran capacidad de condensación, tal y como se puede observar en el siguiente ejemplo:

Ej. **trendLevel(datoscont, pollutant="pm10")** , donde se calcula el promedio de un parámetro de nuestro conjunto de datos, en este caso de PM10, para cada hora del día (desagregación por defecto en el Eje Y) en cada mes del año (desagregación por defecto en el Eje X), y esto elaborado en una gráfica para cada año de nuestro histórico de datos (condición por defecto del comando type), tal y como se puede observar en la gráfica adjunta.



Aunque se encuentra todavía en fase de desarrollo, la función *trendLevel* proporciona las siguientes opciones de configuración:

Variaciones sobre la función:

1-. Seleccionar el parámetro a representar: ***pollutant***

Se debe indicar a la función el contaminante o parámetro del conjunto de datos que se desea representar en la gráfica.

2-. Seleccionar los criterios de desagregación de los datos: ***x***, ***y***, ***type***.

La función *trendLevel* calcula la estadística solicitada para un parámetro dado usando para ello la combinación de los tres criterios de desagregación que dispongamos, asignados mediante los comandos ***type***, que divide la gráfica, y los comandos ***x*** e ***y*** que definen los ejes.

La función trendLevel siempre hará uso de estos tres niveles de desagregación de los datos, asignando valores por defecto a aquellos que el usuario no defina personalmente, tal y como hemos visto en la gráfica de ejemplo anterior. Evidentemente, un requisito a cumplir en cualquier caso es que los niveles de desagregación definidos no deben coincidir nunca unos con otros, si aún siquiera con los asignados por defecto. En caso contrario la función devolverá un error.

Las variables a aplicar a cada uno de los niveles de desagregación de la función son los ya vistos para el comando *type*, en toda su extensión, pudiendo pedirse que se dividan los ejes o la propia gráfica, en función de las horas del día *"hour"*, las horas de luz *"daylight"*, los días de la semana *"weekday"*, o por fines de semana y días laborables *"weekend"*, por meses del año *"month"*, por meses y años *"monthyear"*, por estaciones del año *"season"*, o incluso por años *"year"*.

La desagregación con que se defina cada comando de la función, además de las variables predefinidas vistas con anterioridad, puede también tener en cuenta campo numéricos de nuestra base de datos, tal y como vimos para la instrucción *cutData*. Para ello, tan sólo habrá que indicar al comando el nombre del campo al que debe referirse.

Los niveles de desagregación aplicados se definen mediante los siguientes comandos:

a) *Desagregación a aplicar en el Eje X:* ***x***.

Aplica cualquier tipo de desagregación de las vistas anteriormente al Eje X de la gráfica. Por defecto aplica la desagregación *"month"* (meses del año). Si se utilizase una variable numérica de nuestra base de datos, y ello fuese posible, se realizaría una división de los datos en 10 niveles o cuantiles.

b) *Desagregación a aplicar en el Eje Y:* ***y***

Aplica cualquier tipo de desagregación de las vistas anteriormente al Eje Y de la gráfica. Por defecto se aplica la desagregación *"hour"* (horas del día). Si se utilizase una variable numérica de nuestra base de datos, y ello fuese posible, se realizaría una división de los datos en 10 niveles o cuantiles.

c) *Desagregación a aplicar en la propia gráfica:* ***type***.

Aplica cualquier tipo de desagregación de las vistas anteriormente a la propia gráfica, dividiendo la misma en varias. Por defecto se aplica la desagregación más genérica *"year"* (años). Si se utilizase una variable numérica, la gráfica se dividirá en función de sus cuatro cuartiles.

Destacar además que el comando *type*, al igual que en otras gráficas de *Openair*, se puede utilizar combinando varios de los criterios vistos hasta el momento para segregar los datos.

3-. Número de niveles en los que dividir los ejes: ***n.levels***

Cuando la variable seleccionada para los comandos de desagregación *x*, *y*, o *type* es numérica, correspondiéndose a uno de los campos de nuestro conjunto de datos, ya hemos visto anteriormente que por defecto la función procura dividir en 10 niveles o cuantiles los ejes *x* e *y*, y en los cuatro cuartiles habituales a la propia gráfica, en lo que sería por defecto ***n.levels=c(10,10,4)***. No obstante, se podría indicar a la función que utilizase un número distinto de divisiones modificando los valores preestablecidos para el comando *n.levels*.

4-. Dar inclinación a la rotulación de los ejes: ***rotate.axis***

En ocasiones, y al objeto de favorecer una mejor rotulación de los ejes de la gráfica, es preciso modificar la inclinación de las variables incluidas en los ejes. Por defecto la gráfica optimiza la representación, dando a las variables del eje *X* una orientación vertical, mientras que las del eje *Y* mantienen una orientación horizontal, en lo que sería la configuración del comando como ***rotate.axis=c(90,0)***. No obstante, el usuario podrá disponer de la configuración que considere más adecuada o mejor se adapte a las especiales circunstancias de su gráfica.

5-. Límites en los que se moverá la escala de colores: ***limits***

Por defecto, la gráfica acomoda la escala de colores establecida al rango de valores disponible para el contaminante o parámetro seleccionado. Sin embargo, en ocasiones puede ser preciso indicar a la función los valores mínimo y máximo que debe escoger el comando *limits*.

6-. Colores a utilizar para la escala: ***cols***

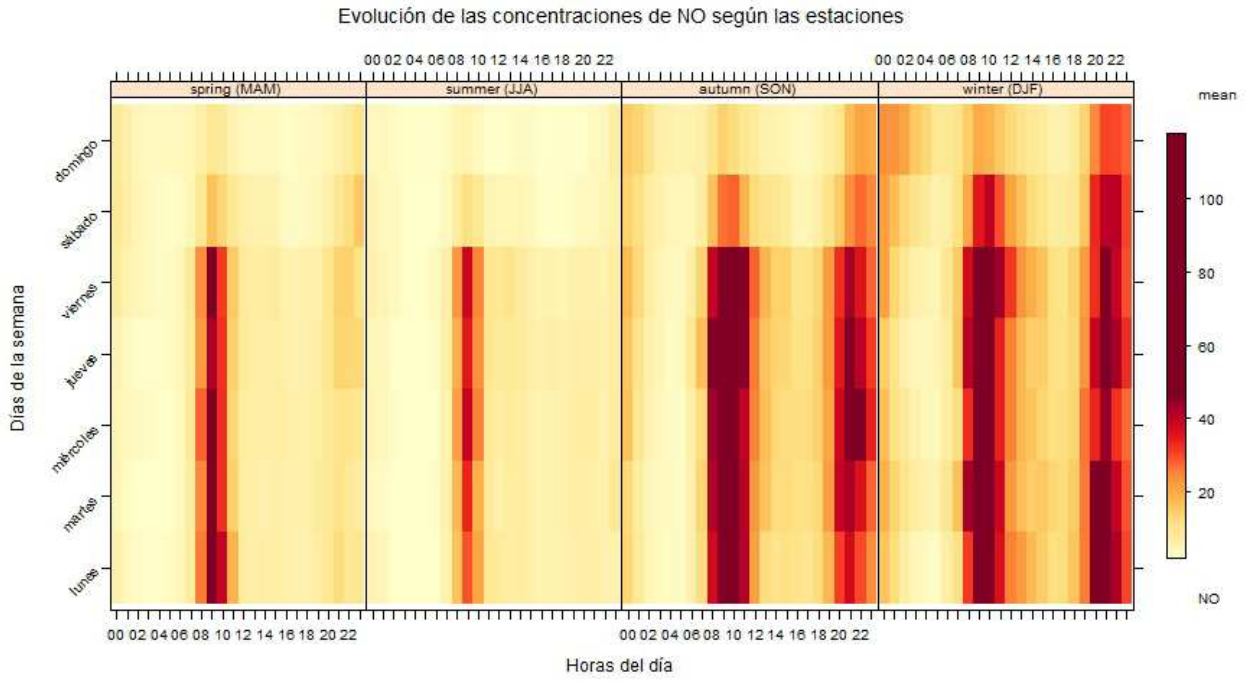
Como hemos visto en múltiples ocasiones, la gráfica podrá hacer uso de combinaciones de colores predeterminadas, o incluso establecer combinaciones personalizadas por el usuario a través del comando cols.

7-. Cálculo estadístico a realizar: ***statistic***

La función *trendLevel* utiliza por defecto para el cálculo estadístico de los datos la media *"mean"*, aunque mediante el comando *statistic* podría también solicitar a la función que realizase el cálculo del máximo *"max"*, o del número de datos o frecuencia de aparición *"frequency"*.

El cálculo estadístico de los datos es uno de esos comando de *trendLevel* que se encuentra en la actualidad aún en desarrollo. No obstante, permite la asignación directa de aquellas funciones que codifique el propio usuario previamente, para lo cual habrá que tener un especial cuidado. En todo caso esto sería objeto de un desarrollo más completo de la programación en R que escapa a los objetivos de este manual.

Una correcta utilización de la función *trendLevel* permite obtener interesantes gráficas como la expuesta a continuación para la evolución del NO en nuestro conjunto de datos. En ella se puede observar la afección del tráfico rodado en los días laborales, así como la estacionalidad del contaminante debido a los condicionantes meteorológicos dados en la zona.




```
> trendLevel (datoscont, pollutant="no", x="hour", y="weekday", type="season", rotate.axis=c(0,45), cols="heat", xlab="Horas del día", ylab="Días de la semana", main="Evolución de las concentraciones de NO según las estaciones del año")
```

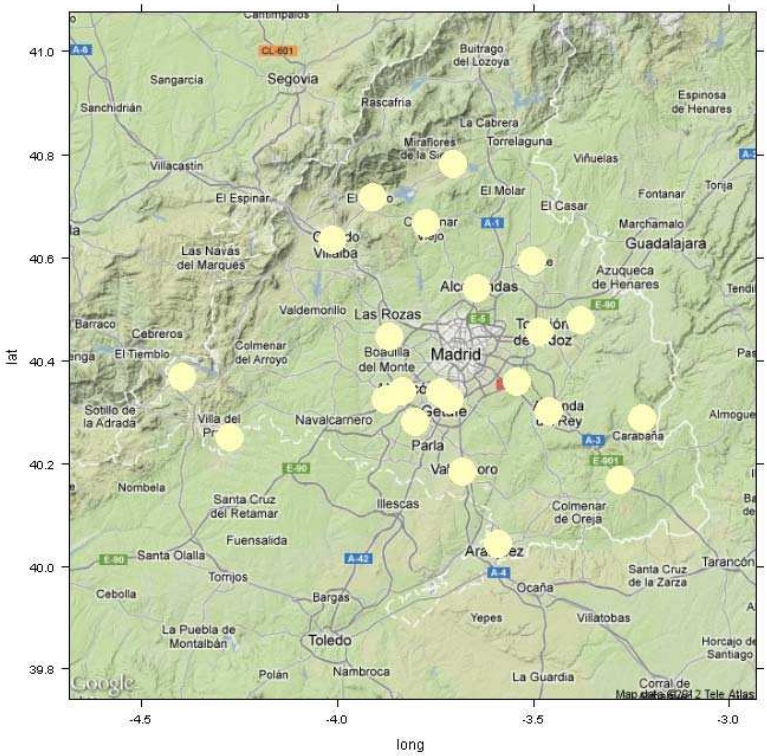
» La Función de Representación Geográfica: *GoogleMapsPlot*

Como última herramienta gráfica de especial interés para el trabajo diariamente desarrollado por los técnicos en materia de calidad del aire, este manual quiere presentar la función de representación geográfica, cuyo propósito básico es permitir al usuario utilizar los mapas de Google para realizar representaciones espaciales de la calidad del aire.

La función GoogleMapsPlot se basa en mapas estáticos simples de Google (API de Google Maps), lo que comparado con cualquier Sistema de Información Geográfica (GIS) convencional hará que resulte bastante limitado. Sin embargo, no por ello deja de ser muy interesante para la representación de localizaciones por puntos. Así, la función GoogleMapsPlot resulta muy útil para representar datos sobre distintos tipos de planos, permitiendo comprobar la distribución geográfica de las concentraciones de un determinado contaminante.

El modo de funcionamiento de la función es muy sencillo, siguiendo el concepto de otras funciones de gráficas de Openair. La función GoogleMapsPlot representa, como si fuese un gráfico X-Y habitual, las coordenadas de latitud y longitud geográfica que le demos en nuestro conjunto de datos, siendo para esta función la información aportada algo similar a la función ScatterPlot de representación de pares de datos. Una vez representados los pares de coordenadas en la gráfica, lo que hace es añadir de fondo el mapa de Google Maps que se adapte a las coordenadas que tengamos en nuestros ejes X e Y. De esta forma, sólo con un conjunto de datos que disponga de la ubicación de las estaciones y los comandos básicos de la función, se representaría la ubicación de los puntos de control, tal y como se puede comprobar en el siguiente ejemplo adjunto.

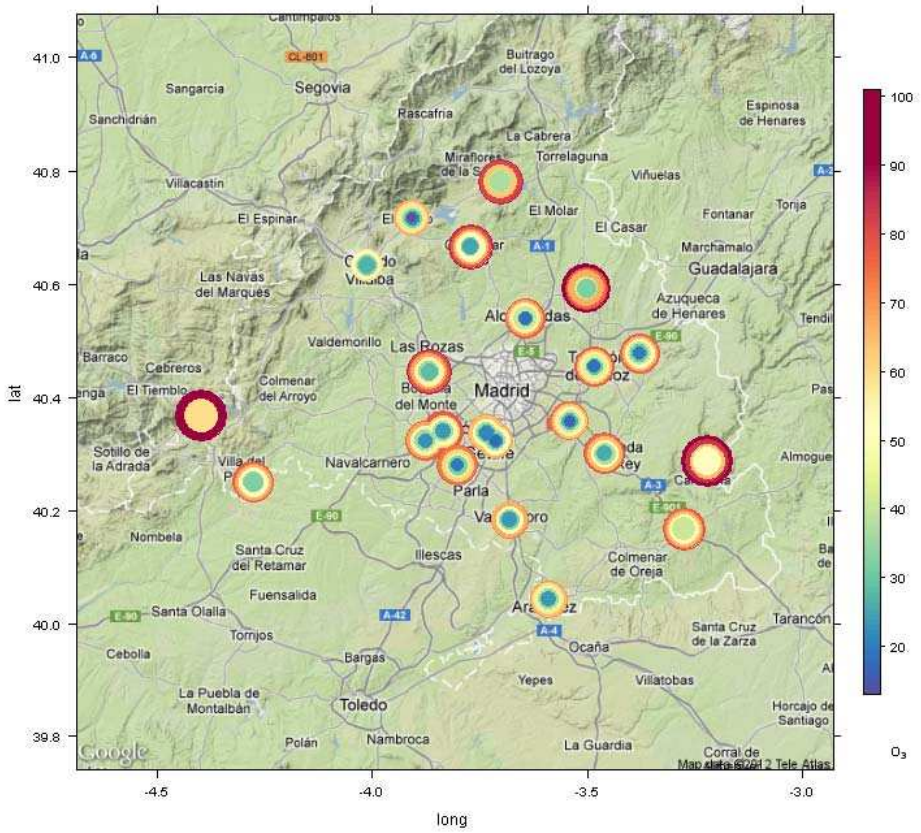
Ej. **GoogleMapsPlot(datoscontub, latitude="lat", longitude="long")** , donde lo que hace la función es representar los distintos pares de coordenadas de las estaciones de la Comunidad Autónoma de Madrid de que disponga el archivo de datos que hemos modificado, datoscontub para que disponga de las coordenadas de ubicación.



Si a esta configuración básica de representación le añadimos a cada punto de ubicación la posibilidad de que represente un parámetro, solicitando a la función que lo obtenga desde nuestro conjunto de datos, como por ejemplo el ozono, nos encontramos que para cada punto de la gráfica (par de datos de longitud y latitud), Openair modifica

la representación del punto de ubicación, variándolo de color y tamaño para representar la concentración dada del parámetro solicitado en el punto de coordenadas establecido, tal y como podemos ver en el siguiente ejemplo:

Ej. **GoogleMapsPlot(datoscontub, latitude="lat", longitude="long", pollutant="o3")** , donde lo que hace la función es representar los distintos pares de coordenadas de las estaciones de la Comunidad Autónoma de Madrid de que disponga el archivo de datos que hemos modificado, datoscontub para que disponga de las coordenadas de ubicación, modificando los puntos en tamaño y color para incluir la representación de la concentración de Ozono dada para cada punto.



Un aspecto importante a tener en cuenta, tal y como puede comprobarse en el ejemplo adjunto, es que GoogleMapsPlot no dispone actualmente de ninguna capacidad de cálculo estadístico o integración de los datos por lo que representa para cada punto todos los datos que encuentre del parámetro.

Esto supone que no basta con añadir las coordenadas geográficas a nuestro conjunto de datos sino que, tal y como podemos imaginar, debemos tratar previamente los mismos, antes de ejecutar la función de representación geográfica, de forma que la representación se haga para los datos pretendidos tanto desde el punto de vista de integración temporal como desde el punto de vista de su desagregación, si queremos aplicar el comando type.

Variaciones sobre la función:

1-. Introducir las coordenadas geográficas a representar: ***latitude***, ***longitude***

La función requiere obligatoriamente que se le indiquen los campos de conjunto de datos que disponen de las coordenadas geográficas que posicionan cada uno de los puntos a representar para lo cual utiliza los comandos latitude y longitude, a los que habrá que indicarles los campos que disponen de los datos de latitud y longitud, respectivamente.

En este sentido, conviene tener en cuenta que Google puede posicionar y representar cualquier punto en el planeta, siempre que se tenga en cuenta que las coordenadas que usa se corresponden con un sistema de proyección cilíndrica simple con un Datum WGS84. Si disponemos de nuestras coordenadas en otro sistema de ubicación geográfica, como por ejemplo UTM (Universal Transversal de Mercator) deberemos de proceder previamente a su transformación. Así mismo, la coordenada se puede proporcionar a Openair en formato numérico decimal, no siendo necesaria su transformación a grados, minutos y segundos.

2-. Añadir un parámetro o contaminante al punto: ***pollutant***

Con la función *GoogleMapsPlot*, tal y como hemos visto anteriormente, es posible suministrar un campo adicional relativo a un parámetro o contaminante que se representará para cada punto de ubicación dispuesto. La representación de dicho parámetro se llevará a cabo modificando el tamaño y/o color del punto en cuestión, en función de los comandos que establezca el usuario para la función.

3-. Modificar los márgenes de la gráfica: ***xlim***, ***ylim***

La función *GoogleMapsPlot* adopta automáticamente los valores de los Ejes X e Y que mejor se adapten a la representación de los puntos que se hayan suministrado, ajustando los límites superior e inferior de ambos ejes a un cuadrado que contenga todos los puntos con la máxima resolución posible del plano de fondo.

En cualquier caso, como en otras gráficas de *Openair*, el usuario podrá modificar dichos valores con los comandos *xlim* o *ylim*, a los que tendrá que dotarse con los valores mínimo y máximo que se desea para las coordenadas de longitud (para el Eje X) o de latitud (para el Eje Y). Se debe tener en cuenta que, en la actual versión de *Openair,* el área de representación de la gráfica es siempre cuadrada, por lo que el usuario sólo podrá modificar el Eje X o el Eje Y, adaptándose automáticamente el otro eje a los valores que correspondan.

4-. Desagregar los datos para representar distintas gráficas: ***type***

Al igual que con otras gráficas de *Openair*, la función *GoogleMapsPlot* dispone del comando *type* que le permite desagregar los datos para representarlos en distintas gráficas. Si bien esta función conserva todas las características del comando *type*, vistas en gráficas anteriores, se debe recordar al usuario que el conjunto de datos a utilizar debe encontrarse previamente tratado, lo cual incluye también las posibles desagregaciones que se vayan a hacer con dicho comando. De lo contrario, la gráfica realizará para el mismo punto y la misma gráfica varias representaciones (tantas como valores adopte el parámetro para dicha ubicación), tal y como ocurría en la gráfica de la página anterior.

Es importante recordar siempre que la función *GoogleMapsPlot* basa su representación en los pares de coordenadas y no en el campo date, como otras gráficas. Se trata de representar coordenadas con la posibilidad de incluir una tercera variable, y no de representar sólo parámetros contaminantes.

5-. Etiquetar los puntos geográficos representados: ***labels***

Si el conjunto de datos, además de las coordenadas geográficas, aporta un campo de texto descriptivo de los puntos, es posible indicar a la gráfica, mediante el comando *labels*, el campo a utilizar para identificar y etiquetas cada uno de los puntos.

6-. Personalizar los colores de la gráfica: ***cols***

Al igual que en otras gráficas de *Openair*, el comando *cols* permite personalizar los colores de representación del parámetro o contaminante, ya sea utilizando cualquiera de los paquetes de colores predeterminados de *Openair*, o los colores que el usuario considere más adecuados.

En caso de utilizar el paquete predeterminado *"greyscale",* no sólo se afectará a la escala de colores de representación del parámetro o contaminante, sino que se transformará toda la gráfica a escala de grises, con inclusión del plano de fondo de Google.

7-. Cambiar los límites de la escala de colores: ***limits***

Por defecto la escala de colores para representar el parámetro o contaminante en el plano se adecua automáticamente al rango de valores a representar. Sin embargo, el usuario podría precisar modificar dicho rango para adaptarlo a sus propias necesidades (como por ejemplo referir el mismo a valores límite), en cuyo caso puede utilizar el comando *limits* para modificar los limites inferior y superior de dicho rango de representación.

8-. Modificar el símbolo a utilizar para representar los puntos: ***pch***

En el fondo de su diseño, la función *GoogleMapsPlot* no deja de ser una gráfica bivariable normal, similar a la gráfica *plot* vista para *R* en su apartado correspondiente. Así, los puntos de la función se representan atendiendo a los códigos ya comentados en el apartado de gráficas R correspondiente al comando *pch*. Por defecto, la función utilizar el símbolo equivalente a ***pch=20***, pero podría utilizar cualquier otro que el usuario considerase más conveniente.

9-. Tamaño con que se representa el punto en la gráfica: ***cex***, ***cex.range***

Si representamos en la gráfica tan sólo ubicaciones, el valor por defecto del tamaño del punto es *NULL* y se establece en función del rango de representación geográfica que se obtenga en la gráfica. En todo caso, se puede formar a los puntos a un tamaño distinto, utilizando para ello el comando *cex*, al que se le podrá asignar el tamaño que el usuario estime más adecuado, mediante un número entero.

Si el punto a representar en la gráfica busca a su vez representar el valor o concentración de un parámetro del conjunto de datos, la función *GoogleMapsPlot* varía por defecto su tamaño y color en función de la concentración o valor que adquiera el parámetro en cada caso. En este caso, el valor de *cex* es igual al valor de la concentración, lo cual puede afectar a la calidad de la representación, especialmente en aquellos casos en los que la ubicación de los puntos es muy próxima. En estos casos, el usuario puede optar por anular la representación del valor por tamaños, estableciendo un valor único con el comando *cex*, o escalar el valor de representación del punto mediante el comando *cex.range*, al que se le asignará el valor de escala que se quiera en función de las concentraciones a representar, como por ejemplo: ***cex.range=c(1,3)***.

10-. Eliminar la representación de los ejes en la gráfica: ***axes***

Mediante el comando lógico *axes* es posible forzar a la gráfica a que represente los Ejes X e Y, opción por defecto, o hacer que los elimine de la gráfica, mediante el comando ***axes=FALSE***.

11-. Incluir el mapa como un objeto o imagen rasterizada: ***map.raster***

Por defecto la generación del plano se realiza como una imagen rasterizada (imagen matricial o en mapa de bits), por ser esta generación la más ágil y la que mejor facilita la elaboración de la gráfica. En cualquier caso, puede ser necesario indicarle a *GoogleMapsPlot* que no rasterice la imagen, siempre que nos esté permitido a nivel de software, en cuyo caso se deberá utilizar el comando ***map.raster=FALSE***.

12-. Modificar el tipo de mapa que se muestra en la gráfica: ***maptype***

Por defecto, la función *GoogleMapsPlot* utilizará el plano del terreno de que dispone Google como mapa de base en función de las coordenadas que se disponga. Sin embargo, la opción de terreno *"terrain"* no es la única disponible, y el usuario podría precisar del uso de otros planos como: los mapas de satélite o fotografía aérea *"satellite",* muy útiles cuando la representación se va a hacer para redes de calidad del aire de núcleos urbanos, los mapas de carreteras o vías de tráfico *"roadmap"* o *"mobile"*, o un plano híbrido entre el de terreno y el de carreteras *"hybrid"*.

13-. Realizar un zoom al plano representado: ***zoom***

En un punto anterior hemos visto cómo se puede actuar sobre las dimensiones del plano actuando sobre los límites de las coordenadas dispuestas por los ejes con el comando típico de *R xlim* o *ylim*. Sin embargo, el paquete *RgoogleMaps* sobre el que se basa la función *GoogleMapsPlot* de *Openair*, también permite modificar el tamaño de la imagen sin actuar directamente sobre los ejes, sino sobre el propio plano, haciendo que con el comando *zoom* se pueda ampliar o reducir la imagen de una forma sencilla y rápida.

Por defecto la función aplicará el *zoom* que mejor se adapte a la representación solicitada y a los márgenes de coordenadas disponibles, tal y como podemos ver en el resumen de parámetros que sobre la consola arrojará la función una vez ejecutada. Sin embargo, el usuario podrá actuar para mejorar o adaptar estos parámetros mediante comandos como *zoom*, *center* o *size*, que veremos en los siguientes puntos.

14-. Establecer el centro del plano: ***center***

De forma automática, uy en función de las coordenadas por puntos que le hayamos dado, la función GoogleMapsPlot establece el centro de la imagen del plano para ubicarla dentro del cuadrado que sirve para representar los puntos de coordenadas de la gráfica. Sin embargo, el usuario podría requerir de dicha imagen esté desplazada del centro del cuadrado, por lo que podrá personalizar las coordenadas que sirven para centrar la imagen con el comando center, al que se deberá asignar tanto la latitud como la longitud, en este orden, tal y como marca el siguiente ejemplo: ***center=c(40.2,-4.5)***

15-. Cambiar el tamaño del plano descargado: ***size***

En combinación con otros comandos de la gráfica ya vistos, el usuario puede cambiar el tamaño en *pixels* del plano descargado a la gráfica. Se debe recordar siempre que los valores disponer deben ser siempre iguales para el tamaño marcado, tanto para *x* como para *y*, de forma que se correspondan con el cuadrado que representa el gráfico.

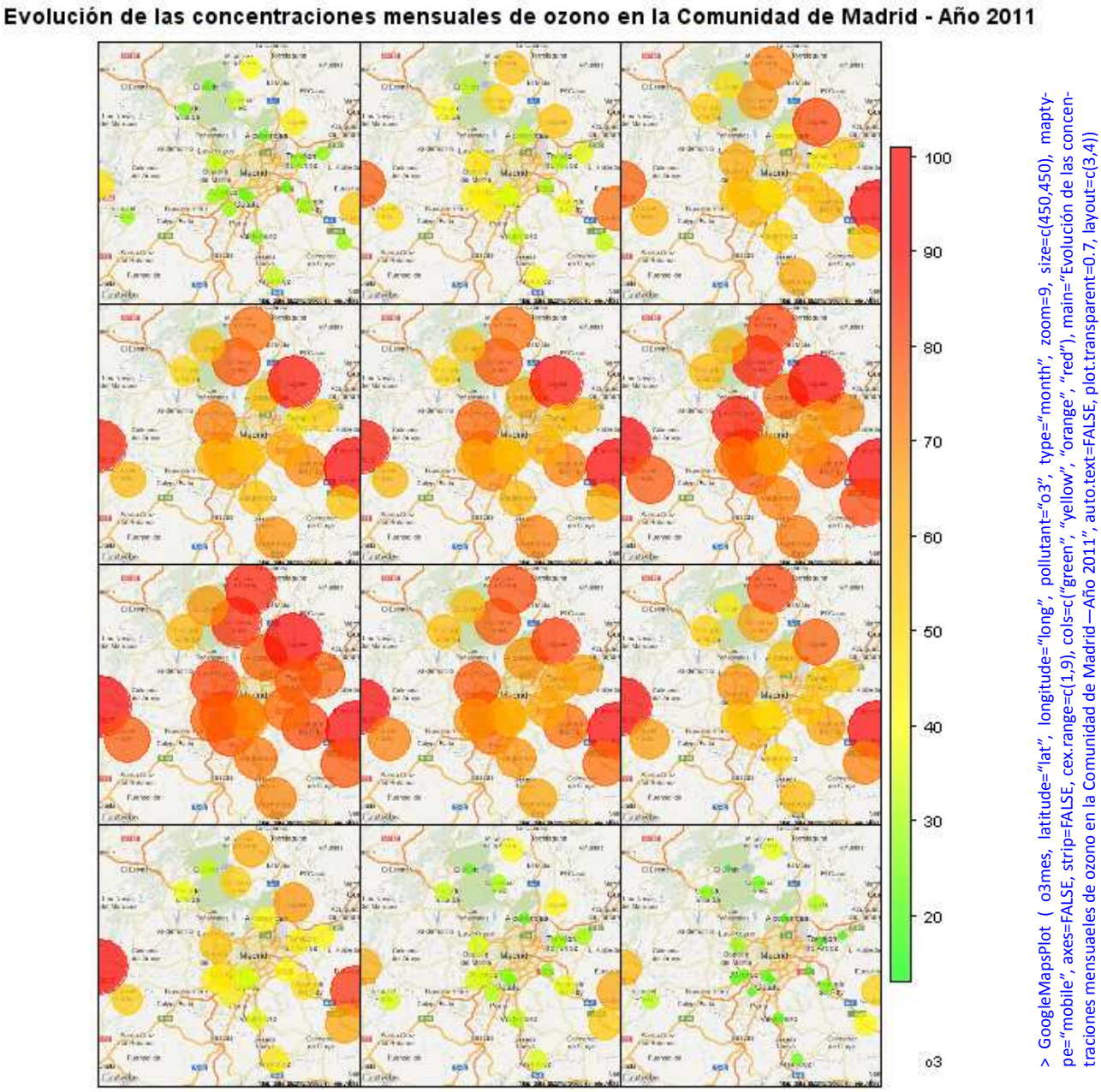
16-. Cambiar el aspecto o proporción de la gráfica: ***aspect***

Se puede cambiar el aspecto de una gráfica modificando las proporciones con las que se representa. De esta forma, si ***aspect=1*** la proporción de la gráfica es la normal, mientras que si el valor es menor la gráfica se ensancha, y si es mayor se alarga. Esta función permite al usuario deformar la gráfica para adaptarla a las necesidades futuras de representación.

17-. Modificar el nivel de transparencia de la representación: ***plot.transparent***

El comando *plot.transparent* nos permite regular el nivel de transparencia con que se representa el punto geográfico en la gráfica, pudiendo adoptar valores que vayan de ***plot.transparent=0***, para hacer el punto invisible, a ***plot.transparent=1***, para hacer la representación totalmente opaca.

Además de las opciones de configuración vistas hasta el momento, la función gráfica GoogleMapsPlot permite el uso de las opciones de configuración clásica de R para gráficas plot, tales como xlab, ylab, main, sub, etc. e incluso opciones de configuración vistas en otras gráficas como strip o layout, para influir sobre el modo de representación de la gráfica cuando se procede a su desagregación con type, tal y como podemos ver en el ejemplo final.



INSTRUCCIONES PARA LA EVALUACIÓN DE MODELOS :

En calidad del aire es muy habitual que tanto organismos oficiales como entidades privadas utilicen modelos que permiten establecer la calidad del aire en un sitio determinado a partir de un cálculo matemático complejo basado en una serie de datos de partida como puedan ser focos de emisión, histórico disponible de datos, datos meteorológicos, etc.

La modelización de la calidad del aire para la predicción de niveles en inmisión puede deberse a la necesidad de:

a) Evaluar la calidad del aire en zonas no controladas de forma automática y que, en función de lo establecido en la normativa aplicable, pueden ser evaluadas mediante este tipo de modelos.
b) Complementar las mediciones de control en continuo de determinadas zonas mediante modelos matemáticos que establezcan la dispersión de la contaminación en la zona.
c) Disponer de una previsión de los niveles de la calidad del aire en una zona determinada, con anterioridad a su registro efectivo, de cara a procurar la adopción de las medidas preventivas que corresponda.

En cualquier caso, el objetivo fundamental de cualquiera de estos modelos de calidad del aire es ofrecer un dato de uno o varios parámetros contaminantes en un punto o zona determinada y una fecha en concreto, buscando en todo momento que el dato calculado sea lo más fiel posible al dato real o esperado para esa zona.

El resultado final de estos modelos es pues un conjunto de datos brutos, referidos a una serie de fechas determinadas, que de por sí podrían ser en cuanto a su estructura, formato y contenidos a los propios conjuntos de datos procedentes del control en continuo, pudiendo incluso gestionarse y tratarse de idéntica manera.

Sin embargo, estos datos son fruto de un modelo matemático y, en muchas ocasiones, uno de los aspectos de mayor interés y complejidad en materia de calidad del aire es la evaluación de su efectividad, contrastando los datos procedentes del modelizado con los datos reales que se obtengan mediante determinación *"in situ"*.

La evaluación de modelos no está todavía estandarizada, y en muchas ocasiones la evaluación realizada se limita a un estudio cualitativo de los resultados obtenidos o, como mucho, a la realización de un estudio simple con estadísticos básicos como el error cuadrático medio, el error absoluto o el sesgo.

Por otro lado, la normativa actualmente vigente en esta materia tan sólo exige que se cumplan una serie de incertidumbres máximas en cuanto a los distintos periodos de integración de los datos, por lo que tampoco apunta mayor información al respecto.

Sin embargo, una buena evaluación de un modelo de calidad del aire debe venir acompañada siempre por un análisis estadístico serio y riguroso de sus resultados, que permita medir la precisión de la simulación, y que complemente el análisis cualitativo realizado, garantizando así sus conclusiones y su reproducibilidad.

Openair pone a disposición del técnico una serie de herramientas de enorme utilidad para la realización del análisis estadístico de los modelos y su evaluación con datos reales, aun cuando dichos datos provengan de grandes series históricas. Dichas herramientas permiten al técnico simplificar al máximo el cálculo estadístico por un lado, reduciéndolo a una simple programación de una instrucción, y por otro a disponer de funciones gráficas específicas que faciliten la visualización e interpretación de los resultados.

Tan sólo se requerirá un tratamiento previo de los datos para ubicar en un único conjunto de datos los datos correspondiente al modelo y a los datos reales utilizados para contrastarlo. A partir de ahí, *Openair* ofrece instrucciones de gran utilidad como la siguiente:

♦ Cálculo de las variables estadísticas más comunes en evaluación de modelos: ***modStats***

Mediante una única instrucción Openair permite al usuario realizar el cálculo de los estadísticos más habituales en la evaluación de un modelo predictivo, atendiendo a los siguientes comandos básicos de configuración.

1-. Introducir el nombre del campo que contiene los datos del modelo: ***mod***

Mediante el comando *mod* se indica a la instrucción el campo de nuestro conjunto de datos que dispone de los datos del modelo.

2-. Introducir el nombre del campo que contiene los datos reales: ***obs***

Mediante el comando *obs* se introduce el nombre del campo de nuestro conjunto de datos que contiene los datos observados , o datos reales, frente a los que se quiere comparar los datos del campo correspondiente al modelo.

3-. Desagregar los datos estadísticos calculados: ***type***

El comando *type* nos permite desagregar el conjunto de datos disponible en función de distintos periodos temporales de interés, como *"season"*, *"weekday"*, *"year"*, etc.

El comando *type* también se puede asignar a cualquier otro campo de nuestro conjunto de datos, en cuyo caso la función *modStats* elaborará los estadísticos en función de los contenidos de dicho campo.

Esta última propiedad, por ejemplo, resulta de enorme interés para la evaluación de diversos modelos, ya que si caracterizamos previamente un campo de nuestro conjunto de datos (con una función del tipo ***splitByDate***), clasificando los datos del campo del modelo (*mod*) en función de los distintos modelos utilizados, o de las configuraciones adoptadas para un mismo modelo a lo largo del tiempo, podemos calcular las estadísticas para cada modelo.

Por otro lado, si utilizamos el comando *type*, y dividimos los cálculos desagregando los datos del modelo, es posible indicarle a la función que elabore un ranking de los mejores modelos en función de su *Índice de Ajuste (IOA)* utilizado como indicador, lo que puede ser de gran utilidad si el número de modelos o configuraciones es muy elevado. Este ranking se consigue dando al comando ***rank.name*** el mismo valor que al comando *type*.

Ej. **modStats (datosconmod, mod="o3mod", obs="o3", type="modelo", rank.name="modelo")** , va a seleccionar los datos de ozono del modelo (o3mod) , para calcular los estadísticos obtenidos frente a los datos de ozono reales (o3), teniendo en cuenta que el modelo corrió en el tiempo utilizando tres configuraciones básicas distintas, según queda caracterizado en el campo (modelo) de nuestro conjunto de datos. El resultado en nuestra consola de R nos ofrece los datos estadísticos calculados para estas tres configuraciones, ordenando los modelos del mejor (modelo 1) al peor (modelo 2), tal y como se puede ver a continuación.

```
    modelo    n      FAC2        MB      MGE         NMB      NMGE     RMSE
1 Modelo 1 4854 0.8388958 -3.964823 12.18557 -0.06615625 0.2033260 17.02517
3 Modelo 3 5212 0.8161934 -9.443592 13.92517 -0.17975706 0.2650631 18.13421
2 Modelo 2 4573 0.8784168 -8.216051 14.21080 -0.11954488 0.2067695 18.26780
          r       IOA
1 0.9057127 0.8035025
3 0.9157546 0.7822444
2 0.8998785 0.7716784
```

Tal y como podemos comprobar por el ejemplo adjunto, al ejecutar la función *modStats* , junto con los comandos básicos descritos , R devuelve a la consola los siguientes resultados estadísticos:

- ***n***, o número total de pares de datos completos (modelo—real) de que consta la muestra utilizada para el cálculo de los estadísticos.
- ***FAC2***, predicción fraccionada, o factor de predicciones dentro de un factor de dos de las observaciones.
- ***MB***, el sesgo medio, nos proporciona información útil sobre la tendencia del modelo a subestimar o sobreestimar el valor de un parámetro sobre el valor real, cuantificando así el error sistemático del modelo. Recomendaciones de la US.EPA establecen un sesgo recomendado medio (BIAS) de entre ± 5 y ± 15%.
- ***RMSE***, o error cuadrático medio, que nos da la medida del promedio de las diferencias entre los valores pronosticados y los observados, aportando información sobre la precisión del modelo.

- ***NMB***, el sesgo medio normalizado, que se utiliza para tener en cuenta el peso del error sistemático encontrado frente al valor de la variable de medida.
- ***MGE***, o error relativo normalizado, que es el cociente entre el error absoluto y el valor exacto de cada medida en su media aritmética, permitiendo así comprobar la existencia y entidad de errores aleatorios.
- ***NMGE***, o error relativo normalizado, que se utiliza para comprobar el pesos del error aleatorio frente a la variable de medición. Recomendaciones de la US.EPA establecen un error relativo de entre el 30 y el 35% con un umbral de 60 ppb.
- ***r***, o coeficiente de correlación de Pearson, que es un índice que mide la relación lineal existente entre dos variables, dando una idea del grado de relación existente entre ambas, de forma que r=1 implica que existe una relación directa perfecta, y r=0 que no se puede comprobar que exista relación lineal entre las variables.
- ***IOA***, conocido como índice de ajuste o concordancia, complementa la información aportada por el resto de estadísticos, al informar con un índice general sobre el comportamiento del modelo al compararlo con datos reales.

La herramienta *modStats* se encuentra actualmente en fase de desarrollo, por lo que probablemente en un futuro incorpore nuevos estadísticos de relevancia para la evaluación de modelos que en la actualidad quedan fuera de la herramienta.

Por otro lado, además de las funciones matemáticas y estadísticas vistas hasta el momento, *Openair* dispone de tres funciones gráficas adicionales que permiten analizar y evaluar gráficamente los modelos, proporcionando así una herramienta fundamental de apoyo para el trabajo de evaluación de modelos de predicción de la calidad del aire. Las gráficas proporcionadas por *Openair* se basan en el *Diagrama de Taylor* y en la *función de cuantiles condicionales*, derivada de la *Gráfica Q-Q*, tal y como veremos a continuación con más detalle.

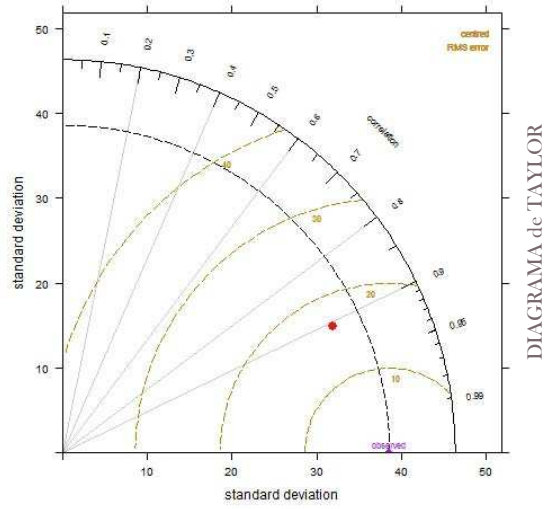
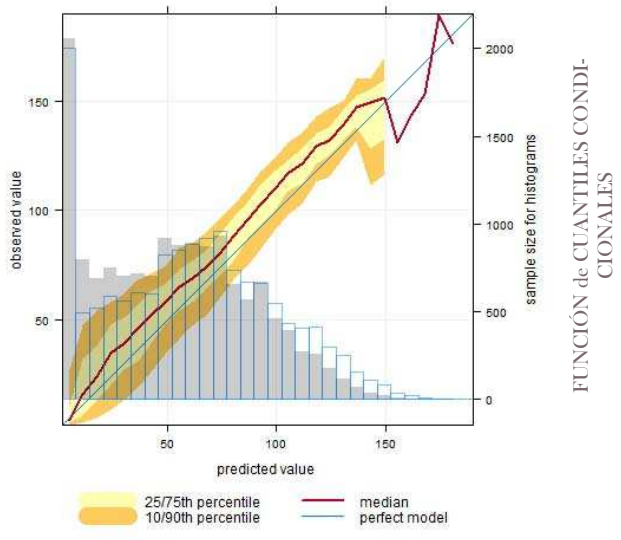


DIAGRAMA de TAYLOR



FUNCIÓN de CUANTILES CONDICIONALES

» El Diagrama de Taylor: *TaylorDiagram*

Una de las herramientas de mayor utilidad para evaluar el rendimiento de un modelo es el conocido como Diagrama de Taylor. Este diagrama utiliza la ley de cosenos para representar en una sola gráfica cómo varían simultáneamente los tres estadísticos más representativos del rendimiento de un modelo, como son:

a) El error cuadrático medio, que matemáticamente es la raíz cuadrada de la media aritmética de los cuadrados de los errores calculados, y que permite obtener una idea de la media del error detectado para el modelo, en la unidad de medida original, y sin tener en cuenta los efecto del signo del error, lo cual resulta muy útil para comprobar la precisión del modelo.

b) La desviación estándar, que permite comprobar cuál es la variabilidad existente en ambas muestras de datos, pudiendo ver además si dicha variabilidad se conserva o varía en el modelo respecto a lo observado para los datos reales.

c) El Coeficiente de Correlación de Pearson, r, que muestra cómo de estrecha es la relación lineal existente entre los pares de datos formados por el modelo y las determinaciones reales.

Gráficamente los datos reales y del modelo, junto con los datos estadísticos comentados, se representan en el Diagrama de Taylor según se muestra en el siguiente ejemplo explicativo.

a) En la parte inferior de la gráfica se representa el valor real o valor observado, correspondiéndose con el valor cero del error cuadrático medio, la unidad para el coeficiente de correlación de Pearson, y la desviación estándar que se corresponda al conjunto de datos reales, que en nuestro ejemplo será de 38,6, que es el valor que finalmente adquiere el punto de datos observados en el Eje X.

b) Los valores del modelo o modelos evaluados se representan con puntos de distintos colores, dentro del Diagrama de Taylor, de forma que a más próximo que se encuentre el punto del modelo a la ubicación del punto de observación, dicho modelo presenta una mayor exactitud, precisión y reproducibilidad con respecto a los datos observados.

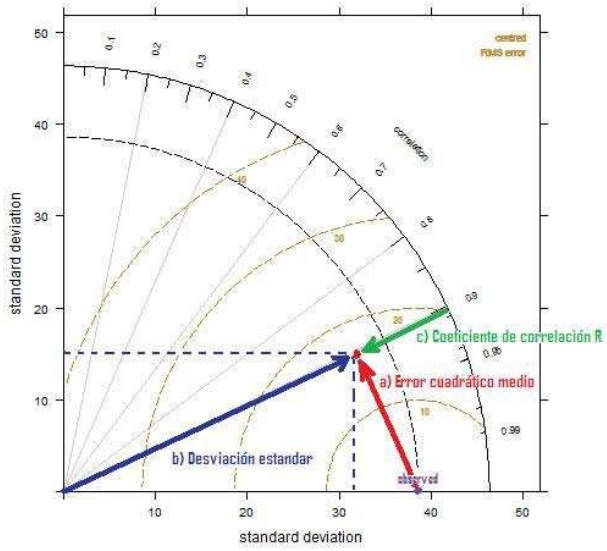
   A efectos de interpretación, se debe tener en cuenta que el punto que representa los datos del modelo se ubica en la gráfica atendiendo a los estadísticos calculados, tal y como se detalla en el ejemplo gráfico explicativo:

   - El Error Cuadrático Medio, que se representa como círculos concéntricos elaborados a partir del punto de observación (valor real), y que se corresponde con la distancia radial desde el punto de observación hasta el punto del modelo.

   - La Desviación Estándar de los datos del modelo, que se corresponde con la distancia radial desde el origen de la gráfica (cruce de los ejes x e y) hasta el punto del modelo (que en nuestro ejemplo sería de 35,26).

   - El Coeficiente de Correlación de Pearson, r, que se representa con un escalado semicircular, frente al cual se ubica el punto del modelo, que adopta como punto de origen radial el origen de la gráfica. De esta forma, cuanto más cercano se encuentre el punto del modelo al eje X más fuerte será la correlación existente entre los datos.

Ej. **TaylorDiagram (datoscontmod, obs="o3", mod="o3mod")** , donde vamos a confrontar los datos de ozono que arroja un modelo frente a los que disponemos de nuestra estación de control en continuo. Sobre la gráfica resultante se ha añadido en esta ocasión un esquema que muestra la interpretación de los estadísticos vista anteriormente.



La Función *TaylorDiagram* dispone, al igual que otras funciones gráficas de *Openair*, de comandos adicionales que incrementan su flexibilidad y añaden utilidades que pueden resultar de interés para el estudio realizado del modelo.

Variaciones sobre la función:

1-. Campo de datos que se corresponde con los datos reales: ***obs***

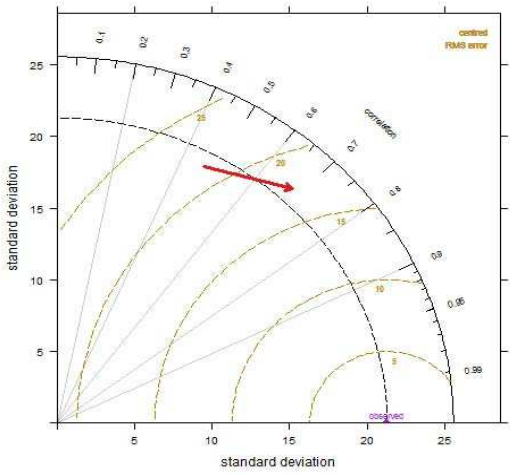
Una vez indicado el conjunto de datos a la función, es necesario señalar a la misma qué campo de dicho conjunto dispone de los datos reales u observados, frente a los cuales se desea contrastar los datos del modelo. Esto se consigue asignando el nombre del campo al comando obs.

2-. Campo de datos que se corresponde con los datos del modelo: ***mod***

Frente a los datos reales, fijados con el comando anterior, se deben disponer los datos correspondientes al modelo que se hayan registrado, indicando al comando *mod* el campo del conjunto donde se encuentran dichos datos.

Destacar en este punto, que el comando *mod* admite la inclusión de hasta dos campos distintos, representando el resultado mediante un flecha que va del primer campo al segundo. Este aspecto es de gran utilidad cuando queremos conocer cómo corre un modelo en comparación con otro, asumiendo los mismos datos de entrada y criterios de funcionamiento, o las diferencias existentes en aquellos casos en los que un mismo modelo genera varias predicciones en función de determinados criterios de revisión y/o corrección.

Ej. **TaylorDiagram (datoscontmod, obs="no2", mod=c ("no2mod", "no2cor"))** , donde representamos en la gráfica los resultados brutos que arroja el modelo para la predicción del NO2 (no2mod), presentados como origen de la flecha, frente a los resultados una vez revisados y corregidos (no2cor), presentados como la punta de flecha, de forma que podemos ver cómo el resultado del modelo una vez corregido mejora considerablemente en sus estadísticos.



3-. Representar los resultados obtenidos para distintos modelos: ***group***

El trabajo con modelos predictivos de la calidad del aire no es, por lo general, un trabajo que se consiga desarrollar en un solo día, para un único modelo, o seleccionando una configuración acertada del mismo a la primera, por lo que es más que probable que finalmente dispongamos de datos de predicción de varios modelos en distintos periodos temporales, o del mismo modelo, que se habría ejecutado con distintas configuraciones o datos de entrada.

La función *TaylorDiagram* permite agrupar todos estos modelos dentro de la misma representación gráfica, asignando a cada uno de ellos un punto con distinto color, de forma que se puede evaluar rápidamente cuál de los modelos presenta una mejor combinación de sus variables estadísticas.

La agrupación de los modelos requiere, sin embargo, que un campo del conjunto de datos identifique el modelo al que pertenecen los distintos datos del campo del modelo, tal y como ya vimos para la instrucción *modStats*. Esto requerirá seguramente del pretratamiento de nuestro conjunto de datos para asignar a cada periodo su modelo. Cabe recordar en este sentido que, herramientas de *Openair* como *splitByDate*, específicamente diseñadas para este tipo de trabajos, nos permiten realizarlo de una forma cómoda y sencilla.

4-. Dividir la gráfica desagregando los datos: ***type***

Como en otras gráficas de *Openair*, el comando *type* puede dividir la gráfica desagregando los datos del modelo o modelos a evaluar en función de distintos periodos temporales de interés como *"weekday"*, *"weekend"*, *"season"*, etc, o incluso en función de otros campos del propio conjunto de datos.

El comando *type* puede servir también para mostrar el resultado de diversos modelos en gráficas separadas, lo que se consigue al seleccionar el campo descriptivo del modelo (el mismo que se seleccionaba con el comando *group*).

5-. Normalizar los estadísticos calculados: ***normalise***

En ocasiones, si queremos comparar gráficas de distintos parámetros contaminantes para el mismo o distintos modelos, es preferible que los estadísticos que vienen expresados en unidades de concentración (como son la desviación estándar y el error cuadrático medio) queden normalizados y sean comparables. En el caso del Diagrama de Taylor la normalización se produce dividiendo el error cuadrático medio y la desviación estándar por la desviación estándar de los datos reales. Esto hace que el punto del dato observado (datos reales) se desplace a la unidad dentro del Eje X, y el resto de puntos de los modelos representados queden referidos a dicha unidad.

6-. Cambiar los colores de representación de la gráfica: ***cols***, ***rms.col***, ***cor.col***

La función *TaylorDiagram* dispone de diversas combinaciones predeterminadas para colorear la representación de los estadísticos de los modelos mediante el comando ***cols***, entre las que podemos destacar *"Accent", "Dark2", "Paried", "Pastel1", "Pastel2", "Set1", "Set2", y "Set3"*. En todo caso, el usuario podrá modificar los colores de la representación gráfica, para personalizarlos a su gusto, tal y como ya se ha visto en anteriores ocasiones, disponiendo los colores que considere más adecuados.

Además, se puede personalizar también el color utilizado para la representación de los círculos concéntricos y el texto del error cuadrático medio, mediante el comando ***rms.col***, o el color utilizado para representar la escala de correlación y su texto, con el comando ***cor.col***.

7-. Cambiar el grosor de las flechas: ***arrow.lwd***

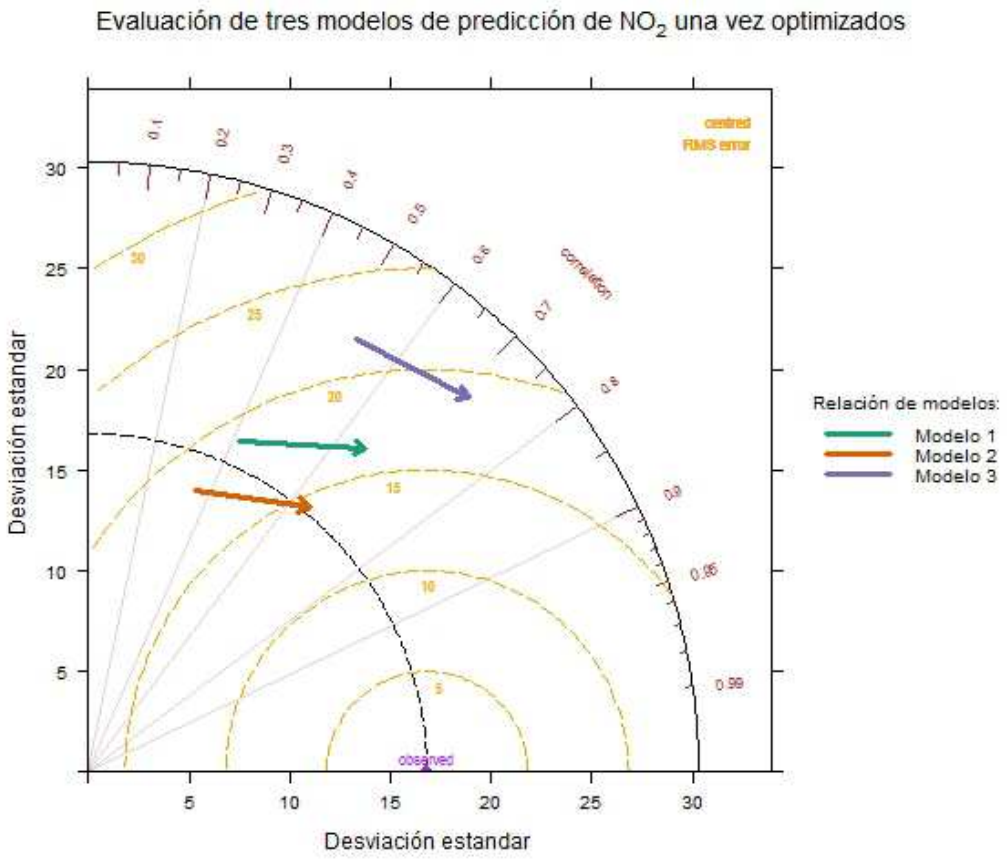
Cuando representamos dos campos de datos con el comando *mod* (utilizado para seleccionar los datos del modelo) tal y como hemos visto anteriormente, la gráfica adopta la forma de flechas para representar el campo de origen y el campo final.

Con el comando *arrow.lwd* es posible personalizar el grosor de la flecha representada, asignándole el número correspondiente. Por defecto la gráfica presenta un grosor de ***arrow.lwd=3***.

8-. Dibujar encabezados a las gráficas divididas: ***strip***

Cuando dividimos la gráfica *TaylorDiagram* con el comando *type*, automáticamente se presenta una banda superior con la que se identificada cada una de las gráficas elaboradas en función del parámetro seleccionado para *type*. Con el comando lógico strip se permite al usuario desactivar este encabezado, ***strip=FALSE***, que viene activado por defecto.

La Función *TaylorDiagram* permite además las opciones clásicas de configuración de la leyenda, mediante el comando *key* y sus distintas variantes, así como las clásicas opciones de configuración de los textos en los ejes *xlab* o *ylab*, entre otras, tal y como se puede observar a continuación en el ejemplo gráfico final.




```
> TaylorDiagram (datoscontmod, obs="no2", mod=c("no2", "no2cor"), group="modelo", cols="Dark2", rms.col="orange", cor.col="brown", xlab="Desviación estandar", ylab="Desviación estandar", main="Evaluación de tres modelos de NO2 una vez optimizados", key.tittle="Relación de modelos")
```

» La Función de Cuantiles Condicionales: *conditionalQuantile*

La función de Cuantiles Condicionales utilizada por *Openair* para la evaluación de modelos de calidad del aire, deriva de la función gráfica Q-Q (del inglés Quantile-Quantile), muy utilizada en estadística también para el diagnóstico de las diferencias entre una población de datos y otra.

No obstante, mientras que la *gráfica Q-Q* compara la distribución de la probabilidad de dos poblaciones o conjuntos de datos, comprobando si la distribución de las muestras es la misma o varía para alguno de sus cuantiles, saliéndose de la línea recta, la función de cuantiles condicionales no considera de forma separada ambas poblaciones de datos, sino que analiza los resultados reales frente a los intervalos de predicción, vinculando unos con otros.

De esta forma, la función de Cuantiles Condicionales será capaz de encontrar discrepancias en conjuntos de datos que para la función *gráfica Q-Q* son indetectables, por no afectar a la distribución de datos (como por ejemplo aquellos casos en los que exista un desplazamiento en las horas, y un conjunto de datos se encuentre en GMT, mientras que el otro se encuentra en horario local). Así, con la función *conditionalQuantile Openair* se asegura que la evaluación del modelo sea completa y adecuada.

El modo de funcionamiento detallado de la función *conditionalQuantile* es muy sencillo:

En primer lugar, divide los datos de la predicción en intervalos fijos abarcando todo el rango de datos de dicha predicción, como si elaborase un histograma (que de hecho es lo que aparece en la gráfica final para ambos conjuntos de datos).

Para cada uno de los intervalos elaborados para los datos de predicción, la función busca la correspondencia entre los valores reales, o dicho de otra forma, busca el valor real que se corresponde con cada uno de los valores predichos que forman el intervalo que corresponda.

Finalmente, para los valores reales identificados en cada intervalo calcula la mediana y los datos que se encuentran entre los percentiles 25/75 y 10/90, para luego representarlos en la gráfica, frente a una línea recta que representará el ajuste perfecto del modelo.

Para el desarrollo de la función se dispone de distintos comandos de configuración, según vemos a continuación.

Variaciones sobre la función:

1-. Campo de datos que se corresponde con los datos reales: ***obs***

Una vez indicado el conjunto de datos a la función, es necesario señalar a la misma qué campo de dicho conjunto dispone de los datos reales u observados, frente a los cuales se desea contrastar los datos del modelo. Esto se consigue asignando el nombre del campo al comando *obs*.

2-. Campo de datos que se corresponde con los datos del modelo: ***mod***

Frente a los datos reales, fijados con el comando anterior, se deben disponer los datos correspondientes al modelo que se hayan registrado, indicando al comando *mod* el campo del conjunto donde se encuentran dichos datos. En esta ocasión, al contrario de lo que ocurría con la función *TaylorDiagram*, el comando *mod* no admitirá la inclusión de más de un campo, por lo que deberemos seleccionar aquellos campos correspondientes al modelo que queremos evaluar finalmente.

3-. Dividir la gráfica desagregando los datos: ***type***

Como en casi todas las instrucciones gráficas de Openair, la instrucción conditionalQuantile dispone del comando type para desagregar los datos, elaborar las estadísticas correspondientes y representarlos en distintas gráficas. Mediante el comando type se puede dividir la gráfica en función de cualquiera de los parámetros contemplados por la función cutData.

También es posible, como hemos visto en múltiples ocasiones, asignarle al comando type cualquier otro campo del conjunto de datos, lo que puede resultar de enorme utilidad si, por ejemplo, queremos observar en distintas gráficas el resultados obtenido para diversos modelos.

4-. Número de intervalos a utilizar: ***bins***

La función permite personalizar el número de intervalos a utilizar para los valores predichos, de forma que se actúe sobre el propio cálculo de los cuantiles, incrementando y/o disminuyendo la precisión en el cálculo de la función. Al comando *bins* habrá que indicarle así el número final de intervalos deseados.

5-. Número mínimo de datos para calcular los percentiles: ***min.bin***

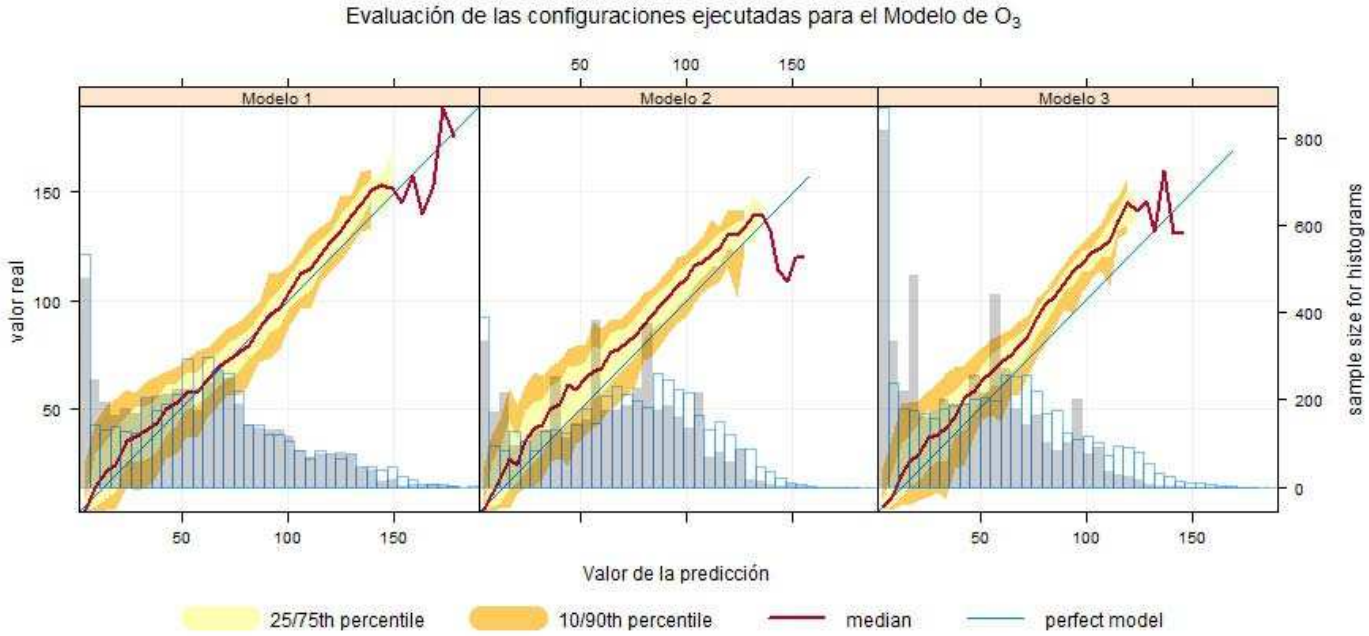
La gráfica calcula siempre la mediana de los datos observados en cada uno de los intervalos. Sin embargo, dicho cálculo puede carecer de representatividad para los valores de los percentiles 25/75 y 10/90 si el número de datos no es suficiente, motivo por el que habitualmente podemos observar cómo en los últimos intervalos del diagrama (donde existe un menor número de datos) es habitual no encontrar representados dichos percentiles.

El usuario podrá personalizar el número mínimo de datos que considera representativo, en primer lugar para la estimación del percentil 25/75, y en segundo lugar para el percentil 10/90, para lo cual deberá asignar dos números al comando min.bin, según se muestra a continuación: ***min.bin=c(20,20)***.

6-. Personalizar los colores de la gráfica: ***col***

La función gráfica conditionalQuantile no tiene previstos en la actualidad ningún paquete predeterminado de configuración de colores, por lo que de querer personalizarlos, el usuario deberá introducir manualmente, mediante el comando col, hasta un mínimo de cinco colores a utilizar en la gráfica, en función de la combinación más deseada.

La Función *conditionalQuantile* permite, además de lo visto hasta el momento, personalizar la leyenda de la gráfica interviniendo en su configuración mediante los comandos key.columns y key.position, o incluir y modificar los títulos de sus ejes mediante los comandos xlab e ylab, tal y como se puede ver en el siguiente ejemplo gráfico final.




```
> conditionalQuantile ( datoscontmod, mod="o3mod", obs="o3", type="modelo", bins=40, xlab="valor de la predicción", ylab="Valor real", key.columns=4, main="Evaluación de las configuraciones ejecutadas para el Modelo de O3")
```

» La Función de Cuantiles Condicionales Ampliados: *conditionalEval*

Esta función fue introducida en la última versión del paquete *Openair* evaluada por el autor de este manual, aproximadamente en mayo de 2012, y permite completar la evaluación de modelos realizada hasta el momento por las funciones vistas en este apartado.

La función *conditionalEval* viene de hecho a mejorar la función de Cuantiles Condicionales al considerar cómo pueden variar otros parámetros en los mismos intervalos, lo cual puede ser muy útil a la hora de evaluar la intervención de terceros parámetros en la predicción realizada por un modelo, permitiendo así determinar las razones que llevan a que un modelo sea mejor o peor en determinadas situaciones.

Así, el funcionamiento de la función *conditionalEval* será igual que en el caso de la función *conditionalQuantile*, salvo que en este caso se adjuntará una gráfica adicional, utilizada para la evaluación del parámetro o parámetros adicionales proporcionados por el usuario. De hecho, el *quid* de la función conditionalEval está realmente en esta gráfica adicional y en la forma en la que esta se puede representar en función del valor que el usuario otorgue al comando *statistic*, tal y como podremos ver más adelante.

Para el desarrollo de la función se dispone de distintos comandos de configuración, según vemos a continuación.

Variaciones sobre la función:

1-. Campo de datos que se corresponde con los datos reales: ***obs***

Una vez indicado el conjunto de datos a la función, es necesario señalar a la misma qué campo de dicho conjunto dispone de los datos reales u observados, frente a los cuales se desea contrastar los datos del modelo. Esto se consigue asignando el nombre del campo al comando *obs*.

2-. Campo de datos que se corresponde con los datos del modelo: ***mod***

Frente a los datos reales, fijados con el comando anterior, se deben disponer los datos correspondientes al modelo que se hayan registrado, indicando al comando *mod* el campo del conjunto donde se encuentran dichos datos, de la misma forma que sucedía en *conditionalQuantile*.

3-. Otros parámetros adicionales observados: ***var.obs***

Otra serie de variables observadas para un parámetro adicional que se desee contrastar con el parámetro principal y para las cuales se calcularán los estadísticos que el usuario demande con el comando *statistic* que veremos más adelante. En este caso, además, el usuario podrá seleccionar varios parámetros a utilizar con la función *conditionalEval*.

4-. Otras variables correspondientes al modelo: ***var.mod***

Frente al parámetro o parámetros observados, que el usuario haya establecido en el comando *var.obs*, se deberán contraponer el parámetro o parámetros que se correspondan con el modelo, adjuntándolos al comando var.mod.

5-. Dividir la gráfica desagregando los datos: ***type***

Como en casi todas las instrucciones gráficas de Openair, la instrucción conditionalQuantile dispone del comando type para desagregar los datos, elaborar las estadísticas correspondientes y representarlos en distintas gráficas. Mediante el comando type se puede dividir la gráfica en función de cualquiera de los parámetros contemplados por la función cutData.

6-. Número de intervalos a utilizar: ***bins***

La función permite personalizar el número de intervalos a utilizar para los valores predichos, de forma que se actúe sobre el propio cálculo de los cuantiles, incrementando y/o disminuyendo la precisión en el cálculo de la función. Al comando *bins* habrá que indicarle así el número final de intervalos deseados.

7-. Estadística a graficar en la gráfica adicional que acompaña a la de cuantiles condicionales: ***statistic***

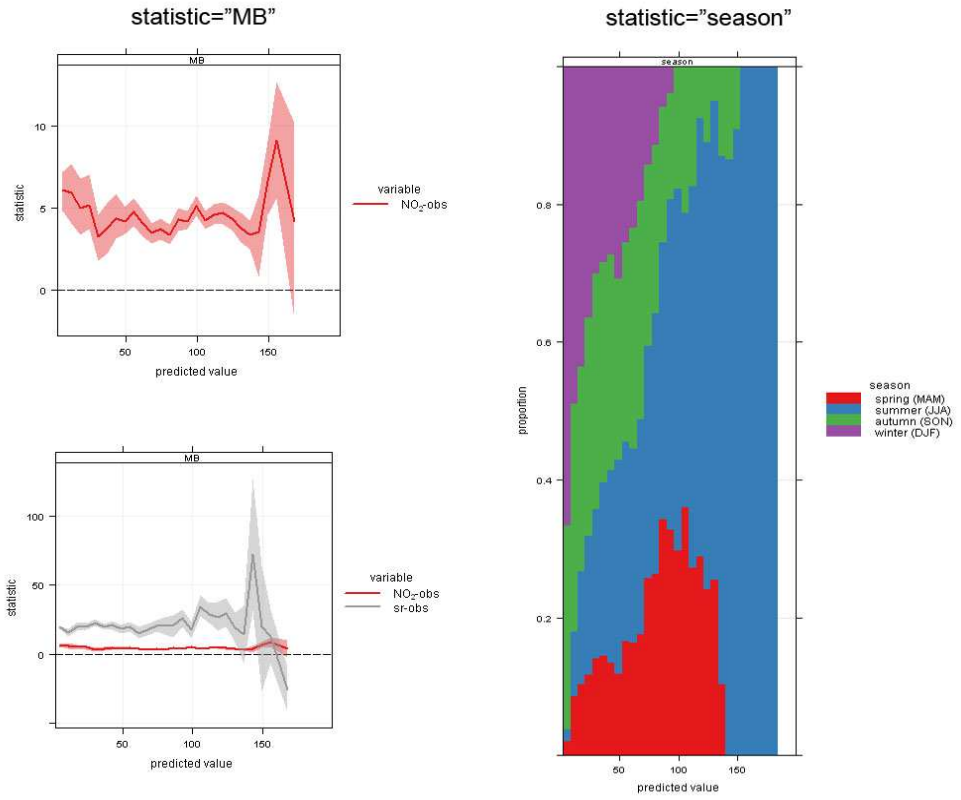
Tal y como se ha comentado antes, el comando *statistic* es la clave en el desarrollo de esta función gráfica, proporcionando gran parte del valor añadido de la misma. Este comando establece la forma en la que se va a representar la gráfica adicional de forma que se pueden suceder dos casuísticas distintas:

a) Que proporcionemos al comando *statistics* como variable cualquiera de los parámetros estadísticos que contempla la función *aqStat*s vista anteriormente en este capítulo, tales como el sesgo medio *"MB",* el error cuadrático medio *"RMSE",* etc. En este caso la función representará en la gráfica adicional la evolución del estadístico seleccionado para el dato observado adicional frente al dato del modelo adicional, ambos proporcionados por el usuario mediante los comandos *var.obs* y *var.mod*.

a)El resultado obtenido se mostrará como una gráfica para el parámetro adicional seleccionado por el usuario, tal y como se muestra en el siguiente ejemplo para el sesgo medio "MB" y el NO2. Si los parámetros adicionales seleccionados son varios el resultado se mostrará en la misma gráfica, tal y como se muestra en el mismo ejemplo para el NO2 y la radiación solar, lo que es importante tener en cuenta dado que ambos parámetros compartirán escala.

b) Que proporcionemos al comando *statistics* una variable de corte de datos, como las proporcionadas al comando *type* o a la instrucción *cutData* vistas hasta ahora en multitud de ocasiones, pudiendo ser una variable de temporal de corte, tipo *"season",* o un parámetro del propio conjunto de datos. Al hacerlo así la gráfica adicional representará, para cada intervalo del modelo, la proporción de las predicciones que se corresponden con la variable marcada por el comando statistics, quedando fuera de la representación, eso sí, las variables adicionales seleccionadas por var.obs y var.mod.

Por ejemplo, utilizar una variable tipo "season" para determinar la distribución de los valores del ozono, tal y como se muestra en el siguiente ejemplo, representará para cada intervalo del modelo de predicción de ozono (valores del eje X) la proporción de las predicciones que se realizan en primavera, verano, otoño o invierno. Por lógica, tal y como podemos ver en el ejemplo, las predicciones de valores más altos se corresponden con los periodos de mayor radiación solar (verano y otoño) expresado en tanto por 1, lo cual puede resultar muy útil para comprobar la fiabilidad en la predicción del modelo en distintos periodos, al contrastar la distribución de las predicciones frente a la evolución del gráfico de cuantiles condicionales.



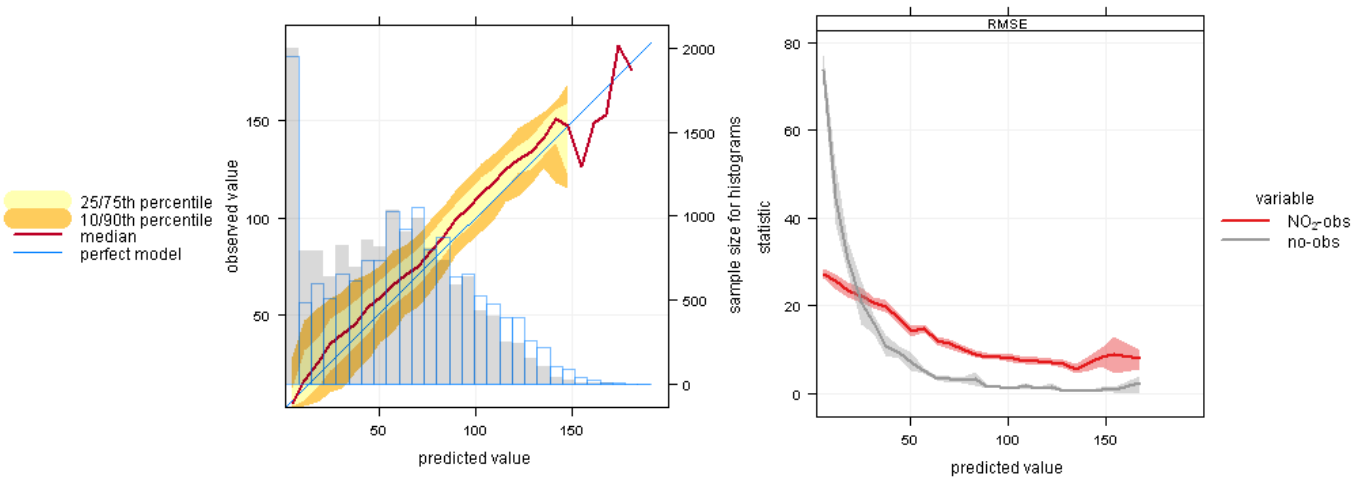
8-. Personalizar los colores de la gráfica: ***col***

La función gráfica *conditionalEval*, al igual que su original *conditionalQuantile*, no tiene previstos en la actualidad ningún paquete predeterminado de configuración de colores, por lo que de querer personalizarlos, el usuario deberá introducir manualmente, mediante el comando col, hasta un mínimo de cinco colores a utilizar en la gráfica, en función de la combinación más deseada.

Sin embargo, la función *conditionalEval* permite asignar colores a la gráfica adicional mediante el comando ***col.var***, al que le podremos indicar los colores que se requieren para representar las líneas en la gráfica adjunta.

9-. Personalizar los nombres de las variables adicionales introducidas: ***var.names***

Se puede indicar a la función los nombres que deseamos que se utilicen para las variables adicionales utilizadas, debiendo indicar al comando *var.names* los nombres que se desean para la variable observada y la variable correspondiente al modelo, sucesivamente.




```
> conditionalEval ( datoscontmod, mod="o3-mod", obs="o3-obs", var.obs="no2-obs", var.mod="no2-mod" , bins=30,
statistic="RMSE")
```