Distributed lag non-linear models in R: the package dlnm

Antonio Gasparrini and Ben Armstrong
London School of Hygiene and Tropical Medicine, UK

dlnm version 1.2.2 , 2010-04-28

Contents

[1]This document is included as a vignette (a LaTeX document created using the R function `Sweave()`) of the package dlnm. It is automatically dowloaded together with the package and can be accessed through R typing *`vignette("dlnmOverview")`*.

1 Preamble

The R package dlnm provides some facilities to run *distributed lag non-linear models* (DLNM's), a modelling framework to describe simultaneously non-linear and delayed effects between predictors and an outcome in time-series data.

The aim of this document is to provide an overview of the capabilities of the package, together with extensive examples of application with real data. Some information on installation procedures and on the data included in the package are given in Section 2. The theory underlying the DLNM methodology is briefly illustrated in Section 3, while the functions included in the package are described in Section 4. Some examples of applications are provided in Section 5: users mainly interested in the application can skip the previous Sections and and start with these examples. Finally, Section 6 offers some conclusions.

The DLNM's methodology, together with a thorough algebraical development, has been previously described in Gasparrini et al. (2010). This framework was originally conceived and proposed to investigate the health effect of temperature in Armstrong (2006).

Type `citation("dlnm")` in R to cite the dlnm package. A list of changes included in the current and previous versions can be found typing `file.show(system.file("ChangeLog", package = "dlnm"))`.

Please send comments or suggestions and report bugs to `antonio.gasparrini@lshtm.ac.uk`.

2 Installation and data

2.1 Installing the package dlnm

The dlnm package is installed in the standard way for CRAN packages, for example using the `install.packages()` function or directly through the menu in R (from version 2.9.0 onwards), clicking on *Packages* and then on *Install package(s)...*. The package can be alternatively installed using the .zip file containing the binaries, via *Packages* and then *Install package(s) from local zip files...*.

The functionalities of dlnm depend on other packages whose commands are called to specify the dlnm functions. This hierarchy is ruled by the field *Imports* of the file `description` included in the package. The functions are imported from the packages splines (functions `ns()` and `bs()`) and tsModel (function `Lag()`). While splines is present in the basic installation of R, the package tsModel is automatically downloaded if dlnm is installed through R using the CRAN, but must be independently installed if a .zip file is used.

2.2 Data

Until the version 0.4.1, the package dlnm did not contain any data, and used the datasets stored in the package NMMAPSlite.

In this version the package contains its own dataset `chicagoNMMAPS`, with daily mortality (all causes, CVD, respiratory), weather (temperature, dew point temperature, relative humidity) and pollution data (PM10 and ozone) for Chicago in the period 1987-2000. The data were assembled from publicly available data sources as part of the National Morbidity, Mortality, and Air Pollution Study (NMMAPS) sponsored by the Health Effects Institute (Samet et al., 2000a,b). They are downloadable from the Internet-based Health and Air Pollution Surveillance System (iHAPSS) website (`http://www.ihapss.jhsph.edu`) or through the packages NMMAPSdata or NMMAPSlite. See `?chicagoNMMAPS` for additional information on the variables included.

3 Distributed lag non-linear models (DLNM's)

The aim of this Section is to provide a methodological summary of the DLNM framework. A detailed description of this methodology and the algebraical development have been published elsewhere (Armstrong, 2006; Gasparrini et al., 2010).

3.1 The issue

The main purpose of a statistical regression model is to define the relationship between a predictor and an outcome, and then to estimate the related effect. A further complexity arises when the dependency shows some *delayed effects*: in this case, a specific occurrence of the predictor (let us call it an *exposure event*) affects the outcome for a certain period in the future. This step requires the definition of more complex models to characterize the association, specifying the temporal structure of the dependency. The main feature of DLNM's is their *bi-dimensional* structure: the model describes simultaneously the potentially non-linear relationship in the space of the predictor and along the new temporal dimension.

3.2 The concept of basis

Several different methods have been adopted to specify non-linear effects in a regression models. A simple solution is to generate strata variables, applying specific cut-off points along the range of the predictor in order to define specific intervals, and then specifying new variables through a dummy parameterization.

Other types of manipulations of the original variable are applied when there are specific assumptions on the shape of the relationship, for example when the effect is likely to exist and be linear only above or below a specific threshold (*hockey-stick* model). An extension of this model assumes two distinct linear effects below a first threshold and above a second threshold, with a null effect in between them.

An alternative to the strata or threshold approaches is to include in the model some terms allowing a true non-linear relationship, describing a smooth curve between the predictor and the outcome. The traditional methods include a quadratic term or higher degree polynomials. Recently, spline functions have been favoured, especially through a natural cubic parameterization.

A generalization may be provided assuming that all the approaches above imply the choice of a *basis*, defined as a *space of functions* used to define the relationship (Wood, 2006). The choice of the basis defines the related *basis functions*, completely known transformations of the original predictor generating a new set of transformed variables, defined *basis variables*. Independently from the basis chosen, the final result will be a matrix of transformed variables which can be included in the design matrix of a model in order to estimate the related parameters. The choice of different bases leads to the specification of different matrices, but the mechanism is common.

3.3 Delayed effect: DLM's

In the specific context of time series analysis, given the ordered series of the predictor values, a delayed (or lagged) effect is present when the outcome in a specific time is influenced by the level of the predictor in previous times, up to a maximum lag. Therefore, the presence of delayed effects requires to take into account the *time dimension* of the relationship, specifying the additional virtual dimension of the *lags*.

A very simple model to deal with delayed effects considers the moving average of the predictor up to a certain lag, specifying a transformed predictor which is the average of the values in that specific lag period. Although simple, this model is limited if the purpose is to assess the temporal structure of the effects.

These limitations have been addressed using a more elegant approach based on distributed lag models (DLM's). The main advantage of this method is the possibility to depict a detailed description of the time-course of the relationship. Originally developed in econometrics (Almon, 1965), this method has recently been used to quantify the health effect in studies on environmental factors (Braga et al., 2001; Schwartz, 2001; Welty and Zeger, 2005; Zanobetti et al., 2000).

In the basic formulation, a DLM is fitted by the inclusion of a parameter for each lagged predictor occurrence. An estimate of the overall effect is given by the sum of the single lag effects upon the whole lag period considered (Hajat et al., 2005; Schwartz, 2000).

This *unconstrained* version of DLM does not require any assumption on the shape of the effect along lags, and consequently on the relationship between parameters. In order to define a more parsimonious model, it is possible to specify some assumptions on the shape of the distributed effect, applying some constraint. The simplest solution is to group the lags in different strata (Pattenden et al., 2003; Welty and Zeger, 2005), while a more complex option is to force the curve along lags to follow a specific smooth function, for example polynomials (Baccini et al., 2008; Schwartz et al., 2004; Zanobetti and Schwartz, 2008) or splines (Zanobetti et al., 2000).

Following the general approach used in Section 3.2, it may be shown that all the different DLM's above can be described by the same equation, where different models are specified through different basis functions to be applied to the vector of lags, building a new basis matrix (see Gasparrini et al., 2010, Eq. 4). Again, the choice of different bases generates different matrices, but the mechanism is general.

3.4 The extension to DLNM's

A general approach to specify non-linear but un-lagged effects has been introduced in Section 3.2, while the methods to define distributed lag functions for simple linear effects have been presented in Section 3.3. An obvious extensions is to combine these approaches to define distributed lag non-linear models (DLNM's), a family of models which can deal at the same time with non-linear and delayed effects.

The different issues of non-linearity and delayed effects share a common feature: in both cases the solution is to choose a basis to describe the shape of the relationship in the relative dimension. This step leads to the concept of *cross-basis*: following the idea of basis in 3.2, a cross-basis can be imagined as a bi-dimensional space of functions describing on the same time the shape of the relationship and the distributed lag effects. The algebraic notation to define the cross-basis and then the DLNM can be quite complex, involving tensor products of 3-dimensional arrays, and has been presented elsewhere (Gasparrini et al., 2010, Section 4.2). Nonetheless, the basic concept is straightforward: choosing a cross-basis amounts to choosing two independent set of basis functions, which will be combined to generate the specific cross-basis functions. The DLM's described in 3.3 can be considered as special cases of DLNM's with a simple linear function in the dimension of the predictor.

The result of a DLNM can be interpreted building a grid of predictions for each lag and for suitable values of the predictor, using three dimensional plots to provide an overall picture of the effects varying along the two dimensions. In addition, it is possible to compute the effects for single predictor levels or lags, simply cutting a "slice" of the grid along specific values of predictor or lags, respectively. Finally, an estimate of the overall effect can be computed by summing all the contributions at different lags. The effects are usually reported versus a reference value of the predictor, centering the basis functions for this space to their corresponding transformed values (Cao et al., 2006).

The choice of the two set of basis functions for each space is perfectly independent, and should be based on a-priori assumptions or on a compromise between complexity and generalizability. Linear, threshold, strata, polynomial or splines functions can be used to define the relationship along the space of predictor, while unconstrained, strata, polynomial or splines functions can be applied to specify the shape along lags.

4 The functions in the package dlnm

The functions included in the package can be used to complete all the steps required to specify and interpret a DLNM. The data are assumed to represent an equally-spaced, complete and ordered series of observations.

The internal functions `mkbasis()` and `mklagbasis()` are called in order to build the basis matrices for the dimension of the predictor and lags, respectively. In concrete terms, they apply a transformation to the vector of predictor and to the vector of lags, and stored the transformed variables in two matrix objects. Several different choices are available, for example splines, polynomials, stratification and threshold parameterization. Details on the basis specification are given in Section 5.1. These two internal commands are called by other functions, and they are not meant to be run by the users. However, they are included in the namespace of the package and therefore made accessible, with the intention to keep the process more transparent and give the opportunity to change or improve them.

The main function in the package dlnm is `crossbasis()`. It calls the internal functions `mkbasis()` and `mklagbasis()` and combines the two basis matrices in order to create the cross-basis matrix which specifies the dependency simultaneously in the two dimensions. The arguments are set to some default values, and can be automatically changed for nonsensical combinations, or set to null if not required. Anyway, meaningless combinations of arguments (for example strata where no observation lies) could lead to collinear variables, with identifiability problems in the model. The user is advised to test the result with the function `summary.crossbasis()`, which provides a summary of the choices made for the two bases and the final cross-basis.

The cross-basis matrix should be included in the model formula of default regression functions. More than one cross-basis matrix can be included in the same formula. The name of the object containing the cross-basis matrix will be used to extract the estimated parameters, and must not match the names of other predictors in the model formula. Several different regression commands can be used: `lm()`, `glm()`, `gam()` (package mgcv), `glm.nb()` (package MASS), `gee()` (package geepack), `clogit()` and `coxph()` (package survival). Other commands may be included in the future.

The function `crosspred()` generates the predicted effects for a set of values of the original predictor, given the applied cross-basis functions and the parameters estimated by the model. It stores them in matrices with specific effects for each combination of predictor values and lags, and in vectors of overall effects (summed up along lags). Cumulative effects may be included, and exponentiated values are returned for model with log or logit link.

Finally, the function `crossplot()` provides some options to visualize the predicted effects.

See the related help pages (for example *`help(crossbasis)`* or *`?crossbasis`*) for the details on the usage and the arguments of these functions.

5 Some examples

This Section provides some examples of the use of the functions included in the dlnm package, described in Section 4. In spite of the specific application on the health effects of air pollution and temperature, these examples are easily generalized to different topics. The results included in this Section are not meant to represent scientific findings, but are reported with the only purpose to illustrate the capabilities of the dlnm package.

First, some simple examples of the internal functions are showed in Section 5.1. Although these commands are not expected to be performed directly by the user, but are commonly called through `crossbasis()`, these codes can shine a light on the process to build the basis functions for the two dimensions (predictor and lags), and clarify the meaning of the arguments of the function `crossbasis()`.

Then, 3 different examples of the application of DLNM's are illustrated in the Sections 5.2-5.4, using the NMMAPS dataset for the city of Chicago in the period 1987-2000 included in the package, which has been described in Section 2.2. These different cases cover most of the functionalities of the package, providing a detailed overview of its capabilities and a basis to perform analyses on this dataset or on other data sources.

The package is assumed to be present in the R library (see Section 2.1) and loaded in the session, together with the data, typing:

```
> library(dlnm)
> data(chicagoNMMAPS)
```

5.1 Examples for `mkbasis()` and `mklagbasis()`

As a first step, we provide an example of the use of the function `mkbasis()`. We build a basis matrix applying the selected basis functions to the vector of integers going form 1 to 5. We leave many of the arguments at their default values, apart from the selection of the degrees of freedom `df`.

```
> basis.var <- mkbasis(1:5, knots=3)
> basis.var

$basis
             b1          b2
[1,] -0.56626284  0.21084190
[2,] -0.20921622 -0.00635585
[3,]  0.00000000  0.00000000
[4,] -0.03716777  0.37894518
[5,] -0.22216593  0.98144395

$type
[1] "ns"

$df
[1] 2

$knots
[1] 3

$bound
[1] 1 5

$int
[1] FALSE
```

```
$cen
[1] TRUE

$cenvalue
[1] 3
```

The result is a simple *"list"* object. The chosen basis is a natural cubic B-splines (default `type="ns"`) with 1 knot and `df=2` (`df` is equal to `length(knots)+1+int` for `type="ns"`). Apart from the fact that the basis variables are centered at `cenvalue=3` (the mean of the predictor values, the default for this argument), the same results could be obtained by the command *`ns(1:5, knots=3)`*. The basis matrix is stored in the object `basis.var$basis`, while the arguments specifying it are included as other objects in the list, and can be called directly (for example, try *`basis.var$knots`*).

Alternative choices may be specified through the following code (results not shown, the user can try to run the commands):

```
> mkbasis(1:5, type="bs", df=4, degree=2)
> mkbasis(1:5, type="lin", cenvalue=4)
```

In the first case the result is a quadratic spline where the number and location of `knots` are chose automatically, and fixed to 2 (`df` is `length(knots)+degree+int` for this `type`) and at equally spaced quantiles, respectively. The second line returns a simple linear function, where the only transformation is the centering at the value of 4.

The function `mklagbasis()` calls `mkbasis()` to create a basis matrix for the space of the lag. The basis functions are applied to the vector `0:maxlag` expressly created by the function. This is an example of application:

```
> mklagbasis(maxlag=5, type="poly", degree=3)

$basis
     b1 b2 b3  b4
lag0  1  0  0   0
lag1  1  1  1   1
lag2  1  2  4   8
lag3  1  3  9  27
lag4  1  4 16  64
lag5  1  5 25 125

$type
[1] "poly"

$df
[1] 4

$degree
[1] 3

$int
[1] TRUE

$maxlag
[1] 5
```

The command specifies a 3rd degree polynomial. Differently from the bases for the space of the predictor build above, this matrix contains an intercept (`int=TRUE` by default), in this case a vector of 1's (see *?crossbasis*), and is never centered. `df` is equal to the degree of the polynomial plus 1 when an intercept is included. In this case, for a polynomial basis, the argument `knots` is not included.

Other examples (results not shown):

```
> mklagbasis(maxlag=5, type="integer")
> mkbasis(1:5, type="dthr", knots=c(2,3))
```

In the first line, the command applies a specific transformation in the space of lags in order to define unconstrained distributed lag effects (see *?crossbasis*), simply returning an identity matrix. The second choice returns a double threshold basis which can be applied to describe linear effects below 2 and above 3, with a null effect in between them.

A basis matrix of `type="strata"` with and without intercept is created by (results not shown):

```
> mklagbasis(maxlag=10, type="strata", knots=c(4,7))
> mklagbasis(maxlag=10, type="strata", knots=c(4,7), int=F)
```

In this case, the intercept is represented by the dummy variable for the first stratum (see *?crossbasis*). The values in `knots` specify the cut-off point for the strata, and represent the lower boundaries for the right-open intervals.

The effect of centering is illustrated below (results not shown):

```
> mkbasis(0:10, type="poly", degree=3)
> mkbasis(0:10, type="poly", degree=3, cen=F)
```

Each basis function is centered on the relative transformation of `cenvalue`, which is placed at the mean of the predictor values by default, or defined by the user.

5.2 Example 1: a simple DLM

In this first example, we specify a simple DLM, assessing the effect of PM_{10} on overall mortality, while adjusting for the effect of temperature. In order to do so, we first build two cross-basis matrices for the two predictors, and then include them in a model formula of a regression function. The effect of PM_{10} is assumed linear in the dimension of the predictor, so, from this point of view, we can define this as a simple DLM even if it estimates also the distributed lag function for temperature, which is included as a non-linear term.

First, we run `crossbasis()` to build the two cross-basis matrices, saving them in two objects. The names of the two objects must be different in order to predict the effects separately for each of them (see *?crosspred*). This is the code:

```
> basis.pm <- crossbasis(chicagoNMMAPS$pm10, vartype="lin", lagtype="poly",
          lagdegree=4, cen=F, maxlag=15)
> basis.temp <- crossbasis(chicagoNMMAPS$temp, vardf=5, lagtype="strata",
          lagknots=1, cenvalue=21, maxlag=3)
```

The function `crossbasis()` calls the two internal functions `mkbasis()` and `mklagbasis()` to build the basis matrices. It passes the arguments with stub `var-` to the former, in order to specify the basis functions for the predictor (in this case `chicagoNMMAPS$pm10` and `chicagoNMMAPS$temp`), and the arguments with stub `lag-` to the latter, specifying the basis functions for the expressly created vector `0:maxlag`. Then it combines the two basis matrices to create the final cross-basis variables included in the objects of class *"crossbasis"* (`basis.pm` and `basis.temp`). As highlighted above, the data are assumed to be composed by equally-spaced, complete and ordered series.

In this case, we assume that the effect of PM_{10} is linear (`vartype="lin"`), while we model the relationship with temperature through a natural cubic spline with 5 degrees of freedom (`vartype="ns"`, chosen by default). In this space, the internal knots (if not provided) are placed by default at equally spaced quantiles, while the boundary knots are located at the range of the observed values, so we need to specify only `vardf`. We did not center PM_{10}, in order to compute the predicted effects versus a reference value of 0 μgr/m^3 (the same results could be obtained setting `cen=TRUE` and `cenvalue=0`). The reference value for temperature is set to 21°C.

The basis for the space of the lags is chosen through the same arguments but with stub `lag-`. We specify the lagged effect of PM_{10} up to 15 days of lag with a 4th degree polynomial function (setting `lagdegree=4`). The delayed effect of temperature are defined by two lag strata (0 and 1-3), assuming the effects as constant within each stratum. The argument `varknots=1` defines the lower boundary of the second interval.

An overview of the specifications for the cross-basis (and the related bases in the two dimensions) is provided by the function `summary.crossbasis`, which calls the attributes of the `crossbasis` object:

```
> summary(basis.pm)

CROSSBASIS FUNCTIONS
observations: 5114
range: -3.049835 , 356.1768
total df: 5
maxlag: 15

BASIS FOR VAR:
type: lin
df: 1

BASIS FOR LAG:
type: poly with degree 4
df: 5
with intercept
```

Now the two `crossbasis` objects can be included in a model formula in order to fit the DLM. In this case we model the effect assuming an overdispersed Poisson distribution, including a smooth function of time with 7 df/year (in order to correct for seasonality and long time trend) and day of the week as factor:

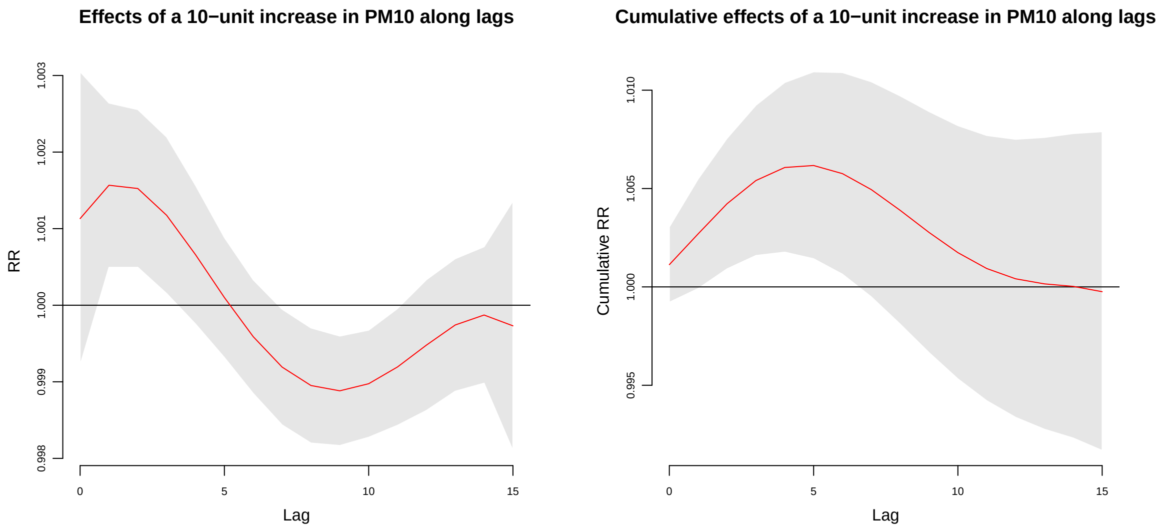
```
> model <- glm(death ~  basis.pm + basis.temp + ns(time, 7*14) + dow,
          family=quasipoisson(), chicagoNMMAPS)
```

The effects of specific levels of PM_{10} on overall mortality, predicted by the model above, can be computed by the function `crosspred()` and saved in an object with the same class:

```
> pred.pm <- crosspred(basis.pm, model, at=0:20, cumul=T)
```

Figure 1

(a) (b)



The function includes the `basis.pm` and `model` objects used to estimate the parameters as the first two arguments, while `at=0:20` states that the prediction must be computed for each integer value from 0 to 20 μgr/m^3. The argument `cumul` (default to `FALSE`) indicates that also cumulative effects along lags must be included. Now that the predicted effects have been stored in `pred.pm`, they can be plot by the function `crossplot()`:

```
> crossplot(pred.pm, "slices", var=10,
          title="Effects of a 10-unit increase in PM10 along lags")
> crossplot(pred.pm, "slices", var=10, cumul=T,
          title="Cumulative effects of a 10-unit increase in PM10 along lags")
```

The function includes the `pred.pm` object with the stored results, and the argument `"slices"` defines that we want to graph the relationship at specific values of the two dimensions (predictor and lag). With `var=10` we specify this relationship along lags for a specific value of PM_{10}, i.e. 10 μgr/m^3. This effect is compared to the reference value of 0 μgr/m^3, giving the lag-specific effects for a 10-unit increase. The argument `cumul` indicates if cumulative effect, previously saved in `pred.pm`, must be plotted. The results are shown in Figures 1a-1b. The interpretation is twofold: the curve represents the increase in risk in each future day following an increase of 10 μgr/m^3 in PM_{10} in a specific day (*forward interpretation*), or otherwise the contributions of each past day with the same PM_{10} increase to the risk in a specific day (*backward interpretation*). The plots in Figures 1a-1b suggest that the initial increase in risk of PM_{10} is reversed at longer lags. The overall effect for a 10-unit increase in PM_{10} over 15 days of lag (i.e. summing all the effects up to the maximum lag), together with its 95% confidence intervals can be extracted by the objects `allRRfit`, `allRRhigh` and `allRRlow` included in `pred.pm`, typing:

```
> pred.pm$allRRfit["10"]
```

```
       10
0.9997563
```

```
> cbind(pred.pm$allRRlow, pred.pm$allRRhigh)["10",]
```

```
[1] 0.991687 1.007891
```

5.3 Example 2: seasonal analysis

The purpose of the second example is to illustrate an analysis where the data are restricted to a specific season. The main feature of these analysis is that the data are assumed to be composed by multiple equally-spaced and ordered series of the same season for each year, and do not represent a single continuous series. In this case, we assess the effect of ozone and temperature on overall mortality up to 5 and 10 days of lag, respectively, using the same steps already seen in Section 5.2.

First, we create the new data restricting to the summer period (June-September) the dataframe `chicagoNMMAPS`:

```
> chicagoNMMAPSseas <- subset(chicagoNMMAPS, month %in% 6:9)
```

Again, we first create the cross-basis matrices:

```
> basis.o3 <- crossbasis(chicagoNMMAPSseas$o3, group=chicagoNMMAPSseas$year,
          vartype="hthr", varknots=40.3, lagtype="integer", maxlag=5)
> basis.temp <- crossbasis(chicagoNMMAPSseas$temp, group=chicagoNMMAPSseas$year,
          vartype="dthr", varknots=c(15,25), lagtype="strata", lagknots=c(2,6),
          maxlag=10)
```

The argument `group` indicates the variable which defines multiple series: the function then breaks the series at the end of each group and replaces the first `maxlag` rows of the cross-basis matrix in the following series with `NA`. Here we make the assumption that the effect of O_3 is null up to 40.3 μgr/m^3 and then linear, applying an high threshold parameterization. For temperature, we use a double threshold with the assumption that the effect is linear below 10°C and above 25°C, and null in between. Regarding the lag dimension, we specify an unconstrained function for O_3, applying one parameter for each lag (`lagtype="integer"`) up to a 5 days. For temperature, we define 3 strata intervals at lag 0-1, 2-5, 6-10. A summary of the choices made for the cross-bases can be shown by the function `summary.crossbasis()`.

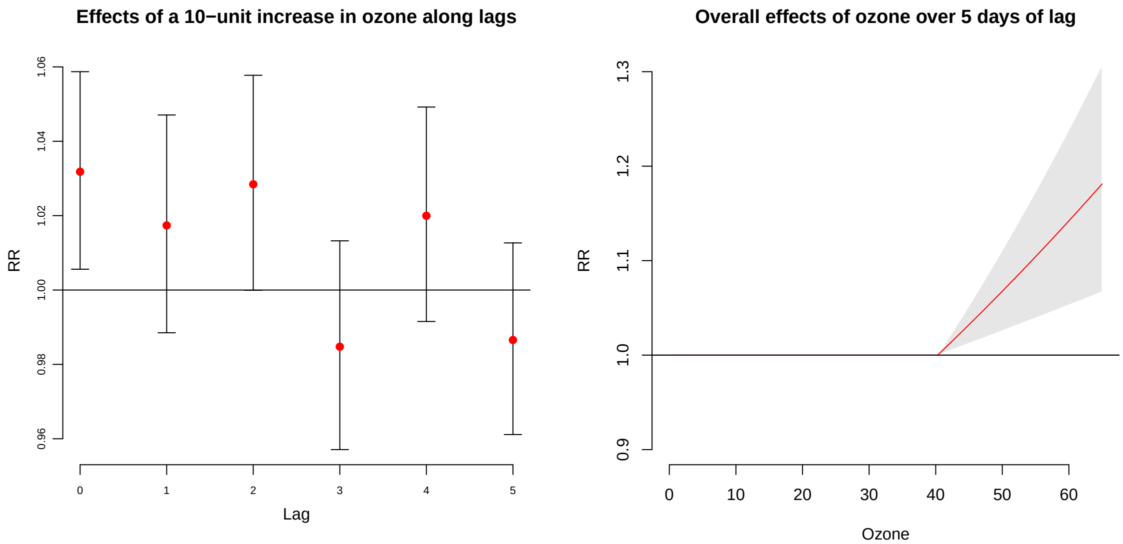
The regression model includes a natural spline for day of the year (with 4 df) in order to describe the seasonal effect within each year. Apart from that, the estimates and predictions are carried out in the same way as in Section 5.2. The code is:

```
> model <- glm(death ~  basis.o3 + basis.temp + ns(doy, 4) + dow,
          family=quasipoisson(), chicagoNMMAPSseas)
> pred.o3 <- crosspred(basis.o3, model, at=c(0:65,40.3,50.3))
```

The values for which the prediction must be computed are specified in `at`: here we define the integers from 0 to 65 μgr/m^3 (approximately the range of ozone distribution), plus the threshold and the value 50.3 μgr/m^3 corresponding to a 10-unit increase above the threshold, which is automatically set as the reference point for `type="hthr"` (see *?crossbasis*). The vector is automatically ordered. We can plot the lag-specific effects, similarly to Section 5.2, and also the overall effect of a 10-unit increase in O_3 with 95% confidence intervals. The commands are (results in Figures 2a-2b):

Figure 2

(a) (b)




```
> crossplot(pred.o3, "slices", var=50.3, ci="bars",
          title="Effects of a 10-unit increase in ozone along lags")
> crossplot(pred.o3,"overall",label="Ozone", ylim=c(0.9,1.3),
          title="Overall effects of ozone over 5 days of lag")
```

In the first command, the argument `ci="bars"` states that, differently from the default `"area"` seen in Figures 1a-1b, the confidence intervals are represented by bars. In the second command, the argument `type="overall"` indicates that the overall effects (summed upon lags) must be plotted, with `ylim` defining the range of the y-axis.

Similarly to the previous example, we can extract from `pred.o3` the estimated overall effect for a 10-unit increase in ozone above the threshold, together with its 95% confidence intervals:

```
> pred.o3$allRRfit["50.3"]

    50.3
1.069768

> cbind(pred.o3$allRRlow, pred.o3$allRRhigh)["50.3",]

[1] 1.026562 1.114792
```

The same plots and computation can be applied to the cold and heat effects of temperatures. For example, we can describe the increase in risk for 1°C beyond the low or high thresholds. The user can perform this analysis repeating the steps above.

5.4 Example 3: a complex DLNM

In the previous examples, the effects of air pollution (PM_{10} and O_3, respectively) were assumed completely linear or linear above a threshold. This assumption facilitates both the interpretation and the representation of the association: the dimension of the predictor is never considered, and the lag-specific or overall effects for a 10-unit increase are easily plotted. In contrast, when considering the non-linear effects of temperature, we need to adopt a bi-dimensional perspective in order to represent effects which vary non-linearly along the space of the predictor and lags.

In this last example we specify a more complex DLNM, where the effects are estimated using smooth non-linear functions for both dimensions . Despite the higher complexity of the relationship, we will see how the steps required to specify and fit the model and predict the results are exactly the same as for the simpler models see before in Sections 5.2-5.3, only requiring different plotting choices. The user can apply the same steps to investigate the effects of temperature in previous examples, and extend the plots for PM_{10} and O_3. In this case we run a DLNM to investigate the effects of temperature and PM_{10} on overall mortality up to lag 30 and 1, respectively.

These are the cross-basis matrices:

```
> basis.pm <- crossbasis(chicagoNMMAPS$pm10,vartype="lin", lagtype="strata",
          cen=F, maxlag=1)
> basis.temp <- crossbasis(chicagoNMMAPS$temp, vartype="bs", vardf=5, vardegree=2,
          lagdf=5, cenvalue=21, maxlag=30)
```

The chosen basis functions for the space of the predictor are a linear function for the effect of PM_{10} and a quadratic B-spline (`vartype="bs"`) with 5 degrees of freedom for temperature (with `varknots` placed by default at equally spaced quantiles in the space of the predictor). The basis for temperature is centered at 21°C, which will represent the reference point for the predicted effects. Regarding the space of lags, we assume a simple lag 0-1 parameterization for PM_{10} (i.e. a single strata up to lag 1, keeping the default values of `lagdf=1`), while we define another cubic spline, this time with the natural constraint (`lagtype="ns"` by default) for the lag dimension of temperature. For this space, `lagknots` are located by default at equally spaced values in the log scale of lags, while the boundary knots are set to 0 and `maxlag`. The estimation, prediction and plotting of the effects of temperature are performed by:

```
> model <- glm(death ~  basis.pm + basis.temp + ns(time, 7*14) + dow,
          family=quasipoisson(), chicagoNMMAPS)
> pred.temp <- crosspred(basis.temp, model, at=-26:33)
> crossplot(pred.temp, label="Temperature",
          title="3D graph of temperature effects")
> crossplot(pred.temp, "contour", label="Temperature",
          title="Contour graph of temperature effects")
```

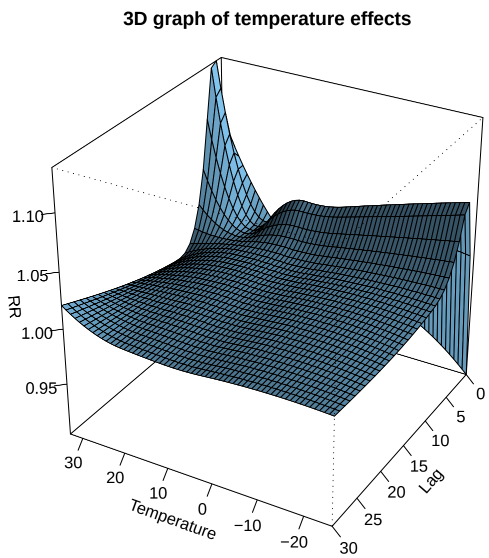
The 3-D and contour plots obtained by the commands above are represented in Figures 3a-3b. The plot of the overall effects can be obtained by (result not shown):

```
> crossplot(pred.temp, "overall", label="Temperature",
          title="Overall effects of temperature over 30 days of lag")
```

More comprehensively, Figure 4 shows the effects by temperature at multiple specific lags (left) and the effect by lag at multiple specific temperatures (right). The arguments `var` and `lag` are used to define the specific predictor and lag values for which the effects must be computed. These plots can

Figure 3

(a)



(b)

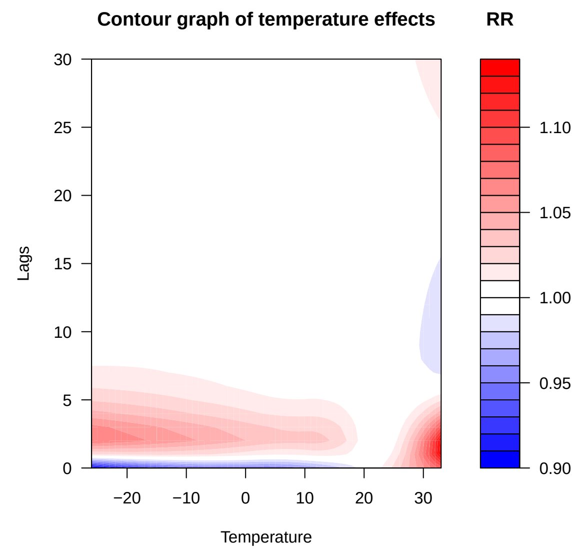
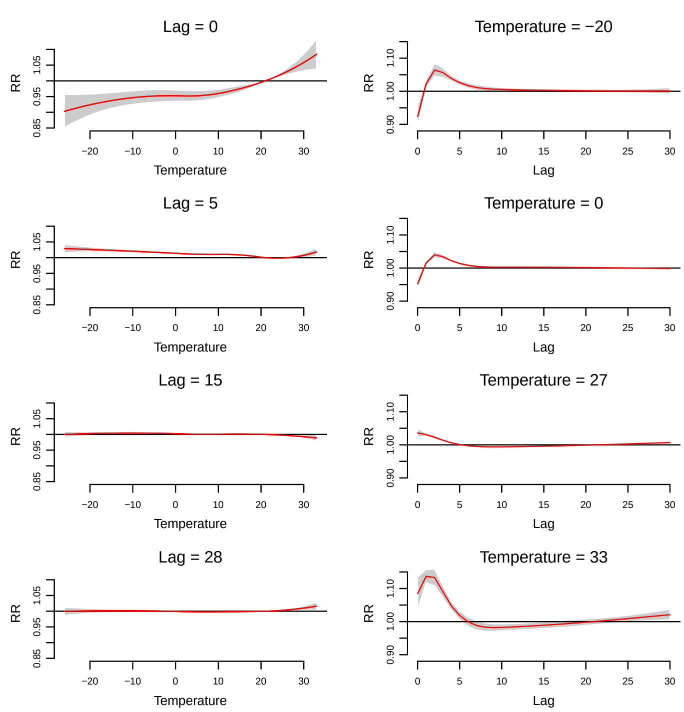


Figure 4



be imagined as the results of cutting *slices* on the effect surface shown in Figures 3a-3b at the specific values of temperature and lags, and provide a detailed overview of the effects surface together with the related confidence intervals. This plot is carried out by:

```
> crossplot(pred.temp, "slices", var=c(-20,0,27,33),
          lag=c(0,5,15,28), label="Temperature")
```

6 Conclusions

This document illustrates the functionalities of the dlnm package, providing a detailed overview of the process to specify and run a DLNM and then to predict and plot its results. The main advantage of this family of models is to unify many of the previous methods to deal with delayed effects in a unique framework, also providing more flexible alternatives regarding the shape of the relationships. Section 3 provides a brief summary of the theory underpinning DLNM's: a more detailed overview has been published elsewhere (Armstrong, 2006; Gasparrini et al., 2010), together with a complete specification of the algebra (Gasparrini et al., 2010).

The flexibility is kept when this framework is implemented in the dlnm package: several different models with an increasing level of complexity can be performed using a simple and general procedure, as showed in the examples in Section 5. As already explained, this method is not limited to the examples on the effect of air pollution and temperature on mortality, but can be applied to investigate the relationship between any predictor and outcomes in time-series data.

The choice of keeping separated the two steps of cross-basis specification and parameters estimation offers several advantages. First, as illustrated in the example, more than one variable showing delayed effects can be transformed through cross-basis functions and included in the model. Second, standard regression commands can be used for estimation, with the default set of diagnostic tools and related functions. More importantly, this implementation provides an open platform where additional models specified with different regression commands can be included as well, aiding the development of these methodology in other contexts or study designs.

The DLNM's framework introduced here is developed for time series design. The general expression of the model in allows this methodology to be applied for any family distribution and link function within generalized linear models (GLM), with extensions to GAM or models based on generalized estimating equations (GEE). Anyway, the current implementation of of DLNM's requires single series of equally-spaced and ordered data. Preliminary tests on the application of the functions included in the package dlnm in case-control, cohort and longitudinal data are promising. Further development may lead to a general framework to describe delayed effects, which spans different study designs.

7 Acknowledgements

This work was supported by the Medical Research Council (UK) through the Research Grant RES-G0707030.

We gratefully acknowledge the valuable suggestions of Fabio Frascati regarding the procedures to build and document this package. Other package vignettes were used as examples (in particular gnm by Heather Turner and David Firth). We are thankful to the colleagues who tested the beta-version of this package and suggested some important improvements included in the current version (in particular Marie-France Valois).

Finally, we express our gratitude to all the people working to develop and maintain the R Project.

References

S. Almon. The distributed lag between capital appropriations and expenditures. *Econometrica*, 33: 178–196, 1965.

B. Armstrong. Models for the relationship between ambient temperature and daily mortality. *Epidemiology*, 17(6):624–31, 2006.

M. Baccini, A. Biggeri, G. Accetta, T. Kosatsky, K. Katsouyanni, A. Analitis, H. R. Anderson, L. Bisanti, D. D'Ippoliti, J. Danova, B. Forsberg, S. Medina, A. Paldy, D. Rabczenko, C. Schindler, and P. Michelozzi. Heat effects on mortality in 15 European cities. *Epidemiology*, 19(5):711–9, 2008.

A. L. Braga, A. Zanobetti, and J. Schwartz. The time course of weather-related deaths. *Epidemiology*, 12(6):662–7, 2001.

J. Cao, M. F. Valois, and M. S. Goldberg. An S-Plus function to calculate relative risks and adjusted means for regression models using natural splines. *Computer Methods and Programs in Biomedicine*, 84(1):58–62, 2006.

A. Gasparrini, B. Armstrong, and M. G. Kenward. Distributed lag non-linear models. *Statistics in Medicine*, (In Press), 2010.

S. Hajat, B. G. Armstrong, N. Gouveia, and P. Wilkinson. Mortality displacement of heat-related deaths: a comparison of Delhi, Sao Paulo, and London. *Epidemiology*, 16(5):613–20, 2005.

S. Pattenden, B. Nikiforov, and B. G. Armstrong. Mortality and temperature in Sofia and London. *Journal of Epidemiology and Community Health*, 57(8):628–33, 2003.

J. M. Samet, S. L. Zeger, F. Dominici, F. Curriero, I. Coursac, and D. W. Dockery. The National Morbidity, Mortality, and Air Pollution Study (NMMAPS). Part 2. Morbidity and mortality from air pollution in the United States. Technical report, Health Effects Institute, 2000a.

J. M. Samet, S. L. Zeger, F. Dominici, D. Dockery, and J. Schwartz. The National Morbidity, Mortality, and Air Pollution Study (NMMAPS). Part 1. Methods and methodological issues. Technical report, Health Effects Institute, 2000b.

J. Schwartz. The distributed lag between air pollution and daily deaths. *Epidemiology*, 11(3):320–6, 2000.

J. Schwartz. Is there harvesting in the association of airborne particles with daily deaths and hospital admissions? *Epidemiology*, 12(1):55–61, 2001.

J. Schwartz, J. M. Samet, and J. A. Patz. Hospital admissions for heart disease: the effects of temperature and humidity. *Epidemiology*, 15(6):755–61, 2004.

L. J. Welty and S. L. Zeger. Are the acute effects of particulate matter on mortality in the National Morbidity, Mortality, and Air Pollution Study the result of inadequate control for weather and season? A sensitivity analysis using flexible distributed lag models. *American Journal of Epidemiology*, 162(1):80–8, 2005.

S.N. Wood. *Generalized additive models: an introduction with R*. Chapman & Hall/CRC, 2006.

A. Zanobetti and J. Schwartz. Mortality displacement in the association of ozone with mortality: an analysis of 48 cities in the United States. *American Journal of Respiratory and Critical Care Medicine*, 177(2):184–9, 2008.

A. Zanobetti, M. P. Wand, J. Schwartz, and L. M. Ryan. Generalized additive distributed lag models: quantifying mortality displacement. *Biostatistics*, 1(3):279–92, 2000.