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 1.5.2 , 2012-01-05

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 offers 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. This document complements the description provided in Gasparrini (2011) (freely available at http://www.jstatsoft.org/v43/i08/), which represents the main reference to the package.

The aim of this contribution is to provide an extended overview of the capabilities of the package, together with additional 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 has been previously described in Gasparrini et al. (2010), together with a detailed algebraical development. This framework was originally conceived and proposed to investigate the health effect of temperature by Armstrong (2006).

Type `citation("dlnm")` in R to cite the dlnm package after installation (see Section 2). 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 from version 2.9.0 onwards, for example typing `install.packages("dlnm")` or directly through the R menu in Windows, 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()`). The former 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, 2011; 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 regression 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 occurs when the outcome in a specific time is determined 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. However, 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 related 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 by 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 summarize the effects for single predictor levels or lags, simply cutting a "slice" of the grid along specific values. These summaries are defined lag-specific association, including the effects along the predictor space for a given lag value, and predictor-specific association, including the effects along lags for a given predictor values, respectively. Finally, an estimate of the overall effect can be computed by summing all the contributions at different lags for each predictor value. 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).

Although prediction aids interpretation of a complex bi-dimensional DLNM, its fit is still expressed in terms of the estimated parameters of cross-basis functions obtained through the tensor product of two bases. It turns out that it is possible to reduce the expression of summaries of the association to modified parameters of one-dimensional basis functions (Gasparrini et al., 2012). In practice, the overall or lag-specific effects, defined in the predictor space, may be re-expressed only in terms of the one-dimensional basis functions originally chosen for that dimension. Similarly, effects along lags at specific predictor values may be reduced to parameters of the original basis functions chosen for the lag space. The reduction of the parameters is obtained through appropriate transformation matrices.

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**

This section describes the main functions included in the package dlnm. Here we provide a description of all the stages involved in the definition, estimation and interpretation of DLNMs, summarizing the conceptual and analytical steps. In addition, we illustrate the structure of the functions and discuss specific issues about their usage. Examples of applications to real time series data are described in Section 5. Additional information is provided in Gasparrini (2011).

4.1 The function `onebasis()`

This function generates the basis matrix for a predictor vector, choosing among a set of possible basis functions. Its usage is general, and not limited to DLNMs. Within the package, it is called by `crossbasis()` to build the one-dimensional basis matrices for both predictor and lag spaces, which are then combined in a cross-basis matrix. It has replaced the old functions `mkbasis()` and `mklagbasis()` since version 1.5.1. Differently from these old internal functions, `onebasis()` is also available to the user to specify one-dimensional (un-lagged) relationships in regression models. Prediction and plotting dlnm functions described in Section 4.3 – 4.5 are also available for these simpler models.

Its first argument is `x`, representing the original predictor vector. Different types of basis may be chosen through the argument `type`: the possible options are natural cubic or simple B-splines (`type="ns"` or `"bs"`), strata through dummy variables (`"strata"`), polynomials (`"poly"`), threshold-type functions such as low, high or double threshold or piecewise parameterization (`"lthr"-"hthr"-"dthr"`), strata variables for each integer values (`"integer"`, used in unconstrained DLMs) and simply linear (`"lin"`).

The argument `"df"` defines the dimension of the basis (the number of its columns, basically the number of transformed variables), which, in completely parametric models, corresponds to the number of degrees of freedom spent to define the relationship in the regression model including the basis. This value may depend on the argument `knots` (which overcomes `df`), specifying the position of the internal knots for `"ns"` and `"bs"` (with boundary knots specified in `bound`), the cut-off points for `"strata"` (defining right-open intervals) and the thresholds/cut-off points for `"lthr"`, `"hthr"` and `"dthr"`. The argument `degree` select the degree of polynomial for `"bs"` and `"poly"`.

The arguments `cen` (numeric or logical) states if the basis must be centered, or the centering value to be used. The basis variables are centered by default for continuous functions (types `"ns"`, `"bs"`, `"poly"` and `"lin"`). The default centering point is the predictor mean, if not set with `cen`. The choice of the reference value does not affect the fit of the model, and should be based on interpretational

issues. The reference in non-continuous functions is automatically set to the first interval in `strata` and `integer`, or to the flat region in `lthr-hthr-dthr`.

The presence of an intercept in the basis matrix is determined by the argument `int`. Actually, the concept of intercept is different between bases: types `"ns"` and `"bs"` apply a complex parameterization where the intercept is implicitly built within the basis variables (see the related help pages typing *`?ns`* and *`?bs`*); in type `"strata"` the intercept corresponds to the dummy variable for the baseline stratum (the first one by default), which is excluded if `int=FALSE`; the intercept is the usual vector of 1's in the other types.

The function returs a matrix object of class *"onebasis"*, with attributes corresponding to the arguments above, which uniquely define the basis transformation. The results can be checked with the related `summary()` method function.

4.2 The function `crossbasis()`

This is the main function in the package `dlnm`. It calls `onebasis()` to generate the basis matrices for predictor and lags, and combines them through a tensor product in order to create the cross-basis, which specifies the dependency simultaneously in the two dimensions. See Gasparrini et al. (2010, Sections 4.1 – 4.2) for details.

The usage of the function has changed since version 1.5.1. Its first argument is `x`, assumed to represent an equally-spaced, complete and ordered series of observations, in order for the function to be coherently applied. The second argument is `lag`, a positive integer vector of length 1 or 2, defining the maximum lag (with minimum set to 0 by default) or the lag range, respectively. The vector of lag values `lag[1]:lag[2]` is then generated to build the basis matrix for the lag space. The two arguments `argvar` and `arglag` contain a list of objects `type-df-knots-bound-degree-int-cen`, each of them to be passed to `onebasis()` to build the matrices for the 2 spaces, respectively (see Section 4.1). The additional argument `group` defines groups of observations to be considered as individual unrelated series, and may be useful for example in seasonal analyses (see Section 5.3). In this case, each series must be consecutive, complete and ordered.

The function returns an matrix object of class *"crossbasis"*, together with attributes defining the choices for the two basis functions. The arguments are set to some default values, and can be automatically changed for nonsensical combinations, or set to null if not required. Meaningless combinations of arguments (for example knots defined outside the predictor range) could lead to collinear variables, with identifiability problems in the model. The function applies some coherence checks and fix some specific problem (for example discarding strata intervals where no observation lies), but other problem may arise. The user is advised to test the result with the method function `summary()`, which provides a summary of the choices made for the two bases and the final cross-basis.

The values in `x` are expected to be equally-spaced (with the interval defining the lag unit) and ordered in time. The series must be complete. Each value in the series of transformed variables is computed also using previous observations included in the lag period considered: therefore, the first observations in the transformed variables up to the maximum lag are set to `NA`. Missing values in `x` are allowed, but, for the same reason, the same and the next transformed values up to the maximum lag will be set to `NA`. Although correct, this could generate computational problems for DLNMs with long lag periods in the presence of scattered missing observations.

The basis variables for the space of the predictor are centered by default for specific types (see Section 4.1). It is strongly recommended to avoid the inclusion of an intercept in the basis for the predictor, otherwise a rank-deficient cross-basis matrix will be specified, causing some of the cross-variables to be excluded in the regression model. Different default values are chosen for the arguments related to the basis for lags, if compared to those set by `onebasis()`. Specifically, the basis is never centered

(cen=FALSE), an intercept is included by default (int=TRUE), and the knots are placed at equally-spaced values in the log scale of lags.

4.3 The function crosspred()

The cross-basis matrix produced by crossbasis(), or simply a basis matrix obtained by onebasis(), need to be included in a regression model formula in order to fir the model. The interpretation of the estimated related parameters, is usually complex for non-trivial basis transformations, and virtually impossible in bi-dimensional DLNMs. The association is summarized through the function crosspred(), which predicts the effects for a set of values of the original predictor, and return the results for each combination of predictor values and lags. The function creates the same basis or cross-basis functions for the chosen predictor values, based on the attributes of the original basis or cross-basis matrix, and generates estimated effects and standard errors by extracting the related parameters estimated in the model (see Gasparrini et al. (2010, Section 4.3) for details).

The first two arguments of the function are basis (the matrix object of class *"onebasis"* or *"crossbasis"*) and model (the regression model object which includes basis). The function extracts the information about the basis or cross-basis from the attributes of the former, and links each basis or cross-basis variables with the estimated parameters in the latter through their names. Multiple basis or cross-basis matrices associated with different predictors may be included in model: in this case, the user must specify different names for the basis objects.

One of the main advantages of the dlnm package is that the user can perform DLNMs with standard regression functions, simply including the cross-basis matrix in the model formula. The current implementation only works with time series data, basically involving an equally-spaced and ordered predictor series, and its use is straightforward with the functions lm(), glm() or gam() (package mgcv). However, the user can apply different regression functions, compatibly with the time series structure of the data. Alternative use beyond time series analysis, such as in case-control or cohort designs, is in development. The function crosspred() exploits coef() and vcov() methods to extract the coefficients and related (co)variance matrix from model, respectively: for classes of regression functions without these methods, the user needs to manually extract the parameters and include them in the arguments coef and vcov. In this case, their dimensions and order must match the variables included in basis.

The predictor values used for prediction are selected with the argument at, or alternatively with from-to-by. If specified by at, the values are automatically ordered and made unique. If at and by are not provided, approximately 50 equally-spaced rounded values are returned using pretty().

The function returns an object of class *"crosspred"*, simply a list of objects including the vector of prediction values, coefficients and associated (co)variance matrix, matrices of lag-specific effects and standard errors for combinations of each prediction value and lag, plus vectors of overall effects (summed up along lags) and standard errors. Matrices of cumulative effects and standard errors are included for cumul=TRUE (default to FALSE), which represent the sum of the lag-specific effects at each lag. Exponentiated effects are added if the link of the regression model is equal to log or logit, together with confidence intervals computed using a normal approximation and a confidence level selected by ci.level. The model link is automatically selected from model for classes *"lm"*, *"glm"*, *"gam"* (package mgcv) and *"clogit"* and *"coxph"* (package survival), but needs to be provided through model.link for different classes or if arguments coef-vcov are used to input the parameters.

4.4 The function `crossreduce()`

As described in Section 3.4, the fit of a DLNM may be reduced to a single dimension of predictor or lags, summarizing overall, lag-specific or predictor-specific effects only in terms of the original basis functions chosen for the related space, and modified associated parameters (Gasparrini et al., 2012). This computation is carried out through the function `crossreduce()`, which provides the modified parameters and basis matrix, plus the estimated effects, similarly to `crosspred()`, as described in Section 4.3.

The first two arguments `basis` and `model`, similarly to `crosspred()`, specify the cross-basis matrix and the model object for which the computation need to be performed. The type of reduction is defined by `type`, with options `"overall"`-`"lag"`-`"var"` for summarizing overall, lag-specific or predictor-specific associations, respectively. The first two are expressed in the dimension of predictor, the third in the dimension of lags. The single value of predictor or lags for which predictor-specific or lag-specific functions must be defined is chosen by the argument `value`. The other arguments (see `?crossreduce`) have the same meaning and specification as in `crosspred()` (see Section 4.3).

The function returns a list object of class *"crossreduce"*, including the modified parameters and associated (co)variance matrix, the vector of values used for prediction and associated one-dimensional basis matrix, and the vectors of predicted effects and associated standard errors, optionally exponentiated, similarly to `crosspred()`.

4.5 Plotting functions

Interpretation of the one-dimensional or bi-dimensional predicted effects are aided by graphical representation. High and low-level plotting functions are provided through the method functions `plot()`, `lines()` and `points()`. The method `plot()` calls high-level functions `plot.default()`, `persp()` and `filled.contour()` to produce scatter plots, 3-D and contour plots of overall, lag-specific or predictor-specific effects. These methods have replaced the old function `crossplot()` since version 1.3.0, providing the user the chance to specify the whole range or arguments of the plotting functions above, allowing complete flexibility in the choices of colours, axes, labels and other graphical parameters. See the help pages of the original high-level functions for additional details and a complete list of the arguments. Methods `lines()` and `points()` may be used as low-level plotting functions to add lines or points to an existing plot.

Method functions are defined for classes *"crosspred"* and *"crossreduce"*. The first argument of the functions is `x`, a list object of related class. For `crosspred` objects, the argument `ptype` specifies the type of plot, choosing among `"3d"`, `"contour"`, `"overall"` and `"slices"`, the latter selecting lag-specific effects along the predictor space or predictor-specific effects along lags. These are chosen through the additional arguments `lag-var`, respectively. For un-lagged relationships defined by `onebasis()`, only the plotting overall effects is meaningful and possible. Cumulative effects along lags are reported if `cumul=TRUE`: in this case, the same option must have been set to obtain the prediction saved in `x` (see Section 4.3). For `crossreduce` objects, the specific type of plot is automatically defined by the type of reduction.

Confidence intervals are optionally plotted for `"overall"` and `"slices"`. The type is chosen by the argument `ci` among `"area"`, `"bars"` and `"lines"`. Low-level plotting functions `polygon()`, `segments()` and `lines()` are called, respectively, whose arguments are passed by a list specified with the argument `ci.arg`. See the help of these low-level functions for additional details and a complete list of the arguments.

All the effects are reported versus a reference value. For continuous functions, this is specified by the centering point defined in the `onebasis` or `crossbasis` object (see Section 4.1). Exponentiated effects are automatically returned if the component `model.link` of `x` is equal to log or logit, or forced with

the argument `exp=TRUE`.

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 use of `onebasis()` to build basis matrices are showed in Section 5.1. Then, 4 different examples of the application of DLNM's are illustrated in the Sections 5.2 – 5.5, 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, typing:

```
> library(dlnm)
```

5.1 Examples for `onebasis()`

As a first step, we provide an example of the use of the function `onebasis()`. We build different basis matrices applying the selected basis functions to a vector of integers. In the first example we leave many of the arguments at their default values, apart from the selection of the knots:

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

```
              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
attr(,"range")
[1] 1 5
attr(,"type")
[1] "ns"
attr(,"df")
[1] 2
attr(,"degree")
[1] 3
attr(,"knots")
[1] 3
attr(,"bound")
[1] 1 5
attr(,"int")
```

```
[1] FALSE
attr(,"cen")
[1] 3
attr(,"class")
[1] "onebasis" "matrix"
```

The result is matrix object with attributes returning the chosen arguments. Here the 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 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)`*.

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

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

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

Other examples (results not shown):

```
> onebasis(1:5, type="poly", degree=3, int=TRUE)
> onebasis(1:5, type="integer")
> onebasis(1:5, type="dthr", knots=c(2,3))
```

The argument `degree=3` specifies a 3$^{\text{rd}}$ degree polynomial. This matrix contains an intercept (obtained through `int=TRUE`), in this case a vector of 1's (see Section 4.1). Here `df` is equal to `degree+1` when an intercept is included. In this case, for a polynomial bases, the argument `knots` is not included. In the second example, the function applies a specific transformation used to define unconstrained distributed lag effects (see Section 3.3), simply returning an identity matrix. The third 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):

```
> onebasis(1:10, type="strata", knots=c(4,7), int=TRUE)
> onebasis(1:10, type="strata", knots=c(4,7))
```

In this case, the intercept is represented by the dummy variable for the first stratum (see Section 4.1). 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):

```
> onebasis(0:10, type="poly", degree=3)
> onebasis(0:10, type="poly", degree=3, cen=FALSE)
```

Each basis function is centered on the relative transformation of `cen` (if numeric), or placed at the mean of the predictor values by default (if `cen=TRUE`).

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. As highlighted above, the data are assumed to be composed by equally-spaced, complete and ordered series.

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 Section 4.3). This is the code:

```
> cb1.pm <- crossbasis(chicagoNMMAPS$pm10, lag=15, argvar=list(type="lin",
    cen=FALSE), arglag=list(type="poly",degree=4))
> cb1.temp <- crossbasis(chicagoNMMAPS$temp, lag=3, argvar=list(df=5,cen=21),
    arglag=list(type="strata",knots=1))
```

The function calls `onebasis()` and passes the arguments in `argvar` and `arglag` to build the basis for predictor and lags, respectively. In this case, we assume that the effect of PM_{10} is linear (`type="lin"`), while we model the relationship with temperature through a natural cubic spline with 5 degrees of freedom (`type="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 `df`. 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=0`). The reference value for temperature is set to 21°C.

Regarding the bases for the space of the lags, we specify the lagged effect of PM_{10} up to 15 days of lag (minimum lag equal to 0 by default), with a 4^{th} degree polynomial function (setting `degree=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 `knots=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 method function `summary()` for this class:

```
> summary(cb1.pm)

CROSSBASIS FUNCTIONS
observations: 5114
range: -3.049835 , 356.1768
total df: 5
lag range: 0 15

BASIS FOR VAR:
type: lin
df: 1
not centered
without intercept

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. The packages splines is loaded, as it is needed in the examples. 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:

```
> library(splines)
> model1 <- glm(death ~ cb1.pm + cb1.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:

```
> pred1.pm <- crosspred(cb1.pm, model1, at=0:20, cumul=TRUE)
```

The functions include the `basis1.pm` and `model1` 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 `pred1.pm`, they can be plot by the methods functions described in Section 4.5. For example:

```
> plot(pred1.pm, "slices", var=10, col=3, ylab="RR", ci.arg=list(density=15,lwd=2),
    main="Effect of a 10-unit increase in PM10 along lags")
> plot(pred1.pm, "slices", var=10, cumul=TRUE, ylab="Cumulative RR",
          main="Cumulative effect of a 10-unit increase in PM10 along lags")
```

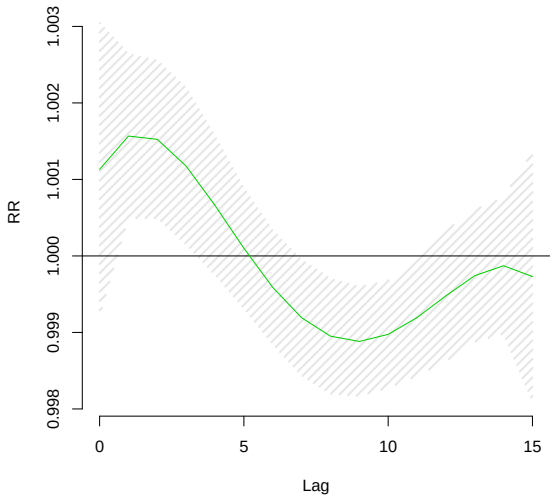
The function includes the `pred1.pm` object with the stored results, and the argument `"slices"` defines that we want to graph the relationship at specific values of the two dimensions (predictor and lag). With `var=10` we specify this relationship along lags for a specific value of PM_{10}, i.e. 10 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. We also chose a different colour for the first plot. The argument `cumul` indicates if cumulative effect, previously saved in `pred1.pm`, must be plotted. The results are shown in Figures 1a – 1b. Confidence intervals are set to the default value `"area"` for the argument `ci`. In the left panel, additional arguments are passed to the low-level plotting function `polygon()` through `ci.arg`, to draw instead shading lines as confidence intervals.

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 `pred1.pm`, typing:

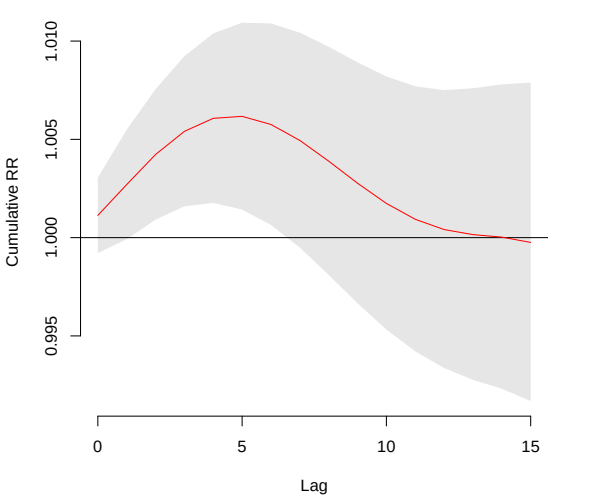
```
> pred1.pm$allRRfit["10"]

       10
0.9997563

> cbind(pred1.pm$allRRlow, pred1.pm$allRRhigh)["10",]

[1] 0.9916871 1.0078911
```

Figure 1

(a)



(b)



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 peculiar feature of this 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)
```

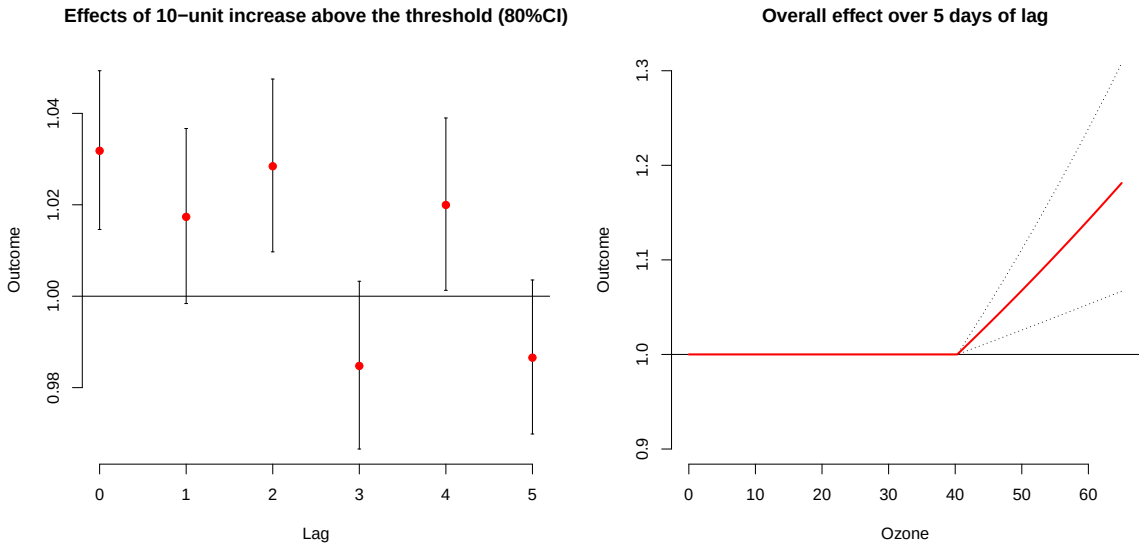
Again, we first create the cross-basis matrices:

```
> cb2.o3 <- crossbasis(chicagoNMMAPSseas$o3, lag=5, argvar=list(type="hthr",
    knots=40.3), arglag=list(type="integer"), group=chicagoNMMAPSseas$year)
> cb2.temp <- crossbasis(chicagoNMMAPSseas$temp, lag=10,
    argvar=list(type="dthr",knots=c(15,25)), arglag=list(type="strata",
    knots=c(2,6)), group=chicagoNMMAPSseas$year)
```

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 rows up to the maximum lag of the cross-basis matrix in the following series with `NA`. Each series must be consecutive, complete and ordered. 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

Figure 2

(a) (b)



specify an unconstrained function for O_3, applying one parameter for each lag (`type="integer"`) up to a 5 days (with minimum lag equal to 0 by default). 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 method `summary()`.

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:

```
> model2 <- glm(death ~  cb2.o3 + cb2.temp + ns(doy, 4) + dow,
    family=quasipoisson(), chicagoNMMAPSseas)
> pred2.o3 <- crosspred(cb2.o3, model2, 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 Section 4.1). The vector is automatically ordered. We can plot the lag-specific effects, similarly to Section 5.2 but with 80% confidence intervals, and also the overall effect of a 10-unit increase in O_3. The related code is (results in Figures 2a – 2b):

```
> plot(pred2.o3, "slices", var=50.3, ci="bars", type="p", pch=19, ci.level=0.80,
         main="Effects of 10-unit increase above the threshold (80%CI)")
> plot(pred2.o3,"overall",xlab="Ozone", ci="lines", ylim=c(0.9,1.3), lwd=2,
    ci.arg=list(col=1,lty=3), main="Overall effect over 5 days of lag")
```

In the first statement, the argument `ci="bars"` dictates that, differently from the default `"area"` seen in Figures 1a – 1b, the confidence intervals are represented by bars. In addition, the argument

`ci.level=0.80` states that 80% confidence intervals must be plotted. Finally, we chose points, instead of the default line, with specific symbol, by the arguments `type` and `pch`. In the second statement, the argument `type="overall"` indicates that the overall effects (summed upon lags) must be plotted, with confidence intervals as lines, `ylim` defining the range of the y-axis, `lwd` the thickness of the line. Similarly to the previous example, the display of confidence intervals are refined through the list of arguments specified by `ci.arg`, passed in this case to the low-level function `lines()`.

Similarly to the previous example, we can extract from `pred2.o3` the estimated overall effect for a 10-unit increase in ozone above the threshold ($50.3 - 40.3$ μgr/m^3), together with its 95% confidence intervals:

```
> pred2.o3$allRRfit["50.3"]

    50.3
1.069768

> cbind(pred2.o3$allRRlow, pred2.o3$allRRhigh)["50.3",]

[1] 1.026563 1.114791
```

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 bi-dimensional 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 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:

```
> cb3.pm <- crossbasis(chicagoNMMAPS$pm10, lag=1, argvar=list(type="lin",
    cen=FALSE), arglag=list(type="strata"))
> cb3.temp <- crossbasis(chicagoNMMAPS$temp, lag=30, argvar=list(type="bs",
    df=5,degree=2,cen=21), arglag=list(df=5))
```

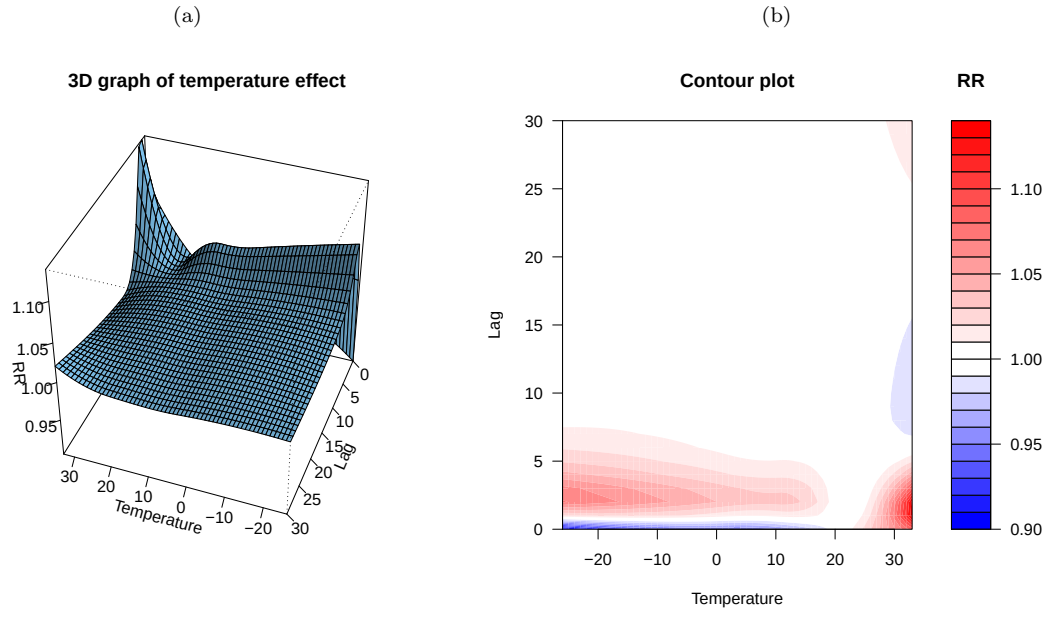
The chosen basis functions for the space of the predictor are a linear function for the effect of PM_{10} and a quadratic B-spline (`type="bs"`) with 5 degrees of freedom for temperature (with `knots` placed by default at equally spaced quantiles in the space of the predictor). The basis for temperature 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, with minimum lag equal to 0 by default, keeping the default values of `df=1`), while we define another cubic spline, this time with the natural constraint (`type="ns"` by default) for the lag dimension of temperature. For this space, `knots` are located by default at equally spaced values in the log scale of lags, while the boundary knots are set to 0 and `lag=30`. The estimation, prediction and plotting of the effects of temperature are performed by:

```
> model3 <- glm(death ~  cb3.pm + cb3.temp + ns(time, 7*14) + dow,
    family=quasipoisson(), chicagoNMMAPS)
> pred3.temp <- crosspred(cb3.temp, model3, by=1)
> plot(pred3.temp, xlab="Temperature", zlab="RR", theta=200, phi=40, lphi=30,
         main="3D graph of temperature effect")
> plot(pred3.temp, "contour", xlab="Temperature", key.title=title("RR"),
         plot.title=title("Contour plot",xlab="Temperature",ylab="Lag"))
```

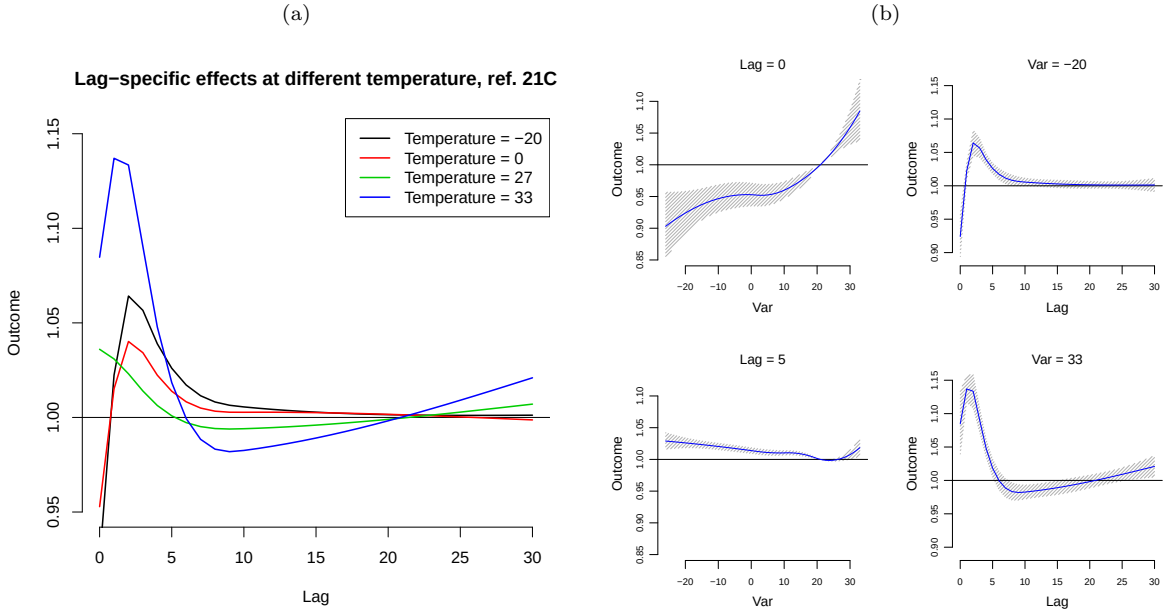
Note that prediction values are chosen only with the argument `by=1` in `crosspred()`, defining all the integer values within the predictor range. The first plotting expression produces a 3-D plot illustrated in Figure 3a, with non-default choices for perspective and lightning obtained through the arguments `theta-phi-lphi`. The second plotting expression specifies the contour plot in Figure 3b with titles and axis labels chosen by arguments `plot.title` and `key.title`. The user can find additional information and a complete list of arguments in the help pages of the original high-level plotting functions (typing *`?persp`* and *`?filled.contour`*).

Figure 3



Plots in Figures 3a – 3b offer a comprehensive summary of the bi-dimensional relationship, but are limited in their ability to inform on effects at specific values of predictor or lags. In addition, they are also limited for inferential purposes, as the uncertainty of the estimated effects is not reported in 3-D

Figure 4



and contour plots. A more detailed analysis is provided by plotting "slices" of the effect surface for specific predictor and lag values. The code is:

```
> plot(pred3.temp, "slices", var=-20, ci="n", col=1, ylim=c(0.95,1.15), lwd=1.5,
          main="Lag-specific effects at different temperature, ref. 21C")
> for(i in 1:3) lines(pred3.temp, "slices", var=c(0,27,33)[i], col=i+1, lwd=1.5)
> legend("topright",paste("Temperature =",c(-20,0,27,33)), col=1:4, lwd=1.5)
> plot(pred3.temp, "slices", var=c(-20,33), lag=c(0,5), col=4,
    ci.arg=list(density=40,col=grey(0.7)))
```

The results are reported in Figures 4a – 4b. Figure 4a illustrates lag-specific effects for mild and extreme cold and hot temperatures of -20°C, 0°C, 27°C, and 33°C (with reference at 21°C). Figures 4b depicts both effects along the predictor range at lag 0 and 5 (left column), and effects along lags at temperatures -20°C and 33°C (right column). The arguments `var` and `lag` define the "slices" to be cut in the effect surface in Figure 3a – 3b. The argument `ci="n"` in the first expression states that confidence intervals must not be plotted. In the multi-panel Figure 4b, the list argument `ci.arg` is used to plot confidence intervals as shading lines with increased grey contrast, more visible here.

The preliminary interpretation suggests that cold temperatures are associated with longer mortality risk than heat, but not immediate, showing a "protective" effect at lag 0. This analytical proficiency would be hardly achieved with simpler models, probably losing important details of the association.

5.5 Example 4: reduce a DLNM

In this last example, we show how we can reduce the fit of a bi-dimensional DLNM to summaries expressed by parameters of one-dimensional basis, using the function `crossreduce()`. First, we specify a new cross-basis matrix, run the model and predict the effects in the usual way:

```
> cb4 <- crossbasis(chicagoNMMAPS$temp, lag=30, argvar=list(type="dthr",
    knots=c(10,25)), arglag=list(df=5))
> model4 <- glm(death ~  cb4 + ns(time, 7*14) + dow,
    family=quasipoisson(), chicagoNMMAPS)
> pred4 <- crosspred(cb4, model4, by=1)
```

The specified cross-basis for temperature is composed by double-threshold functions with cut-off points at 10°C and 25°C for the dimension of the predictor, and a natural cubic splines with 5 df and knots at default equall-spaced values in the log scale for lags, respectively. The reduction may be carried out to 3 specific summaries, namely overall, lag-specific and predictor-specific associations. This is the code:

```
> redall <- crossreduce(cb4, model4)
> redlag <- crossreduce(cb4, model4, type="lag", value=5)
> redvar <- crossreduce(cb4, model4, type="var", value=33)
```

The reduction for specific effects is computed for lag 5 and 33°C in the two spaces, repectively. The 3 objects of class *"crossreduce"* contain the modified reduced parameters for the one-dimensional basis in the related space, which can be compared with the original model:

```
> length(coef(pred4))

[1] 10

> length(coef(redall)) ; length(coef(redlag))

[1] 2

[1] 2

> length(coef(redvar))

[1] 5
```

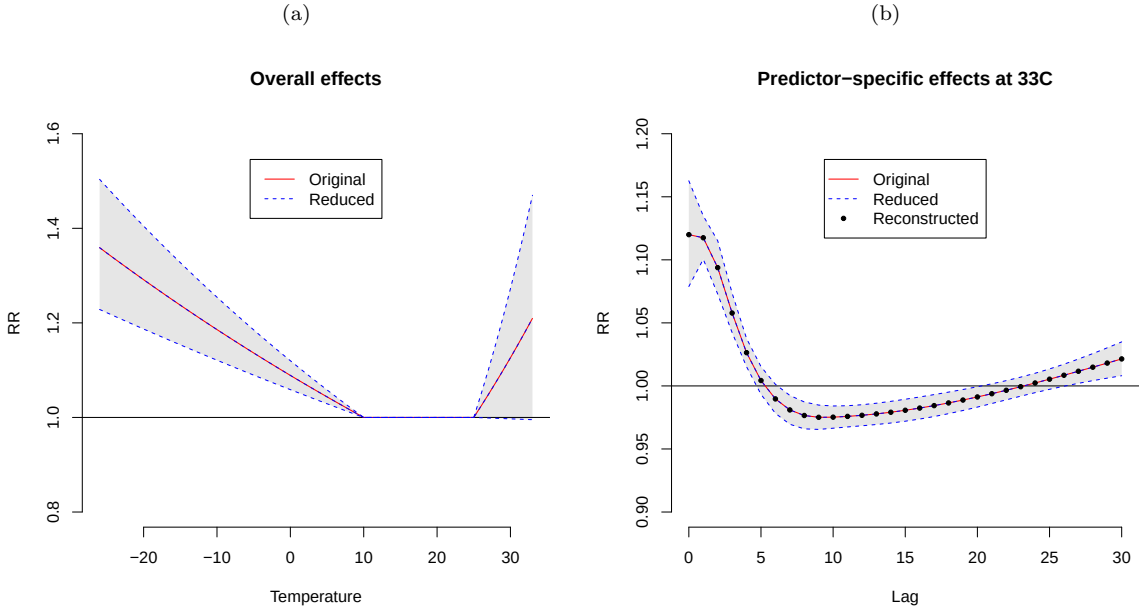
As expected, the number of parameters has been reduced to 2 for the space of the predictor (coherently with the double-threshold parameterization), and to 5 for the space of lags (coherently with the dimension of the natural cubic spline basis). However, the prediction from the original and reduced fit is identical, as illustrated in Figure 5a produced by:

```
> plot(pred4, "overall", xlab="Temperature", ylab="RR",
    ylim=c(0.8,1.6), main="Overall effects")
> lines(redall, ci="lines",col=4,lty=2)
> legend("top",c("Original","Reduced"),col=c(2,4),lty=1:2,ins=0.1)
```

The process may also be clarified by re-constructing the orginal one-dimensional basis and predicting the effects given the modified parameters. As an example, we reproduce the natural cubic spline for the space of the lag using `onebasis()`, and predict the results, with:

```
> b4 <- onebasis(0:30,knots=attributes(cb4)$arglag$knots,int=TRUE,cen=FALSE)
> pred4b <- crosspred(b4,coef=coef(redvar),vcov=vcov(redvar),model.link="log",by=1)
```

Figure 5



The spline basis is computed on the integer values corresponding to lag `0:30`, with knots at the same values as the original cross-basis, and uncentered with intercept as the default for basis for lags (see Section 4.2). Prediction is computed using the modified parameters reduced to predictor-specific association for 33°C. The identical fit of the original, reduced and re-constructed prediction is illustrated in Figure 5b, produced by:

```
> plot(pred4, "slices", var=33, ylab="RR", ylim=c(0.9,1.2),
    main="Predictor-specific effects at 33C")
> lines(redvar, ci="lines", col=4, lty=2)
> points(pred4b, col=1, pch=19, cex=0.6)
> legend("top",c("Original","Reduced","Reconstructed"),col=c(2,4,1),lty=c(1:2,NA),
    pch=c(NA,NA,19),pt.cex=0.6,ins=0.1)
```

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, 2011; 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 is supported by the Medical Research Council (UK), Research Grants RES-G0707030 and RES-G1002296.

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). The data used in the package was collected and made freely available by the NMMAPS researchers, and reproduced with their permission. We are thankful to the colleagues who tested beta-versions of this package and suggested some important improvements included in the current version (in particular Marie-France Valois, Adrian Barnett and Michela Leone).

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. Distributed lag linear and non-linear models in R: the package dlnm. *Journal of Statistical Software*, 43(8):1–20, 2011. URL http://www.jstatsoft.org/v43/i08/.

A. Gasparrini, B. Armstrong, and M. G. Kenward. Distributed lag non-linear models. *Statistics in Medicine*, 29(21):2224–2234, 2010.

A. Gasparrini, B. Armstrong, and M. G. Kenward. Reducing and meta-analyzing estimates from distributed lag non-linear models. (Submitted), 2012.

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.