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

Antonio Gasparrini
London School of Hygiene and Tropical Medicine, UK

dlnm version 2.2.6 , 2016-05-10

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 downloaded together with the package and can be simply accessed through R by typing `vignette("dlnmOverview")`.

1 Preamble

The R package dlnm offers some facilities to run *distributed lag non-linear models*, a modelling framework to describe simultaneously non-linear and delayed effects between predictors and an outcome, a dependency defined as *exposure-lag-response association*. This document complements the description of the package provided in Gasparrini [2011] (freely available at `http://www.jstatsoft.org/v43/i08/`), which represents the main reference to the package. The DLNM's methodology has been previously described in Gasparrini [2014], Gasparrini and Armstrong [2013], 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].

This document **dlnmOverview** is the main of three vignettes documenting the package. Its aim is to describe the methodology and to provide an overview of the main functions. Two other vignettes offer more specific examples. Specifically, the vignette **dlnmTS** illustrates the application of the package dlnm for time series analysis, where distributed lag linear and non-linear models (DLMs and DLNMs) where originally conceived, as described in Gasparrini et al. [2010]. The vignette **dlnmExtended** demonstrates how the same functions in dlnm can be applied to extend the DLNM methodology beyond time series, for example to study exposure-lag-response associations in cohort or case-control designs. This extension is illustrated in Gasparrini [2014]. Each vignette, included in the package installation, can be opened in R by typing *`vignette("namevignette")`*.

Type *`citation("dlnm")`* in R to cite the dlnm package after installation (see Section 2). General information on the development and applications of the DLNM modelling framework, together with an updated version of the R scripts for running the examples in published papers, can be found at www.ag-myresearch.com.

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

2 Installation and data

2.1 Installing and loading the package dlnm

The dlnm package is installed in the standard way for CRAN packages from R version 2.9.0 onwards, for example typing *`install.packages("dlnm")`* or directly through the R menu in Windows. The package can be alternatively installed using the source code (.tar.gz) or binaries for Windows or MacOS, available at http://cran.r-project.org/web/packages/dlnm/index.html.

The package is loaded in the R session by:

```
> library(dlnm)
```

A list of changes included in the current and previous versions can be found typing:

```
> file.show(system.file("ChangeLog", package="dlnm"))
```

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. In this version, the functions imported from contributed packages are `ns()` and `bs()` from splines, `Lag()` from tsModel, and `fixef()` from nlme, three of the recommended packages included in the main distribution of R.

2.2 Data

This version the package includes the three data sets `chicagoNMMAPS`, `nested` and `drug`. The former is used in the vignette **dlnmTS** as an example of use in time series analysis. The other two are used in the vignette **dlnmExtended** as an example of the extension of the methodology and package in other study designs.

The data set `chicagoNMMAPS` contains 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 used to be downloadable from the package NMMAPSlite (now archived) and from Internet-based Health and Air Pollution Surveillance System (iHAPSS) website (www.ihapss.jhsph.edu).

The data set `nested` contains simulated data from an hypothetical nested case-control study on the association between a time-varying occupational exposure and a cancer outcome. The study includes 250 risk sets, each with a case and a control matched by age year. The data on the exposure is collected on 5-year age intervals between 15 and 65 years.

The data set `drug` contains simulated data from an hypothetical randomized controlled trial on the effect of time-varying doses of a drug. The study includes 200 randomized subject, each receiving daily doses of drug for 28 days, varying each week. The exposure level is reported on 7-day intervals.

3 The DLNM methodology

The conceptual and methodological development of distributed lag linear and non-linear models (DLMs and DLNMs) is thoroughly described in a series of publications. Here I provide a brief summary to introduce concepts and definitions. The user can refer to the articles provided below for a more detailed description.

3.1 Exposure-lag-response associations

The modelling class of DLNMs is applied to describe associations in which the dependency between an exposure and an outcome is delayed in time. This process can be described using two different and complementary perspectives. Using a *forward perspective*, we can may say that an exposure event at time t determines the risk in the future at times $t + \ell$. Using a *backward perspective*, the risk at time t is determined by a series of exposures experienced in the past at times $t - \ell$. Here ℓ is the *lag*, expressing the delay between exposure and measured outcome.

The lag dimension represents a new space over which the association is defined, by describing a *lag-response relationship* in addition to the usual *exposure-response relationship* over the space of the predictor. The dependency, characterized in the bi-dimensional space of predictor and lag, is defined here as *exposure-lag-response* association, revising a terminology previously proposed by Thomas [1988].

Exposure-lag-response associations are modelled through an extended version of DLNM. Gasparrini [2014] provides the methodological development and the definitions above.

3.2 A statistical model for DLNMs

The DLNM class provides a conceptual and analytical framework for the description and estimation of exposure-lag-response associations. A statistical development of DLNMs is based on the choice

of a *basis* for each dimension of predictor and lag [Wood, 2006], to describe an exposure-lag-response associations through known transformations of predictor and lag vectors, combining the usual *exposure-response function* with the additional *lag-response* function.

A simple case of exposure-lag-response associations is when the relationship in the space of the predictor, namely the exposure-response relationship, is linear. This type of relationship can be modelled through a DLM. In this case, the association only depends on the lag-response function, which models how the linear risk changes along lags. When such function is applied to the data, it provides a *lag-basis function* and related matrix. Different choices for the lag-response function (splines, polynomials, strata, threshold, among others) lead to the specification of different DLMs, and imply alternative assumptions on the lag-response relationship.

The DLNM class for full exposure-lag-response associations is based on an intuitive extension of DLMs, with the definition of a non-linear exposure-response function. Similarly, this step is provided by the independent choice of another basis for the space of the predictor. The simultaneous application of the two sets of basis functions and their combination through a special tensor product provides *cross-basis functions* and related matrix. A lag-basis function is a special case of cross-basis function with a linear exposure-response function. Again, the choice of the bases for exposure-response and lag-response functions implies alternative assumptions on how the risk changes along the two dimensions.

The lag or cross-basis matrix can be included in the design matrix of a regression model in order to estimate the related parameters. The algebraic definitions of the models above are provided in Gasparrini et al. [2010] and Gasparrini [2014].

3.3 Interpretation of DLNMs

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 3-D plots to provide an overall picture of the association varying along the two dimensions. Also, three *summaries* of the bi-dimensional association are of interest, each of which can be interpreted using the forward and backward perspective defined in Section **3.1**.

The first is the *lag-response curve* associated with a specific exposure value, defined as a *predictor-specific association*. Using a forward perspective, this is interpreted as the series of contribution to risk at times $t + \ell$ associated to a specific exposure at time t. Using a backward perspective, this is interpreted as the contribution to the risk at time t associated to the exposures experiences at times $t - \ell$.

The second is the *exposure-response curve* associated with a specific lag value, defined as a *lag-specific association*. Using a forward perspective, this is interpreted as the exposure-response relationship at time $t + \ell$ associated to exposure values occurring at time t. Using a backward perspective, this is interpreted as the exposure-response relationship at time t associated to exposure values experienced at time $t - \ell$.

The third and most important is the *exposure-response curve* associated to the whole *exposure history* experienced within the lag period considered, defined as a *overall cumulative association*. Using a forward perspective, this is interpreted as the exposure-response relationship representing the net risk experienced over the period $[t, t + L]$ for a given exposure that occurred at time t. Using a backward perspective, this is interpreted as the exposure-response relationship at time t for a constant exposure experiences during the period $[t - L, t]$.

Again, the algebraic definitions of the models above are provided in Gasparrini et al. [2010] and Gasparrini [2014].

A fitted DLNM can be re-expressed in terms of such summaries (predictor-specific, lag-specific or overall cumulative association) by *reducing* its parameters to those of the uni-dimensional lag-response

or exposure-response function only. This step is described in Gasparrini and Armstrong [2013].

3.4 Generalization beyond time series design

Distributed lag models were firstly conceived in econometric time series analysis long ago [Almon, 1965], and then re-proposed in time series data within environmental epidemiology Schwartz [2000]. The extension to DLNMs were conceived by Armstrong [2006]. A re-evaluation of this modelling framework for time series data is given in Gasparrini et al. [2010].

Interestingly, models for such exposure-lag-response associations have been proposed in different research fields. The general idea is to *weight* past exposures through specific functions whose parameters are estimated by the data. Models for linear-exposure-response relationships similar to DLMs were illustrated in cancer epidemiology [Hauptmann et al., 2000, Langholz et al., 1999, Richardson, 2009, Thomas, 1983, Vacek, 1997] and pharmaco-epidemiology [Abrahamowicz et al., 2012, Sylvestre and Abrahamowicz, 2009]. Extensions to non-linear exposure-responses were also proposed [Abrahamowicz and MacKenzie, 2007, Berhane et al., 2008, Vacek, 1997]

The extension of DLNM beyond time series data and the implementation in the package dlnm provide a general conceptual and modelling framework which generalizes all the developments above. The methodology and software can be applied in different study designs and data structures. This extension of DLNMs is described in Gasparrini [2014].

4 The functions in the package **dlnm**

This section describes the main functions included in the package dlnm. Here I provide a brief description of all the stages involved in the definition, estimation and interpretation of DLNMs, summarizing the conceptual and analytical steps. In addition, I illustrate the structure of the functions and discuss specific issues about their usage. Examples of applications to real data, with a more detailed overview of the use of the functions in time series analysis and beyond, are described in the vignettes **dlnmTS** and **dlnmExtended** respectively. Examples involving the old usage of the functions are also provided in Gasparrini [2011].

4.1 Basic functions

The package dlnm contains basic functions to specify standard exposure-response and lag-response relationships, such polynomials, step or threshold functions. These functions are not exported to the namespace in order to prevent conflicts with other existing functions in recommended packages, and are only meant to be used internally through `onebasis()` (see Section 4.2) and `crossbasis()` (see Section 4.3).

Other existing or user-defined functions can also be used within the framework of dlnm, as shown in Section 4.2. For example, splines are specified by the functions `ns()` and `bs()` included in the recommended package splines. The package is installed together with the main R distribution. The user can refer to the related help pages of these functions for further info (type *?ns* or *?bs* in R).

Polynomials are obtained through the function `poly()`. As mentioned earlier, this function is not exported in the namespace in order to avoid conflicts with the function with the same name included in the package stats. The user can call this or other functions described below by using the triple colon operator ':::'. This is an example of transformation of a simple vector:

```
> dlnm:::poly(1:5,degree=3)
```

```
       1    2     3
[1,] 0.2 0.04 0.008
[2,] 0.4 0.16 0.064
[3,] 0.6 0.36 0.216
[4,] 0.8 0.64 0.512
[5,] 1.0 1.00 1.000
attr(,"degree")
[1] 3
attr(,"scale")
[1] 5
attr(,"intercept")
[1] FALSE
attr(,"class")
[1] "poly"   "matrix"
```

The result is a basis matrix with additional class *"poly"*, with attributes storing the info on the arguments exactly defining the transformation. The first unnamed argument `x` specifies the vector to be transformed, while the argument `degree` sets the degree of the polynomial. Other arguments let to default values are `scale` (a scaling factor) and `intercept` (a logical value specifying if an intercept should be included). See *`?poly`* for further info.

Step functions defining strata are specified through `strata()` (again, be aware that this function is different from the other one with the same name in the recommended package survival). An example is shown below, with square brackets preventing the printing of attributes to save space:

```
> dlnm:::strata(1:5,breaks=c(2,4))[,]

     1 2
[1,] 0 0
[2,] 1 0
[3,] 1 0
[4,] 0 1
[5,] 0 1
```

The result is a basis matrix with additional class *"strata"*. The transformation is a dummy parameterization defining contrasts. The argument `breaks` defines lower boundaries for right-open intervals of the strata. Other arguments of `strata()` are `df`, which sets the number of intervals (with cut-offs at equally-spaced percentiles) when `breaks` is undefined, `ref` to select the refence interval, and `intercept` to include the intercept. See *`?strata`* for further info.

Threshold functions are specified through `thr()`. An example:

```
> dlnm:::thr(1:5,thr.value=3,side="d")[,]

     1 2
[1,] 2 0
[2,] 1 0
[3,] 0 0
[4,] 0 1
[5,] 0 2
```

The result is a basis matrix with additional class *"thr"*. The argument `thr.value` defines a vector with one or two thresholds, while `side` is used to specify high (`"h"`, the default), low (`"l"`) or double (`"d"` threshold parameterizations. The default value for `side` is `"h"` or `"d"` depending on the length of `thr.value`. As above, an intercept can be included with the argument `intercept`. See *`?thr`* for further info.

Two special functions are `lin()` and `integer()`. The former returns the untransformed variable (optionally with intercept), and is used to specify DLM with linear exposure-response relationships via `crossbasis()`. The latter returns an identity matrix (minus the first column if an intercept is not included) for specifying unconstrained DLMs and DLNMs, where the lag-response functions includes one parameter per lag. Again, this is different from the function with the same name included in the package base. See *`?lin`* and *`?integer`* for further info.

4.2 The function `onebasis()`

This function represents the workhorse for basis transformation in dlnm, and it is applied for the specification of exposure-response and lag-response relationships. It has replaced the old functions `mkbasis()` and `mklagbasis()` since version 1.5.1 of dlnm. Its role is to apply chosen transformations and generate basis matrices in a format suitable for other functions such as `crossbasis()` and `crosspred()`. However, the function can be also used directly for generating uni-dimensional functions in regression models, with predictions and graphs derived by `crosspred()` (see Section 4.4) and plotting methods (see Section 4.6).

Since version 2.0.0 of dlnm, `onebasis()` simply acts as a wrapper to other functions, such as those described in Section 4.1. The following example replicate the polynomial transformation shown in that section:

```
> onebasis(1:5,fun="poly",degree=3)

      b1   b2    b3
[1,] 0.2 0.04 0.008
[2,] 0.4 0.16 0.064
[3,] 0.6 0.36 0.216
[4,] 0.8 0.64 0.512
[5,] 1.0 1.00 1.000
attr(,"fun")
[1] "poly"
attr(,"degree")
[1] 3
attr(,"scale")
[1] 5
attr(,"intercept")
[1] FALSE
attr(,"class")
[1] "onebasis" "matrix"
attr(,"range")
[1] 1 5
```

The result is a basis matrix with additional class *"onebasis"*. Again, the first unnamed argument `x` specifies the vector to be transformed, while the second argument `fun` defines the name of the function to be called for applying the transformation, as a character. The result is very similar to the call to

`poly()`, although other attributes are stored, and used later for simplifying predictions and plotting. Specifically, the basis matrix includes the attributes `fun` and `range`, plus the arguments of the called function which exactly define the transformation.

As mentioned earlier, `onebasis()` can also call other existing or user-defined functions, with specific requirements. A simple example:

```
> mylog <- function(x) log(x)
> onebasis(1:5,"mylog")

            b1
[1,] 0.0000000
[2,] 0.6931472
[3,] 1.0986123
[4,] 1.3862944
[5,] 1.6094379
attr(,"fun")
[1] "mylog"
attr(,"range")
[1] 1 5
attr(,"class")
[1] "onebasis" "matrix"
```

The called function must have `x` as its first argument and being a closure (*i.e.* primitive functions such as `log()` itself are not accepted). In addition, it must return a vector or matrix of transformed basis variables together with attributes storing the arguments defining the function. More detailed examples of the usage with user-defined functions are illustrated in the vignette **dlnmExtended**.

A method function `summary()` for objects of class *"onebasis"* summarizes the function and the content and the basis matrix. See *`?onebasis`* for further info.

4.3 The function `crossbasis()`

This is the main function in the package dlnm. It internally calls `onebasis()` to generate the basis matrices for exposure-response and lag-response relationships, and combines them through a special tensor product in order to create the cross-basis, which specifies the exposure-lag-response dependency simultaneously in the two dimensions. See Gasparrini [2014, Sections 2.1–2.2] and Gasparrini et al. [2010, Sections 4.1–4.2] for details.

The class of its first argument `x` defines how the data are interpreted. If a n-dimensional vector, `x` is assumed to represent an equally-spaced, complete and ordered series of observations in a time series framework. If a $n \times (L - \ell_0 + 1)$ matrix, `x` is assumed to represent a series of exposure histories for n observations over the lag period from ℓ_0 to L. The latter can be used to extend DLNMs beyond time series data, as thoroughly illustrated in the vignette **dlnmExtended**. The lag period can be modified with the second argument `lag`. The two arguments `argvar` and `arglag` contain lists of arguments, each of them to be passed to `onebasis()` to build the matrices for the exposure-response and lag-response relationships respectively (see Section 4.2). The additional argument `group`, used only for time series data defines groups of observations to be considered as individual unrelated series, and may be useful for example in seasonal analyses (see the vignette **dlnmTS**).

As a simple example, I simulate a matrix of exposure histories for 3 subjects over the lag period 2–5:

```
> hist <- matrix(sample(1:12),3,dimnames=list(paste("sub",1:3,sep=""),
```

```
    paste("lag",2:5,sep="")))
> hist

     lag2 lag3 lag4 lag5
sub1    1    3    5    6
sub2    7    8    9    4
sub3   10    2   11   12
```

Then, I apply the cross-basis parameterization, with a quadratic polynomial as the exposure-response function and a step function defining strata 2–3 and 4–5 as the lag-response function:

```
> crossbasis(hist,lag=c(2,5),argvar=list(fun="poly",degree=2),
    arglag=list(fun="strata",breaks=4))[,]

        v1.l1     v1.l2     v2.l1     v2.l2
sub1 1.250000 0.9166667 0.4930556 0.4236111
sub2 2.333333 1.0833333 1.4583333 0.6736111
sub3 2.916667 1.9166667 2.5625000 1.8402778
```

The function returns a matrix object of class *"crossbasis"* (again, printing the long series of attributes if prevented by th use of the square brackets). It first calls `onebasis()` with arguments in the lists `argvar` and `arglag` to build the matrix bases for exposure-response and lag-response spaces. In particular, the basis matrix for the lag space by default includes an intercept (see `?poly` and `?strata` for this specific example). The two matrices of dimension 2 are then combined through a special tensor product in the final cross-basis matrix of dimension $2 \times 2 = 4$, with column labels identifying the product. The cross-basis object also included the attributes `df`, `range`, `lag`, `argvar` and `arglag`, not printed here by using the square brackets. The latter two can be different from the specified arguments due to internal checks and setting of default values.

The function can also be used with time series data, simply including a vector with the series as the first argument `x`. In another example, I apply `crossbasis()` to the variable `temp` in the data set `chicagoNMMAPS`, representing the series of daily mean temperature in Chicago during the period 1987–2000:

```
> cb <- crossbasis(chicagoNMMAPS$temp,lag=30,argvar=list("thr",thr.value=c(10,20)),
    arglag=list(knots=c(1,4,12)))
> summary(cb)

CROSSBASIS FUNCTIONS
observations: 5114
range: -26.66667 to 33.33333
lag period: 0 30
total df:  10

BASIS FOR VAR:
fun: thr
thr.value: 10 20
side: d
intercept: FALSE

BASIS FOR LAG:
```

```
fun: ns
knots: 1 4 12
intercept: TRUE
Boundary.knots: 0 30
```

Here the exposure-response is modelled as a a double threshold function with thresholds at 10 and 20, as the argument `side` of `thr()` is set by default to `"d"` if two values are provided in `thr.value`. The lag period is fixed to 0 to 30. The lag-response function is left to the default natural cubic spline (`fun="ns"`) with knots at 1, 4 and 12 lags. As the default for the space of lag, this function includes the intercept. The method function `summary()` for `crossbasis` objects provides an overview of the transformation, and can be used to check the results.

Missing values are properly handled by `crossbasis()`. Specifically, if `x` is provided as a matrix of exposure history, a missing value causes the related row in the transformed matrix to be set to `NA`. In time series data, a missing value in the vector series causes all the following rows corresponding to the lag period to be set to `NA`.

The usage of the function has repeatedly changed in different versions of the package dlnm. The user is advised to follow the usage in the last available version.

4.4 The function `crosspred()`

The cross-basis matrix produced by `crossbasis()` needs to be included in a regression model formula in order to fit a 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 association for a grid of combinations of predictor and lag values, chosen by default or directly by the user. The function creates the same basis or cross-basis functions for the chosen predictor and lag values, based on the attributes of the original basis or cross-basis matrix, and generates predictions (with associated standard errors and confidence intervals) by extracting the related parameters estimated in the model (see Gasparrini [2014, Section 2.3] and Gasparrini et al. [2010, Section 4.3] for algebraic details).

As an example, I use the cross-basis matrix `cb` created in Section 4.3 to study the association between temperature and cardiovascular mortality, using the time series data in the data set `chicagoNMMAPS`. First, I fit a simple linear model with the cross-basis matrix included in the model formula (the model proposed here is only chosen for illustrative purposes). Then, I obtain the predictions by calling `crosspred()` with the cross-basis and regression model objects as the first two arguments:

```
> model <- lm(cvd~cb,chicagoNMMAPS)
> pred <- crosspred(cb,model,at=-20:30)
```

The result is a list object of class *"crosspred"*, with components storing the predictions and other information about the model, such as coefficients and part of associated (co)variance matrix related to the parameters of the cross-basis. The grid of predictions can be chosen for specific predictor-lag combinations. The values of the predictor are selected with the argument `at` as above, 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 arguments `lag` and `bylag` (not used here) determine instead the range and increment of the sequence of lag values used for prediction, by default the series of integers used for estimation.

Predictions are computed versus a reference value, with default values dependent on the function used for modelling the exposure-response, or manually set through the argument `cen` of `crosspred()`.

Briefly, sensible default values are automatically defined for `strata()`, `thr()` and `integer()` (corresponding to the reference region), and for `lin()` (corresponding to 0). For other choices, such as `ns()`, `bs()`, `poly()` or other existing or user-defined functions, the centering value is set by default to the mid-range. In this case, no centering is needed, as the function `thr()` has an default selection of the reference.

The predictions can be accessed directly through the components of the `crosspred` list. In particular, components with names starting with `mat-` store the matrices of prediction over the grid, namely rows with lag-response relationships for specific predictor values or columns with exposure-response relationships for specific lag-values. Instead, components `all-` include vectors with the overall cumulative predictions. As an example, I extract the prediction and confidence interval for temperature -10°C and lag 5, and then the overall cumulative prediction for 25°C:

```
> c(pred$matfit["-10","lag5"],pred$matlow["-10","lag5"],pred$mathigh["-10","lag5"])

[1] 0.9464757 0.6773841 1.2155673

> pred$allfit["25"]

      25
1.108262
```

The results are interpreted following the forward and backward perspectives illustrated in Section 3.1. The first result indicates that a temperature of -20°C in a given day produces an estimated increase of 0.95 cardiovascular deaths five days later (forward), or that the number of cardiovascular deaths in a given day are increased by 0.95 from a temperature of -20°C experienced five days earlier (backward), compared to the reference temperature 10–20°C. The second figure indicates an overall increase of 1.11 cardiovascular deaths within the next 30 days for an exposure to 25°C in a given day (forward), or an overall increase of 1.11 cardiovascular deaths in a given day following a constant exposure to 25°C in the previous 30 days (backward).

Other types of predictions can be obtained through `crosspred()`. In particular, exponentiated predictions are automatically returned if the model link is equal to `log` or `logit`, and stored in components `matRR-` and `allRR-`. Matrices `cum-` of incremental cumulative predictions are optionally included if the argument `cum` is set to `TRUE`, and stored in components `cum-`. The user can refer to *`help(crosspred)`* for a complete list of components.

An alternative use of `crosspred()` is to predict the effect of specific sets of exposure histories. This can be achieved by inputting a matrix of exposure histories as the argument `at`. For example, we can predict from the fitted model which is the overall cumulative increase in cardiovascular deaths following an exposure to 30°C in the last 10 days and 22°C in the rest of the lag period:

```
> histpred <- t(c(rep(30,10),rep(22,21)))
> crosspred(cb,model,at=histpred)$allfit

       1
5.934992
```

The user should be careful in inputting a matrix as the argument `at`, which is interpreted by `crosspred()` as a series of exposure histories and not as a vector of predictor values as above. A more detailed overview of this prediction method is given in the vignette **dlnmExtended**.

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 function `crosspred()` automatically works with model objects from regression function `lm()` and `glm()`, `gam()` (package mgcv), `coxph()` and `clogit()` (package survival), `lme()` and `nlme()` (package nlme), `lmer()` and `glmer()` and `nlmer()` (package lme4), `gee()` (package gee), `geeglm()` (package geepack). The function also works with any regression function for which `coef()` and `vcov()` methods are available. Otherwise, the user needs to input the coefficients and associated (co)variance matrix related to the parameters of the cross-basis as arguments `coef` and `vcov`, and the model link in `model.link`.

Multiple cross-basis matrices associated with different predictors may be included in `model`, and predictions for each of them can be computed with `crosspred()`. Also, the function `crosspred()` can produce predictions for simple uni-dimensional functions produced by `onebasis()`.

4.5 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, expressing summaries such as overall cumulative exposure-responses, lag-specific exposure-responses, or predictor-specific lag-responses only in terms of the original basis functions chosen for the related space, and modified associated parameters. The method is thoroughly illustrated in Gasparrini and Armstrong [2013].

This computation is carried out through the function `crossreduce()`, which works very similarly to `crosspred()` (see Section 4.4). The first two arguments `basis` and `model` 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 cumulative exposure-responses, lag-specific exposure-responses or predictor-specific lag-responses, respectively. The single value of predictor or lags for which predictor-specific or lag-specific summaries 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 4.4).

The function `crossreduce()` 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 uni-dimensional basis matrix, and the vectors of predictions and associated standard errors, optionally exponentiated.

An illustrative example of the use of the function in given in the vignette **dlnmTS**.

4.6 Plotting functions

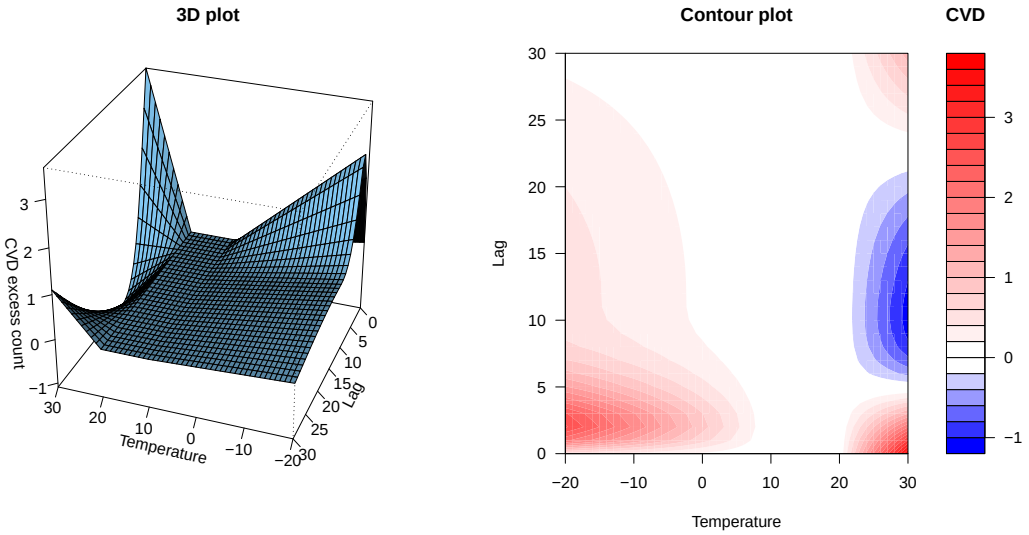
Interpretation of one-dimensional or bi-dimensional associations is aided by graphical representation. High and low-level plotting functions are provided through the method functions `plot()`, `lines()` and `points()` for classes *"crosspred"* and *"crossreduce"*. These methods have replaced the old function `crossplot()` since version 1.3.0, providing the user with the possibility of specifying the whole range or arguments of the plotting functions above, allowing complete flexibility in the choices of colours, axes, labels and other graphical parameters. As an example, I use the predictions stored in the object `pred` defined in Section 4.4.

The `plot()` method can generate different types of plots for *"crosspred"* objects through the argument `ptype`. Specifically, it generates graphs of the entire bi-dimensional exposure-lag-response association, or of exposure-response or lag-response summaries defined in Section 3.3. Bi-dimensional associations are plotter as 3-D or contour graphs, as for example:

```
> plot(pred,ptype="3d",main="3D plot",xlab="Temperature",zlab="CVD excess count",
```

Figure 1

(a) (b)




```
    theta=200, ltheta=180)
> plot(pred,ptype="contour",key.title=title("CVD"),
    plot.title=title("Contour plot",xlab="Temperature",ylab="Lag"))
```

The two plots are shown in Figures 1a and 1b. The `plot()` method function internally calls the functions `persp()` and `filled.contour()` when `ptype` is equal to `"3d"` and `"contour"`, respectively. The user can include in the function call also other arguments specific to each function, such as `main`-`xlag`-`zlag`-`theta`-`ltheta` and `key.title`-`plot.title` above. See *?persp* and *?filled.contour* for a complete list of arguments.

The two plots in Figure 1 summarize the whole exposure-lag-response association. The risk surface reflects the choice of a threshold function for the exposure-response relationship and a smooth spline function for the lag-response relationship. Summaries of the associations defined in Section 3.3 can be obtained by a different choice of `ptype`. In particular, `ptype="overall"` is used for plotting the exposure-response relationship expressing the overall cumulative association:

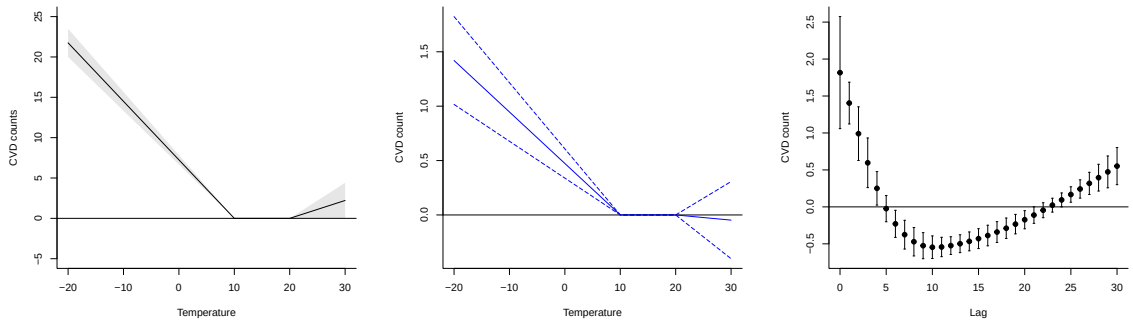
```
> plot(pred,"overall",xlab="Temperature",ylab="CVD counts",ylim=c(-5,25))
```

The plot is shown in Figure 2a. In this case, the method function `plot()` internally calls the function `plot.default()`, and as shown in the example above other specific arguments can be added to the function call. Lag-specific and predictor-specific associations can be plotted as exposure-response and lag-response relationships, respectively, by setting `ptype="slices"`, as they are *slices* in the 3-D surface cut along specific dimensions. For example:

```
> plot(pred,"slices",lag=5,xlab="Temperature",ylab="CVD count",
    col="blue",ci="lines",ci.arg=list(lty=5))
> plot(pred,"slices",var=25,xlab="Lag",ylab="CVD count",type="p",pch=19,ci="bars")
```

Figure 2

(a) (b) (c)



The two plots are shown in Figures 2b and 2c, and represent the exposure-response specific to lag 5 and the lag-response specific to a temperature of 25°C, respectively. The arguments `lag` and `var` specify the value for which lag and predictor-specific associations must be plotted, respectively. Again, additional arguments of the function `plot.default()` can be called as well, such as `col`, `type` and `pch`. The method function `plot()` also includes two additional arguments for plotting confidence intervals: the type is chosen by the argument `ci` among `"area"` (default), `"bars"` and `"lines"`, as shown in the three plots in Figure 2. 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. Incremental cumulative associations along lags are reported if `cumul=TRUE`: in this case, the same option must have been set to obtain the predictions (see Section 4.4).

Methods `lines()` and `points()` may be used as low-level plotting functions to add lines or points to an existing plot. When `crosspred()` is used for predictions of one-dimensional associations defined by `onebasis()`, only the plotting of the overall summary is meaningful and possible. The same plotting method functions are defined also for objects of class *"crossreduce"*. Examples are included in the vignette **dlnmTS**.

All the predictions are reported versus a reference value. For continuous functions, this is specified by the centering point, directly defined or set by default in `crosspred()`. Exponentiated predictions are automatically plotted if generated in the predictions, or forced with the argument `exp=TRUE`.

4.7 Other functions

The two functions `equalknots()` and `logknots()` are used to select knots or cut-off values for spline or strata functions at equally-spaced values and log-values, respectively. In particular, the latter is used to select knots for lag-response spline functions following the default used up to version 2.0.0 of dlnm, based on equally-spaced log values of lags.

The package dlnm also contains a set of functions which are called internally by the other functions illustrated above, in particular `onebasis()`, `crossbasis()` and `crosspred()`. Some of this functions are documented, and help pages are opened with the usual call (results not shown):

```
> help(getcoef)
```

The users bold enough to go through the source code of dlnm can access internal documented and undocumented functions through the use of the triple colon operator ':::' or through the function `getAnywhere()`. For example (results not shown):

```
> dlnm:::fci
> getAnywhere(fci)
```

5 Changes in the package dlnm

The package dlnm has experienced several changes since the first version 0.1.0, uploaded on CRAN on the 1st of July 2009, up to the current version 2.2.6. In particular, the new functions have been added or replaced and, more importantly, the usage of some of the existing functions has changed. An important change first occurred in the version 1.5.1, where the long set of arguments of `crossbasis()` were grouped in `argvar` and `arglag`. The second important development occurred instead in version 2.0.0, where the extension of the package described in the vignette **dlnmExtended** required a substantial restructure of the internal code of `onebasis()` and `crossbasis()`. This included changes in the default value of some arguments, the most important being the placement of knots values for spline bases in the space of lag to equally-spaced quantiles in the original scale, instead than equally-spaced values in the log scale. Another important change occurred in version 2.2.0, with the moving of the centering procedure from `onebasis()-crossbasis()` to `crosspred()-crossreduce()`. This change avoids the need to refit the model to obtain predictions with different reference values.

These changes are documented in the change-log and more extensively in two additional documents, accessed through:

```
> file.show(system.file("ChangeLog", package="dlnm"))
> file.show(system.file("Changesince151", package="dlnm"))
> file.show(system.file("Changesince200", package="dlnm"))
> file.show(system.file("Changesince220", package="dlnm"))
```

In particular, some of the changes may cause some of the R code written for old versions to produce different results with the updated versions. This marginally applies also to code included as supplementary material of published papers. An updated version of published code is available at www.ag-myresearch.com.

Such changes became unavoidable for the development of the dlnm package. Although further changes cannot be excluded in future versions, these will become less likely as long as the the package will take a definite structure.

6 Acknowledgements

The development of the package dlnm has been supported by the Medical Research Council (UK), through research grants with ID MR/M022625/1, G1002296 and G0701030.

I 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 the package gnm by Heather Turner and David Firth). The data used in the package were collected and made freely available by the NMMAPS researchers, and reproduced with their permission. I am thankful to the colleagues who have tested different versions of this package, suggesting improvements or identifying bugs (in particular Marie-France Valois, Adrian Barnett, Michela Leone, Yasushi Honda). Distributed

lag non-linear models were originally conceived by Ben Armstrong, who contributed greatly to the development of the dlnm package.

Finally, I express my gratitude to all the people working to develop and maintain the R Project.

References

M. Abrahamowicz and T. A. MacKenzie. Joint estimation of time-dependent and non-linear effects of continuous covariates on survival. *Statistics in Medicine*, 26(2):392–408, 2007.

M. Abrahamowicz, M. E. Beauchamp, and M. P. Sylvestre. Comparison of alternative models for linking drug exposure with adverse effects. *Statistics in Medicine*, 31(11-12):1014–1030, 2012.

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.

K. Berhane, M. Hauptmann, and B. Langholz. Using tensor product splines in modeling exposure-time-response relationships: Application to the Colorado Plateau Uranium Miners cohort. *Statistics in Medicine*, 27(26):5484–5496, 2008.

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. Modeling exposure-lag-response associations with distributed lag non-linear models. *Statistics in Medicine*, 33(5):881–899, 2014.

A. Gasparrini and B. Armstrong. Reducing and meta-analyzing estimates from distributed lag non-linear models. *BMC Medical Research Methodology*, 13(1):1, 2013.

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

M. Hauptmann, J. Wellmann, J. H. Lubin, P. S. Rosenberg, and L. Kreienbrock. Analysis of exposure-time-response relationships using a spline weight function. *Biometrics*, 56(4):1105–1108, 2000.

B. Langholz, D. Thomas, A. Xiang, and D. Stram. Latency analysis in epidemiologic studies of occupational exposures: application to the Colorado Plateau uranium miners cohort. *American Journal of Industrial Medicine*, 35(3):246–256, 1999.

D. B. Richardson. Latency models for analyses of protracted exposures. *Epidemiology*, 20(3):395–399, 2009.

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.

M. P. Sylvestre and M. Abrahamowicz. Flexible modeling of the cumulative effects of time-dependent exposures on the hazard. *Statistics in Medicine*, 28(27):3437–3453, 2009.

D. C. Thomas. Statistical methods for analyzing effects of temporal patterns of exposure on cancer risks. *Scandinavian Journal of Work, Environment & Health*, 9(4):353–366, 1983.

D. C. Thomas. Models for exposure-time-response relationships with applications to cancer epidemiology. *Annual Review of Public Health*, 9:451–482, 1988.

P. M. Vacek. Assessing the effect of intensity when exposure varies over time. *Statistics in Medicine*, 16(5):505–513, 1997.

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