Extensions of the dlnm package

Antonio Gasparrini
London School of Hygiene & Tropical Medicine, UK

dlnm version 2.4.7 , 2021-10-07

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("dlnmExtended")`*.

1 Preamble

This vignette DLNMEXTENDED illustrates recent extensions to the R package dlnm, some of them implementing development of the modelling framework of distributed lag linear and non-linear models (DLMs and DLNMs). Primarily, this document describes the generalization of the DLM/DLNM methodology beyond time series data, described in more detail in Gasparrini [2014]. In addition, this vignette illustrates other developments, specifically the definition of extended prediction summaries, the flexible application of existing or user-defined functions, and a more general use of the functions for regression analysis. The results included in this document are not meant to represent scientific findings, but are reported with the only purpose of illustrating the capabilities of the dlnm package.

A general overview of functions included in the package, with information on its installation and a brief summary of the DLNM methodology are included in the vignette DLNMOVERVIEW, which represents the main documentation of dlnm. The user can refer to that vignette for a general introduction to the package.

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

2 Data

These extensions of the software are illustrated mainly through two examples, using the data sets `drug` and `nested` included as data frames objects in the package. In particular, these data are ideal for illustrating the main development presented in this vignette, namely the extension of the modelling framework beyond time series.

These data sets contain simulated data from an hypothetical trial on a drug and a nested case-control study, respectively, both including measures of time-varying exposures. They are described in the related help pages, available by typing *help(drug)* or *help(nested)*, and in the main vignette DLNMOVERVIEW.

After loading the package in the R session, let's have a look at the first three observations of the data frame `drug`:

```
> library(dlnm)
> head(drug, 3)

  id out sex day1.7 day8.14 day15.21 day22.28
1  1  46   M      0       0       40       37
2  2  50   F      0      47       55        0
3  3   7   F     56      22        0        0
```

The data set contains data from a trial, with records for 200 randomized subjects, each receiving doses of a drug for two out of four random weeks, with daily doses varying each week. The exposure level is reported on 7-day intervals corresponding to each week. The data set contains also information on the outcome measured on the 28^{th} day and the sex of the subject.

The second data frame `nested` includes one record for each of 300 cancer cases and 300 controls matched by age. The first four observations are:

```
> head(nested, 4)

  id case age riskset exp15 exp20 exp25 exp30 exp35 exp40 exp45 exp50 exp55
1  1    1  81     240     5    84    34    45   128    81    14    52    11
```

```
2  2     1  69      129     11      8     25      6      8     12     19     60     16
3  3     1  73      180     14     15      7     69     10    143     18     19     44
4  4     0  52       19     10     16      5     30     24     33     14    122     NA
  exp60
1    16
2    10
3    23
4    NA
```

The variable `case` defines the case/control status, while other variables report the age of the subject and the risk set he/she belongs to. The time-varying occupational exposure profiles are stored in the variables `exp15`–`exp60`, corresponding to average yearly exposure experienced in age intervals 15–19, 20–24 and so on up to 65 years. Note how the fourth subject, a control sampled at the age of 52, has the exposure profile set to `NA` from age 55 on.

3 The matrix of exposure histories

The main difference between the extended and standard DLNM framework is the definition of a *matrix of exposure histories*, namely the series of exposures experienced at lag ℓ for each of the n observations, with $\ell = \ell_0, \ldots, L$ and ℓ_0 and L as minimum and maximum lag, respectively. This $n \times (L - \ell_0 + 1)$ matrix needs to be put together in different ways depending on the study design and the information available on the time-varying exposure. The same process applies to time series data, although in this case the matrix is reconstructed internally from the vector of exposure series. In time series data the value for the entry $[t, \ell]$ of the matrix of exposure histories is equal to the entry at $[t+1, \ell+1]$, due to the ordered nature of time series data. This correspondence does not apply any more in the extended framework, as the exposure histories for two observations can be completely unrelated.

In the first example, I build the matrix of exposure histories for the trial data in the data frame `drug` (see Section 2). The exposure profile for each subject is used to reconstruct the matrix of exposure histories. In this case, the exposure at lag 0 corresponds to that experienced on the 28^{th} day when the outcome is measured for all the subjects. The rest of the exposure history is traced backward up to lag 27, corresponding to exposure in the first day. This is a simple code to expand and reverse the exposure profiles stored by week into a matrix of daily exposure histories:

```
> Qdrug <- as.matrix(drug[,rep(7:4, each=7)])
> colnames(Qdrug) <- paste("lag", 0:27, sep="")
> Qdrug[1:3,1:14]

  lag0 lag1 lag2 lag3 lag4 lag5 lag6 lag7 lag8 lag9 lag10 lag11 lag12 lag13
1   37   37   37   37   37   37   37   40   40   40    40    40    40    40
2    0    0    0    0    0    0    0   55   55   55    55    55    55    55
3    0    0    0    0    0    0    0    0    0    0     0     0     0     0
```

The exposure histories for lag 0–13 are reported above for the first three subject. The first seven lags (0–6) correspond to exposures during the last week, while lags 7–13 correspond to the third week, and so on.

In a second example, I reconstruct the matrix of exposure histories for the data frame `nested` using the exposure profiles stored in 5-year intervals. These data are expanded to a matrix of exposure histories over lag 3–40, with lag unit equal to a year. However, in this case the computation is more complex, as each subject is sampled at a different age. Specifically, the exposure history is computed backward along the exposure profile starting from the age of the subject. This step requires some additional computation and data manipulation. The function `exphist()`, which derives an exposure history at a given time of an exposure profile, may be of help:

```
> Qnest <- t(apply(nested, 1, function(sub) exphist(rep(c(0,0,0,sub[5:14]),
    each=5), sub["age"], lag=c(3,40))))
> colnames(Qnest) <- paste("lag", 3:40, sep="")
> Qnest[1:3,1:11]

  lag3 lag4 lag5 lag6 lag7 lag8 lag9 lag10 lag11 lag12 lag13
1    0    0    0    0    0    0    0     0     0     0     0
2    0   10   10   10   10   10   16    16    16    16    16
3    0    0    0    0    0   23   23    23    23    23    44
```

The exposure histories for lag 0–10 are reported above for the first three subject. The first subject, sampled at the age of 81, is assumed to experience the exposure at lag 0 between 80 and 81, the exposure at lag 1 between 79 and 80, and so on. As his/her last exposure is at age 65, the exposure history up to lag 10 is set to 0. The second subject, sampled at the age of 69, has the exposure history set to 0 for lag 3, corresponding to the exposure event at 66, and then to 10 for lags 4–8 and 16 for lags 9–10, corresponding to exposure experienced at age periods 60–65 and 55–60, respectively. These exposure histories are consistent with the exposure profiles and age shown in Section 2.

An example of computation of the matrix of exposure histories for time-to-event analysis using cohort data is illustrated in the code provided as supplementary material in Gasparrini [2014]. In that case, multiple exposure histories are computed for each subject at the times he/she contributed to different risk sets, using the same exposure profile.

In general, the computation of this matrix depends on study design, information on exposure, lag unit and desired level of approximation. This prevents the definition of functions in the dlnm package applicable for this purpose. Nonetheless this issue represents the only additional computational step for using the extended DLNM methodology beyond time series analysis. As shown in the next sections, the use of the functions and interpretation of the results are mostly identical to the standard applications in time series data illustrated in the vignette dlnmTS.

4 Applications beyond time series

4.1 A simple DLM

In this first example, I analyse the temporal dependency between the daily doses of a drug and an unspecified health outcome, applying the functions in the dlnm package to the data set `drug`. Specific information on the use of the functions is provided in the related help pages and the vignette dlnmOverview.

The first step is the definition of a cross-basis function and the derivation of a cross-basis matrix. This is obtained through the function `crossbasis()`:

```
> cbdrug <- crossbasis(Qdrug, lag=27, argvar=list("lin"),
    arglag=list(fun="ns",knots=c(9,18)))
```

The results is stored in the object `cbdrug`, namely a matrix of transformed variables with special attributes. The first unnamed argument `x`, differently from the original applications in time series described in the vignette DLNMTS, is the matrix of exposure histories. However, the rest of the syntax is identical. The argument `lag` specifies the lag period, with minimum lag placed by default at 0. The lag period must be consistent with the dimension (*i.e.* number of columns) of the matrix `Qdrug`. The arguments `argvar` and `arglag` define the exposure-response and lag-response functions, respectively, chosen here as a simple linear function and a natural cubic spline with knots at lag 9 and 18. An intercept is included by default in the lag-response function if not otherwise stated. Given the linearity assumption, this can be technically defined as a DLM. See *?crossbasis* for a complete list of options and additional details.

A summary of the transformation can be obtained by the method function `summary()` for objects of class *"crossbasis"*:

```
> summary(cbdrug)

CROSSBASIS FUNCTIONS
observations: 200
range: 0 to 100
lag period: 0 27
total df:  4

BASIS FOR VAR:
fun: lin
intercept: FALSE

BASIS FOR LAG:
fun: ns
knots: 9 18
intercept: TRUE
Boundary.knots: 0 27
```

The matrix `cbdrug` can be included in the formula of a regression model, in this case a simple linear model assuming a Gaussian distribution, controlling for the effect of sex. This simplified approach does not consider any inter-subject variability, which is beyond the scope of this illustrative example. The estimated exposure-lag-response association can be interpreted by predicting specific effect summaries through the function `crosspred()`:

```
> mdrug <- lm(out~cbdrug+sex, drug)
> pdrug <- crosspred(cbdrug, mdrug, at=0:20*5)
```

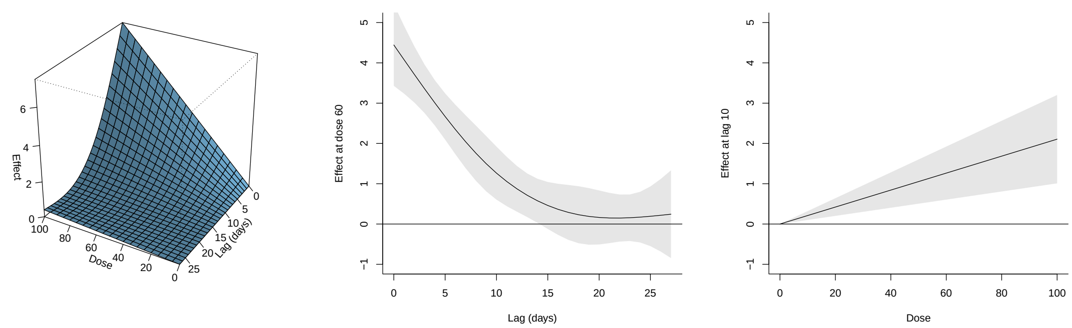
The function `crosspred()` accepts the cross-basis matrix and the related model object as the first two arguments. The argument `at`, if provided as a numeric vector, determines the predictor levels at which predictions should be computed. The reference value, not directly defined here, is set by default to 0 for the function `lin()`. The effect summaries are saved in the object `pdrug` of class *"crosspred"*, from which can be extracted:

```
> with(pdrug,cbind(allfit,alllow,allhigh)["50",])

  allfit   alllow  allhigh
30.29584 20.12871 40.46298
```

The code above extracts the estimate for the *overall cumulative effects* associated with an exposure to 50, interpreted using two perspectives: either as the overall increase in the outcome after a constant

Figure 1

(a) (b) (c)



exposure to 50 sustained throughout the lag period of 28 days (*backward* perspective), or as the sum of the contributions of an exposure to 50 in the next 28 days (*forward* perspective). The 95% confidence intervals are also included, with the confidence level that can be changed with the argument `ci.level` in `crosspred()`. Alternatively, specific combinations of exposure levels and lag values can be extracted from different effect summaries with prefix `mat-` stored in `pdrug`:

```
> pdrug$matfit["20","lag3"]
```

```
[1] 1.118139
```

This is interpreted as the increase in the outcome associated with an intake of a dose level of 20 three days earlier. See *?crosspred* for a full list of the predicted effect summaries. Alternatively, the `plot()` methods for objects of class *"crosspred"* can be used to generate graphs:

```
> plot(pdrug, zlab="Effect", xlab="Dose", ylab="Lag (days)")
> plot(pdrug, var=60, ylab="Effect at dose 60", xlab="Lag (days)", ylim=c(-1,5))
> plot(pdrug, lag=10, ylab="Effect at lag 10", xlab="Dose", ylim=c(-1,5))
```

The first line of code produces the graph in Figure 1a, namely the bi-dimensional exposure-lag-response association estimated by the regression model, showing how the effect varies across the range of dose and lag values. This type of graph is obtained by leaving the argument `ptype` unselected, thus choosing the default value `"3d"`. The graphs suggests that the effect of a dose of the drug is pronounced in the first days after the intake and then tends to disappear after 15-20 days.

The second and third lines of code produce the graphs in Figures 1b–1c, respectively, showing the lag-response curve specific to exposure 60 and the exposure-response curve specific to lag 10. The shape of these curves depends on the specific choices for the basis functions selected for producing the cross-basis `cbdrug`. In particular, the lag-response curve in Figure 1b indicates an exponential decay in the effects. This type of graphs represents slices cut in the 3-D surface of Figure 1a, with the argument `ptype` set by default to `"slices"` if one of the argument `var` or `lag` is specified. Additional argument such as `xlab` and `ylim` are internally passed to `plot.default` to control the graphical parameters. See `?plot.default` and *?par* for a complete list, and generally `?plot.crosspred` for using plotting functions.

4.2 A more complex DLNM

In a second example, I assess how protracted exposures to an occupational agent affect the risk of occurrence of cancer, using the data set `nested`. The steps of the analysis are the same illustrated in Section 4.1.

An initial assumption is that the exposures sustained in the last three years, corresponding to lag 0–2, are not affecting the risk of occurrence of cancer. Consistently with this assumption, the matrix of exposure history `Qnest` has been derived for lag period 3–40. The cross-basis matrix can therefore be created by:

```
> cbnest <- crossbasis(Qnest, lag=c(3,40), argvar=list("bs",degree=2,df=3),
    arglag=list(fun="ns",knots=c(10,30),intercept=F))
```

The chosen basis functions are a quadratic spline for the dimensions of predictor and a natural cubic spline for lags. In the former, only the number of degrees of freedom are chosen, and the single knot is placed by default at the median. Note that in the spline for the lag-response the intercept is excluded, so the function is forced to predict a null effect at the beginning of the lag period, consistently with the assumption above. The command *`summary(cbnest)`* can show additional details.

The cross-basis objects can be included again in the formula of a regression model. Compatibly with the nested case-control design, a conditional logistic regression is performed through the function `clogit()` included in the package survival, which needs to be loaded into the session. Effect summaries are then predicted. The code is:

```
> library(survival)
> mnest <- clogit(case~cbnest+strata(riskset), nested)
> pnest <- crosspred(cbnest, mnest, cen=0, at=0:20*5)
```

Note how in this case the centering value must be selected directly through the argument `cen`, as no straightforward reference exist for non-linear functions such as `bs()`. Similarly to the previous example, estimates of the effect summaries can be extracted from the object `pnest`, although this time using the objects with prefix `allRR-` and `matRR-` which store the exponentiated predictions in the scale of OR (see *`?crosspred`*). The same types of graphs displayed in Figure 1 are obtained by:

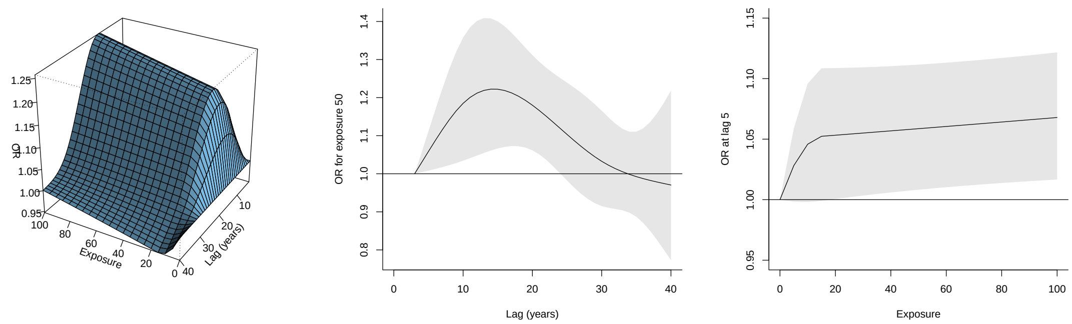
```
> plot(pnest, zlab="OR", xlab="Exposure", ylab="Lag (years)")
> plot(pnest, var=50, ylab="OR for exposure 50", xlab="Lag (years)", xlim=c(0,40))
> plot(pnest, lag=5, ylab="OR at lag 5", xlab="Exposure", ylim=c(0.95,1.15))
```

The 3-D graph in Figure 2a is again interpreted as the bi-dimensional exposure-lag-response association between the occupational exposure and the risk of cancer. Note how the lag period is expressed in years in this example. The graph suggests an initial increase in risk, measured as odds ratio (OR), followed by a decrease.

The slice graphs in Figures 2b–2c provide additional details. Specifically, the estimated lag-response curve in Figure 2b displays a peak in risk 10 to 15 years after the exposure, with the risk then returning to the baseline level 30 years after the exposure, although the confidence intervals are quite wide. The exposure-response curve in Figure 2c suggests an attenuation of the effect at higher exposures, although again the confidence intervals do not rule out a linear association.

Figure 2

(a) (b) (c)



5 Extended prediction summaries

The usual effect summaries obtained by `crosspred()` are computed over a grid of exposure and lag values, specified directly or through default selection. In particular, a vector of exposure values is usually passed through the argument `at`, while the lag period is selected through `lag`. The function then computes *overall cumulative* summaries, stored in vectors with prefix `all-`, and *specific* summaries associated to combinations of exposure and lag values, stored in matrices with prefix `mat-`. The former refer either to effects associated to a constant exposure throughout the lag period backward in time, or to the total effect contribution of an exposure event within the lag period forward in time. The latter can be used to display exposure-response and lag-response curves.

However, in the extended setting described in this vignette it is useful to define alternative and complementary effect summaries. In particular, it may be of interest to predict what is the overall cumulative effect associated to a specific exposure history, possibly characterized by time-varying exposures. These new effect summaries can be easily computed using the function `exphist()` used in Section 3, which produces a matrix of exposure histories given an exposure profile. This matrix can be directly passed as the argument `at` of `crosspred()`, rather than a vector of exposure values.

As an example, we can use the nested case-control analysis in Section 4.2 for computing the overall cumulative OR for an hypothetical subject exposed to exposure 10 for five years, then unexposed for five more years, then exposed to 13 for ten years. From this exposure profile, we can compute the exposure history at the end of the exposure period, looking backward in time. Specifically:

```
> expnested <- rep(c(10,0,13), c(5,5,10))
> hist <- exphist(expnested, time=length(expnested), lag=c(3,40))
> hist

   lag3 lag4 lag5 lag6 lag7 lag8 lag9 lag10 lag11 lag12 lag13 lag14 lag15 lag16
20   13   13   13   13   13   13   13     0     0     0     0     0    10    10
   lag17 lag18 lag19 lag20 lag21 lag22 lag23 lag24 lag25 lag26 lag27 lag28
20    10    10    10     0     0     0     0     0     0     0     0     0
   lag29 lag30 lag31 lag32 lag33 lag34 lag35 lag36 lag37 lag38 lag39 lag40
20     0     0     0     0     0     0     0     0     0     0     0     0
```

The function `exphist()` produces the exposure history at time 20 over lag 3–40. The specific time is set through the argument `time` and in this case corresponds to the end of the exposure period in `expnested`. The last 21 exposures to 0 are included to complete the exposure history up to 40 years. Now we can predict the overall cumulative effect by using `hist` as the argument `at` of `crosspred()`. Note that the lag period must be consistent with that used in estimation. This is the code:

```
> pnesthist <- crosspred(cbnest, mnest, cen=0, at=hist)
> with(pnesthist, c(allRRfit,allRRlow,allRRhigh))

      20       20       20
3.503928 1.240109 9.900351
```

The estimated OR is 3.5 (95%CI: 1.2–9.9) compared to a subject with no exposure throughout the whole lag period.

The same approach can be used to obtain *dynamic predictions* along time for a specific exposure profile. The idea behind this more complex effect summary is that the risk can be predicted dynamically in time given time-varying exposure histories, based on an assumed exposure-lag-response association. In practice, for each given time, moving forward, the exposure history changes as specific exposure events refer to different lag periods.

As an example, I show how the dynamic predicted effect following a specific drug prescription can be estimated using the analysis of the trial data illustrated in Section 4.1. Let's assume that a patient is treated with a dose 10 for two weeks, then he/she increases to 50 for one week, then stops for 1 week and starts again with a dose 20 for two weeks. First, I create the daily exposure profile:

```
> expdrug <- rep(c(10,50,0,20),c(2,1,1,2)*7)
```

The function `exphist()` can now be used sequentially along the exposure profile to create the matrix of exposure histories for all the time points:

```
> dynhist <- exphist(expdrug, lag=27)
```

The argument `time` of `exphist()` by default takes the values of all the time points of `expdrug`, creating the matrix of exposure histories for all of them. This matrix can now be used in `crosspred()` to obtain the dynamic predictions:

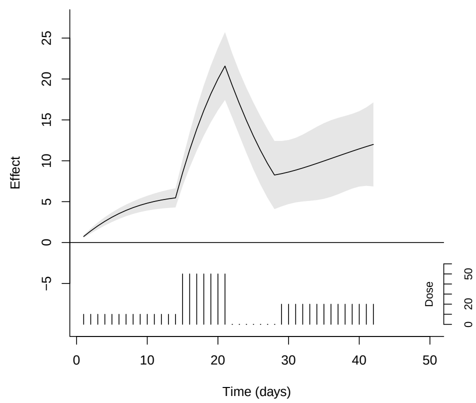
```
> pdyndrug <- crosspred(cbdrug, mdrug, at=dynhist)
```

The object can now be used to plot the dynamic prediction:

```
> plot(pdyndrug,"overall", ylab="Effect", xlab="Time (days)", ylim=c(-10,27),
   xlim=c(1,50), yaxt="n")
> axis(2, at=-1:5*5)
> par(new=TRUE)
> plot(expdrug, type="h", xlim=c(1,50), ylim=c(0,300), axes=F, ann=F)
> axis(4, at=0:6*10, cex.axis=0.8)
> mtext("Dose", 4, line=-1.5, at=30, cex=0.8)
```

The unnamed argument `ptype` in `plot()` is set to `"overall"`, thus plotting this extended version of overall cumulative association in Figure 3. This graph displays the variation from the baseline outcome associated with the drug prescription profile detailed above, represented as histogram-like vertical lines. As expected, the effect changes dynamically in time, depending on the doses but with a delay determined by the lag structure.

Figure 3



6 Applying user-defined functions

Since version 2.0.0 of the dlnm package, important changes have also been implemented in the use of the main functions. In particular, the functions `onebasis()`, called independently or internally through `crossbasis()` for applying transformations, now simply acts as a wrapper to other functions (see *onebasis*). The new functions `strata()`, `poly()`, `thr()` and `integer()` in the dlnm package (see the related help pages), together with the functions `ns()` and `bs()` in the splines package, offer all the previous options of basis transformation.

However, this flexible approach offers the possibility of using different functions available in other R packages, or functions directly defined by the user. The called function must have `x` as its first argument, and it must return a vector or matrix of transformed variables with attributes storing the arguments which exactly define the transformation. This information will be used later by `crosspred()` to produce the predictions. Also, the function must be defined as a closure containing formal arguments, meaning that primitive functions such as `exp()`, `sin()` or `log()` cannot be used directly (see the example below).

As a first example of applying user-defined functions within the DLNM framework, we can revise the previous analysis illustrated in Section 4.2. Figure 2c suggested a possible attenuation of the effect at high exposures. This fact and the skewness of the exposure distribution can be addressed through a logarithmic transformation. As shown in Gasparrini [2014], this is equivalent to apply a logarithm as exposure-response function in the cross-basis transformation. First, let's define a new log function:

```
> mylog <- function(x) log(x+1)
```

This step is required as `log()` is a primitive function and cannot be used directly. The original exposure is summed to 1 to prevent problems with 0 values in the logarithm. The new function `mylog()` can now be used directly in `crossbasis()` in place of `bs()` to model the exposure-response curve:

```
> cbnest2 <- crossbasis(Qnest, lag=c(3,40), argvar=list("mylog"),
    arglag=list(fun="ns",knots=c(10,30),intercept=F))
> summary(cbnest2)

CROSSBASIS FUNCTIONS
```

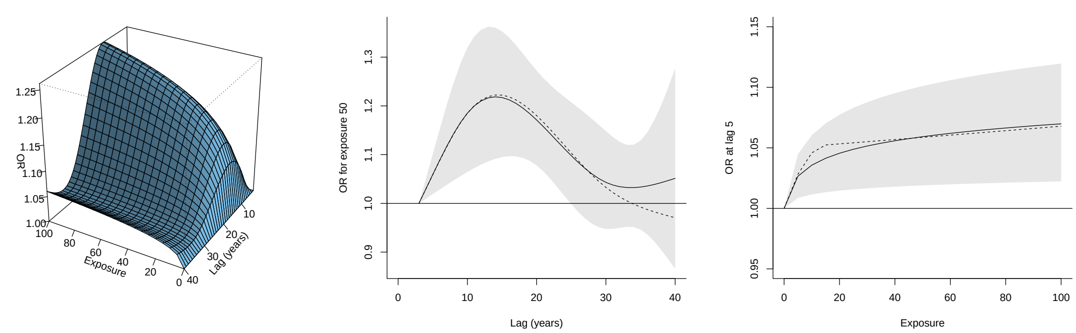
```
observations: 600
range: 0 to 1064
lag period: 3 40
total df:  3

BASIS FOR VAR:
fun: mylog

BASIS FOR LAG:
fun: ns
knots: 10 30
intercept: FALSE
Boundary.knots: 3 40
```

Note how the total *df* of the cross-basis are now 3, if compared to the 9 in the original transformation, as the exposure-response is modelled with only 1 *df*. The rest of the code is almost identical, just substituting the newly created objects:

```
> mnest2 <- clogit(case~cbnest2+strata(riskset), nested)
> pnest2 <- crosspred(cbnest2, mnest2, cen=0, at=0:20*5)
> plot(pnest2, zlab="OR", xlab="Exposure", ylab="Lag (years)")
> plot(pnest2, var=50, ylab="OR for exposure 50", xlab="Lag (years)", xlim=c(0,40))
> lines(pnest, var=50, lty=2)
> plot(pnest2, lag=5, ylab="OR at lag 5", xlab="Exposure", ylim=c(0.95,1.15))
> lines(pnest, lag=5, lty=2)
```

Figure 4

(a) (b) (c)



The results presented in Figure 4 can be compared with those originally displayed in Figure 2 of Section 4.2. The method function `lines()` are used to add the original curves to the new plots. The comparison also indicates how the assumption of a logarithmic shape produces a substantial increase in precision.

Another example of application of user-defined functions is provided by extending the analysis illustrated in Section 4.1. The inspection of Figure 1b suggested that the lag-response curve follows an exponential decay trajectory. It may be reasonable to apply a function modelling this shape instead than a natural cubic spline. This decay function can be defined as:

```
> fdecay <- function(x,scale=5) {
    basis <- exp(-x/scale)
    attributes(basis)$scale <- scale
    return(basis)
  }
```

The argument `scale`, with default value 5, is used to control the degree of decay. Note how this must be included as an attribute of the vector returned by the new function `fdecay()`. Again, we can use this new function to obtain the alternative cross-basis transformation:

```
> cbdrug2 <- crossbasis(Qdrug, lag=27, argvar=list("lin"),
    arglag=list(fun="fdecay",scale=6))
> summary(cbdrug2)

CROSSBASIS FUNCTIONS
observations: 200
range: 0 to 100
lag period: 0 27
total df:  1

BASIS FOR VAR:
fun: lin
intercept: FALSE

BASIS FOR LAG:
fun: fdecay
scale: 6
```

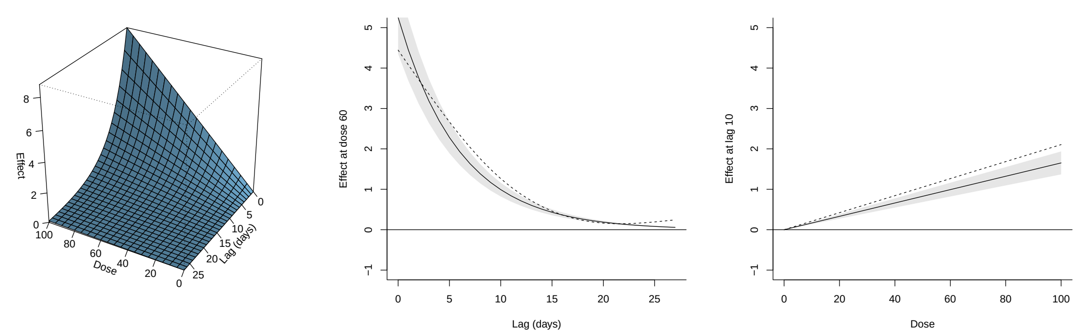
Again, the computational step used in Section 4.1 can be repeated to perform the modified analysis:

```
> mdrug2 <- lm(out~cbdrug2+sex, drug)
> pdrug2 <- crosspred(cbdrug2, mdrug2, at=0:20*5)
> plot(pdrug2, zlab="Effect", xlab="Dose", ylab="Lag (days)")
> plot(pdrug2, var=60, ylab="Effect at dose 60", xlab="Lag (days)", ylim=c(-1,5))
> lines(pdrug, var=60, lty=2)
> plot(pdrug2, lag=10, ylab="Effect at lag 10", xlab="Dose", ylim=c(-1,5))
> lines(pdrug, lag=10, lty=2)
```

The results are reported in Figure 5. The comparison with results in Figure 1 of Section 4.1, included again as dashed lines, shows a dramatic increase in precision, as a strict structure is assumed for the exposure-lag-response surface, which is entirely estimated with only 1 *df*. Note how the argument `scale` is selected a priori to 6 and not estimated here, as the function `fdecay()` is non-linear for this parameter. However, the DLNM framework can also be used with non-linear regression functions such as `nls()` for estimating the scale parameter. More generally, a critical discussion and some guidance on inference and model selection in the DLNM framework are offered in Gasparrini [2014].

Figure 5

(a) (b) (c)



7 A general tool for regression analysis

The functions in the package dlnm can also be used as a general tool for regression analysis. In particular, the facilities for prediction and graphical representation, developed for assessing bi-dimensional exposure-lag-response associations, can be more generally applied for unlagged exposure-response relationships.

Standard method for estimating these uni-dimensional associations include the use of regression splines in unpenalized models, such as generalized linear models (GLMs) and Cox proportional hazard models, or penalized splines in generalized additive models (GAMs). The user can extract predictions in the usual way using the function `crosspred()`, which can be applied with unpenalized models in conjunction with the function `onebasis()`, or directly with regression outputs of penalized models fitted using the function `gam()` in the mgcv package.

In order to illustrate these options, I replicate examples illustrated in the help pages of the functions `ns()` in the package splines and `gam()` in the package `mgcv()`.

The first example deomonstrates the use of regression splines with the regression function `lm()` to assess the relationship between average height (in inches) and weight (in pounds) in a sample of American women aged 30–39. The code is reported in the Examples section of the help page of `ns()` (see `help(ns)`). The same transformation can be obtained by applying the wrapper function `onebasis()`:

```
> library(splines)
> oneheight <- onebasis(women$height, "ns", df=5)
> mwomen <- lm(weight ~ oneheight, data=women)
```

The use of `onebasis()` allows the use of other functions in dlnm for obtaining predictions and plots, using a simple code:

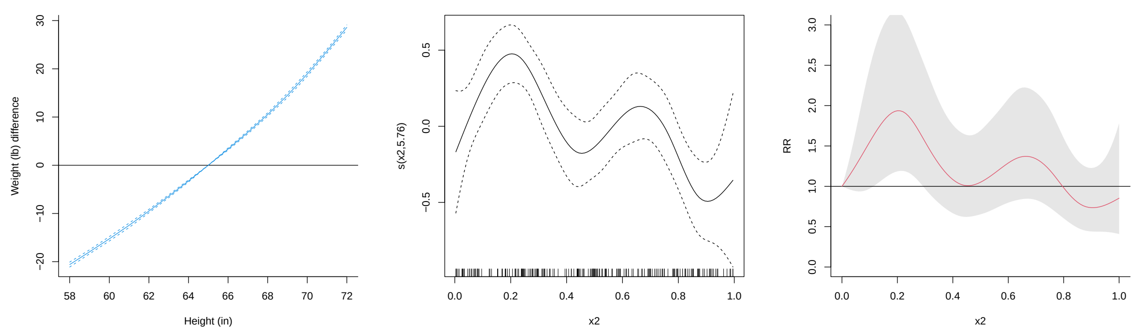
```
> pwomen <- crosspred(oneheight, mwomen, cen=65, at=58:72)
> with(pwomen, cbind(allfit, alllow, allhigh)["70",])

  allfit   alllow  allhigh
18.92287 18.46545 19.38030
```

```
> plot(pwomen, ci="l", ylab="Weight (lb) difference", xlab="Height (in)", col=4)
```

The function `crosspred()` can be applied as usual, just including the object of class *"onebasis"* as its first argument. The results are reported using a height of 65 inches as references. The estimated association, with confidence intervals, can be retrieved simply by accessing the `all-` components. Note that, as no lagged effect is allowed, these are identical to the `mat-` components, which are however reported as 1-column matrices. The association can be plotted in the usual way with the method function `plot()`, with the graph shown in Figure 6a. Note how it is not important to select the type of the plot with the argument `ptype`, as only uni-dimensional graphs can be created.

Figure 6

(a) (b) (c)



The second example illustrates the application of functions in the package dlnm to facilitate the analysis of smooth associations using penalized splines. The (slightly modified) original code, reported in the Examples section of the help page of `gam()` (see `help(gam)`), is:

```
> library(mgcv)
> dat <- gamSim(1,n=200,dist="poisson",scale=.1)

Gu & Wahba 4 term additive model

> b2 <- gam(y ~ s(x0,bs="cr") + s(x1,bs="cr") + s(x2,bs="cr") + s(x3,bs="cr"),
    family=poisson, data=dat, method="REML")
> plot(b2, select=3)
```

The code performs a GAM estimating smoothed relationships in simulated data with several variables using penalized cubic regression splines through the function `s()`, and generates the graph for the variable `x2`, displayed in Figure 6b. Predictions and plots can also be obtained using dlnm functions, with:

```
> pgam <- crosspred("x2", b2, cen=0, at=0:100/100)
> with(pgam, cbind(allRRfit, allRRlow, allRRhigh)["0.7",])

 allRRfit  allRRlow allRRhigh
1.3405415 0.8309798 2.1625694
```

```
> plot(pgam, ylim=c(0,3), ylab="RR", xlab="x2", col=2)
```

As shown above, the `crosspred()` function also works directly with associations defined through the function `s()` within `gam()`, simply including the character string with the name of the variable as its first argument. This usage is similar to the more complex cross-basis parameterization obtained using the related smooth `cb` smooth constructor (see the vignette dlnmPenalized). This step simplifies the computations of estimated assocations, together with measures of uncertainty. In addition, it is possible to plot the smoothed relationship in the response scale or relative risk (RR), and using a reference value that makes easier to interpret the association, as shown in Figure 6c.

References

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