Distributed lag linear and non-linear models for time series data

Antonio Gasparrini
London School of Hygiene & Tropical Medicine, UK

dlnm version 2.3.9 , 2019-03-11

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("dlnmTS")`.

1 Preamble

This vignette DLNMTS illustrates the use of the R package dlnm for the application of distributed lag linear and non-linear models (DLMs and DLNMs) in time series analysis. The development of DLMs and DLNMs and the original software implementation for time series data are illustrated in Gasparrini et al. [2010] and Gasparrini [2011].

The examples described in the next sections cover most of the standard applications of the DLNM methodology for time series data, and explore the capabilities of the dlnm package for specifying, summarizing and plotting this class of models. In spite of the specific application on the health effects of air pollution and temperature, these examples are easily generalized to different topics, and form a basis for the analysis of this data set or other time series data sources. 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

The examples included in vignette explore the associations between air pollution and temperature with mortality, using a time series data set with daily observations for the city of Chicago in the period 1987–2000. This data set is included in the package as the data frame chicagoNMMAPS, and is described in the related help page (see help(chicagoNMMAPS) and the vignette DLNMOVERVIEW).

After loading the package in the R session, let's have a look at the first three observations:

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

        date time year month doy      dow death cvd resp       temp   dptp
1 1987-01-01    1 1987     1   1 Thursday   130  65   13 -0.2777778 31.500
2 1987-01-02    2 1987     1   2   Friday   150  73   14  0.5555556 29.875
3 1987-01-03    3 1987     1   3 Saturday   101  43   11  0.5555556 27.375
   rhum     pm10       o3
1 95.50 26.95607 4.376079
2 88.25       NA 4.929803
3 89.50 32.83869 3.751079
```

The data set is composed by a complete series of equally-spaced observations taken each day in the period 1987–2000. This represents the required format for applying DLNMs in time series data.

3 Example 1: a simple DLM

In this first example, I specify a simple DLM, assessing the effect of PM_{10} on mortality, while adjusting for the effect of temperature. In order to do so, I 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 the regression model estimates also the distributed lag function for temperature, which is included as a non-linear term.

First, I 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 associations separately for each of them. This is the code:

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

In applications with time series data, the first argument `x` is used to specify the vector series. The function internally passes the arguments in `argvar` and `arglag` to `onebasis()` in order to build the basis for predictor and lags, respectively. In this case, we assume that the effect of PM_{10} is linear (`fun="lin"`), while modelling the relationship with temperature through a natural cubic spline with 5 degrees of freedom (`fun="ns"`, chosen by default). The internal knots (if not provided) are placed by `ns()` at the default equally spaced quantiles, while the boundary knots are located at the temperature range, so only `df` must be specified.

Regarding the bases for the space of the lags, I specify the lagged effect of PM_{10} up to 15 days of lag (minimum lag equal to 0 by default), with a 4th 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 `breaks=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 to 356.1768
lag period: 0 15
total df:  5

BASIS FOR VAR:
fun: lin
intercept: FALSE

BASIS FOR LAG:
fun: poly
degree: 4
scale: 15
intercept: TRUE
```

Now the two `crossbasis` objects can be included in a model formula of a regression model. The packages `splines` is loaded, as it is needed in the examples. In this case I fit the time series model 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 estimated association with specific levels of PM_{10} on mortality, predicted by the model above, can be summarized by the function `crosspred()` and saved in an object with the same class:

```
> pred1.pm <- crosspred(cb1.pm, model1, at=0:20, bylag=0.2, cumul=TRUE)
```

The function includes 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 $\mu\mathrm{gr}/\mathrm{m}^3$. By setting `bylag=0.2`, the prediction is computed along the lag space with an increment of 0.2. This finer grid is meant to produce a smoother lag curve when the results are plotted. The argument `cumul` (default to `FALSE`) indicates that also incremental cumulative associations along lags must be included. No centering is defined through the argument `cen`, and the reference value is therefore set at value 0 by default (this happens for the function `lin()`). Now that the predictions have been stored in `pred1.pm`, they can be plot by specific method functions. For example:

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

Figure 1

(a)   (b)
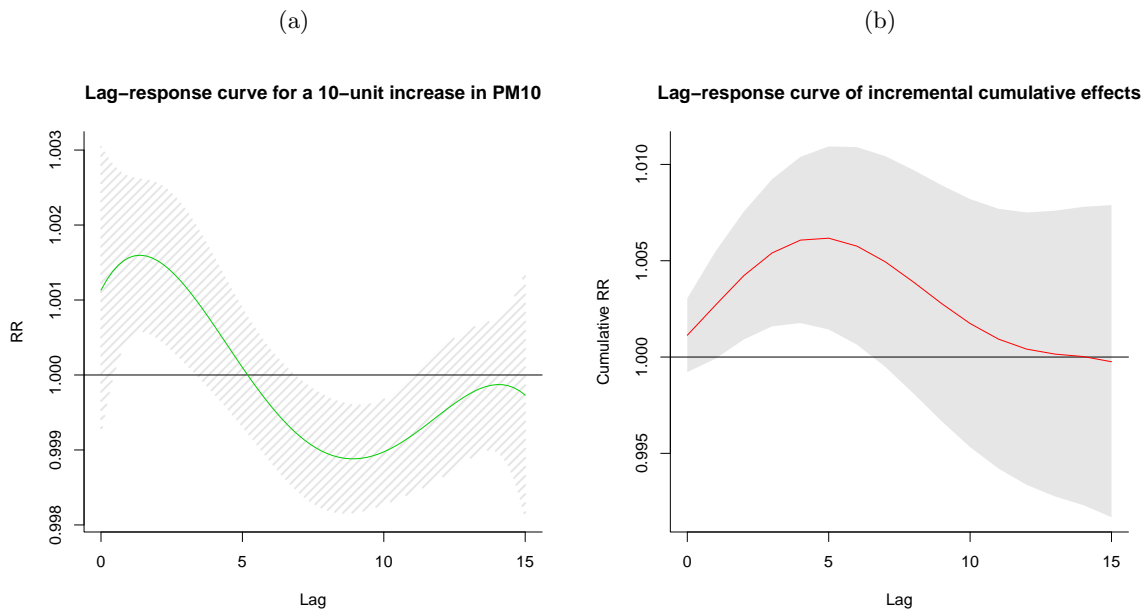
The function includes the `pred1.pm` object with the stored results, and the argument `"slices"` defines that we want to graph relationship corresponding to specific values of predictor and lag in the related dimensions. With `var=10` I display the lag-response relationship for a specific value of PM_{10}, i.e.

10 μgr/m^3. This association is defined using the reference value of 0 μgr/m^3, thus providing the predictor-specific association for a 10-unit increase. I also chose a different colour for the first plot. The argument `cumul` indicates if incremental cumulative associations, 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 of these plots is twofold: the lag 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 cumulative effect of a 10-unit increase in PM$_{10}$ over 15 days of lag (i.e. summing all the contributions 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
```

4   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 multiple seasons in different years, and do not represent a single continuous series. In this case, I assess the effect of ozone and temperature on mortality up to 5 and 10 days of lag, respectively, using the same steps already seen in Section 3.

First, I create a seasonal time series data set obtained by restricting to the summer period (June-September), and save it in the data frame `chicagoNMMAPS`:

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

Again, I first create the cross-basis matrices:

```
> cb2.o3 <- crossbasis(chicagoNMMAPSseas$o3, lag=5,
    argvar=list(fun="thr",thr=40.3), arglag=list(fun="integer"),
    group=chicagoNMMAPSseas$year)
> cb2.temp <- crossbasis(chicagoNMMAPSseas$temp, lag=10,
    argvar=list(fun="thr",thr=c(15,25)), arglag=list(fun="strata",breaks=c(2,6)),
    group=chicagoNMMAPSseas$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 I 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 (`fun="thr"`). For temperature, I use a double threshold with the assumption that the effect is linear below 15°C and above 25°C, and null in between. The threshold values are chosen with the argument `thr.value` (abbreviated to `thr`), while the un-specified argument `side` is set to the default value `"h"` for the first cross-basis and to `"d"` for the second one (given two threshold values are provided). Regarding the lag dimension, I specify an unconstrained function for O_3, applying one parameter for each lag (`fun="integer"`) up to a 5 days (with minimum lag equal to 0 by default). For temperature, I 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 natural splines for day of the year and time, in order to describe the seasonal effect within each year, and the long-time trend, respectively. In particular, the latter has far less degrees of freedom, if compared to the previous analysis, as it only needs to capture a smooth annual trend. Apart from that, the estimates and predictions are carried out in the same way as in Section 3. The code is:

```
> model2 <- glm(death ~  cb2.o3 + cb2.temp + ns(doy, 4) + ns(time,3) + 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 I define the integers from 0 to 65 $\mu gr/m^3$ (approximately the range of ozone distribution), plus the threshold and the value 50.3 $\mu gr/m^3$ corresponding to a 10-unit increase above the threshold. The vector is automatically ordered. A reference is automatically selected exposure-response curve modelled by `thr()`, and the argument `cen` can be left undefined.

I plot the predictor-specific lag-response relationship for a 10-unit increase in O_3, similarly to Section 3 but with 80% confidence intervals, and also the overall cumulative exposure-response relationship. The related code is (results in Figures 2a–2b):

```
> plot(pred2.o3, "slices", var=50.3, ci="bars", type="p", col=2, pch=19,
    ci.level=0.80, main="Lag-response a 10-unit increase above threshold (80CI)")
> plot(pred2.o3,"overall",xlab="Ozone", ci="l", col=3, ylim=c(0.9,1.3), lwd=2,
    ci.arg=list(col=1,lty=3), main="Overall cumulative association for 5 lags")
```

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, I 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 cumulative association must be plotted, with confidence intervals as lines, `ylim` defining the range of the y-axis, `lwd` the thickness of the line. In this case, confidence intervals are displayed as lines, selected through an abbreviation `"l"` in the argument `ci`. 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 cumulative effect for a 10-unit increase in ozone above the threshold ($50.3 - 40.3$ $\mu gr/m^3$), together with its 95% confidence intervals:

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

    50.3
1.047313
```

Figure 2

(a)   (b)

Lag–response a 10–unit increase above threshold (80CI)   **Overall cumulative association for 5 lags**
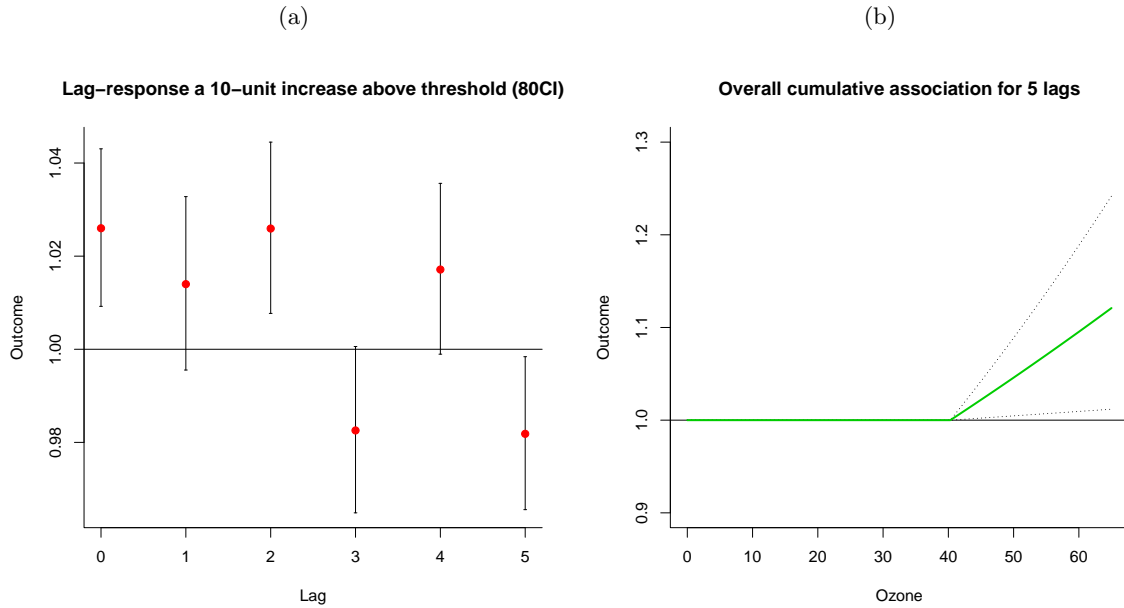

```
> cbind(pred2.o3$allRRlow, pred2.o3$allRRhigh)["50.3",]

[1] 1.004775 1.091652
```

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   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 relationship: the dimension of the predictor is never considered, and the lag-specific or overall cumulative associations with a 10-unit increase are easily plotted. In contrast, when allowing for a non-linear dependency with temperature, we need to adopt a bi-dimensional perspective in order to represent associations which vary non-linearly along the space of the predictor and lags.

In this example I specify a more complex DLNM, where the dependency is 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 3–4, 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 I run a DLNM to investigate the effects of temperature and PM_{10} on mortality up to lag 30 and 1, respectively.

First, I define the cross-basis matrices. In particular, the cross-basis for temperature is specified through a natural and non-natural splines, using the functions `ns()` and `bs()` from the package splines. This

is the code:

```
> cb3.pm <- crossbasis(chicagoNMMAPS$pm10, lag=1, argvar=list(fun="lin"),
    arglag=list(fun="strata"))
> varknots <- equalknots(chicagoNMMAPS$temp,fun="bs",df=5,degree=2)
> lagknots <- logknots(30, 3)
> cb3.temp <- crossbasis(chicagoNMMAPS$temp, lag=30, argvar=list(fun="bs",
    knots=varknots), arglag=list(knots=lagknots))
```

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 (`fun="bs"`) with 5 degrees of freedom for temperature, with `knots` placed by default at equally spaced value in the space of the predictor, selected through the function `equalknots()`. Regarding the space of lags, I 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 I define another cubic spline, this time with the natural constraint (`fun="ns"` by default) for the lag dimension of temperature. The knots for the spline for lags are placed at equally-spaced values in the log scale of lags, using the function `logknots()`. This used to be the default values in versions of the package earlier than 2.0.0.

The estimation, prediction and plotting of the association between temperature and mortality are performed by:

```
> model3 <- glm(death ~  cb3.pm + cb3.temp + ns(time, 7*14) + dow,
    family=quasipoisson(), chicagoNMMAPS)
> pred3.temp <- crosspred(cb3.temp, model3, cen=21, 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 centered here at 21°C, the point which represents the reference for the interpretation of the estimated effects. This step is needed here, as the relationship is modelled with a non-linear function with no obvious reference value. The 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`).

Plots in Figures 3a–3b offer a comprehensive summary of the bi-dimensional exposure-lag-response association, but are limited in their ability to inform on associations at specific values of predictor or lags. In addition, they are also limited for inferential purposes, as the uncertainty of the estimated association is not reported in 3-D 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(0.95,1.25), lwd=1.5,
    main="Lag-response curves for different temperatures, 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)))
```
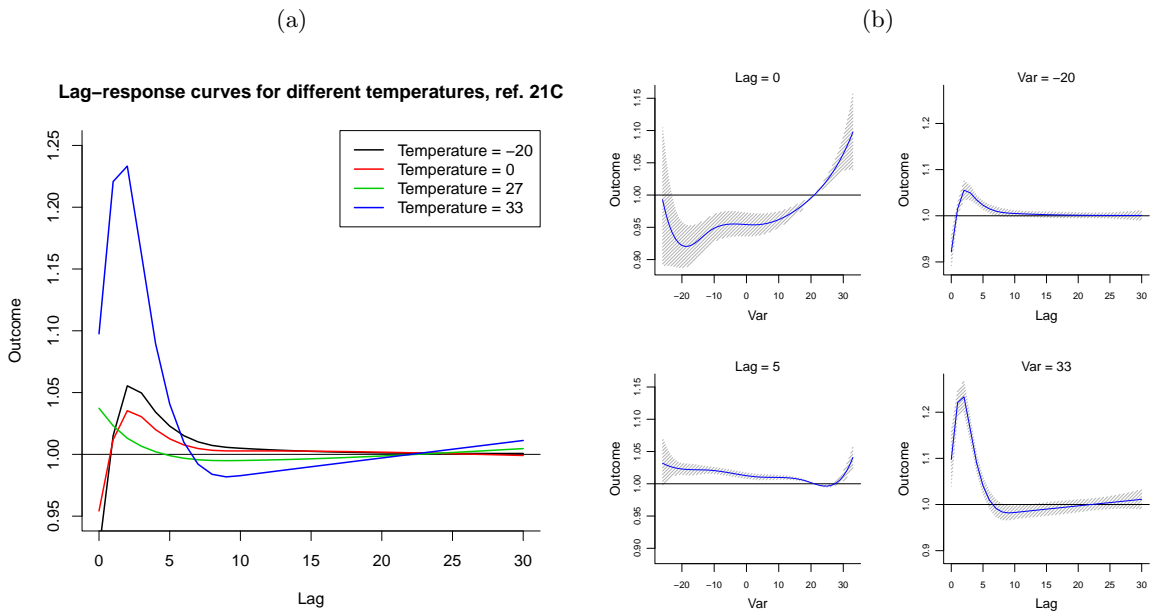
Figure 3

(a)   (b)

3D graph of temperature effect   **Contour plot**   **RR**
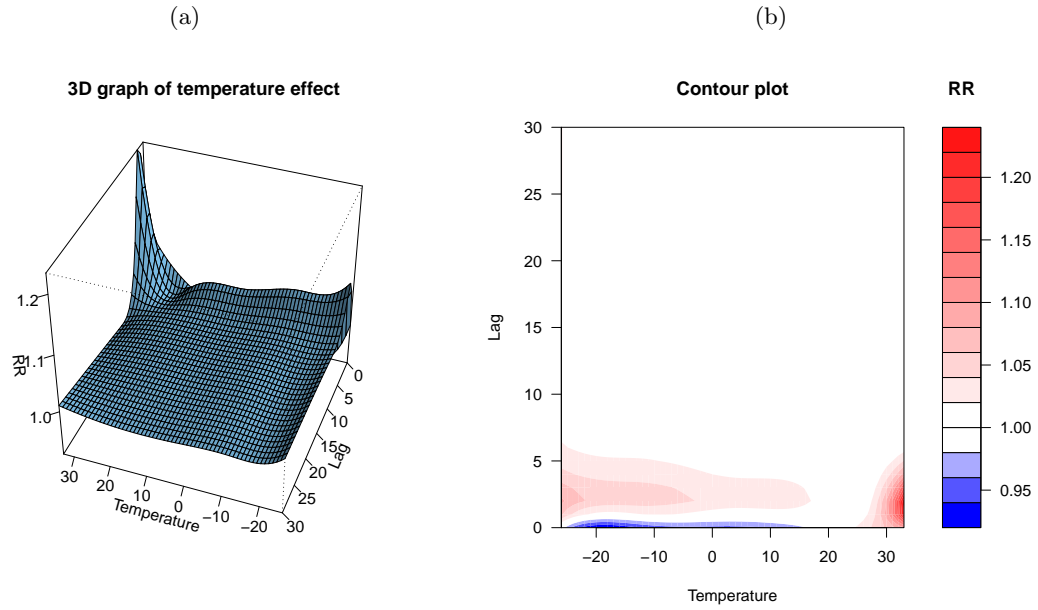
The results are reported in Figures 4a–4b. Figure 4a illustrates lag-response curves specific to mild and extreme cold and hot temperatures of -20°C, 0°C, 27°C, and 33°C (with reference at 21°C). Figures 4b

Figure 4

(a)   (b)

Lag–response curves for different temperatures, ref. 21C

depicts both exposure-response relationships specific to lag 0 and 5 (left column), and lag-response relationships specific to temperatures -20°C and 33°C (right column). The arguments var and lag define the values of temperature and lag for "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.

6   Example 4: reducing a DLNM

In this last example, I 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(). This method is thoroughly illustrated in Gasparrini and Armstrong [2013]. First, I specify a new cross-basis matrix, run the model and predict in the usual way:

```
> cb4 <- crossbasis(chicagoNMMAPS$temp, lag=30,
    argvar=list(fun="thr",thr=c(10,25)), arglag=list(knots=lagknots))
> 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 knots at equally-spaced values in the log scale for lags as in the previous example, respectively. The reduction may be carried out to 3 specific summaries, namely overall cumulative, lag-specific and predictor-specific associations. The first two represent exposure-response relationships, while the third one represents a lag-response relationship. 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 associations is computed at lag 5 and 33°C in the two spaces, respectively. 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:
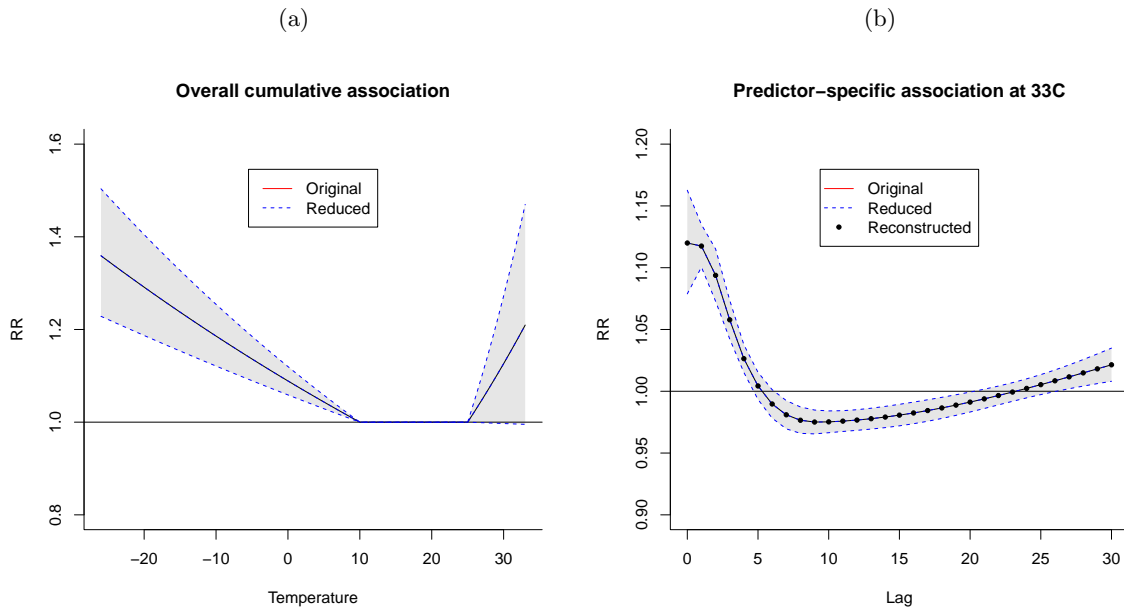
```
> length(coef(pred4))

[1] 10

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

[1] 2

[1] 2

> length(coef(redvar))
```

Figure 5

(a)

(b)

Overall cumulative association

Predictor–specific association at 33C

[1] 5

As expected, the number of parameters has been reduced to 2 for the space of the predictor (consistently with the double-threshold parameterization), and to 5 for the space of lags (consistently 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 cumulative association")
> 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 association given the modified parameters. As an example, I 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,intercept=TRUE)
> pred4b <- crosspred(b4,coef=coef(redvar),vcov=vcov(redvar),model.link="log",by=1)
```

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. Predictions are 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 association at 33C")
```

11

```
> 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)
```

References

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 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.