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

Antonio Gasparrini
*London School of Hygiene & Tropical Medicine, UK*

dlnm version 2.4.7 , 2021-10-07

## Contents

**1 Preamble** **2**

**2 Installation** **2**

**3 Methodology and applications** **2**
3.1 The DLM and DLNM framework . . . 3
3.2 Standard application in time series analysis . . . 3
3.3 Generalization beyond time series design . . . 3
3.4 A penalized version of DLMs and DLNMs . . . 4
3.5 A general tool for regression analysis . . . 4

**4 Functions and data in the package dlnm** **4**
4.1 Basis functions . . . 4
4.2 The function `onebasis()` . . . 5
4.3 The function `crossbasis()` . . . 5
4.4 Functions for penalized models . . . 6
4.5 The function `crosspred()` . . . 6
4.6 The function `crossreduce()` . . . 7
4.7 Plotting functions . . . 7
4.8 Other functions . . . 8
4.9 Data . . . 8

**5 Changes in the package dlnm** **9**

**6 Acknowledgements** **9**

**Bibliography** **11**

[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* (DLNMs), a modelling framework to describe simultaneously non-linear and delayed effects between predictors and an outcome, a dependency defined as *exposure-lag-response association*. These include previously described *distributed lag models* (DLMs) for linear relationships as special cases. This document complements the description of the package provided in Gasparrini [2011], now with an updated syntax, which represents the main reference to the package. The DLMs/DLNMs methodology has been previously described in Gasparrini [2014], Gasparrini and Armstrong [2013], Gasparrini et al. [2010, 2017], together with a detailed algebraical development. This framework was originally conceived and proposed to investigate the health effects of temperature by Armstrong [2006].

This document DLNMOVERVIEW is the main of four vignettes documenting the package. Its aim is to describe the methodology and to provide an overview of the main functions. Three other vignettes offer more specific examples, as detailed below. Each vignette, included in the package installation, can be opened in R by typing `vignette("namevignette")`.

The dlnm package is available on the Comprehensive R Archive Network (CRAN), with info at the related web page (CRAN.R-project.org/package=dlnm). A development website is available on GitHub (github.com/gasparrini/dlnm). General information on the development and applications of the DLM/DLNM modelling framework, together with an updated version of the R scripts for running the examples in published papers, can be found on GitHub (github.com/gasparrini) or at the personal web page of the package maintainer (www.ag-myresearch.com).

Type `citation("dlnm")` in R to cite the dlnm package after installation (see Section 2).

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

# 2 Installation

The last version of the dlnm package officially released on CRAN can be downloaded and installed following standard procedures, for example by typing:

```
> install.packages("dlnm")
```

or directly through the R or RStudio menu. The package can be alternatively installed from its webpage within CRAN (https://CRAN.R-project.org/package=dlnm), using the source code (.tar.gz) or binaries for Windows or MacOS. In this case, also the other packages which functions in dlnm are dependent from, defined by the fields *Imports* and *Suggests* of `description`, must be installed.

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:

```
> news(package="dlnm")
```

# 3 Methodology and applications

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,

focusing also on specific extensions and applications of the methodology and software. The user can refer to the articles and vignettes provided below for a more detailed description.

### 3.1 The DLM and DLNM framework

The modelling class of DLMs and DLNMs is applied to describe associations in which the dependency between an exposure and an outcome is lagged in time. This 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 [Gasparrini, 2014], revising a terminology previously proposed by Thomas [1988].

A statistical development of DLMs and DLNMs is based on the application of *basis* functions for parameterizing the *exposure history*, namely the set of lagged exposures. Specifically, two set of basis functions are chosen independently for modelling the exposure and lag-response relationships. These are then combined through a special tensor product defined by bi-dimensional *cross-basis* functions [Gasparrini, 2014, Gasparrini et al., 2010]. The choice of the two sets of functions determines the shape of the relationship in each dimension. DLMs can be considered special cases of the more general DLNMs, when the exposure-response is assumed linear.

Estimation is performed using standard regression models and R functions, simply including the matrix storing the cross-basis variables in a model formula. Result 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 [Gasparrini et al., 2010]. Also, *summaries* of the bi-dimensional association can be derived, namely exposure-reponses at specific lags, lag-responses at specific predictor values, and the overall cumulative exposure-response as the net effect across the whole lag period [Gasparrini and Armstrong, 2013]. These summaries can be interpreted using either a forward or backward interpretation, as explained in Gasparrini and Leone [2014].

### 3.2 Standard application in time series analysis

Simpler DLMs 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 full extension to DLNMs was conceived by Armstrong [2006]. A conceptual and methodological re-evaluation of this modelling framework for time series data is given in Gasparrini et al. [2010]. DLMs and DLNMs are now commonly used in time series analysis, and the functions in the package dlnm offer a simple way for deriving the complex parameterization and for producing predictions and plots, as illustrated in Gasparrini [2011].

The vignette DLNMTS provides a thorough overview of the use of the dlnm package for performing DLMs and DLNMs in time series analysis.

### 3.3 Generalization beyond time series design

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 have also been proposed [Abrahamowicz and MacKenzie, 2007, Berhane et al., 2008, Vacek, 1997]. A general unifying framework

based on DLMs and DLNMs is described in Gasparrini [2014].

The vignette DLNMEXTENDED illustrates this extension and the application of the dlnm software in study designs and data structures beyond time series.

### 3.4 A penalized version of DLMs and DLNMs

In the standard definition of the DLM/DLNM modelling framework, models are fitted with common regression methods, such as generalized linear models (GLMs) or Cox proportional hazard models. Here, the bi-dimensional shape of the exposure-lag-response relationship depends entirely on the parametric form of the basis functions applied in each of the two spaces of predictor and lag. Recently, Gasparrini et al. [2017] has described an extension of DLNMs based on the use of penalized splines through generalized additive models (GAMs), where potentially flexible shapes are smoothed with the application of specific penalties [Wood, 2006]. This extension generalizes similar methods previously proposed for simpler models assuming linear [Obermeier et al., 2015, Zanobetti et al., 2000] and linear threshold [Muggeo, 2008] exposure-response shapes.

The vignette DLNMPENALIZED offers an overview of the implementation of penalized DLMs and DLNMs in the dlnm package and the use of the functions in illustrative examples.

### 3.5 A general tool for regression analysis

The functions in the dlnm package can be used more generally to facilitate the computation and interpretation of associations estimated from regression models, beyod the specific case of distributed lag modelling. Specifically, the functions can be applied to obtain predictions and plots of point estimates and measures of uncertainty for linear or non-linear unlagged relationships, estimated from either unpenalized (*e.g.*, GLMs and Cox models) or penalized (GAMs) models.

The vignette DLNMEXTENDED provides examples of the use of the functions to derive predictions and plots of exposure-response associations from standard regression models.

## 4 Functions and data in the package dlnm

This section describes the main functions and data included in the package dlnm. The functions are grouped consistently with the various steps in the definition, estimation and interpretation of DLMs and DLNMs. Only a general summary is provided here. For details on the usage of the functions and for real-data examples, the user can refer to the related help pages and the other vignettes, respectively.

### 4.1 Basis functions

The first step for performing DLMs or DLNMs consists of the choice of two sets of basis functions for modelling the exposure-lag-response association. Any kind of function determining completely known parametric transformations of the predictor can be used, and several options are available in the dlnm package. All the functions below are meant to be called internally by `onebasis()` and `crossbasis()` (see sections below) and not directly run by the users.

First, the package contains basic functions to specify standard relationships. Specifically, the functions `strata()`, `thr()` and `poly()` can be applied to obtain indicator variables defining intervals through dummy parameterization, high, low or double-threshold relationships, and polynomial variables, respectively. The function `lin()` returns the un-transformed predictor variable and it is applied to

specify simple DLMs, while the function `integer()` produces indicator variables for each integer value and it is used in unconstrained DLMs and DLNMs [Gasparrini et al., 2010]. These functions are not exported to the namespace in order to prevent conflicts with other existing functions in recommended packages.

Functions from other packages can also be called for deriving more complex transformations. For instance, the functions `ns()` and `bs()` from the recommended package splines can be called for deriving spline parameterizations. In addition, the functions `ps()` and `cr()`, available in dlnm, are used to specify penalized splines, as described in Section 4.4 and in the vignette DLNMPENALIZED.

More generally, user-defined functions defining any type of basis transformations can also be used within dlnm. The only requirements is that the function returns a matrix of basis variables univocally determined by its arguments, and that all these parameters are stored as attributes in order to reproduce exactly the same transformations. The vignette DLNMEXTENDED provides more details and some examples.

The user can refer to the related help pages of these functions for further info on their usage (for instance, type `?strata` or `?bs` in R).

## 4.2 The function `onebasis()`

This function represents the workhorse for basis transformation in dlnm. It has replaced the old functions `mkbasis()` and `mklagbasis()` since version 1.5.1 of the package. Its main role is to apply chosen transformations and generate basis matrices in a format suitable for other functions such as `crossbasis()` and `crosspred()`.

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 function has a first argument `x` for the predictor variable, and another argument `fun` for specifying the basis function to be called internally, whose arguments are passed through the ellipsis argument '`...`'.

Interestingly, `onebasis()` can also be used more generally for generating uni-dimensional functions in regression models, with predictions and graphs derived by `crosspred()` (see Section 4.5) and plotting methods (see Section 4.7). Examples are provided in the vignette DLNMEXTENDED.

The function `onebasis()` returns a basis matrix with additional class *"onebasis"*, with attributes determining the chosen parameterization. See `?onebasis` for further info.

## 4.3 The function `crossbasis()`

This is the main function in the package dlnm. It calls `onebasis()` internally 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], Gasparrini et al. [2010, Sections 4.1–4.2], and Gasparrini et al. [2017, Section 2] for details.

The class of the first argument `x` of `crossbasis()` defines how the data are interpreted. If a vector, `x` is assumed to represent an equally-spaced, complete and ordered series of observations in a time series framework. If a matrix, `x` is assumed to represent a series of exposure histories for each observation (rows) and lag (columns). This second format can be used to extend DLNMs beyond time series data, as illustrated in the vignette DLNMEXTENDED. The lag period can be defined through 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 Sections 4.1–4.2). 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).

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

The function returns a matrix object of class *"crossbasis"*, with attributes storing information on the original variable and the parameters of the chosen parameterization in the two spaces. See `?crossbasis` for further info. The cross-basis matrix needs to be included in a regression model formula in order to fit a model.

## 4.4 Functions for penalized models

More flexible versions of DLMs and DLNMs can be specified through penalized splines by embedding functions of the packages dlnm and mgcv [Gasparrini et al., 2017]. Here I briefly introduce the functions available in dlnm, while a more comprehensive overview is provided in the vignette DLNMEXTENDED.

There are two approaches for defining penalized DLMs and DLNMs. Using the *external* method, the cross-basis parameterization is derived as usual by calling the function `ps()` and/or `cr()` through `crossbasis()`. These two functions (see Section 4.1) create a basis matrix with specific spline parameterizations, and a related penalty matrix stored as an additional attribute `S`. The function `cbPen()` is then called on the cross-basis object to generate the bi-dimensional penalty matrices using consistent tensor product transformations. Additional penalties on the lag dimension can be added through the argument `addSlag` of `cbPen()`. The models are fitted by the regression function `gam()` of mgcv, including the cross-basis object in the formula and the penalty matrices as the argument `paraPen`.

Using instead the *internal* method, the cross-basis transformation is not generated by `crossbasis()`, but using the function `smooth.construct.cb.smooth.spec()` available in dlnm. This smooth constructor is called internally as a smooth term using `s(X,L,bs="cb",...)` within the formula of the regression function `gam()` of mgcv, with `X` and `L` as a matrix of exposure histories and a matrix of lags, respectively. Additional info can be passed to the constructor using a named list as the argument `xt` of `s()`, for instance lists `argvar` and `arglag` to define othe types of transformations in the two spaces, and `addSlag` for additional penalties on the lag dimension.

Note that, in both methods, it is possible to define unpenalized functions in either the predictor or lag space through `argvar` and `arglag`. Indeed, the function `lin()` can be called through the former for performing penalized DLMs.

The user can refer to the help pages for additional details, and to the vignette DLNMPENALIZED for an overview of penalized DLMs and DLNMs using the package dlnm.

## 4.5 The function `crosspred()`

The interpretation of estimated parameters is usually complex for non-trivial basis transformations in DLMs, and virtually impossible in bi-dimensional DLNMs. The function `crosspred()` facilitates the interpretation by predicting the association for a grid 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, extracts the related parameters estimated in the regression model, and generates predictions with associated standard errors and confidence intervals (see Gasparrini [2014, Section 2.3] and Gasparrini et al. [2010, Section 4.3] for algebraic details).

The first two arguments `basis` and `model` of `crosspred()` are usually the cross-basis matrix and the fitted model object. When using the internal method for penalized models (see Section 4.4), the argument `basis` must be a character string identifying the first argument of `s()` in `gam()`. The function

`crosspred()` automatically extract the parameters related to the cross-basis from several regression functions, or alternatively these can be directly inputted using the arguments `coef` and `vcov`. The predictor values used for prediction can be selected with the argument `at` or alternatively with `from-to-by`. The arguments `lag` and `bylag` determine instead the range and increment of the sequence of lag values. 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`.

An alternative use of `crosspred()` is to predict the results for specific sets of lagged exposures. This can be achieved by inputting a matrix of exposure histories as the argument `at`. The function `exphist()` can be used to simplify the computation (see the vignette DLNMEXTENDED for example of these extended prediction summaries).

Multiple cross-basis matrices associated with different predictors may be included in `model`, and predictions for each of them can be computed with `crosspred()`. More generally, the function `crosspred()` can be used to predict estimated effects from regression models defining standard uni-dimensional exposure-response relationships with no lag, either as penalized functions in `gam()` or in other regression functions through `onebasis()` (see the vignette DLNMEXTENDED for some examples).

The function returns a list object of class *"crosspred"*, with components storing the predictions and other information about the model. The user can refer to ***`help(crosspred)`*** for a complete list of argument and returned list components.

## 4.6 The function `crossreduce()`

As described in Section 3.1, results from DLMs and DLNMs can be expressed as one-dimensional summaries, namely overall cumulative exposure-responses, lag-specific exposure-responses, or predictor-specific lag-responses. The function `crossreduce()` reduces the fit of DLMs and DLNMs consistently with these summaries, and re-expresses it in terms of modified parameters of the one-dimensional basis functions chosen for that space. Algebraic details are provided in Gasparrini and Armstrong [2013].

The function works very similarly to `crosspred()` (see Section 4.5), with similar usage and arguments. 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 have the same meaning and specification as in `crosspred()` (see Section 4.5 and `?crossreduce`).

The function returns a list object of class *"crossreduce"*, againg similar to that returned by `crosspred()`. An illustrative example of the use of the function in given in the vignette DLNMTS.

## 4.7 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.

The `plot()` method can produce different types of plots through the argument `ptype`. Specifically, it can generate 3-D or countour graphs of the entire bi-dimensional exposure-lag-response association (`ptype="3d"` and `ptype="contour"` calling `persp()` and `filled.contour()` internally, respectively), or uni-dimensional exposure-response or lag-response summaries defined in Section 3.1 (`ptype="slices"` calling default `plot()` functions). Methods `lines()` and `points()` for may be used as low-level plotting functions to add lines or points to an existing plot.

The argument `ci` (with options `"area"`, the default, and `"bars"` and `"lines"`) and `ci.arg` can be used to add a graphical representation of confidence intervals. Exponentiated predictions are automatically plotted if generated in the predictions, or forced with the argument `exp=TRUE`. Additional arguments of plotting functions called internally can be specified through the ellipsis argument '`...`', allowing complete flexibility in the choices of colours, axes, labels and other graphical parameters.

## 4.8 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 function `exphist()`, mentioned in Section 4.5, builds a matrix of exposure histories given an exposure profile, and is used in particular for data management tasks in applications beyond time series analysis (see the vignette DLNMEXTENDED).

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 these 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)
```

Other method functions, such as `summary()`, `coef()` and `vcov()`, are provided for objects of class *"crossbasis"*, *"onebasis"*, *"crosspred"* and *"crossreduce"*.

## 4.9 Data

This version the package includes the three data sets `chicagoNMMAPS`, `nested` and `drug`. The former is used to illustrate the use of DLMs and DLNMs in time series analysis (in particular in the vignette DLNMTS), while the other two are used in examples of the extension of the methodology and package in other study designs (in particular in the vignette DLNMEXTENDED).

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.

# 5 Changes in the package dlnm

A GitHub page of the package dlnm (github.com/gasparrini/dlnm) includes information on ongoing developments. In addition, changes in the last version 2.4.7, and in previous ones since the first version 0.1.0 uploaded on CRAN on the 1st of July 2009, are documented in the NEWS file, visualized with:

```
> news(package="dlnm")
```

In some versions, new functions have been added and existing functions replaced, and, more importantly, the usage of some of them has changed. These important changes are detailed more extensively in three additional documents, accessed through:

```
> 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 applies also to code included as supplementary material of published papers. An updated version of published code is available at the GitHub page (github.com/gasparrini) or personal web page (www.ag-myresearch.com) of the package maintainer.

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 https://doi.org/10.18637/jss.v043.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 and M. Leone. Attributable risk from distributed lag models. *BMC Medical Research Methodology*, 14(1):55, 2014.

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

A Gasparrini, F Scheipl, B Armstrong, and M G Kenward. A penalized framework for distributed lag non-linear models. *Biometrics*, (In Press), 2017.

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.

V. M. Muggeo. Modeling temperature effects on mortality: multiple segmented relationships with common break points. *Biostatistics*, 9(4):613–620, 2008.

Viola Obermeier, Fabian Scheipl, Christian Heumann, Joachim Wassermann, and Helmut Kuhchenhoff. Flexible distributed lags for modelling earthquake data. *Journal of the Royal Statistical Society: Series C*, 64(2):395–412, 2015.

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.

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