

Package ‘dlnm’

April 19, 2012

Type Package

Title Distributed Lag Non-linear Models

Version 1.6.2

Date 2012-04-17

Author Antonio Gasparrini and Ben Armstrong

Maintainer Antonio Gasparrini <antonio.gasparrini@lshtm.ac.uk>

Imports stats, graphics, grDevices, utils, splines, tsModel

Depends R (>= 2.8.0)

Suggest splines, mgcv, NMMAPSLite

Description The package dlnm contains functions to specify and run distributed lag linear and non-linear models.

License GPL (>= 2)

LazyLoad yes

LazyData yes

Repository CRAN

Date/Publication 2012-04-19 15:46:55

R topics documented:

dlnm-package	2
chicagoNMMAPS	3
coef.crosspred	4
crossbasis	5
crosspred	8
crossreduce	12
onebasis	15
plot.crosspred	18
plot.crossreduce	21

`dlnm-package`*Distributed Lag Non-linear Models (DLNM)*

Description

The package **dlnm** contains functions to specify and run distributed lag linear and non-linear models. These functions are used to specify basis and cross-basis matrices among a set of available options, then to predict and plot the results for a fitted model.

Details

Distributed lag non-linear models (DLNM) represent a modelling framework to describe simultaneously non-linear and delayed dependencies in time-series data. This methodology is based on the definition of a *cross-basis*, a bi-dimensional space of functions specifying the dependency along the space of the predictor and along lags. The cross-basis functions are built combining the basis functions for the two dimensions, chosen among a set of possible bases. This family includes simple distributed lag models (DLM) as a special case. A thorough methodological overview is given in the references below.

Given a series of observations ordered and equally spaced in time, `crossbasis` creates two set of basis functions to define the relationship in the two dimensions of predictor and lags, by calling the function `onebasis`, which provides a list of possible options (splines, strata, and threshold functions, among others). The two matrices are then combined in a matrix object of class "crossbasis", containing the transformed variables to be included in the model formula. After the model fitting, `crosspred` generate predictions for a set of suitable values of the original predictor and stores them in a "crosspred" object. The fit of a DLNM can be reduced and re-expressed as the chosen function of one of the two dimensions through the function `crossreduce`. It returns a "crossreduce" object storing the new parameters and predictions.

Method functions are available for both objects "crosspred" and "crossreduce". The plotting functions `plot`, `lines` and `points`, offer a set of choices to plot the results, while `coef` and `vcov` return the coefficients and associated (co)variance matrix for a (optionally reduced) DLNM.

The DLNM framework is developed for time series data: the data are assumed to be an equally-spaced, complete and ordered series of observations, where the interval defines the lag unit. The estimation can be carried out with the default regression functions, such as `lm`, `glm` or `gam` (package **mgcv**). The function `crosspred` extracts the relevant parameters directly from the model object if methods for `coef` and `vcov` are available for that model class. Alternatively, the user must extract the parameters manually.

Potential applications to other study desing and data structures are currently under evaluation. Preliminary results for matched case-control and cohort data are promising, and the functions included in the package already works with `clogit` and `coxph` (package **survival**).

Additional information on the package **dlnm** are available in the vignette included in the installation. This document offers a detailed description of the capabilities of the package, and some examples of application to real data, with an extensive illustration of the use of the functions. The vignette is available by typing:

```
vignette("dlnmOverview")
```

Use `citation("dlnm")` to cite this package.

A list of changes included in the current and previous versions can be found by typing:
`file.show(system.file("ChangeLog", package="dlnm"))`

Author(s)

Antonio Gasparri and Ben Armstrong

Maintainer: Antonio Gasparri, <antonio.gasparrini@lshtm.ac.uk>

References

Gasparri A. Distributed lag linear and non-linear models in R: the package dlnm. *Journal of Statistical Software*. 2011; **43**(8):1-20. [freely available [here](#)].

Gasparri A., Armstrong, B., Kenward M. G. Distributed lag non-linear models. *Statistics in Medicine*. 2010; **29**(21):2224-2234. [freely available [here](#)]

Armstrong, B. Models for the relationship between ambient temperature and daily mortality. *Epidemiology*. 2006, **17**(6):624-31.

See Also

[onebasis](#) to generate simple basis matrices. [crossbasis](#) to generate cross-basis matrices. [crosspred](#) to obtain predictions after model fitting. [crossreduce](#) to reduce the fit to one dimension. [plot.crosspred](#) and [plot.crossreduce](#) to plot several type of graphs.

Type `'vignette(dlnmOverview)'` for a detailed description.

chicagoNMMAPS

Daily Mortality Weather and Pollution Data for Chicago

Description

The dataset 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 from the National Morbidity, Mortality and Air Pollution Study (NMMAPS)

Usage

```
data(chicagoNMMAPS)
```

Format

A data frame with 5114 observations on the following 14 variables.

`date` Date in the period 1987-2000

`time` The sequence of observations

`year` Year

month Month (numeric)
 doy Day of the year
 dow Day of the week (factor)
 death Counts of all cause mortality excluding accident
 cvd Cardiovascular Deaths
 resp Respiratory Deaths
 temp Mean temperature (in Celsius degrees)
 dptp Dew point temperature
 rhum Mean relative humidity
 pm10 PM10
 o3 Ozone

Details

These data represents a subsample of the variables included in the NMMAPS dataset for Chicago.

The variable temp is derived from the original tmpd after a transformation from Fahrenheit to Celsius. The variables pm10 and o3 are an approximated reconstruction of the original series, adding the de-trended values and the median of the long term trend. This is the reason they include negative values. See vignette('PollutantProcess') from the packages **NMMAPSdata** or **NMMAP-Slite**.

Source

The complete dataset is available at the Internet-based Health and Air Pollution Surveillance System (iHAPSS) website:

<http://www.ihapss.jhsph.edu>

or through the packages **NMMAPSdata** or **NMMAPSlite**.

coef.crosspred

Model Coefficients and their (Co)Variance Matrix of a DLNM

Description

These method functions extract the estimated model coefficients and their (co)variance matrix from a DLNM from objects of class "crosspred" and "crossreduce".

Usage

```
## S3 method for class 'crosspred'  
coef(object, ...)  
  
## S3 method for class 'crosspred'  
vcov(object, ...)  
  
## S3 method for class 'crossreduce'  
coef(object, ...)  
  
## S3 method for class 'crossreduce'  
vcov(object, ...)
```

Arguments

object an object of class "crosspred" or "crossreduce".
... further arguments passed to or from other methods.

Author(s)

Antonio Gasparriani, <antonio.gasparrini@lshtm.ac.uk>

See Also

See [dlnm-package](#) for an overview of the package and type 'vignette(dlnmOverview)' for a detailed description.

crossbasis

Generate a Cross-Basis Matrix for a DLNM

Description

Generate the basis matrices for the two dimensions of predictor and lags, choosing among a set of possible basis functions. Then, these functions are combined in order to create the related cross-basis matrix, which can be included in a model formula to fit a distributed lag non-linear model (DLNM).

Usage

```
crossbasis(x, lag=c(0,0), argvar=list(), arglag=list(), group=NULL, ...)  
  
## S3 method for class 'crossbasis'  
summary(object, ...)
```

Arguments

x	the predictor variable, defined as a numeric vector representing a complete series of ordered observations.
lag	either an integer scalar or vector of length 2, defining the the maximum lag or the lag range, respectively. If a scalar, the minimum is automatically set of 0.
argvar, arglag	lists of arguments to be passed to the function onebasis for generating the two basis matrices for predictor and lags, respectively. See Details below.
group	a factor defining groups of observations, representing multiple series. Each series must be consecutive, complete and ordered.
object	a object of class "crossbasis".
...	additional arguments. See Details below.

Details

Until version 1.5.0, the function adopted a completely different usage, with different arguments. The compatibility of the old code is retained by the additional arguments passed through ... Users are however suggested to adopt the current usage.

The arguments in `argvar` and `arglag` (optionally including `type`, `df`, `degree`, `knots`, `bound`, `int`, `cen`) define two set of basis functions for each dimension. The function [onebasis](#) is called internally, to build the related basis matrices. The `argvar` list is applied to `x`, in order to generate the matrix for the space of the predictor. The `arglag` list is applied to a new vector given by the sequence obtained by `lag`, in order to generate the matrix for the space of lags. Then, the two set of basis matrices are combined in order to create the related cross-basis matrix. See [onebasis](#) for additional information on how to specify each basis.

Results from DLNM are interpreted relatively to a reference value of the predictor, determined automatically or through a centering point. See [onebasis](#) for further details.

The basis functions for lags are defined with different default arguments than in [onebasis](#): specifically, the knots are placed at equally spaced values on the log scale, an intercept is always included (see Warnings below), and the basis is never centered. Some arguments can be automatically changed for not sensible combinations, or set to NULL if not required. Use [summary.crossbasis](#) to check the result.

The argument `group` defines groups of observations representing independent series. Each series must be consecutive, complete and ordered. `crossbasis` is run on each of them applying the same cross-basis functions: default choices (knots position, range, etc.) are taken considering the pooled distribution.

For a detailed illustration of the use of the function, see:

```
vignette("dlnmOverview")
```

Value

A matrix object of class "crossbasis" which can be included in a model formula in order to fit a DLNM. It contains the attributes `range` (range of the original vector of observations), `lag` (lag range), `argvar` and `arglag` (lists of arguments defining the basis functions in each space, which can be modified if compared to the arguments above). The function [summary.crossbasis](#) returns

a summary of the cross-basis matrix and the related attributes, and can be used to check the options for the basis functions chosen for the two dimensions.

Warnings

Meaningless combinations of arguments (for example the inclusion of knots lying outside the range for type equal to "strata" or thr-type) could lead to collinear variables, with identifiability problems in the model and the exclusion of some of them.

It is strongly recommended to avoid the inclusion of an intercept in the basis for x (`int` in `argvar` should be `FALSE`, as default), otherwise a rank-deficient cross-basis matrix will be specified, causing some of the cross-variables to be excluded in the regression model. Conversely, an intercept is included by default in the basis for the space of lags.

Note

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

The name of the crossbasis object will be used by `crosspred` in order to extract the related estimated parameters. If more than one variable is transformed through cross-basis functions in the same model, different names must be specified.

Author(s)

Antonio Gasparrini, <antonio.gasparrini@lshtm.ac.uk>

References

- Gasparrini A. Distributed lag linear and non-linear models in R: the package `dlm`. *Journal of Statistical Software*. 2011; **43**(8):1-20. [freely available [here](#)].
- Gasparrini A., Armstrong, B.,Kenward M. G. Distributed lag non-linear models. *Statistics in Medicine*. 2010; **29**(21):2224-2234. [freely available [here](#)]

See Also

`onebasis` to generate one-dimensional basis matrices. `crosspred` to obtain predictions after model fitting. `plot.crosspred` to plot several type of graphs.

See `dlm-package` for an overview of the package and type '`vignette(dlmOverview)`' for a detailed description.

Examples

```

### simple DLM
### space of predictor: linear relationship for PM10
### space of predictor: 5df natural cubic spline for temperature
### lag function: 4th degree polynomial for PM10 up to lag15
### lag function: strata intervals at lag 0 and 1-3 for temperature

# CREATE THE CROSS-BASIS FOR EACH PREDICTOR AND CHECK WITH SUMMARY
cb1.pm <- crossbasis(chicagoNMAPS$pm10, lag=15, argvar=list(type="lin",cen=0),
  arglag=list(type="poly",degree=4))
cb1.temp <- crossbasis(chicagoNMAPS$temp, lag=3, argvar=list(df=5,cen=21),
  arglag=list(type="strata",knots=1))
summary(cb1.pm)
summary(cb1.temp)

# RUN THE MODEL AND GET THE PREDICTION FOR PM10
library(splines)
model1 <- glm(death ~ cb1.pm + cb1.temp + ns(time, 7*14) + dow,
  family=quasipoisson(), chicagoNMAPS)
pred1.pm <- crosspred(cb1.pm, model1, at=0:20, bylag=0.2, cumul=TRUE)

# PLOT THE LINEAR ASSOCIATION OF PM10 ALONG LAGS
plot(pred1.pm, "slices", var=10, col=3, ylab="RR", ci.arg=list(density=15,lwd=2),
  main="Effects of a 10-unit increase in PM10 along lags")
plot(pred1.pm, "slices", var=10, cumul=TRUE, ylab="Cumulative RR",
  main="Cumulative effects of a 10-unit increase in PM10 along lags")

# GET THE FIGURES FOR THE OVERALL EFFECTS, WITH CI
pred1.pm$allRRfit["10"]
cbind(pred1.pm$allRRlow, pred1.pm$allRRhigh)["10",]

### See the vignette 'dlnmOverview' for a detailed explanation of this example

```

crosspred

Generate Predictions for a DLNM

Description

Generate predictions from a distributed lag non-linear model (DLNM) for a set of values of the original predictor. Predictions are computed versus a reference predictor value, and interpreted as effects. Specific effects are computed for each combination of predictor and lag values, plus overall and (optionally) cumulative effects (summed up along lags). The function can be used also for simple basis functions not including lag.

Usage

```

crosspred(basis, model=NULL, coef=NULL, vcov=NULL, model.link=NULL,
  at=NULL, from=NULL, to=NULL, by=NULL, bylag=1, ci.level=0.95, cumul=FALSE)

```

```
## S3 method for class 'crosspred'
summary(object, ...)
```

Arguments

basis	an object of class "onebasis" or "crossbasis".
model	a model object for which the prediction is desired. See Details below.
coef, vcov, model.link	user-provided coefficients, (co)variance matrix and model link for the prediction. See Details below.
at	vector of values used for prediction.
from, to	range of predictor values used for prediction.
by, bylag	increment of the sequences of predictor and lag values used for prediction.
ci.level	confidence level for the computation of confidence intervals.
cumul	logical. If TRUE, cumulative effects are predicted. See Details.
object	an object of class "crosspred".
...	additional arguments to be passed to summary.

Details

model is the model object including basis. It must include methods for `coef` and `vcov`, applied to extract the parameters. For model classes without these methods, the user can manually extract the related parameters and include them in `coef-vcov`, also specifying `model.link`. In this case, the dimensions and order of the first two must match the variables included in `basis`.

The object `basis` must be the same containing the basis or cross-basis matrix included in `model`, preserving its attributes and class. The set of values for which predictions must be computed can be specified by `at` or alternatively by `from/to/by`. If specified by `at`, the values are automatically ordered and made unique. If `at` and `by` are not provided, approximately 50 equally-spaced rounded values are returned using `pretty`. `bylag` determines the lag step (forced to 1 for cumulative effects).

The function can be used to compute predictions for models with simple basis functions not including lag, computed with `onebasis`. In this case, only unlagged predicted effects are returned.

Exponentiated predicted effects are included if `model.link` is equal to `log` or `logit`, together with confidence intervals computed using a normal approximation and a confidence level of `ci.level`. `model.link` is automatically selected from `model` for classes "lm"-`glm`-"gam"-`clogit`-"`coxph`", but needs to be provided for different classes. Matrices with cumulative predicted effects summed upon lags for each values used for prediction are included if `cumul=TRUE`. For a long lag series (i.e. 1000 lags) the routine can be slow.

For a detailed illustration of the use of the functions, see:

```
vignette("dlnmOverview")
```

Value

A list object of class "crosspred" with the following (optional) components:

predvar	vector of observations used for prediction.
---------	---

<code>lag</code>	integer vector defining the lag range.
<code>bylag</code>	increment of the sequence of lag values.
<code>coef, vcov</code>	coefficients and their variance-covariance matrix.
<code>matfit, matse</code>	matrices of predicted effects and related standard errors for each value of <code>predvar</code> and lag.
<code>allfit, allse</code>	vectors of overall predicted effects and related standard errors for each value of <code>predvar</code> .
<code>cumfit, cumse</code>	matrices of cumulative predicted effects (along lags) and related standard errors for each value of <code>predvar</code> and lag. Computed if <code>cumul=TRUE</code> .
<code>matRRfit</code>	matrix of exponentiated predicted effects from <code>matfit</code> .
<code>matRRlow, matRRhigh</code>	matrices with low and high confidence intervals for <code>matRRfit</code> .
<code>allRRfit</code>	vector of exponentiated overall predicted effects from <code>allfit</code> .
<code>allRRlow, allRRhigh</code>	vectors with low and high confidence intervals for <code>allRRfit</code> .
<code>cumRRfit</code>	matrix of exponentiated predicted effects from <code>cumfit</code> . Computed if <code>cumul=TRUE</code> .
<code>cumRRlow, cumRRhigh</code>	matrices with low and high confidence intervals for <code>cumRRfit</code> . Computed if <code>cumul=TRUE</code> .
<code>ci.level</code>	confidence level used for the computation of confidence intervals.
<code>model.class</code>	class of the model command used for estimation.
<code>model.link</code>	a specification for the model link function.

The function `summary.crosspred` returns a summary of the list.

Warnings

In case of collinear variables in the `basis` object, some of them are discarded and the related parameters not included in `model`. Then, `crosspred` will return an error. Check that the specification of the variables is meaningful through `summary.crossbasis` or `summary.onebasis`.

The name of the object `basis` will be used to extract the related estimated parameters from `model`. If more than one variable is transformed by cross-basis functions in the same model, different names must be specified.

Note

All the predictions are generated using a reference value corresponding to the centering point for continuous functions or to the default values for the other options (see `onebasis` and `crossbasis`). Exponentiated predicted effects are included if `model.link` (specified automatically by `model` or selected by the user) is equal to `log` or `logit`.

Author(s)

Antonio Gasparri, <antonio.gasparrini@lshtm.ac.uk>

References

Gasparrini A. Distributed lag linear and non-linear models in R: the package dlnm. *Journal of Statistical Software*. 2011; **43**(8):1-20. [freely available [here](#)].

Gasparrini A., Armstrong, B., Kenward M. G. Distributed lag non-linear models. *Statistics in Medicine*. 2010; **29**(21):2224-2234. [freely available [here](#)]

See Also

[onebasis](#) to generate one-dimensional basis matrices. [crossbasis](#) to generate cross-basis matrices. [crossreduce](#) to reduce the fit to one dimension. [plot.crosspred](#) to plot several type of graphs.

See [dlnm-package](#) for an overview of the package and type 'vignette(dlnmOverview)' for a detailed description.

Examples

```
### seasonal analysis
### space of predictor: linear effect above 40.3 microgr/m3 for O3
### space of predictor: linear effects below 15C and above 25C for temperature
### lag function: integer lag parameterization (unconstrained) for O3 up to lag5
### lag function: strata intervals at lag 0-1, 2-5 and 6-10 for temperature

# SELECT SUMMER MONTHS OF THE SERIES
chicagoNMMAPSseas <- subset(chicagoNMMAPS, month>5 & month<10)

# CREATE THE CROSS-BASIS FOR EACH PREDICTOR, SPECIFYING THE GROUP VARIABLE
cb2.o3 <- crossbasis(chicagoNMMAPSseas$o3, lag=5, argvar=list(type="hthr",
  knots=40.3), arglag=list(type="integer"), group=chicagoNMMAPSseas$year)
cb2.temp <- crossbasis(chicagoNMMAPSseas$temp, lag=10,
  argvar=list(type="dthr", knots=c(15,25)), arglag=list(type="strata",
  knots=c(2,6)), group=chicagoNMMAPSseas$year)
summary(cb2.o3)
summary(cb2.temp)

# RUN THE MODEL AND GET THE PREDICTION FOR O3
library(splines)
model2 <- glm(death ~ cb2.o3 + cb2.temp + ns(doy, 4) + dow,
  family=quasipoisson(), chicagoNMMAPSseas)
pred2.o3 <- crosspred(cb2.o3, model2, at=c(0:65,40.3,50.3))

# PLOT THE LINEAR ASSOCIATION OF O3 ALONG LAGS (WITH 80%CI)
plot(pred2.o3, "slices", var=50.3, ci="bars", type="p", pch=19, ci.level=0.80,
  main="Effects of 10-unit increase above the threshold (80CI)")
# PLOT THE OVERALL ASSOCIATION
plot(pred2.o3,"overall",xlab="Ozone", ci="lines", ylim=c(0.9,1.3), lwd=2,
  ci.arg=list(col=1,lty=3), main="Overall effects over 5 days of lag")

# GET THE FIGURES FOR THE OVERALL ASSOCIATION ABOVE, WITH CI
pred2.o3$allRRfit["50.3"]
cbind(pred2.o3$allRRlow, pred2.o3$allRRhigh)["50.3",]
```

```
# 3-D PLOT WITH DEFAULT AND USER-DEFINED PERSPECTIVE
plot(pred2.o3, xlab="Ozone", main="3D: default perspective")
plot(pred2.o3, xlab="Ozone", main="3D: different perspective", theta=250, phi=40)
### See the vignette 'dlnmOverview' for a detailed explanation of this example
```

crossreduce

Reduce the Fit of a DLNM to One Dimension

Description

Reduce the fit of a bi-dimensional DLNM to summaries defined in the the dimension of predictor or lags only, and re-expresses it in terms of modified parameters of the one-dimensional basis functions chosen for that space. Specifically, it summarizes lag-specific or overall predicted effects in the predictor dimension, or predictor-specific effects in the lag dimension. It returns the new parameters and predictions.

Usage

```
crossreduce(basis, model=NULL, type="overall", value=NULL, coef=NULL, vcov=NULL,
  model.link=NULL, at=NULL, from=NULL, to=NULL, by=NULL, bylag=1, ci.level=0.95)
```

```
## S3 method for class 'crossreduce'
summary(object, ...)
```

Arguments

basis	an object of class "crossbasis".
model	a model object for which the reduction and prediction are desired. See Details below.
coef, vcov, model.link	user-provided coefficients, (co)variance matrix and model link for the reduction and then prediction. See Details below.
type	type of reduction. Possible options are "overall" (default) or "lag" for reduction to the predictor space of overall or lag-specific effects, or "var" for reduction to the lag space of predictor-specific effects. See Details below.
value	the single value of predictor or lag for which predictor-specific or lag-specific functions must be defined, respectively. See Details below.
at	vector of values used for prediction in the dimension of predictor.
from, to	range of predictor values used for prediction.
by, bylag	increment of the sequences of predictor and lag values used for prediction.
ci.level	confidence level for the computation of confidence intervals.
object	an object of class "crossreduce".
...	additional arguments to be passed to summary.

Details

`model` is the model object including `basis`. It must include methods for `coef` and `vcov`, applied to extract the parameters. For model classes without these methods, the user can manually extract the related parameters and include them in `coef-vcov`, also specifying `model.link`. In this case, the dimensions and order of the first two must match the variables included in `basis`.

The dimension to which the fit is reduced is chosen by `type`, computing summaries for overall or lag-specific effects in the predictor dimension, or predictor-specific effects in the lag dimension. For specific effects, the value at which the reduction is computed is chosen by `value`. The function then re-express the original fit of the model, defined by the parameters of the bi-dimensional cross-basis functions, in summaries defined by the one-dimensional basis for the related space and a (usually smaller) set of modified parameters.

Similarly to `crosspred`, the object `basis` must be the same containing the cross-basis matrix included in `object`, including its attributes and class. The optional arguments `at` and `from/to/by` provides the values for predicted effects when the reduction is in the dimension of predictor. `bylag` determines instead the increment of the sequence of lag values. Exponentiated outcomes and confidence intervals are also optionally returned. See `crosspred` for details.

For a detailed illustration of the use of the functions, see:

```
vignette("dlnmOverview")
```

Value

A list object of class "crossreduce" with the following (optional) components:

<code>newcoef</code> , <code>newvcov</code>	reduced parameters of the original fitted model for the chosen dimension.
<code>newbasis</code>	basis matrix computed at <code>predvar</code> or for the sequence of lags defined by <code>lag</code> , depending on the chosen dimension.
<code>type</code> , <code>value</code>	type of reduction and (optional) value, as arguments above.
<code>predvar</code>	vector of observations used for prediction, if the reduction is in the dimension of predictor.
<code>lag</code>	integer vector defining the lag range.
<code>bylag</code>	increment of the sequence of lag values.
<code>fit</code> , <code>se</code>	vectors of predicted effects and related standard errors.
<code>RRfit</code>	vector of exponentiated predicted effects from <code>fit</code> .
<code>RRlow</code> , <code>RRhigh</code>	vectors with low and high confidence intervals for <code>RRfit</code> .
<code>ci.level</code>	confidence level used for the computation of confidence intervals.
<code>model.class</code>	class of the model command used for estimation.
<code>model.link</code>	a specification for the model link function.

Warnings

In case of collinear variables in the `basis` object, some of them are discarded and the related parameters not included in `model`. Then, `crossreduce` will return an error. Check that the specification of the variables is meaningful through `summary.crossbasis`.

The name of the object `basis` will be used to extract the related estimated parameters from object. If more than one variable is transformed by cross-basis functions in the same model, different names must be specified.

Note

All the predictions are generated using a reference value corresponding to the centering point for continuous functions or to the default values for the other options (see [onebasis](#) and [crossbasis](#)). Exponentiated predicted outcomes are included if `model.link` (specified automatically by object or selected by the user) is equal to `log` or `logit`.

Author(s)

Antonio Gasparriani, <antonio.gasparrini@lshtm.ac.uk>

References

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

See Also

[crossbasis](#) to generate cross-basis matrices. [crosspred](#) to obtain predictions after model fitting. [plot.crossreduce](#) to plot several the graphs.

See [dlnm-package](#) for an overview of the package and type `'vignette(dlnmOverview)'` for a detailed description.

Examples

```
# CREATE THE CROSS-BASIS: DOUBLE THRESHOLD AND NATURAL SPLINE
cb4 <- crossbasis(chicagoNMMAPS$temp, lag=30, argvar=list(type="dthr",
  knots=c(10,25)), arglag=list(df=5))

# RUN THE MODEL AND GET THE PREDICTION FOR TEMPERATURE
library(splines)
model4 <- glm(death ~ cb4 + ns(time, 7*14) + dow,
  family=quasipoisson(), chicagoNMMAPS)
pred4 <- crosspred(cb4, model4, by=1)

# REDUCE TO OVERALL ASSOCIATION
redall <- crossreduce(cb4, model4)
summary(redall)
# REDUCE TO LAG-SPECIFIC ASSOCIATION FOR LAG 5
redlag <- crossreduce(cb4, model4, type="lag", value=5)
# REDUCE TO PREDICTOR-SPECIFIC ASSOCIATION AT VALUE 33
redvar <- crossreduce(cb4, model4, type="var", value=33)

# NUMBER OF PARAMETERS OF THE ORIGINAL MODEL
length(coef(pred4))
# REDUCED NUMBER OF PARAMETERS FOR OVERALL AND LAG-SPECIFIC ASSOCIATIONS
length(coef(redall)) ; length(coef(redlag))
```

```

# REDUCED NUMBER OF PARAMETERS FOR PREDICTOR-SPECIFIC ASSOCIATIONS
length(coef(redvar))

# TEST: IDENTICAL FIT BETWEEN ORIGINAL AND REDUCED FIT
plot(pred4, "overall", xlab="Temperature", ylab="RR",
     ylim=c(0.8,1.6), main="Overall effects")
lines(redall, ci="lines",col=4,lty=2)
legend("top",c("Original","Reduced"),col=c(2,4),lty=1:2,ins=0.1)

# RECONSTRUCT THE FIT IN TERMS OF ONE-DIMENSIONAL BASIS
b4 <- onebasis(0:30,knots=attributes(cb4)$arglag$knots,int=TRUE,cen=FALSE)
pred4b <- crosspred(b4,coef=coef(redvar),vcov=vcov(redvar),model.link="log",by=1)

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

```

onebasis

Generate a Basis Matrix for Different Functions

Description

Generate the basis matrix for a predictor vector, choosing among a set of possible basis functions. Specifically, different types of splines, polynomials, strata and linear threshold functions.

Usage

```
onebasis(x, type="ns", df=1, degree=1, knots=NULL, bound, int=FALSE, cen)
```

```
## S3 method for class 'onebasis'
summary(object, ...)
```

Arguments

x	the predictor variable. Missing values are allowed.
type	type of basis. See Details below for the list of possible choices.
df	dimension of the basis, equivalent to number of degrees of freedom. They depend on knots if provided, or on degree for type="poly".
degree	degree of polynomial. Used only for type equal to "bs" (degree of the piecewise polynomial for the B-spline) or "poly" (degree of the polynomial).
knots	knots location for the basis function. They specify the position of the internal knots for "ns" and "bs", the cut-off points for "strata" (defining right-open intervals) and the threshold(s)/cut-off points for "lthr", "hthr" and "dthr". If provided, are automatically ordered and made unique, determining the value of df. If only df is provided, knots are placed at equally spaced quantiles.

bound	boundary knots (sometimes called external knots). Used only for type equal to "ns" and "bs". Default to the range of the variable.
int	logical. If TRUE, an 'intercept' is included in the basis matrix, with different meanings depending on type above: see Details below.
cen	logical or a numeric scalar. It specifies the value the basis functions for the space of predictor are centered at, then used as a reference for predictions. See Note below.
object	a object of class "onebasis".
...	additional arguments to be passed to summary.

Details

The value in type defines the basis function. It must be one of:

"ns": natural cubic B-splines (constrained to be linear beyond the boundary knots). Specified by knots (internal knots) and bound (boundary or external knots). If knots is provided, the dimension df is set to $\text{length}(\text{knots})+1+\text{int}$. An intercept is included if $\text{int}=\text{T}$ (corresponding to a vector of 1's if $df=1$, involving a more complex parameterization for $df>1$). See the functions [ns](#) for additional information. The transformed variables can be centered by cen.

"bs": B-splines characterized by degree (degree of the piecewise polynomial). Specified by knots (internal knots) and bound (boundary or external knots). If knots is provided, the dimension df is set to $\text{length}(\text{knots})+\text{degree}+\text{int}$; if not, df must be higher than $\text{degree}+\text{int}$. An intercept is included if $\text{int}=\text{T}$ (corresponding to a vector of 1's if $df=1$, involving a more complex parameterization for $df>1$). See the functions [bs](#) for additional information. The transformed variables can be centered by cen.

"strata": strata variables (dummy parameterization) determined by internal cut-off values specified in knots, which represent the lower boundaries for the right-open intervals. Intervals containing no observation are automatically discarded. If knots is provided, the dimension df is set to $\text{length}(\text{knots})+\text{int}$. A dummy variable for the reference stratum (the first one by default) is included if $\text{int}=\text{T}$, generating a full rank basis. Never centered.

"poly": polynomial with power specified by degree. The dimension df is set to $\text{degree}+\text{int}$. An intercept, corresponding to a vector of 1's (the power 0 of the polynomial) is included if $\text{int}=\text{T}$. The transformed variables can be centered by cen.

"integer": strata variables (dummy parameterization) for each integer values, expressly created to specify an unconstrained function in the space of lags. df is set automatically to the number of integer values minus 1 plus int . A dummy variable for the reference stratum (the first one by default) is included if $\text{int}=\text{T}$, generating a full rank basis. Never centered.

"hthr", "lthr": high and low threshold parameterization, with a linear relationship above or below the threshold, respectively, and flat otherwise. The threshold is chosen by knots: if more than one is provided, a piecewise linear relationship is applied above the first knot or below the last one, respectively, with the slope changing at each further knot. df is automatically set to $\text{length}(\text{knots})+\text{int}$. An intercept (corresponding to a vector of 1's) is included if $\text{int}=\text{T}$. Never centered.

"dthr": double threshold parameterization (2 independent linear relationships above the second and below the first threshold, flat between them). The thresholds are chosen by knots. If only one is provided, the threshold is unique (V-model). If more than 2 are provided, the first and the last ones are chosen. df is automatically set to $2+\text{int}$. An intercept (corresponding to a vector of 1's) is included if $\text{int}=\text{T}$. Never centered.

"lin": linear relationship (untransformed apart from optional centering). `df` is automatically set to `1+int`. An intercept (corresponding to a vector of 1's) is included if `int=T`. It can be centered by `cen`.

Results from models including basis functions are interpreted here relatively to a reference value of the predictor, determined automatically or through a centering point (see also Note below). In the latter case, the centering value is chosen by `cen` (if a numeric scalar), or fixed at the mean if `cen=TRUE`. The basis is uncentered for `cen=FALSE`. Some other arguments can be automatically changed for not sensible combinations, or set to `NULL` if not required. Use [summary.onebasis](#) to check the result.

For a detailed illustration of the use of the function, see:

```
vignette("dlnmOverview")
```

Value

A matrix object of class "onebasis" which can be included in a model formula in order to estimate the association. It contains the attribute `range` (range of the original vector of observations) and additional attributes corresponding to the arguments above. The function [summary.onebasis](#) returns a summary of the basis matrix and the related attributes, and can be used to check the options for the chosen basis function.

Warnings

Meaningless combinations of arguments (for example the inclusion of knots lying outside the range for type equal to "strata" or thr-type) could lead to collinear variables, with identifiability problems in the model and the exclusion of some of them.

Note

This function offers a wide range of options about modelling the shape of the exposure-response relationship, also extending the use of functions for splines or polynomials, which can be centered here. The function is called by [crossbasis](#) to generate cross-basis matrices. In addition, other functions in the package **dlnm** may be applied with these simple basis functions to obtain prediction (see [crosspred](#)) and to plot the results (see [plot.crosspred](#)).

This function has replaced the two old functions `mkbasis` and `mklagbasis` since version 1.5.0.

The name of the basis object will be used by [crosspred](#) in order to extract the related estimated parameters. If more than one variable is transformed through basis functions in the same model, different names must be specified.

For continuous functions specified with type equal to "ns", "bs", "poly" or "lin", the reference is set at the value specified by `cen`. For the other choices, the reference is automatic: for type equal to "strata" and "integer", the reference is the first interval, while for `vartype` equal to "hthr", "lthr" and "dthr", the reference is the region of null effect below, above or between the threshold(s), respectively. The inclusion of the intercept term nullifies the centering.

Author(s)

Antonio Gasparri, <antonio.gasparrini@lshtm.ac.uk>

See Also

[crossbasis](#) to generate cross-basis matrices. [crosspred](#) to obtain predictions after model fitting. [plot.crosspred](#) to plot several type of graphs.

See [dlnm-package](#) for an overview of the package and type 'vignette(dlnmOverview)' for a detailed description.

Examples

```
### relationship between PM10 and mortality: strata
b <- onebasis(chicagoNMMAPS$pm10, "strata", knots=c(20,40))
summary(b)
model <- glm(death ~ b, family=quasipoisson(), chicagoNMMAPS)
pred <- crosspred(b, model, at=0:60)
plot(pred, xlab="PM10", ylab="RR", main="RR for PM10")

### relationship between temperature and mortality: double threshold
b <- onebasis(chicagoNMMAPS$temp, "dthr", knots=c(10,25))
summary(b)
model <- glm(death ~ b, family=quasipoisson(), chicagoNMMAPS)
pred <- crosspred(b, model, by=1)
plot(pred, xlab="Temperature (C)", ylab="RR", main="RR for temperature")

### extending the example for the 'ns' function in package splines
b <- onebasis(women$height, df=5)
summary(b)
model <- lm(weight ~ b, data=women)
pred <- crosspred(b, model)
plot(pred, xlab="Height (in)", ylab="Weight (lb) difference",
      main="Association between weight and height")
```

plot.crosspred

Plot Predictions for a DLNM

Description

High and low-level method functions for graphs (3d, contour, slices and overall) of predictions from distributed lag non-linear models (DLNM).

Usage

```
## S3 method for class 'crosspred'
plot(x, ptype, var=NULL, lag=NULL, ci="area", ci.arg,
     ci.level=x$ci.level, cumul=FALSE, exp=NULL, ...)

## S3 method for class 'crosspred'
lines(x, ptype, var=NULL, lag=NULL, ci="n", ci.arg,
     ci.level=x$ci.level, cumul=FALSE, exp=NULL, ...)
```

```
## S3 method for class 'crosspred'
points(x, ptype, var=NULL, lag=NULL, ci="n", ci.arg,
       ci.level=x$ci.level, cumul=FALSE, exp=NULL, ...)
```

Arguments

x	an object of class "crosspred".
ptype	type of plot. Default to "3d" for lagged relationship, otherwise "overall". See Details below.
var, lag	vectors (for plot) or numeric scalars (for lines-points) of predictor or lag values for which specific effects must be plotted. Used only if ptype="slices".
ci	type of confidence intervals representation: one of "area", "bars", "lines" or "n". Default to "area" in high level functions, "n" for low-level functions.
ci.arg	list of arguments to be passed to low-level plotting functions to draw the confidence intervals. See Details.
ci.level	confidence level for the computation of confidence intervals.
cumul	logical. If TRUE, cumulative effects along lags are reported. Used only if type="slices". See Details.
exp	logical. It forces the choice about the exponentiation of effects. See Details.
...	optional graphical arguments. See Details.

Details

Different plots can be obtained by choosing the following values for the argument ptype:

"3d": a 3-D plot of predicted effects on the grid of predictor-lag values. Additional graphical arguments can be included, such as `theta-phi` (perspective), `border-shade` (surface), `xlab-ylab-zlab` (axis labelling) or `col`. See [persp](#) for additional information.

"contour": a contour/level plot of predicted effects on the grid of predictor-lag values. Additional graphical arguments can be included, such as `plot.title-plot.axes-key.title` for titles and axis and key labelling. Arguments `x-y-z` and `col-level` are automatically set and cannot be specified by the user. See [filled.contour](#) for additional information.

"overall": a plot of the overall effects (summing up all the single lag contributions). See [plot.default](#), [lines](#) and [points](#) for information on additional graphical arguments.

"slices": a (optionally multiple) plot of predictor-specific effects along the lag space, and/or lag-specific effects along the predictor space. Predictor and lag values are chosen by `var` and `lag`, respectively. See [plot.default](#), [lines](#) and [points](#) for information on additional graphical arguments.

The method function `plot` calls the high-level functions listed above for each ptype, while `lines-points` add lines or points for ptype equal to "overall" or "slices". These methods allow a great flexibility in the choice of graphical parameters, specified through arguments of the original plotting functions. Some arguments, if not specified, are set to different default values than the original functions.

Confidence intervals are plotted for ptype equal to "overall" or "slices". Their type is determined by `ci`, with options "area" (default for plot), "bars", "lines" or "n" (no confidence intervals, default for points and lines). The appearance may be modified through `ci.arg`, a list of

arguments passed to low-level plotting functions: `polygon` for "area", `segments` for "bars" and `lines` for "lines". See the original functions for a complete list of the arguments. This option offers flexibility in the choice of confidence intervals display. As above, some unspecified arguments are set to different default values.

For `ptype="slices"`, up to 4 plots for each dimension of predictor and lags are allowed in `plot`, while for `lines-points` a single plot in one of the two dimension must be chosen. Cumulative effects along lags are reported if `cumul=TRUE`: in this case, the same option must have been set to obtain the prediction saved in the `crosspred` object (see `crosspred`).

For a detailed illustration of the use of the functions, see:

```
vignette("dlnmOverview")
```

Warnings

The values in `var` must match those specified in the object `crosspred` (see `crosspred`), while the values in `lag` must be included in the lag period specified by `crossbasis`.

Note

These methods for class "crosspred" have replaced the old function `crossplot` since version 1.3.0. The old function has been kept in the namespace of the package, but its use is discouraged.

All the predictions are plotted using a reference value corresponding to the centering point for continuous functions or to the default values for the other options (see `onebasis` and `crossbasis`). Exponentiated predicted outcomes are returned by default if `x$model.link` is equal to `log` or `logit`.

Author(s)

Antonio Gasparini, <antonio.gasparini@lshtm.ac.uk>

References

- Gasparini A. Distributed lag linear and non-linear models in R: the package `dlnm`. *Journal of Statistical Software*. 2011; **43**(8):1-20. [freely available [here](#)].
- Gasparini A., Armstrong, B., Kenward M. G. Distributed lag non-linear models. *Statistics in Medicine*. 2010; **29**(21):2224-2234. [freely available [here](#)]

See Also

`crossbasis` to generate cross-basis matrices. `crosspred` to obtain predictions after model fitting.

See `dlnm-package` for an overview of the package and type `'vignette(dlnmOverview)'` for a detailed description.

Examples

```
### complex DLNM
### space of predictor: 5df quadratic spline for temperature
### space of predictor: linear effect for PM10
### lag function: 5df natural cubic spline for temperature up to lag30
### lag function: single strata at lag 0-1 for PM10
```

```

# CREATE THE CROSS-BASIS FOR EACH PREDICTOR AND CHECK WITH SUMMARY
cb3.pm <- crossbasis(chicagoNMMAPS$pm10, lag=1, argvar=list(type="lin",cen=0),
  arglag=list(type="strata"))
cb3.temp <- crossbasis(chicagoNMMAPS$temp, lag=30, argvar=list(type="bs",
  df=5,degree=2,cen=21), arglag=list(df=5))
summary(cb3.pm)
summary(cb3.temp)

# RUN THE MODEL AND GET THE PREDICTION FOR TEMPERATURE
library(splines)
model3 <- glm(death ~ cb3.pm + cb3.temp + ns(time, 7*14) + dow,
  family=quasipoisson(), chicagoNMMAPS)
pred3.temp <- crosspred(cb3.temp, model3, by=1)

# 3-D AND CONTOUR PLOTS
plot(pred3.temp, xlab="Temperature", col="red", zlab="RR", shade=0.6,
  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"))

# MULTIPLE SLICES
plot(pred3.temp, "slices", var=-20, ci="n", col=1, ylim=c(0.95,1.15), lwd=1.5,
  main="Lag-specific effects at different temperature, ref. 21C")
for(i in 1:3) lines(pred3.temp, "slices", var=c(0,27,33)[i], col=i+1, lwd=1.5)
legend("topright",paste("Temperature =",c(-20,0,27,33)), col=1:4, lwd=1.5)
plot(pred3.temp, "slices", var=c(-20,0,27,33), lag=c(0,5,15,28), col=4,
  ci.arg=list(density=40,col=grey(0.7)))

### See the vignette 'dlnmOverview' for a detailed explanation of this example

```

plot.crossreduce

Plot Predictions for a Reduced DLNM

Description

High and low-level method functions for graphs of predictions from reduced distributed lag non-linear models (DLNM).

Usage

```

## S3 method for class 'crossreduce'
plot(x, ci="area", ci.arg, ci.level=x$ci.level, exp=NULL, ...)

## S3 method for class 'crossreduce'
lines(x, ci="n", ci.arg, ci.level=x$ci.level, exp=NULL, ...)

## S3 method for class 'crossreduce'
points(x, ci="n", ci.arg, ci.level=x$ci.level, exp=NULL, ...)

```

Arguments

<code>x</code>	an object of class "crossreduce".
<code>ci</code>	type of confidence intervals representation: one of "area", "bars", "lines" or "n". Default to "area" in high level functions, "n" for low-level functions.
<code>ci.arg</code>	list of arguments to be passed to low-level plotting functions to draw the confidence intervals. See Details.
<code>ci.level</code>	confidence level for the computation of confidence intervals.
<code>exp</code>	logical. It forces the choice about the exponentiation of effects. See Details.
<code>...</code>	optional graphical arguments. See Details.

Details

Differently than for plotting functions for crosspred objects (see [plot.crosspred](#)), the type of the plot is automatically chosen by the dimension and values over which the model has been reduced. Namely, the overall or lag-specific effects along chosen values of the predictor, or predictor-specific effects along lags.

These methods allow a great flexibility in the choice of graphical parameters, specified through arguments of the original plotting functions. See [plot.default](#), [lines](#) and [points](#) for information on additional graphical arguments. Some arguments, if not specified, are set to different default values than the original functions.

Confidence intervals are plotted for ptype equal to "overall" or "slices". Their type is determined by `ci`, with options "area" (default for plot), "bars", "lines" or "n" (no confidence intervals, default for points and lines). The appearance may be modified through `ci.arg`, a list of arguments passed to low-level plotting functions: [polygon](#) for "area", [segments](#) for "bars" and [lines](#) for "lines". See the original functions for a complete list of the arguments. This option offers flexibility in the choice of confidence intervals display. As above, some unspecified arguments are set to different default values.

For a detailed illustration of the use of the functions, see:

```
vignette("dlnmOverview")
```

Note

All the predictions are plotted using a reference value corresponding to the centering point for continuous functions or to the default values for the other options (see [onebasis](#) and [crossbasis](#)). Exponentiated predicted outcomes are returned by default if `x$model.link` is equal to log or logit.

Author(s)

Antonio Gasparri, <antonio.gasparri@lshtm.ac.uk>

References

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

See Also

[onebasis](#) to generate simple basis matrices. [crosspred](#) to obtain predictions after model fitting. [crossreduce](#) to reduce the fit to one dimension.

See [dlnm-package](#) for an overview of the package and type 'vignette(dlnmOverview)' for a detailed description.

Examples

```
# CREATE THE CROSS-BASIS: DOUBLE THRESHOLD AND NATURAL SPLINE
cb4 <- crossbasis(chicagoNMMAPS$temp, lag=30, argvar=list(type="dthr",
  knots=c(10,25)), arglag=list(df=5))

# RUN THE MODEL AND GET THE PREDICTION FOR TEMPERATURE
library(splines)
model4 <- glm(death ~ cb4 + ns(time, 7*14) + dow,
  family=quasipoisson(), chicagoNMMAPS)
pred4 <- crosspred(cb4, model4, by=1)

# REDUCE TO OVERALL ASSOCIATION
redall <- crossreduce(cb4, model4)
summary(redall)
# REDUCE TO LAG-SPECIFIC ASSOCIATION FOR LAG 5
redlag <- crossreduce(cb4, model4, type="lag", value=5)
# REDUCE TO PREDICTOR-SPECIFIC ASSOCIATION AT VALUE 33
redvar <- crossreduce(cb4, model4, type="var", value=33)

# NUMBER OF PARAMETERS OF THE ORIGINAL MODEL
length(coef(pred4))
# REDUCED NUMBER OF PARAMETERS FOR OVERALL AND LAG-SPECIFIC ASSOCIATIONS
length(coef(redall)) ; length(coef(redlag))
# REDUCED NUMBER OF PARAMETERS FOR PREDICTOR-SPECIFIC ASSOCIATIONS
length(coef(redvar))

# TEST: IDENTICAL FIT BETWEEN ORIGINAL AND REDUCED FIT
plot(pred4, "overall", xlab="Temperature", ylab="RR",
  ylim=c(0.8,1.6), main="Overall effects")
lines(redall, ci="lines", col=4, lty=2)
legend("top", c("Original", "Reduced"), col=c(2,4), lty=1:2, ins=0.1)

# RECONSTRUCT THE FIT IN TERMS OF ONE-DIMENSIONAL BASIS
b4 <- onebasis(0:30, knots=attributes(cb4)$arglag$knots, int=TRUE, cen=FALSE)
pred4b <- crosspred(b4, coef=coef(redvar), vcov=vcov(redvar), model.link="log", by=1)

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

Index

- *Topic **aplot**
 - plot.crosspred, 18
 - plot.crossreduce, 21
- *Topic **datasets**
 - chicagoNMMAPS, 3
- *Topic **hplot**
 - plot.crosspred, 18
 - plot.crossreduce, 21
- *Topic **methods**
 - coef.crosspred, 4
- *Topic **package**
 - dlnm-package, 2
- *Topic **smooth**
 - crossbasis, 5
 - crosspred, 8
 - crossreduce, 12
 - onebasis, 15
- *Topic **ts**
 - crossbasis, 5
 - crosspred, 8
 - crossreduce, 12
- bs, 16
- chicagoNMMAPS, 3
- coef, 2, 9, 13
- coef.crosspred, 4
- coef.crossreduce (coef.crosspred), 4
- crossbasis, 2, 3, 5, 10, 11, 14, 17, 18, 20, 22
- crossplot (plot.crosspred), 18
- crosspred, 2, 3, 7, 8, 13, 14, 17, 18, 20, 23
- crossreduce, 2, 3, 11, 12, 23
- dlnm (dlnm-package), 2
- dlnm-package, 2
- filled.contour, 19
- glm, 2
- lines, 2, 19, 20, 22
- lines.crosspred (plot.crosspred), 18
- lines.crossreduce (plot.crossreduce), 21
- lm, 2
- mkbasis (onebasis), 15
- mklagbasis (onebasis), 15
- ns, 16
- onebasis, 2, 3, 6, 7, 9–11, 14, 15, 20, 22, 23
- persp, 19
- plot, 2
- plot.crosspred, 3, 7, 11, 17, 18, 18, 22
- plot.crossreduce, 3, 14, 21
- plot.default, 19, 22
- points, 2, 19, 22
- points.crosspred (plot.crosspred), 18
- points.crossreduce (plot.crossreduce), 21
- polygon, 20, 22
- pretty, 9
- segments, 20, 22
- summary.crossbasis, 6, 10, 13
- summary.crossbasis (crossbasis), 5
- summary.crosspred, 10
- summary.crosspred (crosspred), 8
- summary.crossreduce (crossreduce), 12
- summary.onebasis, 10, 17
- summary.onebasis (onebasis), 15
- vcov, 2, 9, 13
- vcov.crosspred (coef.crosspred), 4
- vcov.crossreduce (coef.crosspred), 4