Package 'dlnm'

October 7, 2021

Type Package

Version 2.4.7

Date 2021-10-07

Title Distributed Lag Non-Linear Models

Description Collection of functions for distributed lag linear and non-linear models.

Author Antonio Gasparrini [aut, cre],
Ben Armstrong [aut],
Fabian Scheipl [ctb]

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

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

Depends R (>= 3.2)

Suggests survival, lme4, gee, geepack, mixmeta

URL https://github.com/gasparrini/dlnm, http://www.ag-myresearch.com/package-dlnm

License GPL (>= 2)

LazyData yes

NeedsCompilation no

Repository CRAN

Date/Publication 2021-10-07 10:40:05 UTC

R topics documented:

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

Description

The package **dlnm** contains functions to specify and interpret distributed lag linear (DLMs) and non-linear (DLNMs) models. These functions are used to build basis and cross-basis matrices and then to predict and plot the results of a fitted model.

Modelling framework

Distributed lag non-linear models (DLNMs) represent a modelling framework to describe simultaneously non-linear and delayed dependencies, termed as *exposure-lag-response associations*. These include models for linear exposure-responses (DLMs) as special cases. The methodology of DLMs and DLNMs was originally developed for time series data, and has been recently extended to other study designs and data structures, compatible with cohort, case-control or longitudinal analyses, amongst others. A thorough methodological overview is given in the references and the package vignettes detailed below.

The modelling framework 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, produced by applying existing or user-defined functions such as splines, polynomials, linear threshold or indicators.

The application of DLMs and DLNMs requires the availability of predictor values measured at equally-spaced time points. In the original development in time series analysis, these are represented by the ordered series of observations. More generally, the data can be stored in a matrix of *exposure histories*, where each row represents the lagged values of the predictor for each observation.

The cross-basis matrix of transformed variables is included in the model formula of a regression model to estimate the associated parameters. The estimation can be carried out with the default regression functions, such as `lm`, `glm`, `gam` (package **mgcv**), `clogit` and `coxph` (package **survival**), `lme` (package **nlme**), `lmer` and `glmer` (package **lme4**). Estimates are then extracted to obtain predictions and graphical representations which facilitate the interpretation of the results.

Functions and data included in the package

In the standard usage, `crossbasis` creates two set of basis functions from a time series vector or a matrix of exposure histories to define the relationship in the two dimensions of predictor and lags. This step is performed through a call to the function `onebasis`, which in turn internally calls existing or user-defined functions and produces a basis matrix of class `"crossbasis"` with specific attributes. Standard choices for the functions in the two dimension are `ns` or `bs` from package **splines**, or the internal functions `poly`, `strata`, `thr`, `integer` and `lin` in **dlnm**. Other existing of user-defined functions can be also chosen. The functions `equalknots` and `logknots` can be used for knot placement. The two basis matrices are then combined in a matrix object of class `"crossbasis"`, containing the transformed variables to be included in the model formula.

In a more recent development, a penalized version of DLMs and DLNMs can be performed using two alternative approaches. In the *external* method, the functions `ps` or `cr` are called in `crossbasis` to derive the spline transformations, and the function `cbPen` is used to form the list of bi-dimensional penalty matrices. In the *internal* method, the cross-basis parameterization and matrix penalization are obtained directly using `smooth.construct.cb.smooth.spec`, a specific smooth constructor of class `"cb"`. This is used within the function `s` in the model formula. In both cases, the model is fitted using the regression function `gam` in **mgcv**.

After the model fitting, `crosspred` generates predictions for a set of suitable values of the original predictor and lag period, and stores them in a `"crosspred"` object. The function `exphist` can be used to generate exposure histories for predictions. The fit of a DLM or 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 objects `"onebasis"`, `"crossbasis"`, `"crosspred"` and `"crossreduce"`. Specific `summary` methods summarize the content of each object. 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) model.

The data set `chicagoNMMAPS` is provided to perform examples of use of **dlnm** in time series analysis. It includes time series data of daily mortality counts, weather and pollution variables for Chicago in the period 1987-2000. The data sets `nested` and `drug` include simulated data to illustrate the extension of **dlnm** to other study designs, specifically nested case-controls and randomized controlled trials. The former contains information on 300 risk sets each with one cancer case and one matched control, and an occupational exposure collected in 5-year periods. The latter contains information on 200 subjects who are randomly allocated a different dose of a drug for two out of four weeks, with their outcome measured after 28 days.

Additional information

Additonal details on the package **dlnm** are available in the vignettes included in the installation. These documents offer 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 **dlnmOverview** offers a general illustration of the DLM/DLNM methodology and the functions included in the package. The vignette **dlnmTS** illustrates specific examples on the use of the functions for time series analysis. The vignette **dlnmExtended** provides some examples on the extension of the methodology and package in other study designs and on the use of user-written functions. The vignette **dlnmPenalized** describes the definition of DLMs and DLNMs through penalized splines.

A vignette is available by typing:

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

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

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

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).

Please use `citation("dlnm")` to cite this package.

Author(s)

Antonio Gasparrini and Ben Armstrong, with contributions from Fabian Scheipl

Maintainer: Antonio Gasparrini <<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, Scheipl F, Armstrong B, Kenward MG. A penalized framework for distributed lag non-linear models. *Biometrics*. 2017;**73**(3):938-948. [freely available here]

Gasparrini A. Modelling lagged associations in environmental time series data: a simulation study. *Epidemiology*. 2016;**27**(6):835-842. [freely available here]

Gasparrini A. Modeling exposure-lag-response associations with distributed lag non-linear models. *Statistics in Medicine*. 2014;**33**(5):881-899. [freely available here]

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

Gasparrini A, Armstrong B, Kenward MG. Reducing and meta-analyzing estimates from distributed lag non-linear models.*BMC Medical Research Methodology*. 2013;**13**(1):1. [freely available here].

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

See Also

`onebasis` to generate simple basis matrices. `crossbasis` to generate cross-basis matrices. `cb` `smooth` `constructor` for a penalized version. `crosspred` to obtain predictions after model fitting.

crossreduce to reduce the fit to one dimension. The methods plot.crosspred and plot.crossreduce to plot several type of graphs.

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

---

cbPen *Generate Penalty Matrices for a DLNM*

Description

This function generates penalty matrices for the two dimensions of predictor and lags, given the functions selected to model the relationship in each space. It can also be used for generating the single penalty matrix for the predictor space of a uni-dimensional basis not accouning for lags.

Usage

```
cbPen(cb, sp=-1, addSlag=NULL)
```

Arguments

cb	an object of class "crossbasis" or "onebasis".
sp	supplied smoothing parameters. See Details below.
addSlag	matrix or vector (or list of matrices and/or vectors) defining additional penalties on the lag structure. See Details below.

Details

This function is used to perform penalized regression models using the *external* method. This involves generating the transformation using crossbasis or onebasis with functions for penalized splines (either ps or cr). The function cbPen is then called to generate a list of the related penalty matrices. The model is performed by penalizing so-called parametric terms in the gam function of **mgcv**, by including the basis or cross-basis matrix in the regression formula and the list of penalty matrices in its paraPen argument.

When cb is a cross-basis object, the penalty matrices for the two spaces of predictor and lags are rescaled and expanded accordingly to its tensor product-type structure. A penalty matrix is not defined when using a function different than ps or cr, thus keeping one of the two dimensions unpenalized.

Additional penalties on the lag dimension can be added through the argument addSlag, either as a single matrix or a list of matrices. If provided as a vector, this is taken as the diagonal of the penalty matrix and expanded accordingly. These objects must have appropriate dimensions in accordance with the basis matrix for the lag space.

All the penalty matrices are also appropriately rescaled to improve the estimation process.

The vector sp must have the same length as the number of penalties, including additional penalties on the lags, and it is replicated accordingly if of length 1. Positive or zero elements are taken as fixed smoothing parameters. Negative elements signal that these parameters need to be estimated.

Value

A list including penalty matrices plus two vectors `rank` and `sp` defining their rank and the smoothing parameters. This list is consistent with the argument `paraPen` in the regression function `gam` function of **mgcv**.

Author(s)

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

References

Gasparrini A, Scheipl F, Armstrong B, Kenward MG. A penalized framework for distributed lag non-linear models. *Biometrics*. 2017;**73**(3):938-948. [freely available here]

Wood S. N. Generalized Additive Models: An Introduction with R. Chapman and Hall/CRC Press, 2006.

See Also

`ps` and `cr` for penalized spline functions. The `cb` `smooth` `constructor` for cross-basis penalized spline smooths.

See `dlnm-package` for an introduction to the package and for links to package vignettes providing more detailed information, in particular the vignette **dlnmPenalized**.

Examples

```
# to be added soon
```

---

chicagoNMMAPS *Daily Mortality Weather and Pollution Data for Chicago*

Description

The data set 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.

Source

The complete dataset used to be available at the Internet-based Health and Air Pollution Surveillance System (iHAPSS) website, or through the packages **NMMAPSdata** or **NMMAPSlite**. Currently, the data are not available any more and the two packages have been archived.

See Also

`nested` for an example of analysing exposure-lag-response associations in a nested case-control study. `drug` for an example of analysing exposure-lag-response associations in a randomized controlled trial.

The application of DLNMs to this data with more detailed examples are given in vignette **dlnmExtended**.

See `dlnm-package` for an introduction to the package and for links to package vignettes providing more detailed information.

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

See Also

See dlnm-package for an introduction to the package and for links to package vignettes providing more detailed information.

cr *Generate a Basis Matrix for Penalized Cubic Regression Splines*

Description

Generate the basis matrix for cubic regression splines with penalties on the second derivatives.

Usage

```
cr(x, df=10, knots=NULL, intercept=FALSE, fx= FALSE, S=NULL)
```

Arguments

x	the predictor variable. Missing values are allowed.
df	degrees of freedom, basically the dimension of the basis matrix. If supplied in the absence of knots, it automatically selects df+1-intercept knots at equally-spaced quantiles of x. The minimum allowed is df=3.
knots	breakpoints that define the spline. These are generally automatically selected, and not defined by the user. See Details below.
intercept	logical. If TRUE, an intercept is included in the basis matrix. See Details below.
fx	logical. If TRUE, it removes the penalization. See Details below.
S	penalty matrix, usually internally defined if NULL (default).

Details

The function has a usage similar to bs and ns in the **splines** package. It produces spline transformations, however using a parameterization that represents the splines fit in terms of values at the knots. A penalty matrix is also defined. The same results are returned by the related smooth constructor in the package **mgcv**, which is in fact called internally.

The argument knots defines a vector of knots within the range of the predictor x, by default at equally-spaced quantiles. The penalization is defined on the second derivative of the function through a penalty matrix S.

Similarly to bs and ns, setting intercept=FALSE (default) determines the exclusion of the first transformed variables, and the corresponding first row and column in S, thus avoiding identifiability issues during the model fitting. Note how the procedure of imposing identifiability constraints is different from that adopted by smoothCon in the package **mgcv**, where a more complex reparameterization is produced.

Value

A matrix object of class "cr". It contains the attributes df, knots, intercept, fx, and S, with values that can be different than the arguments provided due to internal reset.

Note

The function is primarily added here to specify penalized DLMs and DLNMs using the so-called *external* method, *i.e.* by including the penalty matrix in the argument paraPen of the gam regression function in **mgcv** (see cbPen). However, this approach can be also used to fit standard uni-dimensional penalized cubic spline models as an alternative to the use of specific smooth constructor, as it takes advantage of the use of prediction and plotting functions in **dlnm**.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>>, with internall calls to functions included in the package **mgcv** by Simon N. Wood.

References

Gasparrini A, Scheipl F, Armstrong B, Kenward MG. A penalized framework for distributed lag non-linear models. *Biometrics*. 2017;**73**(3):938-948. [freely available here]

Wood S. N. Generalized Additive Models: An Introduction with R. Chapman and Hall/CRC Press, 2006.

See Also

`ps` for P-splines. `bs` and `ns` for B-splines and natural cubic splines, respectively. `cbPen` for defining tensor-type bi-dimensional penalties in DLNMs. The related `smooth constructor` for cubic regression spline smooths in **mgcv**. The `cb smooth constructor` for cross-basis penalized spline smooths.

See `dlnm-package` for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
# to be added soon
```

---

crossbasis *Generate a Cross-Basis Matrix for a DLNM*

Description

The function generates the basis matrices for the two dimensions of predictor and lags, given the functions selected to model the relationship in each space. Then, these one-dimensions basis matrices are combined in order to create the related cross-basis matrix, which can be included in a model formula to fit distributed lag linear (DLMs) and non-linear models (DLNMs).

Usage

```
crossbasis(x, lag, argvar=list(), arglag=list(), group=NULL, ...)

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

Arguments

`x`	either a numeric vector representing a complete series of ordered observations (for time series data), or a matrix of exposure histories over the same lag period for each observation. See Details below.
`lag`	either an integer scalar or vector of length 2, defining the the maximum lag or the lag range, respectively.
`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 or a list of factors defining groups of observations. Only for time series data.
object	a object of class "crossbasis".
...	additional arguments. See Details below.

Details

The argument x defines the type of data. If a n-dimensional vector, the data are interpreted as a time series of equally-spaced and complete observations. If a $n \times (L - \ell_0 + 1)$ matrix, the data are interpreted as a set of complete exposure histories at equally-spaced lags over the same lag period from ℓ_0 to L for each observation. The latter is general and can be used for applying DLMs and DLNMs beyond time series data. Lags are usually positive integers: if not provided, by default the minimum lag $L0$ is set to 0, and the maximum lag L is set to 0 if x is a vector or to ncol(x)-1 otherwise. Negative lags are rarely needed but allowed.

The lists in argvar and arglag are passed to onebasis, which calls existing or user-defined functions to build the related basis matrices. The two lists should contain the argument fun defining the chosen function, and a set of additional arguments of the function. 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. By default, the basis functions for lags are defined with an intercept (if not otherwise stated). Some arguments can be automatically re-set by onebasis. Then, the two set of basis matrices are combined in order to create the related cross-basis matrix.

Common choices for fun are represented by ns and bs from package **splines** or by the internal functions of the package **dlnm**, namely poly, strata, thr, integer and lin. In particular, DLMs can be considered a special case of DLNMs with a linear function in argvar. Functions ps and cr are used to specify penalized models with an external method (see cbPen). See help(onebasis) and the help pages of these functions for information on the additional arguments to be specified. Also, other existing or user-defined functions can be applied.

The argument group, only used for time series data, defines groups of observations representing independent series. Each series must be consecutive, complete and ordered.

Value

A matrix object of class "crossbasis" which can be included in a model formula in order to fit a DLM or DLNM. It contains the attributes df (vector of length 2 with the df for each dimension), 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 lists used in the call). The method 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

In previous versions of the package the function adopted a different usage. In particular, the argvar list should not include a cen argument any more (see Note in this help page or onebasis). Users are strongly suggested to comply with the current usage, as backward compatibility may be discontinued in future versions of the package.

Meaningless combinations of arguments in `argvar` and `arglag` passed to `onebasis` 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` (`intercept` 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

Missing values in `x` are allowed, but this causes the observation (for non-time series data with `x` as a matrix) or the following observations corresponding to the lag period (for time series data with `x` as a vector series) to be set to `NA`. Although correct, this could generate computational problems in the presence of a high number of missing observations.

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.

Before version 2.2.0 of **dlnm**, the `argvar` list could include a `cen` argument to be passed internally to `onebasis` for centering the basis. This step is now moved to the prediction stage, with a `cen` argument in `crosspred` or `crossreduce` (see the related help pages). For backward compatibility, the use of `cen` in `crossbasis` is still allowed (with a warning), but may be discontinued in future versions.

Author(s)

Antonio Gasparrini <<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, Scheipl F, Armstrong B, Kenward MG. A penalized framework for distributed lag non-linear models. *Biometrics*. 2017;**73**(3):938-948. [freely available here]

Gasparrini A. Modeling exposure-lag-response associations with distributed lag non-linear models. *Statistics in Medicine*. 2014;**33**(5):881-899. [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. The `cb smooth constructor` for cross-basis penalized spline smooths. `crosspred` to obtain predictions after model fitting. The method function `plot` to plot several type of graphs.

See `dlnm-package` for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
### example of application in time series analysis - see vignette("dlnmTS")

# create the crossbasis objects and summarize their contents
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))
summary(cb1.pm)
summary(cb1.temp)

# run the model and get the predictions for pm10
library(splines)
model1 <- glm(death ~ cb1.pm + cb1.temp + ns(time, 7*14) + dow,
  family=quasipoisson(), chicagoNMMAPS)
pred1.pm <- crosspred(cb1.pm, model1, at=0:20, bylag=0.2, cumul=TRUE)

# plot the lag-response curves for specific and incremental cumulative effects
plot(pred1.pm, "slices", var=10, col=3, ylab="RR", ci.arg=list(density=15,lwd=2),
  main="Lag-response curve for a 10-unit increase in PM10")
plot(pred1.pm, "slices", var=10, col=2, cumul=TRUE, ylab="Cumulative RR",
  main="Lag-response curve of incremental cumulative effects")

### example of application beyond time series - see vignette("dlnmExtended")

# generate the matrix of exposure histories from the 5-year periods
Qnest <- t(apply(nested, 1, function(sub) exphist(rep(c(0,0,0,sub[5:14]),
  each=5), sub["age"], lag=c(3,40))))

# define the cross-basis
cbnest <- crossbasis(Qnest, lag=c(3,40), argvar=list("bs",degree=2,df=3),
  arglag=list(fun="ns",knots=c(10,30),intercept=FALSE))
summary(cbnest)

# run the model and predict
library(survival)
mnest <- clogit(case~cbnest+strata(risket), nested)
pnest <- crosspred(cbnest,mnest, cen=0, at=0:20*5)

# bi-dimensional exposure-lag-response association
plot(pnest, zlab="OR", xlab="Exposure", ylab="Lag (years)")
# lag-response curve for dose 60
plot(pnest, var=50, ylab="OR for exposure 50", xlab="Lag (years)", xlim=c(0,40))
# exposure-response curve for lag 10
plot(pnest, lag=5, ylab="OR at lag 5", xlab="Exposure", ylim=c(0.95,1.15))

### example of extended predictions - see vignette("dlnmExtended")

# compute exposure profiles and exposure history
expnested <- rep(c(10,0,13), c(5,5,10))
hist <- exphist(expnested, time=length(expnested), lag=c(3,40))
```

```
# predict association with a specific exposure history
pnesthist <- crosspred(cbnest, mnest, cen=0, at=hist)
with(pnesthist, c(allRRfit,allRRlow,allRRhigh))

### example of user-defined functions - see vignette("dlnmExtended")

# define a log function
mylog <- function(x) log(x+1)

# define the cross-basis
cbnest2 <- crossbasis(Qnest, lag=c(3,40), argvar=list("mylog"),
  arglag=list(fun="ns",knots=c(10,30),intercept=FALSE))
summary(cbnest2)

# run the model and predict
mnest2 <- clogit(case~cbnest2+strata(riskset), nested)
pnest2 <- crosspred(cbnest2, mnest2, cen=0, at=0:20*5)

# plot and compare with previous fit
plot(pnest2, zlab="OR", xlab="Exposure", ylab="Lag (years)")
plot(pnest2, var=50, ylab="OR for exposure 50", xlab="Lag (years)", xlim=c(0,40))
lines(pnest, var=50, lty=2)
plot(pnest2, lag=5, ylab="OR at lag 5", xlab="Exposure", ylim=c(0.95,1.15))
lines(pnest, lag=5, lty=2)

### example of penalized models - see vignette("dlnmPenalized")

# to be added soon
```

`crosspred` *Generate Predictions for a DLNM*

Description

The function generates predictions from distributed lag linear (DLMs) and non-linear models (DLNMs). These are interpreted as estimated associations defined on a grid of values of the original predictor and lags, computed versus a reference predictor value. This function can be used more generally to generate predictions and facilitate interpretation for uni-dimensional unlagged models.

Usage

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

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

Arguments

basis	usually an object of class "crossbasis" or "onebasis". Alternatively a character string for penalized models. See Details below.
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	either a numeric vector representing the values of a constant exposure throughout the lag period defined by lag, or a matrix of exposure histories over the same lag period used for estimation.
from, to	range of predictor values used for prediction.
lag	either an integer scalar or vector of length 2, defining the lag range used for prediction. Defalut to values used for estimation.
by, bylag	increment of the sequences of predictor and lag values used for prediction.
cen	logical or a numeric scalar. It specifies the centering value, then used as a reference for predictions. See Details below.
ci.level	confidence level for the computation of confidence intervals.
cumul	logical. If TRUE, incremental cumulative associations along (integer) lags are also predicted. See Details.
object	an object of class "crosspred".
...	additional arguments to be passed to summary.

Details

model is the model object including basis in its formula. basis is usually an object representing the cross-basis or basis matrix included in model, preserving its attributes and class. Alternatively, for penalized models fitted with gam, basis can be a character string identifying the first argument of s in the model formula (see the cb smooth constructor). Examples are provided below in the related section.

The function computes predictions for specific combinations of predictor and lag values, and the net overall predictions accounting for the whole lag period. By default, predictor values are set internally as approximately 50 equally-spaced points within the range, or alternatively directly defined through at or from/to/by. Lag values are are set by default at all the integer values within the lag period, or determined by lag and bylag.

The values in at can be provided as a vector, and in this case they are replicated for each lag. As an alternative usage, at can be provided as a matrix of complete exposure histories over the same lag period used for estimation, in order to compute the association with a specific exposure pattern (see also exphist).

Predictions are computed versus a reference value, with default values dependent on the function used in basis, or manually set through cen. Briefly, sensible default values are automatically defined for strata, thr and integer (corresponding to the reference region), and for lin (corresponding to 0). For other choices, such as ns, bs, poly or other existing or user-defined functions, the centering value is set by default to the mid-range. The inclusion of the intercept in basis term nullifies the centering.

Exponentiated predictions are included if model.link is equal to "log" or "logit". Confidence intervals computed using a normal approximation and a confidence level of ci.level. model.link is automatically selected from model for some classes when set to NULL (default), but needs to be provided for different classes. Matrices with incremental cumulative predicted associations along integer lags at each exposure values used for prediction are included if cumul=TRUE.

The function automatically works with model objects from regression function lm and glm, gam (package **mgcv**), coxph and clogit (package **survival**), lme and nlme (package **nlme**), lmer and glmer and nlmer (package **lme4**), gee (package **gee**), geeglm (package **geepack**). The function also works with any regression function for which coef and vcov methods are available. Otherwise, the user needs to input the coefficients and associated (co)variance matrix related to the parameters of the crossbasis as arguments coef and vcov, and information on the link function in model.link. In this case, dimensions and order must match the variables included in basis.

The function can be used to compute predictions for models with simple uni-dimensional basis functions not including lag, derived either with onebasis or with smooth penalized functions in gam. In this case, only unlagged predicted associations are returned.

Value

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

predvar — vector or matrix of values used for prediction, depending on the format of the argument at (see Details above).

cen — (optional) numeric scalar defining the centering value.

lag — integer vector defining the lag range used for prediction.

bylag — increment of the sequence of lag values.

coefficients, vcov — coefficients and their variance-covariance matrix.

matfit, matse — matrices of predictions and standard errors at the chosen combinations of predictor and lag values.

matlow, mathigh — matrices of confidence intervals for matfit.

allfit, allse — vectors of the overall cumulative predicted association and standard errors.

alllow, allhigh — vectors of confidence intervals for allfit.

cumfit, cumse — matrices of incremental cumulative predicted associations along lags and related standard errors at the chosen combinations of predictor and (integer) lag values. Computed if cumul=TRUE.

cumlow, cumhigh — matrices of confidence intervals for cumfit. Computed if cumul=TRUE.

matRRfit — matrix of exponentiated specific associations from matfit.

matRRlow, matRRhigh — matrices of confidence intervals for matRRfit.

allRRfit — vector of exponentiated overall cumulative associations from allfit.

allRRlow, allRRhigh — vectors of confidence intervals for allRRfit.

`cumRRfit` matrix of exponentiated incremental cumulative associations from `cumfit`. Computed if `cumul=TRUE`.

`cumRRlow, cumRRhigh` matrix of confidence intervals for . Computed if `cumul=TRUE`.

`ci.level` confidence level used for the computation of confidence intervals for `cumRRfit`.

`model.class` class of the model command used for estimation.

`model.link` 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, which if not directly specific by `cen` is given default values corresponding to (approximately) the mid-range point for continuous functions. Before version 2.2.0 of **dlnm**, centering was produced in `onebasis` or `crossbasis` (see the related help pages), and for backward compatibility this information is kept (with a warning) and used in `crosspred` unless `cen` is directly defined as an argument.

Exponentiated predictions are included if `model.link` (selected by the user or specified automatically by `model`) is equal to `"log"` or `"logit"`.

Author(s)

Antonio Gasparrini <<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, Scheipl F, Armstrong B, Kenward MG. A penalized framework for distributed lag non-linear models. *Biometrics*. 2017;**73**(3):938-948. [freely available here]

Gasparrini A. Modeling exposure-lag-response associations with distributed lag non-linear models. *Statistics in Medicine*. 2014;**33**(5):881-899. [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. The method function plot to plot several type of graphs.

See dlnm-package for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
### example of application in time series analysis - see vignette("dlnmTS")

# seasonal analysis: select summer months only
chicagoNMMAPSseas <- subset(chicagoNMMAPS, month>5 & month<10)

# create the crossbasis objects, including info on groups
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)
summary(cb2.o3)
summary(cb2.temp)

# run the model
library(splines)
model2 <- glm(death ~  cb2.o3 + cb2.temp + ns(doy, 4) + ns(time,3) + dow,
  family=quasipoisson(), chicagoNMMAPSseas)

# get the predictions for o3 at specific exposure values
pred2.o3 <- crosspred(cb2.o3, model2, at=c(0:65,40.3,50.3))

# get figures for the overall cumulative association, with ci
pred2.o3$allRRfit["50.3"]
cbind(pred2.o3$allRRlow, pred2.o3$allRRhigh)["50.3",]

# plot the estimated lag-response curve (with 80%CI)
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 the estimated overall cumulative exposure-response curve
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")

# plot the estimated exposure-lag-response surface
plot(pred2.o3, xlab="Ozone", main="3D: default perspective")
plot(pred2.o3, xlab="Ozone", main="3D: different perspective", theta=250, phi=40)

### example of application beyond time series - see vignette("dlnmExtended")

# generate the matrix of exposure histories from the weekly data
Qdrug <- as.matrix(drug[,rep(7:4, each=7)])
```

```
colnames(Qdrug) <- paste("lag", 0:27, sep="")

# define the cross-basis
cbdrug <- crossbasis(Qdrug, lag=27, argvar=list("lin"),
  arglag=list(fun="ns",knots=c(9,18)))

# run the model, predict, and show estimates for specific values
mdrug <- lm(out~cbdrug+sex, drug)
pdrug <- crosspred(cbdrug, mdrug, at=0:20*5)
with(pdrug,cbind(allfit,alllow,allhigh)["50",])
pdrug$matfit["20","lag3"]

# bi-dimensional exposure-lag-response association
plot(pdrug, zlab="Effect", xlab="Dose", ylab="Lag (days)")
plot(pdrug, var=60, ylab="Effect at dose 60", xlab="Lag (days)", ylim=c(-1,5))
plot(pdrug, lag=10, ylab="Effect at lag 10", xlab="Dose", ylim=c(-1,5))

### example of extended predictions - see vignette("dlnmExtended")

# dose 20 for 10 days
histdrug <- exphist(rep(20,10), time=10, lag=27)
pdrug4 <- crosspred(cbdrug, mdrug, at=histdrug)
with(pdrug4,c(allfit,alllow,allhigh))

# define exposure profile with weekly exposures to 10, 50, 0 and 20
expdrug <- rep(c(10,50,0,20),c(2,1,1,2)*7)

# define the exposure histories for all the time points
dynhist <- exphist(expdrug, lag=27)

# predict the effects
pdyndrug <- crosspred(cbdrug, mdrug, at=dynhist)

# plot of the evolution of the effects along time given the doses
plot(pdyndrug,"overall", ylab="Effect", xlab="Time (days)", ylim=c(-5,27),
  xlim=c(1,50))

### example of user-defined functions - see vignette("dlnmExtended")

# define the decay function
fdecay <- function(x, scale=5, ...) {
  basis <- exp(-x/scale)
  attributes(basis)$scale <- scale
  return(basis)
}

# define the cross-basis
cbdrug2 <- crossbasis(Qdrug, lag=27, argvar=list("lin"),
  arglag=list(fun="fdecay",scale=6))
summary(cbdrug2)

# run the model and predict
mdrug2 <- lm(out~cbdrug2+sex, drug)
```

```
pdrug2 <- crosspred(cbdrug2, mdrug2, at=0:20*5)

# plot and compare with previous fit
plot(pdrug2, zlab="Effect", xlab="Dose", ylab="Lag (days)")
plot(pdrug2, var=60, ylab="Effect at dose 60", xlab="Lag (days)", ylim=c(-1,5))
lines(pdrug, var=60, lty=2)
plot(pdrug2, lag=10, ylab="Effect at lag 10", xlab="Dose", ylim=c(-1,5))
lines(pdrug, lag=10, lty=2)

### example of general use for regression models - see vignette("dlnmExtended")

# replicate example illustrated in help(ns)
library(splines)
oneheight <- onebasis(women$height, "ns", df=5)
mwomen <- lm(weight ~ oneheight, data=women)
pwomen <- crosspred(oneheight, mwomen, cen=65, at=58:72)
with(pwomen, cbind(allfit, alllow, allhigh)["70",])
plot(pwomen, ci="l", ylab="Weight (lb) difference", xlab="Height (in)", col=4)

# replicate example illustrated in help(gam)
library(mgcv)
dat <- gamSim(1,n=200,dist="poisson",scale=.1)
b2 <- gam(y ~ s(x0,bs="cr") + s(x1,bs="cr") + s(x2,bs="cr") + s(x3,bs="cr"),
  family=poisson, data=dat, method="REML")
plot(b2, select=3)
pgam <- crosspred("x2", b2, cen=0, at=0:100/100)
with(pgam, cbind(allRRfit, allRRlow, allRRhigh)["0.7",])
plot(pgam, ylim=c(0,3), ylab="RR", xlab="x2", col=2)

### example of penalized models - see vignette("dlnmPenalized")

# to be added soon
```

`crossreduce` *Reduce the Fit of a DLNM to One-Dimensional Summaries*

Description

The function reduces the fit of bi-dimensional distributed lag linear (DLMs) or non-linear (DLNMs) models 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.

Usage

```
crossreduce(basis, model=NULL, type="overall", value=NULL, coef=NULL, vcov=NULL,
  model.link=NULL, at=NULL, from=NULL, to=NULL, by=NULL, lag, bylag=1, cen=NULL,
  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) for reduction to the overall cumulative exposure-response association, "lag" for reduction to a lag-specific exposure-response association, or "var" for reduction to a predictor-specific lag-response association. See Details below.
value	the single value of predictor or lag at which predictor-specific or lag-specific associations 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.
lag	either an integer scalar or vector of length 2, defining the lag range used for prediction. Defalut to values used for estimation.
by, bylag	increment of the sequences of predictor and lag values used for prediction.
cen	logical or a numeric scalar. It specifies the centering value, then used as a reference for predictions. See Details below.
ci.level	confidence level for the computation of confidence intervals.
object	an object of class "crossreduce".
...	additional arguments to be passed to summary.

Details

The dimension to which the fit is reduced is chosen by type, computing summaries for overall cumulative or lag-specific associations defining an exposure-response relationship in the predictor space, or predictor-specific associations defining a lag-response relationship in the lag space. The function re-expresses 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 model, with its attributes and class. The function computes predictions for specific values of predictor (for type equal to "overall" and "lag") or lag (for for type equal to "var"). Values are set to default or chosen thorugh at/from/to/by and lag/bylag, respectively.

Predictions are computed versus a reference value, with default values dependent on the function used in basis, or manually set through cen. Briefly, sensible default values are automatically defined for strata, thr and integer (corresponding to the reference region), and for lin (corresponding to 0). For other choices, such as ns, bs, poly or other existing or user-defined functions, the centering value is set by default to the mid-range. The inclusion of the intercept in basis term nullifies the centering.

Exponentiated predictions are included if model.link is equal to "log" or "logit". Confidence intervals computed using a normal approximation and a confidence level of ci.level. model.link

is automatically selected from model for some classes when set to NULL (default), but needs to be provided for different classes.

The function automatically works with model objects from regression function lm and glm, gam (package **mgcv**), coxph and clogit (package **survival**), lme and nlme (package **nlme**), lmer and glmer and nlmer (package **lme4**), gee (package **gee**), geeglm (package **geepack**). The function also works with any regression function for which coef and vcov methods are available and return appropriately named objects. Otherwise, the user needs to input the coefficients and associated (co)variance matrix related to the parameters of the crossbasis as arguments coef and vcov. In this case, their dimensions and order must match the variables included in basis.

Value

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

coefficients, vcov
: reduced parameters of the original fitted model for the chosen dimension.

basis
: basis matrix computed at predvar or for the sequence of lags defined by lag, depending on the chosen dimension.

type, value
: type of reduction and (optional) value, as arguments above.

cen
: (optional) numeric scalar defining the centering value.

predvar
: vector of observations used for prediction, if the reduction is in the dimension of predictor.

lag
: integer vector defining the lag range.

bylag
: increment of the sequence of lag values.

fit, se
: vectors of the predicted association and related standard errors.

low, high
: vectors of confidence intervals for fit.

RRfit
: vector of exponentiated predicted associations from fit.

RRlow, RRhigh
: vectors of confidence intervals for RRfit.

ci.level
: confidence level used for the computation of confidence intervals.

model.class
: class of the model command used for estimation.

model.link
: 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.

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, which if not directly specific by `cen` is given default values corresponding to (approximately) the mid-range point for continuous functions. Before version 2.2.0 of **dlnm**, centering was produced in `crossbasis` (see the related help page), and for backward compatibility this information is kept (with a warning) and used in `crossreduce` unless `cen` is directly defined as an argument.

Exponentiated predictions are included if `model.link` (selected by the user or specified automatically by `model`) is equal to `"log"` or `"logit"`.

Author(s)

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

References

Gasparrini A., Armstrong, B., Kenward M. G. Reducing and meta-analyzing estimates from distributed lag non-linear models.*BMC Medical Research Methodology*. 2013;**13**(1):1. [freely available here].

See Also

`crossbasis` to generate cross-basis matrices. `crosspred` to obtain predictions after model fitting. The method function `plot` to plot the association.

See `dlnm-package` for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
# create the crossbasis object
lagnk <- 3
lagknots <- exp(((1+log(30))/(lagnk+1) * seq(lagnk))-1)
cb4 <- crossbasis(chicagoNMMAPS$temp, lag=30, argvar=list(fun="thr",
  thr=c(10,25)), arglag=list(knots=lagknots))

# # run the model and get the predictions
library(splines)
model4 <- glm(death ~  cb4 + ns(time, 7*14) + dow, family=quasipoisson(),
  chicagoNMMAPS)
pred4 <- crosspred(cb4, model4, by=1)

# reduce to overall cumulative association
redall <- crossreduce(cb4, model4)
summary(redall)
# reduce to exposure-response association for lag 5
redlag <- crossreduce(cb4, model4, type="lag", value=5)
# reduce to lag-response association for value 33
redvar <- crossreduce(cb4, model4, type="var", value=33)

# compare number of parameters
length(coef(pred4))
```

```
length(coef(redall))
length(coef(redlag))
length(coef(redvar))

# test
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)

# reconstruct the fit in terms of uni-dimensional function
b4 <- onebasis(0:30,knots=attributes(cb4)$arglag$knots,int=TRUE)
pred4b <- crosspred(b4,coef=coef(redvar),vcov=vcov(redvar),model.link="log",by=1)

# test
plot(pred4, "slices", var=33, ylab="RR", ylim=c(0.9,1.2),
  main="Lag-response association at 33C")
lines(redvar, ci="lines", col=4, lty=2)
points(pred4b, 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)
```

`drug` *A Trial on the Effect of Time-Varying Doses of a Drug*

Description

The data set contains simulated data from an hypothetical randomized controlled trial on the effect of time-varying doses of a drug. The study include records for 200 randomized subjects, each receiving doses of a drug randomly allocated in two out of four weeks, with daily doses varying each week. The daily doses are reported on 7-day intervals corresponding to each week.

Usage

```
data(drug)
```

Format

A data frame with 200 observations on the following 7 variables.

- `id`: subject ID.
- `out`: the outcome level measured at day 28.
- `sex`: the sex of the subject.
- `day1.7`: daily dose for the first week.
- `day8.14`: daily dose for the second week.
- `day15.21`: daily dose for the third week.
- `day22.28`: daily dose for the fourth week.

Details

The exposure history for each subject (series of daily doses from day 28 to 1) can be recovered by expanding the values given in `day1.7-day22.28`.

Author(s)

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

Source

This data set only contains simulated data.

See Also

`nested` for an example of nested case-control study data. `chicagoNMMAPS` for an example of time series data.

The application of DLNMs to these data with detailed examples are provided in the vignette **dlnmExtended**.

See `dlnm-package` for an introduction to the package and for links to package vignettes providing more detailed information.

---

equalknots *Define Knots at Equally-Spaced Values*

Description

This function defines the position of knot or cut-off values at equally-spaced values for spline or strata functions, respectively.

Usage

```
equalknots(x, nk=NULL, fun="ns", df=1, degree=3, intercept=FALSE)
```

Arguments

`x`	a vector variable.
`nk`	number of knots or cut-offs.
`fun`	character scalar with the name of the function for which the knots or cut-offs must be created. See Details below.
`df`	degree of freedom.
`degree`	degree of the piecewise polynomial. Only for `fun="bs"`.
`intercept`	logical. If an intercept is included in the basis function.

Details

The number of knots is set with the argument nk, or otherwise determined by the choice of function and number of degrees of freedom through the arguments fun and df. Specifically, the number of knots is set to df-1-intercept for "ns", df-degree-intercept for "bs", or df-intercept for "strata".

Value

A numeric vector of knot or cut-off values.

Author(s)

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

See Also

logknots for placing the knots at equally-spaced log values. crossbasis to generate cross-basis matrices.

See dlnm-package for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
### setting 3 knots for range 0-20
equalknots(20, 3)

### setting knots and cut-offs for different functions
equalknots(20, fun="ns", df=4)
equalknots(20, fun="bs", df=4, degree=2)
equalknots(20, fun="strata", df=4)

### with and without without intercept
equalknots(20, fun="ns", df=4)
equalknots(20, fun="ns", df=4, intercept=TRUE)
```

---

exphist *Define Exposure Histories from an Exposure Profile*

Description

This function builds a matrix of exposure histories given an exposure profile, the time points at which each exposure history is evaluated, and a lag period.

Usage

```
exphist(exp, times, lag, fill=0)
```

Arguments

exp	an exposure profile defined at equally-spaced time units, from time 1 on.
times	either a numeric scalar or vector of integer numbers specifying the time points at which each exposure history is evaluated. By default, all the time points of exp. See Details.
lag	either an integer scalar or vector of length 2, defining the the maximum lag or the lag range, respectively. By default, the lag period from 0 to length(exp)-1.
fill	value used to fill the exposure history. See Details.

Details

This function is used to define matrices of exposure histories (backward in time) given an exposure profile (forward in time). Among other uses, this can be applied to define specific exposure histories for obtaining predictions in crosspred.

The exposure profile in exp is assumed to represent a series of exposure events defined forward in time, starting from time 1 and on. An exposure history is then evaluated backward in time for each point defined by times (rounded to integers) on the lag period defined by lag.

Negative numbers in exp represent time points before the start of the exposure profile, with 0 as the time immediately before, -1 as two times before, and so on. If the values in times are higher than the length of exp, or negative, or if the lag period extends backward before the beginning of the exposure profile, the exposure history is padded with values defined by fill.

Value

A numeric matrix of exposure histories, with named rows corresponding to the values in times and named columns corresponding to the lag period in lag.

Author(s)

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

References

Gasparrini A. Modeling exposure-lag-response associations with distributed lag non-linear models. *Statistics in Medicine*. 2014;**33**(5):881-899. [freely available here]

See Also

crosspred to obtain predictions after model fitting.

See dlnm-package for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
### an exposure history evaluated at a single time
(exp <- sample(1:10))
exphist(exp, 5, 3)
exphist(exp, 5, 12)
```

```
exphist(exp, 15, 3)

### use of argument lag
exphist(exp, 10, c(3,7))

### exposure histories evaluated at multiple times
exphist(exp, 3:5, 12)
exphist(exp, lag=12)

### fill with NA's
exphist(exp, lag=12, fill=NA)

### see the vignette dlnmExtended for further examples
```

`integer` *Generate a Basis Matrix of Indicator Variables for Integer Values*

Description

The function generates a basis matrix including indicator variables defining intervals for integer values. It is meant to be used internally by `onebasis` and `crossbasis` and not directly run by the users.

Usage

```
integer(x, values, intercept=FALSE)
```

Arguments

`x`	the predictor variable. Missing values are allowed.
`values`	the values for which the indicator variables should be computed. Used internally, usually to be left as missing.
`intercept`	logical. If `TRUE`, an intercept is included in the basis matrix. See Details below.

Details

The function returns indicator variables for intervals defined by the integer values within the range of `x`. It is expressly created to specify an unconstrained function in the space of lags for distributed lag linear (DLMs) or non-linear (DLNMs) models, and probably of no use beyond that.

The argument `intercept` determines the presence of an intercept. If `FALSE`, the interval corresponding to the first value in `values` is excluded, and the parameterization is indentical to dummy variables with the first group as a reference.

Value

A matrix object of class `"integer"`. It contains the attributes `values` and `intercept`.

Note

This function is mainly used internally thorugh `onebasis` to create basis matrices. It is not exported in the namespace to avoid conflicts with the function with the same name in the package **base**, and can be accessed through the triple colon operator '`:::`' (see Examples below).

Author(s)

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

See Also

`onebasis` to generate basis matrices and `crossbasis` to generate cross-basis matrices.

See `dlnm-package` for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
### simple use (accessing non-exported function through ':::')
dlnm:::integer(1:5)
dlnm:::integer(1:5, intercept=TRUE)
```

`lin` *Generate a Basis Matrix with a Variable as Linear*

Description

The function generates a basis matrix including a linear un-transformed variable. It is meant to be used internally by `onebasis` and `crossbasis` and not directly run by the users.

Usage

```
lin(x, intercept=FALSE)
```

Arguments

`x`	the predictor variable. Missing values are allowed.
`intercept`	logical. If `TRUE`, an intercept is included in the basis matrix, namely a vector of 1's.

Details

The function returns a basis matrix with the un-transformed variable, optionally with an intercept if `intercept=TRUE`.

Value

A matrix object of class `"lin"`. It contains the attribute `intercept`.

Note

This function is mainly used internally thorugh `onebasis` to create basis matrices. It is not exported in the namespace, and can be accessed through the triple colon operator ':::' (see Examples below).

Author(s)

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

See Also

`onebasis` to generate basis matrices and `crossbasis` to generate cross-basis matrices.

See `dlnm-package` for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
### simple use (accessing non-exported function through ':::')
dlnm:::lin(1:5)
dlnm:::lin(1:5, intercept=TRUE)

### use as an internal function in onebasis (note the centering)
b <- onebasis(chicagoNMMAPS$pm10, "lin")
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")
```

---

logknots *Define Knots for Lag Space at Equally-Spaced Log-Values*

Description

This function defines the position of knot or cut-off values at equally-spaced log-values for spline or strata functions, respectively. It is expressely created for lag-response functions to set the knots or cut-offs placements accordingly with the default of versions of **dlnm** earlier than 2.0.0.

Usage

```
logknots(x, nk=NULL, fun="ns", df=1, degree=3, intercept=TRUE)
```

Arguments

x	an integer scalar or vector of length 2, defining the the maximum lag or the lag range, respectively, or a vector variable.
nk	number of knots or cut-offs.
fun	character scalar with the name of the function for which the knots or cut-offs must be created. See Details below.

df	degree of freedom.
degree	degree of the piecewise polynomial. Only for fun="bs".
intercept	logical. If an intercept is included in the basis function.

Details

This functions has been included for consistency with versions of **dlnm** earlier than 2.0.0, where the default knots or cut-off placements in the lag space for functions ns, bs and strata used to be at equally-spaced values in the log scale. Since version 2.0.0 on, the default is equally-spaced quantiles, similarly to functions defined for the space of predictor. This function can be used to replicate the results obtained with old versions.

The argument x is usually assumed to represent the maximum lag (if a scalar) or the lag range (if a vector of length 2). Otherwise is interpreted as a vector variable for which the range is computed internally.

The number of knots is set with the argument nk, or otherwise determined by the choice of function and number of degrees of freedom through the arguments fun and df. Specifically, the number of knots is set to df-1-intercept for "ns", df-degree-intercept for "bs", or df-intercept for "strata".

An intercept is included by default (intercept=TRUE), consistently with the default for the lag space.

Value

A numeric vector of knot or cut-off values, to be used in the arglag list argument of crossbasis for reproducing the default of versions of **dlnm** earlier than 2.0.0.

Author(s)

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

See Also

equalknots for placing the knots at equally-spaced values. crossbasis to generate cross-basis matrices.

See dlnm-package for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
### setting 3 knots for lag 0-20
logknots(20, 3)
logknots(c(0,20), 3)

### setting knots and cut-offs for different functions
logknots(20, fun="ns", df=4)
logknots(20, fun="bs", df=4, degree=2)
logknots(20, fun="strata", df=4)
```

```
### with and without without intercept
logknots(20, fun="ns", df=4)
logknots(20, fun="ns", df=4, intercept=FALSE)

### replicating an old example in time series analysis
lagknots <- logknots(30, 3)
cb <- crossbasis(chicagoNMMAPS$temp, lag=30, argvar=list(fun="bs",df=5,
  degree=2), arglag=list(knots=lagknots))
summary(cb)
library(splines)
model <- glm(death ~  cb + ns(time, 7*14) + dow,
  family=quasipoisson(), chicagoNMMAPS)
pred <- crosspred(cb, model, cen=21, by=1)
plot(pred, xlab="Temperature", col="red", zlab="RR", shade=0.6,
  main="3D graph of temperature effect")
```

`nested` *Nested Case-Control Study with a Time-Varying Exposure and a Cancer Outcome*

Description

The data set 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 300 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.

Usage

```
data(nested)
```

Format

A data frame with 600 observations on the following 14 variables.

- `id`: subject ID.
- `case`: indicator for case (1) or control (0).
- `age`: age of each subject.
- `riskset`: risk set id.
- `exp15`: yearly exposure in the age period 15-19 year.
- `exp20`: yearly exposure in the age period 20-24 year.
- `...`
- `exp60`: yearly exposure in the age period 60-64 year.

Details

The exposure history for each subject (series of yearly exposures) can be recovered by expanding the values given in `exp15`-`exp60`, and then selecting the values backward from the age of the subject for a given lag period.

Author(s)

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

Source

These nested case-control data were extracted from a simulated cohort with 300 cases of cancer and a time-varying exposure.

See Also

drug for an example of randomized controlled trial data. chicagoNMMAPS for an example of time series data.

The application of DLNMs to these data with detailed examples are provided in the vignette **dlnmExtended**.

See dlnm-package for an introduction to the package and for links to package vignettes providing more detailed information.

---

onebasis *Generate a Basis Matrix for Different Functions*

Description

The function generates the basis matrix for a predictor vector. The function operates as a wrapper to existing or user-defined functions. Amongst other options, main choices include splines, polynomials, strata and linear threshold functions.

Usage

```
onebasis(x, fun="ns", ...)

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

Arguments

x	the predictor variable. Missing values are allowed.
fun	character scalar with the name of the function to be called. See Details below.
...	additional arguments to be passed to the function specified by fun or to summary.
object	a object of class "onebasis".

Details

The function onebasis is a wrapper to existing functions which are called internally to produce different types of basis matrices in a pre-defined format. Its main use in the package **dlnm** is to be called by crossbasis to generate cross-basis matrices for modelling bi-dimensional exposure-lag-response associations in distributed lag linear (DLMs) and non-linear (DLNMs) models. However, it can be used also for simplifying the modelling and plotting of uni-dimensional exposure-response relationships.

The function to be called is chosen through the argument fun. Standard choices are:

- "ns" and "bs": natural cubic B-splines or B-splines of various degree. Performed through a call to functions ns or bs from package **splines**. Arguments passed through ... may include df, knots, intercept, and Boundary.knots.
- "ps" and "cr": penalized splines with different parameterizations and penalties. Performed through a call to functions ps or cr. Arguments passed through ... may include df, knots, degree, intercept, fx, S, and diff.
- "poly": polynomials functions. Performed through a call to the internal function poly (be aware that this is different from poly in the package **stats**). Arguments passed through ... may include degree, scale and intercept.
- "strata": indicator variables defining strata. Performed through a call to the function strata. Arguments passed through ... may include df, breaks, ref and intercept.
- "thr": high, low or double linear threshold functions. Performed through a call to the function thr. Arguments passed through ... may include thr.value, side and intercept.
- "integer": indicator variables for each integer value. Performed through a call to the internal function integer (be aware that this is different from the function integer in the package **base**). Arguments passed through ... may include intercept.
- "lin": linear functions. Performed through a call to the internal function lin. Arguments passed through ... may include intercept.

The help pages of the called functions provides additional information. In particular, the option "lin" and "integer" are usually applied for defining constrained and unconstrained DLMs.

In addition, any other existing or user-defined function can be potentially called through onebasis. The function should have a first argument x defining the vector to be transformed. It also should return a vector or matrix of transformed variables, with attributes including the arguments of the function itself which define the transformations univocally.

Value

A matrix object of class "onebasis" which can be included in a model formula in order to estimate the association. It contains the attributes fun, range (range of the original vector of observations) and additional attributes specific to the chosen function. The method summary.onebasis returns a summary of the basis matrix and the related attributes.

Warnings

Meaningless combinations of arguments could lead to collinear variables, with identifiability problems in the model. The function onebasis does not perform many checks on the arguments provided. The user is expected to provide valid arguments.

Note

This function offers a wide range of options about modelling the shape of the exposure-response relationships, also simplifying or extending the use of existing functions. The function crosspred can be called on objects of class "onebasis" in order to obtain predictions and plotting of such uni-dimensional associations. If more than one variable is transformed through onebasis in the same model, different names must be specified.

Before version 2.2.0 of **dlnm**, onebasis could include a cen argument for centering the basis. This step is now moved to the prediction stage, with a cen argument in crosspred or crossreduce (see the related help pages). For backward compatibility, the use of cen in onebasis is still allowed (with a warning), but may be discontinued in the future.

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

Author(s)

Antonio Gasparrini <<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].

See Also

crossbasis to generate cross-basis matrices. crosspred to obtain predictions after model fitting. The method function plot to plot several type of graphs.

See dlnm-package for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
### a polynomial transformation of a simple vector
onebasis(1:5, "poly", degree=3)

### a low linear threshold parameterization, with and without intercept
onebasis(1:5, "thr", thr=3, side="l")
onebasis(1:5, "thr", thr=3, side="l", intercept=TRUE)

### relationship between PM10 and mortality estimated by a step function
b <- onebasis(chicagoNMMAPS$pm10, "strata", breaks=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")

### changing the reference in prediction (alternative to argument ref in strata)
pred <- crosspred(b, model, cen=30, at=0:60)
plot(pred, xlab="PM10", ylab="RR", main="RR for PM10, alternative reference")

### relationship between temperature and mortality: double threshold
```

```
b <- onebasis(chicagoNMMAPS$temp, "thr", thr=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, cen=65)
plot(pred, xlab="Height (in)", ylab="Weight (lb) difference",
  main="Association between weight and height")

### use with a user-defined function with proper attributes
mylog <- function(x, scale=min(x, na.rm=TRUE)) {
  basis <- log(x-scale+1)
  attributes(basis)$scale <- scale
  return(basis)
}
mylog(-2:5)
onebasis(-2:5,"mylog")
```

`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 linear (DLMs) and non-linear (DLNMs) models.

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". Other options are "slices" and "contour". See Details below.
var, lag	vectors (for plot) or numeric scalars (for lines-points) of predictor or lag values, respectively, at which specific associations 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, incremental cumulative associations along lags are plotted. Used only if type="slices". See Details.
exp	logical. It forces the choice about the exponentiation. 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 associations 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 associations 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 cumulative exposure-response associations over the whole lag period. See plot.default, lines and points for information on additional graphical arguments.

"slices": a (optionally multi-panel) plot of exposure-response association(s) for specific lag value(s), and/or lag-response association(s) for specific predictor value(s). 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). Their appearance may be modified through ci.arg, a list of arguments passed to 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. Incremental

cumulative associations 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` and `lag` must match those specified in the object `crosspred` (see `crosspred`).

Note

All the predictions are plotted using a reference value corresponding to the centering point for continuous functions or different values for the other functions (see the related help pages). This is determined by the argument `cen` in `crosspred`. Exponentiated predictions are returned by default if `x$model.link` is equal to `"log"` or `"logit"`.

These methods for class `"crosspred"` have replaced the old function `crossplot` since version 1.3.0.

Author(s)

Antonio Gasparrini <<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, Scheipl F, Armstrong B, Kenward MG. A penalized framework for distributed lag non-linear models. *Biometrics*. 2017;**73**(3):938-948. [freely available here]

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

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

See `dlnm-package` for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
### example of application in time series analysis - see vignette("dlnmTS")

# create the crossbasis object for pm10
cb3.pm <- crossbasis(chicagoNMMAPS$pm10, lag=1, argvar=list(fun="lin"),
  arglag=list(fun="strata"))

# create the crossbasis object for temperature
```

```
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))

# summarize
summary(cb3.pm)
summary(cb3.temp)

# run the model and get the predictions 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, cen=21, by=1)

# 3-D and contour plots
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"))

# lag-response curves specific to different temperature values
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)

# in one plot
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)))

### example of application beyond time series - see vignette("dlnmExtended")

# generate the matrix of exposure histories from the 5-year periods
Qnest <- t(apply(nested, 1, function(sub) exphist(rep(c(0,0,0,sub[5:14]),
  each=5), sub["age"], lag=c(3,40))))

# define the cross-basis
cbnest <- crossbasis(Qnest, lag=c(3,40), argvar=list("bs",degree=2,df=3),
  arglag=list(fun="ns",knots=c(10,30),intercept=FALSE))
summary(cbnest)

# run the model and predict
library(survival)
mnest <- clogit(case~cbnest+strata(riskset), nested)
pnest <- crosspred(cbnest,mnest, at=0:20*5, cen=0)

# bi-dimensional exposure-lag-response association
plot(pnest, zlab="OR", xlab="Exposure", ylab="Lag (years)")
# lag-response curve for dose 60
plot(pnest, var=50, ylab="OR for exposure 50", xlab="Lag (years)", xlim=c(0,40))
# exposure-response curve for lag 10
plot(pnest, lag=5, ylab="OR at lag 5", xlab="Exposure", ylim=c(0.95,1.15))
```

```
### example of extended predictions - see vignette("dlnmExtended")

# compute exposure profiles and exposure history
expnested <- rep(c(10,0,13), c(5,5,10))
hist <- exphist(expnested, time=length(expnested), lag=c(3,40))

# predict association with a specific exposure history
pnesthist <- crosspred(cbnest, mnest, cen=0, at=hist)
with(pnesthist, c(allRRfit,allRRlow,allRRhigh))

### example of user-defined functions - see vignette("dlnmExtended")

# define a log function
mylog <- function(x) log(x+1)

# define the cross-basis
cbnest2 <- crossbasis(Qnest, lag=c(3,40), argvar=list("mylog"),
  arglag=list(fun="ns",knots=c(10,30),intercept=FALSE))
summary(cbnest2)

# run the model and predict
mnest2 <- clogit(case~cbnest2+strata(riskset), nested)
pnest2 <- crosspred(cbnest2, mnest2, cen=0, at=0:20*5)

# plot and compare with previous fit
plot(pnest2, zlab="OR", xlab="Exposure", ylab="Lag (years)")
plot(pnest2, var=50, ylab="OR for exposure 50", xlab="Lag (years)", xlim=c(0,40))
lines(pnest, var=50, lty=2)
plot(pnest2, lag=5, ylab="OR at lag 5", xlab="Exposure", ylim=c(0.95,1.15))
lines(pnest, lag=5, lty=2)

### example of general use for regression models - see vignette("dlnmExtended")

# replicate example illustrated in help(ns)
library(splines)
oneheight <- onebasis(women$height, "ns", df=5)
mwomen <- lm(weight ~ oneheight, data=women)
pwomen <- crosspred(oneheight, mwomen, cen=65, at=58:72)
with(pwomen, cbind(allfit, alllow, allhigh)["70",])
plot(pwomen, ci="l", ylab="Weight (lb) difference", xlab="Height (in)", col=4)

# replicate example illustrated in help(gam)
library(mgcv)
dat <- gamSim(1,n=200,dist="poisson",scale=.1)
b2 <- gam(y ~ s(x0,bs="cr") + s(x1,bs="cr") + s(x2,bs="cr") + s(x3,bs="cr"),
  family=poisson, data=dat, method="REML")
plot(b2, select=3)
pgam <- crosspred("x2", b2, cen=0, at=0:100/100)
with(pgam, cbind(allRRfit, allRRlow, allRRhigh)["0.7",])
plot(pgam, ylim=c(0,3), ylab="RR", xlab="x2", col=2)

### example of penalized models - see vignette("dlnmPenalized")
```

```
# to be added soon
```

`plot.crossreduce` *Plot Predictions for a Reduced DLNM*

Description

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

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

`x`	an object of class `"crossreduce"`.
`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.
`exp`	logical. It forces the choice about the exponentiation. See Details.
`...`	optional graphical arguments. See Details.

Details

Differently than for plotting functions for `crosspred` objects (see the method function `plot` for objects of class `"crosspred"`), the type of the plot is automatically chosen by the dimension and value at which the model has been reduced. Namely, the lag-specific association at the chosen lag value, the predictor-specific association at the chosen predictor value, or the overall cumulative association.

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). Their appearance may be modified through ci.arg, a list of arguments passed to 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 different values for the other functions (see the related help pages). This is determined by the argument cen in crossreduce. Exponentiated predictions are returned by default if x$model.link is equal to "log" or "logit".

Author(s)

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

References

Gasparrini A., Armstrong, B., Kenward M. G. Reducing and meta-analyzing estimates from distributed lag non-linear models.*BMC Medical Research Methodology*. 2013;**13**(1):1. [freely available here].

See Also

onebasis to generate simple basis matrices. crosspred to obtain predictions after model fitting. crossreduce to reduce the fit ot one dimension.

See dlnm-package for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
# create the crossbasis object
lagnk <- 3
lagknots <- exp(((1+log(30))/(lagnk+1) * seq(lagnk))-1)
cb4 <- crossbasis(chicagoNMMAPS$temp, lag=30, argvar=list(fun="thr",
  thr=c(10,25)), arglag=list(knots=lagknots))

# # run the model and get the predictions
library(splines)
model4 <- glm(death ~  cb4 + ns(time, 7*14) + dow, family=quasipoisson(),
  chicagoNMMAPS)
pred4 <- crosspred(cb4, model4, by=1)

# reduce to overall cumulative association
redall <- crossreduce(cb4, model4)
```

```
summary(redall)
# reduce to exposure-response association for lag 5
redlag <- crossreduce(cb4, model4, type="lag", value=5)
# reduce to lag-response association for value 33
redvar <- crossreduce(cb4, model4, type="var", value=33)

# compare number of parameters
length(coef(pred4))
length(coef(redall))
length(coef(redlag))
length(coef(redvar))

# test
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)

# reconstruct the fit in terms of uni-dimensional function
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)

# test
plot(pred4, "slices", var=33, ylab="RR", ylim=c(0.9,1.2),
  main="Lag-response association at 33C")
lines(redvar, ci="lines", col=4, lty=2)
points(pred4b, 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)
```

---

poly *Generate a Basis Matrix of Polynomials*

Description

The function generates a basis matrix of polynomial transformations. It is meant to be used internally by onebasis and crossbasis and not directly run by the users.

Usage

```
poly(x, degree=1, scale, intercept=FALSE)
```

Arguments

x	the predictor variable. Missing values are allowed.
degree	numerical scalar defining the degree of the polynomial.
scale	scaling factor. Default to the maximum of the absolute value of x.
intercept	logical. If TRUE, an intercept is included in the basis matrix. See Details below.

Details

The predictor vector is scaled by default through the argument `scale` to avoid numerical problem with powers of very high/low values.

If `intercept=TRUE`, an intercept is included in the model, namely an additional variable with a constant value of 1.

Value

A matrix object of class `"poly"`. It contains the attributes `degree`, `scale` and `intercept`, with values which can be different than the arguments provided due to internal reset.

Note

This function is mainly used internally thorugh `onebasis` and `crossbasis` to create basis and cross-basis matrices, respectively. It is not exported in the namespace to avoid conflicts with the function with the same name in the package **stats**, and can be accessed through the triple colon operator ':::' (see Examples below).

In particular, the function `poly` from the package **stats** cannot be used directly, as it does not store as attributes all the parameters need to univocally define the transformation.

Author(s)

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

See Also

`onebasis` to generate basis matrices and `crossbasis` to generate cross-basis matrices.

See `dlnm-package` for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
### simple use (accessing non-exported function through ':::')
dlnm:::poly(1:5, degree=3)
dlnm:::poly(1:5, degree=3, intercept=TRUE)

### use as an internal function in onebasis
b <- onebasis(chicagoNMMAPS$pm10, "poly", degree=3)
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")
```

ps *Generate a Basis Matrix for P-Splines*

Description

Generate the basis matrix for P-splines, namely a B-spline basis with difference penalties.

Usage

```
ps(x, df=10, knots=NULL, degree=3, intercept=FALSE, fx= FALSE, S=NULL, diff=2)
```

Arguments

x	the predictor variable. Missing values are allowed.
df	degrees of freedom, basically the dimension of the basis matrix. If supplied in the absence of knots, it automatically selects df+degree+2-intercept equally-spaced knots (within and beyond the range of x). The minimum df allowed is degree+1-intercept.
knots	breakpoints that define the spline. These are generally automatically selected, and not defined by the user. See Details below.
degree	degree of the piecewise polynomial. Default is 3 for cubic splines.
intercept	logical. If TRUE, an intercept is included in the basis matrix. See Details below.
fx	logical. If TRUE, it removes the penalization. See Details below.
S	penalty matrix, usually internally defined if NULL (default).
diff	order difference of the penalty.

Details

The function has a usage similar to bs and ns in the **splines** package. It produces B-spline transformations through a call to splineDesign, plus a difference matrix to define penalties. The same results are returned by the related smooth constructor in the package **mgcv**.

The argument knots defines a vector of knots or (if of length 2) the lower and upper limits between which the splines can be evaluated. However, knots should be usually left automatically selected, and in particular these P-splines only have sense with equally-spaced knots, due to the nature of the penalization. It is important to highlight that, differently from bs where *internal* and *boundary* knots are defined, this function adopts a standard B-spline parameterization, including by default 2*(degree+1) knots beyond the range of the variable.

The penalization is defined on the difference of adjacent coefficients during fitting procedure through a penalty matrix S. The argument diff selects the order difference (with the default 2 determining a second order difference, and 0 producing a ridge penalty), while setting fx=TRUE removes the penalization.

Similarly to bs and ns, setting intercept=FALSE (default) determines the exclusion of the first transformed variables, and the corresponding first row and column in S, thus avoiding identifiability issues during the model fitting. Note how the procedure of imposing identifiability constraints is different from that adopted by smoothCon in the package **mgcv**, where a more complex reparameterization is produced.

Value

A matrix object of class `"ps"`. It contains the attributes `df`, `knots`, `degree`, `intercept`, `fx`, `S`, and `diff`, with values that can be different than the arguments provided due to internal reset.

Note

The function is primarily added here to specify penalized DLMs and DLNMs using the so-called *external* method, *i.e.* by including the penalty matrix in the argument `paraPen` of the `gam` regression function in **mgcv** (see `cbPen`). However, this approach can be also used to fit standard uni-dimensional P-spline models as an alternative to the use of specific `smooth constructor`, as it takes advantage of the use of prediction and plotting functions in **dlnm**.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>>, adapting code available from functions included in the package **mgcv** by Simon N. Wood.

References

Gasparrini A, Scheipl F, Armstrong B, Kenward MG. A penalized framework for distributed lag non-linear models. *Biometrics*. 2017;**73**(3):938-948. [freely available here]

Eilers P. H. C. and Marx B. D. Flexible smoothing with B-splines and penalties. *Statistical Science*. 1996;**11**(2):89-121.

Wood S. N. Generalized Additive Models: An Introduction with R. Chapman and Hall/CRC Press, 2006.

See Also

`cr` for penalized cubic regression splines. `bs` and `ns` for B-splines and natural cubic splines, respectively. `cbPen` for defining tensor-type bi-dimensional penalties in DLNMs. The related `smooth constructor` for P-spline smooths in **mgcv**. The `cb smooth constructor` for cross-basis penalized spline smooths.

See `dlnm-package` for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
# to be added soon
```

`smooth.construct.cb.smooth.spec`
Cross-Basis Spline Smooths for a DLNM

Description

These are method functions for a smooth class defining bi-dimensional cross-basis splines for penalized distributed lag linear (DLMs) and non-linear (DLNMs) models. The functions are not supposed to be called directly, and the class is usually specified via terms like `s(X,L,bs="cb",...)` in the formula of the `gam` function of the package **mgcv**.

Usage

```
## S3 method for class 'cb.smooth.spec'
smooth.construct(object, data, knots)

## S3 method for class 'cb.smooth'
Predict.matrix(object, data)
```

Arguments

`object`	for `smooth.construct`, an object of class `"cb.smooth.spec"` usually generated by a call `s(X,L,bs="cb",...)`. For `Predict.matrix`, an object of class `cb.smooth` generated by `smooth.construct`.
`data`	a list containing just the data (including any by variable) required by this term, with names corresponding to `object$term`.
`knots`	a list containing any knots supplied for basis setup — in same order and with same names as data. It is usually `NULL`.

Details

These method functions embed tools available in the packages **dlnm** and **mgcv** to perform penalized DLMs and DLNMs. This represent the *internal* approach to perform such models (see Notes below). Specifically, the models are fitted by including a term `s(X,L,bs="cb",...)`, defining a basis `"cb"` for bi-dimensional cross-basis splines, in the formula of the `gam` function. The constructor function for this class turns this smooth terms into a smooth specification object, which includes the cross-basis matrix (see `crossbasis`) and the penalty matrices for the two spaces of predictor and lags used in model fitting. Then, `crosspred` uses the predict matrix function for the `"cb"` basis to obtain predictions, and a graphical representation can be obtained by standard `plotting functions`, similary to unpenalized models.

The first two arguments `X` and `L` in `s` represent a matrix of exposure histories and a matrix of lags. The former, also used in `crossbasis`, needs to be defined directly even with time series data by lagging the exposure series. The matrix `L` must have the same dimensions of `X`, with identical rows representing the sequence of lags. The other arguments of `s` have the same meaning: in particular, `k` (default to 10), `fx` (default to `FALSE`) and `sp` (default to `NULL`) can be provided for each marginal

basis as vectors of length 2, and similarly `m` can be provided as a list (see also `te`). No `by` argument is allowed.

Extra information can be included in the argument `xt` of `s`, which accepts a single object or a list of objects. First, an object `bs` (a vector of length 1 or 2) can be used to specify the smoother for each marginal dimension, with current options restricted to `"ps"` (`P-splines`, used by default) and/or `"cr"` (`cubic regression splines`). Second, list objects `argvar` and `arglag` can be used to build the marginal bases for predictor and lags by calling other functions (see the same arguments in `crossbasis`). In particular, these can be used for a more flexible specification of penalized functions (using `ps` or `cr`) or for using unpenalized functions for one marginal basis, thus limiting the penalization to one of the two dimensions. Third, the object `addSlag` can contain a matrix or vector (or list of matrices and/or vectors) defining additional penalties on the lag structure (see `cbPen`).

Value

The smooth constructor function returns an object of classes `"cb.smooth"` and `"tensor.smooth"`. Specifically, a list with a similar structure of that returned by the smooth constructor for `tensor product smooths` (see also `te`).

The `Predict.matrix` function return a cross-basis matrix evaluated at specific values used for prediction.

Note

Identifiability constraints are applied to marginal basis for predictor (see `smoothCon`), while the marginal basis for the lag dimension is left untransformed. This involves a re-parameterization with the absorption of constraints into the basis that causes its dimension to decrease by 1. Note that this procedure is similar to that in `crossbasis`, while it is different than in standard tensor product smooths (see `te`), where identifiability constraints are not applied to the marginal bases.

Using the default specification with `k=10` in the smooth terms defined by `s`, the dimension of the cross-basis matrix will be $(10 - 1) \times 10 = 90$ (accounting for identifiability constraints). This is consistent with the rationale that this choice is not important as far as the upper limit for the degrees of freedom in each marginal basis is large enough to represent the underlying relationship (see `choose.k`). Smaller values of `k` can be used for speeding up the computation, as long as the underlying relationship can be assumed to be smooth enough.

These method functions provide an *internal* method for performing penalized DLMs and DLNMs, with the cross-basis spline smoother defined directly in the model formula of `gam` through a smooth term specified by `s`. The alternative *external* method relies on the standard use of `crossbasis` and on the penalization of so-called parametric terms through the argument `paraPen` of `gam` (see `cbPen` for details). The two methods are expected to returns almost identical results in most cases. However, while the internal method takes advantage of the full machinery of **mgcv** and plausibly more stable procedures, the external method allows more flexibility and the optional use of user-defined smoothers.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Fabian Scheipl <<fabian.scheipl@stat.uni-muenche

References

Gasparrini A, Scheipl F, Armstrong B, Kenward MG. A penalized framework for distributed lag non-linear models. *Biometrics*. 2017;**73**(3):938-948. [freely available here]

Wood S. N. Generalized Additive Models: An Introduction with R. Chapman and Hall/CRC Press, 2006.

See Also

Smooth constructors for `P-splines` and `cubic regression splines` in **mgcv**. `ps` and `cr` for the same functions available in **dlnm**. `cbPen` for defining tensor-type bi-dimensional penalties in DLNMs.

See `dlnm-package` for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
# to be added soon
```

`strata` *Generate a Basis Matrix of Indicator Variables*

Description

The function generates a basis matrix including indicator variables defining intervals (strata), through dummy parameterization. It is meant to be used internally by `onebasis` and `crossbasis` and not directly run by the users.

Usage

```
strata(x, df=1, breaks=NULL, ref=1, intercept=FALSE)
```

Arguments

`x`	the predictor variable. Missing values are allowed.
`df`	dimension of the basis, equal to the number of strata. They depend on `breaks` if provided.
`breaks`	internal cut-off points defining the strata as right-open intervals. If provided, they determine `df`.
`ref`	interval used as reference category. Default to the first stratum. See Details below.
`intercept`	logical. If `TRUE`, an intercept is included in the basis matrix. See Details below.

Details

The strata are defined by right-open intervals specified through `breaks`. If these are not provided, a number of intervals defined by `df` is placed at equally-spaced quantiles. This step is performed through an internal call to `cut`.

The argument `ref` indentifies the reference category, specified by excluding the related stratum in the dummy parameterization of the basis. This defines control-treatment contrasts, where each interval is compared with the baseline (see `contrast`). If set to 0 (when `intercept=TRUE`), it provides a different parameterization, where each interval has its own baseline.

If `intercept=TRUE`, an intercept is included in the model. The default (when `ref` is different from 0) produces an additional variable with a constant value of 1, representing the baseline.

Value

A matrix object of class `"strata"`. It contains the attributes `df`, `breaks`, `ref` and `intercept`, with values which can be different than the arguments provided due to internal reset.

Note

This function is mainly used internally thorugh `onebasis` and `crossbasis` to create basis and cross-basis matrices, respectively. It is not exported in the namespace to avoid conflicts with the function with the same name in the package **survival**, and can be accessed through the triple colon operator ':::' (see Examples below).

Author(s)

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

See Also

`onebasis` to generate basis matrices and `crossbasis` to generate cross-basis matrices.

See `dlnm-package` for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
### simple use (accessing non-exported function through ':::')
dlnm:::strata(1:5, breaks=3)
dlnm:::strata(1:5, df=3)
dlnm:::strata(1:5, df=3, intercept=TRUE)
dlnm:::strata(1:5, df=3, ref=2, intercept=TRUE)

### use as an internal function in onebasis
b <- onebasis(chicagoNMMAPS$pm10, "strata", breaks=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")
```

thr *Generate a Basis Matrix of Linear Threshold Transformations*

Description

The function generates a basis matrix including transformed variables through high, low or double linear threshold parameterization. It is meant to be used internally by `onebasis` and `crossbasis` and not directly run by the users.

Usage

```
thr(x, thr.value=NULL, side=NULL, intercept=FALSE)
```

Arguments

`x`	the predictor variable. Missing values are allowed.
`thr.value`	numeric scalar or vector defining the threshold value(s).
`side`	type of threshold parameterization: `"l"` for low, `"h"` for high, `"d"` for double. See Details below.
`intercept`	logical. If `TRUE`, an intercept is included in the basis matrix. See Details below.

Details

A linear threshold function defines a linear relationship beyond a specific threshold. A high linear threshold defines a linear increase above the threshold, while a low linear threshold defines a linear increase below. A double linear threshold includes both of them.

The argument `thr.value` is placed at the median if not provided. If `side` is not provided, the default is `side="h"` when `thr.value` is a scalar, `side="d"` otherwise. Only the minimum (for `side="h"` and `side="l"`) and minimum and maximum values (for `side="d"`) of `thr.value` are considered.

If `intercept=TRUE`, an intercept is included in the model, namely an additional variable with a constant value of 1.

Value

A matrix object of class `"thr"`. It contains the attributes `thr.value`, `side` and `intercept`, with values which can be different than the arguments provided due to internal reset.

Note

This function is mainly used internally thorugh `onebasis` and `crossbasis` to create basis and cross-basis matrices, respectively. It is not exported in the namespace, and can be accessed through the triple colon operator ':::' (see Examples below).

Author(s)

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

See Also

onebasis to generate basis matrices and crossbasis to generate cross-basis matrices.

See dlnm-package for an introduction to the package and for links to package vignettes providing more detailed information.

Examples

```
### simple use (accessing non-exported function through ':::')
dlnm:::thr(1:5, thr=3)
dlnm:::thr(1:5, side="d")
dlnm:::thr(1:5, side="d", intercept=TRUE)

### use as an internal function in onebasis
b <- onebasis(chicagoNMMAPS$pm10, "thr", thr.value=20)
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")
```

Index