Package 'dlnm'

January 27, 2015

Type Package

Title Distributed Lag Non-linear Models

Version 2.1.3

Date 2014-08-05

Author Antonio Gasparrini and Ben Armstrong

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

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

Depends R (>= 2.10)

Suggests mgcv, survival, lme4, gee, geepack, mvmeta

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

URL http://www.ag-myresearch.com/package-dlnm

License GPL (>= 2)

LazyData yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-08-05 19:13:21

R topics documented:

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

Description

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

Modelling framework

Distributed lag non-linear models (DLNM) represent a modelling framework to describe simultaneously non-linear and delayed dependencies, termed as *exposure-lag-response associations*. The methodology of 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 studies, 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 DLNM family includes simple distributed lag models (DLM) as a special case.

The application of DLNMs requires the availability of predictor values 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

Given a time series vector or a matrix of exposure histories, `crossbasis` creates two set of basis functions 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.

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 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) DLNM.

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 DLNM methodology and the functions included in the package. The vignette **dlnmTS** includes 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.

A vignette is available by typing:

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

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

```
file.show(system.file("ChangeLog",package="dlnm"))
```

General information on the development and applications of the DLNM modelling framework, together with an updated version of the R scripts for running the examples in published papers, can be found at **www.ag-myresearch.com**.

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

Author(s)

Antonio Gasparrini and Ben Armstrong

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. 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]

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].

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. `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.

`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:

`http://www.ihapss.jhsph.edu`

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.

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

Description

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

Usage

```
crossbasis(x, lag, 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 distributed lag linear and non-linear models in different study designs. 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, optionally the argument cen in argvar, 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. 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. See help(onebasis) and the help pages of these functions provide information on the additional arguments to be specified. Also, other existing or user-defined functions can be applied.

Results from DLNM are interpreted relatively to a reference value of the predictor, determined automatically or through a centering point. See onebasis for further details. By default, the basis functions for lags are defined with an intercept (if otherwise stated) and never centered. Some arguments can be automatically re-set by onebasis. Use summary.crossbasis to check the result.

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 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 the arguments above). The function summary.crossbasis returns a summary of the cross-basis matrix and the related attributes, and can be used to check the options for the basis functions chosen for the two dimensions.

Warnings

In previous versions of the package the function adopted a different usage. Users are invited to comply with the current usage.

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

Note

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 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.

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. 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. 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$pm10, lag=15, argvar=list(fun="lin",cen=0),
  arglag=list(fun="poly",degree=4))
cb1.temp <- crossbasis(chicagoNMMAPS$temp, lag=3, argvar=list(df=5,cen=21),
  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 of specific effects")
plot(pred1.pm, "slices", var=10, 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,cen=0), arglag=list(fun="ns",knots=c(10,30),int=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)

# 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))
```

---

crosspred *Generate Predictions for a DLNM*

Description

The function generates predictions from a distributed lag non-linear model (DLNM). These are interpreted as estimated associations defined on a grid of values of the original predictor and lags, computed versus a reference predictor value.

Usage

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

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

Arguments

basis	an object of class "onebasis" or "crossbasis".
model	a model object for which the prediction is desired. See Details below.
coef, vcov, model.link	user-provided coefficients, (co)variance matrix and model link for the prediction. See Details below.
at	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.
ci.level	confidence level for the computation of confidence intervals.
cumul	logical. If TRUE, incremental cumulative associations along 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. The object basis must be the same containing the basis or cross-basis matrix included in model, preserving its attributes and class. The set of predictor values at which predictions must be computed can be specified by at or alternatively by from/to/by. The argument lag defines the lag values used for prediction, while bylag determines the lag step. The values in at can be provided as a vector, and in this case represent constant exposures experienced within the lag period. If at and by are not provided, approximately 50 equally-spaced rounded values are selected using pretty.

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.

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. In this case, their dimensions and order must match the variables included in basis.

The function can be used to compute predictions for models with simple basis functions not including lag, computed with onebasis. In this case, only unlagged predicted associations are returned. 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, but needs to be provided for different classes. Matrices with incremental cumulative predicted associations along lags at each values used for prediction are included if cumul=TRUE.

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).
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 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 corresponding to the centering point for continuous functions or to the default values for the other options (see `onebasis` and `crossbasis`). Exponentiated predictions are included if `model.link` (specified automatically by `model` or selected by the user) 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. 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

```
### default usage - 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$o3, lag=5, argvar=list(fun="thr",
  side="h",thr=40.3), arglag=list(fun="integer"), group=chicagoNMMAPSseas$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", 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="lines", 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)

### extended usage - 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 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",cen=0),
  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)

# dose 20 for 10 days
histdrug <- exphist(rep(20,10), time=10, lag=27)
pdrug4 <- crosspred(cbdrug2, mdrug2, 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(cbdrug2, mdrug2, 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))
```

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

Description

The function reduces the fit of a bi-dimensional DLNM to summaries defined in the the dimension of predictor or lags only, and re-expresses it in terms of modified parameters of the one-dimensional basis functions chosen for that space.

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,
  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 association, "lag" for reduction to a lag-specific association, or "var" for reduction to a predictor-specific 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.
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 optional arguments at and from/to/by provides the predictor values for predicted associations when the reduction is in the dimension of predictor. lag and bylag determine instead the lag values for predictor-specific associations. For specific associations, the value at which the reduction is computed is chosen by value. Exponentiated predictions and confidence intervals are also optionally returned. See crosspred for details.

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. 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.
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 corresponding to the centering point for continuous functions or to the default values for the other options (see onebasis and crossbasis). Exponentiated predictions are included if model.link (specified automatically by object or selected by the user) 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,cen=FALSE)
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, col=1, pch=19, cex=0.6)
legend("top",c("Original","Reduced","Reconstructed"),col=c(2,4,1),lty=c(1:2,NA),
  pch=c(NA,NA,19),pt.cex=0.6,ins=0.1)
```

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 given in 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, int=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"`.
`int`	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-int` for `"ns"`, `df-degree-int` for `"bs"`, or `df-int` 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, int=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, time, lag)
```

Arguments

`exp`	an exposure profile defined at equally-spaced time units, from time 1 on.
`time`	either a numeric scalar or vector of positive integer numbers specifying the time points at which each exposure history is evaluated. By default, all the time points of `exp`.
`lag`	either an integer scalar or vector of length 2, defining the the maximum lag or the lag range, respectively. Only non-negative lags allowed. By default, the lag period from 0 to `length(exp)-1`.

Details

This function is used to define matrices of exposure histories to be used in `crosspred` for obtaining predictions.

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 `time` (rounded to integers) on the lag period defined by `lag`.

If the values in `time` are higher than the length of `exp`, or if the lag period extends backward before the beginning of the exposure profile, the exposure history is padded with 0's.

Value

A numeric matrix of exposure histories, with rows corresponding to the values in `time` and 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, 15, c(3,7))

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

### see help(drug) and help(nested) for further examples
```

---

integer *Generate a Basis Matrix of with a 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, int=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.
int	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 or non-linear models, and probably of no use beyond that.

The argument int 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 int.

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, int=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, int=FALSE)
```

Arguments

`x`	the predictor variable. Missing values are allowed.
`int`	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 `int=TRUE`.

Value

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

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, int=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, int=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".
int	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-int for "ns", df-degree-int for "bs", or df-int for "strata".

An intercept is included by default (int=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, int=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,cen=21), arglag=list(knots=lagknots))
summary(cb)
library(splines)
model <- glm(death ~  cb + ns(time, 7*14) + dow,
  family=quasipoisson(), chicagoNMMAPS)
pred <- crosspred(cb, model, 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 given in 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, optionally centering on a specific predictor value. The function operates as a wrapper to existing or user-defined functions. Amongst other options, default choices include splines, polynomials, strata and linear threshold functions.

Usage

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

## 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.
cen	logical or a numeric scalar. It specifies the centering value, then used as a reference for predictions. Setting cen=FALSE is never recommended. See Note 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 is to be called by crossbasis to generate cross-basis matrices for modelling bi-dimensional exposure-lag-response associations thorugh distributed lag non-linear models. However, it can be used also for simplifying the modelling of uni-dimensional exposure-response relationship.

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.

- "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 and int.
- "strata": indicator variables defining strata. Performed through a call to the function strata. Arguments passed through ... may include breaks, df, ref and int.
- "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 int.
- "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 int.
- "lin": linear functions. Performed through a call to the internal function lin. Arguments passed through ... may include int.

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 distributed lag models.

In addition, any other existing or user-defined function can be potentially called through onebasis. The function should have the 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 exactly define the transformations.

Results from models including basis functions are interpreted here relatively to a reference value of the predictor, determined automatically or through a centering point (see also Note below). In the latter case, the centering value is chosen by cen (if a numeric scalar), or fixed at the mean if cen=TRUE. The basis is uncentered for cen=FALSE. Use summary to check the result.

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, cen, range (range of the original vector of observations) and additional attributes specific to the chosen function. The function 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 extending the use of existing functions, which can be centered here. The function crosspred can be called on objects of class "onebasis" in order to obtain predictions and plotting of such unidimensional associations. If more than one variable is transformed through onebasis in the same model, different names must be specified.

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

Centering is always recommended for continuous functions such as splines or polynomials. Centering is prevented for functions `strata`, `thr` and `integer`. The inclusion of the intercept term nullifies the centering.

Author(s)

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

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, centered and uncentered
onebasis(1:5, "poly", degree=3)
onebasis(1:5, "poly", degree=3, cen=FALSE)

### 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", int=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")

### 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)
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)+1) {
  basis <- log(x+scale)
  attributes(basis)$scale <- scale
  return(basis)
```

```
}
mylog(-2:5)
onebasis(0:5,"mylog",cen=FALSE)
onebasis(0:5,"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 non-linear models (DLNM).

Usage

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

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

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

Arguments

`x`	an object of class `"crosspred"`.
`ptype`	type of plot. Default to `"3d"` for lagged relationship, otherwise `"overall"`. See Details below.
`var, lag`	vectors (for `plot`) or numeric scalars (for `lines-points`) of predictor or lag values 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 associations in the whole lag period. See plot.default, lines and points for information on additional graphical arguments.

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

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

Confidence intervals are plotted for ptype equal to "overall" or "slices". Their type is determined by ci, with options "area" (default for plot), "bars", "lines" or "n" (no confidence intervals, default for points and lines). The appearance may be modified through ci.arg, a list of arguments passed to 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 to the default values for the other options (see onebasis and crossbasis). 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. 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",cen=0),
  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,cen=21), 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, by=1)

# 3-D and contour plots
plot(pred3.temp, xlab="Temperature", col="red", zlab="RR", shade=0.6,
  main="3D graph of temperature effect")
plot(pred3.temp, "contour", xlab="Temperature", key.title=title("RR"),
  plot.title=title("Contour plot",xlab="Temperature",ylab="Lag"))

# 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 associations at 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,cen=0), arglag=list(fun="ns",knots=c(10,30),int=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)

# 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))
```

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

Description

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

Usage

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

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

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

Arguments

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). The 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 to the default values for the other options (see onebasis and crossbasis). 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,int=TRUE,cen=FALSE)
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, col=1, pch=19, cex=0.6)
legend("top",c("Original","Reduced","Reconstructed"),col=c(2,4,1),lty=c(1:2,NA),
  pch=c(NA,NA,19),pt.cex=0.6,ins=0.1)
```

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, int=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.
int	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 int=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 int, 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).

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, int=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")
```

---

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, int=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.
`int`	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 equal to `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.

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

Value

A matrix object of class "strata". It contains the attributes df, breaks, ref and int, 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, int=TRUE)
dlnm:::strata(1:5, df=3, ref=2, int=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, int=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.
int	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 int=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 int, 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", int=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