

# Package ‘MortalitySmooth’

February 14, 2012

**Type** Package

**Title** Smoothing and forecasting Poisson counts with P-splines

**Version** 2.2

**Date** 2012-02-05

**Author** Carlo G Camarda

**Maintainer** Carlo G Camarda <camarda@demogr.mpg.de>

**Depends** R (>= 2.10), svcm

**Description** Smoothing one- and two-dimensional Poisson counts with P-splines specifically tailored to mortality data. Extra-Poisson variation can be accounted as well as forecasting. Collection of mortality data and a specific function for selecting those data by country, sex, age and years.

**License** GPL (>= 2)

**LazyLoad** yes

**LazyData** yes

**Repository** CRAN

**Date/Publication** 2012-02-06 15:07:14

## R topics documented:

MortalitySmooth-package	2
HMDdata	3
Mort1Dsmooth	4
Mort1Dsmooth_checker	9
Mort1Dsmooth_estimate	10
Mort1Dsmooth_optimize	11
Mort1Dsmooth_update	13

Mort2Dsmooth	14
Mort2Dsmooth_checker	19
Mort2Dsmooth_estimate	20
Mort2Dsmooth_optimize	22
Mort2Dsmooth_se	23
Mort2Dsmooth_update	24
MortSmooth_bbase	25
MortSmooth_BcoefB	26
MortSmooth_BWB	27
MortSmooth_tpower	28
plot.HMDdata	29
plot.Mort1Dsmooth	30
plot.Mort2Dsmooth	32
predict.Mort1Dsmooth	33
predict.Mort2Dsmooth	36
residuals.Mort1Dsmooth	39
residuals.Mort2Dsmooth	40
selectHMDdata	42
summary.Mort1Dsmooth	43
summary.Mort2Dsmooth	44

**Index** **46**

MortalitySmooth-package

*Smoothing Poisson counts with P-splines*

### Description

Smoothing one- and two-dimensional Poisson counts with P-splines specifically tailored to mortality data. Extra-Poisson variation can be accounted as well as mortality forecasting. Collection of mortality data and a specific function for selecting those data by country, sex, age and years.

### Details

Package:	MortalitySmooth
Type:	Package
Version:	2.2
Date:	2012-02-05
License:	What license is it under?
LazyLoad:	yes

**Author(s)**

Carlo G Camarda

Maintainer: Carlo G Camarda <camarda@demogr.mpg.de>

**References**

Eilers and Marx (1996). Flexible Smoothing with B-splines and Penalties. *Statistical Science*. Vol. 11. 89-121.

Currie, Durban and Eilers (2004). Smoothing and forecasting mortality rates. *Statistical Modelling*. Vol. 4. 279-298.

Currie, Durban and Eilers (2006). Generalized linear array models with applications to multidimensional smoothing. *JRSS*. Vol. 68. 259-280.

---

HMDdata

*Population and Mortality Data*

---

**Description**

Age-specific population, deaths, exposures and rates from the Human Mortality Database.

**Usage**

```
data(HMDdata)
```

**Format**

Object of class HMDdata containing, for each country, the following components:

**country** country name.

**year** vector of years.

**age** vector of ages.

**pop** list of matrices containing population with one age group per row and one column per year.  
Matrices: female, male, total.

**death** list of matrices containing deaths in same form as pop.

**exposure** list of matrices containing exposure in same form as pop.

**rate** list of matrices containing mortality rate in same form as pop.

**Details**

The list contained 4 countries (Denmark, Japan, Sweden and Switzerland). Data taken from the Human Mortality Database on 30 August 2011. Data are accessible either manually or using [selectHMDdata](#).

For a given country, the matrices have the same dimensions. The vector age is the same for each country (0:110), whereas years depends on the availability.

pop indicates population size on January 1st of each age and year.

death are death counts occurred during one age-year interval.

exposure population is approximated by the average of the population size in the beginning and at the end of the year.

rate are computed as ratio between death counts and exposures.

**Author(s)**

Carlo G Camarda

**Source**

Human Mortality Database [www.mortality.org](http://www.mortality.org)

**References**

Human Mortality Database (2011). University of California, Berkeley (USA), and Max Planck Institute for Demographic Research (Germany). Available at [www.mortality.org](http://www.mortality.org).

**See Also**

[selectHMDdata](#).

**Examples**

```
data(HMDdata)
plot(HMDdata$Japan$year, HMDdata$Japan$pop$female[81,],
     main="Japan population, female, age 80")
## list of available countries
names(HMDdata)
```

---

Mort1Dsmooth

*Fit One-dimensional Poisson P-splines*

---

**Description**

Returns an object of class Mort1Dsmooth which is a P-splines smooth of the input data of degree and order fixed by the user. Specifically tailored to mortality data.

**Usage**

```
Mort1Dsmooth(x, y, offset, w,
             overdispersion=FALSE,
             ndx = floor(length(x)/5), deg = 3, pord = 2,
             lambda = NULL, df = NULL, method = 1,
             coefstart = NULL,
             control = list())
```

**Arguments**

x	a vector with the values of the predictor variable. These must be at least 2 <code>ndx</code> + 1 of them.
y	a vector with a set of counts response variable values. <code>y</code> must be a vector of the same length as <code>x</code> .
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length either one or equal to the number of cases.
w	an optional vector of weights to be used in the fitting process. This should be NULL or a numeric vector of length equal to the number of cases.
overdispersion	logical on the accounting for overdispersion in the smoothing parameter selection criterion. See Details. Default: FALSE.
ndx	number of internal knots -1. Default: <code>floor(length(x)/5)</code> .
deg	degree of the B-splines. Default: 3.
pord	order of differences. Default: 2.
lambda	smoothing parameter (optional).
df	a number which specifies the degrees of freedom (optional).
method	the method for controlling the amount of smoothing. <code>method = 1</code> (default) adjusts the smoothing parameter so that the BIC is minimized. <code>method = 2</code> adjusts lambda so that the AIC is minimized. <code>method = 3</code> uses the value supplied for lambda. <code>method = 4</code> adjusts lambda so that the degrees of freedom is equal to the supplied df.
coefstart	an optional vector of starting coefficients.
control	a list of control parameters. See Details.

**Details**

The method fits a P-spline model with equally-spaced B-splines along `x`. The response variables must be Poisson distributed counts, though overdispersion can be accounted. Offset can be provided, otherwise the default is that all weights are one.

The function is specifically tailored to smooth mortality data in one-dimensional setting. In such case the argument `x` would be either the ages or the years under study. Death counts will be the argument `y`. In a Poisson regression setting applied to actual death counts the `offset` will be the logarithm of the exposure population. See example below.

The function can obviously account for zero counts and definite offset. In a mortality context, the user can apply the function to data with zero deaths, but it has to take care that no exposures are equal to zero, i.e. offset equal to minus infinitive. In this last case, the argument `w` can help. The user would need to set weights equal to zero when exposures are equal to zero leading to interpolation of the data. See example below.

Regardless the presence of exposures equal to zero, the argument `w` can also be used for extrapolation and interpolation of the data. Nevertheless see the function [predict.Mort1Dsmooth](#) for a more comprehensive way to forecast mortality rates.

The method produces results similar to function `smooth.spline`, but the smoothing function is a B-spline smooth with discrete penalization directly on the differences of the B-splines coefficients. The user can set the order of difference, the degree of the B-splines and number of them. Nevertheless, the smoothing parameter `lambda` is mainly used to tune the smoothness/model fidelity of the fitted values.

The range in which `lambda` is searched is given in `control - RANGE`. Though it can be modified, the default values are suitable for most of the application.

There are [print.Mort1Dsmooth](#), [summary.Mort1Dsmooth](#), [plot.Mort1Dsmooth](#), [predict.Mort1Dsmooth](#) and [residuals.Mort1Dsmooth](#) methods available for this function.

Four methods for optimizing the smoothing parameter are available. The BIC is set as default. Minimization of the AIC is also possible. BIC will give always smoother outcomes with respect to AIC, especially for large sample size. Alternatively the user can directly provide the smoothing parameter (`method=3`) or the degree of freedom to be used in the model (`method=4`). Note that `Mort1Dsmooth` uses approximated degree of freedom, therefore `method=4` will produce fitted values with degree of freedom only similar to the one provided in `df`. The tolerance level can be set via `control - TOL2`.

Note that the 'ultimate' smoothing with very large `lambda` will approach to a polynomial of degree `ord`.

Starting coefficients for the B-spline basis can be provided by the user. This feature can be useful when a grid-search is manually performed by the user.

The argument `overdispersion` can be set to `TRUE` when possible presence of over(under)dispersion needs to be considered in the selection of the smoothing parameter. Mortality data often present overdispersion also known, in demography, as heterogeneity. Duplicates in insurance data can lead to overdispersed data, too. Smoothing parameter selection may be affected by this phenomenon. When `overdispersion=TRUE`, the function uses a penalized quasi-likelihood method for including an overdispersion parameter (`psi2`) in the fitting procedure. With this approach expected values are assumed equal to the variance multiplied by the parameter `psi2`. See reference. Note that the inclusion of the overdispersion parameter within the estimation might lead to select higher `lambda`, leading to smoother outcomes. When `overdispersion=FALSE` (default value) or `method=3` or `method=4`, `psi2` is estimated after the smoothing parameter have been employed. Overdispersion parameter considerably larger (smaller) than 1 may be a sign of overdispersion (underdispersion).

The `control` argument is a list that can supply any of the following components:

`MON`: Logical. If `TRUE` tracing information on the progress of the fitting is produced. Default: `FALSE`.

`TOL1`: The absolute convergence tolerance for each completed scoring algorithm. Default: `1e-06`.

`TOL2`: Difference between two adjacent smoothing parameters in the (pseudo) grid search, log-scale. Useful only when `method` is equal to 1, 2 or 4. Default: `0.5`.

RANGE: Range of smoothing parameters in which the grid-search is applied, commonly taken in log-scale. Default:  $[10^{-4}; 10^6]$ .

MAX.IT: The maximum number of iterations for each completed scoring algorithm. Default: 50.

The arguments MON, TOL1 and MAX.IT are kept during all the grid search when method is equal to 1, 2 or 4. Function `cleversearch` from package `svcm` is employed to speed the grid search. See [Mort1Dsmooth\\_optimize](#) for details.

## Value

An object of the class `Mort1Dsmooth` with components:

<code>coefficients</code>	vector of fitted (penalized) B-splines coefficients.
<code>residuals</code>	the deviance residuals.
<code>fitted.values</code>	vector of fitted counts.
<code>linear.predictor</code>	vector of fitted linear predictor.
<code>logmortality</code>	fitted mortality rates in log-scale.
<code>lev</code>	diagonal of the hat-matrix.
<code>df</code>	effective dimension.
<code>deviance</code>	Poisson Deviance.
<code>aic</code>	Akaike's Information Criterion.
<code>bic</code>	Bayesian Information Criterion.
<code>psi2</code>	overdispersion parameter.
<code>lambda</code>	the selected (given) smoothing parameter $\lambda$ .
<code>call</code>	the matched call.
<code>n</code>	number of observations.
<code>tolerance</code>	the used tolerance level.
<code>B</code>	the B-splines basis.
<code>ndx</code>	the number of internal knots -1.
<code>deg</code>	degree of the B-splines.
<code>pord</code>	order of difference.
<code>x</code>	values of the predictor variable.
<code>y</code>	set of counts response variable values.
<code>offset</code>	vector of the offset.
<code>w</code>	vector of weights used in the model.

## Author(s)

Carlo G Camarda

## References

Eilers and Marx (1996). Flexible Smoothing with B-splines and Penalties. *Statistical Science*. Vol. 11. 89-121.

**See Also**

[predict.Mort1Dsmooth](#), [plot.Mort1Dsmooth](#), [Mort1Dsmooth\\_optimize](#).

**Examples**

```
## selected data
years <- 1950:2006
death <- selectHMDdata("Japan", "Deaths", "Females",
                      ages = 80, years = years)
exposure <- selectHMDdata("Japan", "Exposures", "Females",
                          ages = 80, years = years)

## various fits
## default using Bayesian Information Criterion
fitBIC <- Mort1Dsmooth(x=years, y=death,
                      offset=log(exposure))

fitBIC
summary(fitBIC)
## subjective choice of the smoothing parameter lambda
fitLAM <- Mort1Dsmooth(x=years, y=death,
                      offset=log(exposure),
                      method=3, lambda=10000)

## plot
plot(years, log(death/exposure),
     main="Mortality rates, log-scale.
         Japanese females, age 80, 1950:2006")
lines(years, fitBIC$logmortality, col=2, lwd=2)
lines(years, fitLAM$logmortality, col=3, lwd=2)
legend("topright", c("Actual", "BIC", "lambda=10000"),
      col=1:3, lwd=c(1,2,2), lty=c(-1,1,1),
      pch=c(1,-1,-1))

## about Extra-Poisson variation (overdispersion)
## checking the presence of overdispersion
fitBIC$psi2 # quite larger than 1
## fitting accounting for overdispersion
fitBICover <- Mort1Dsmooth(x=years, y=death,
                          offset=log(exposure),
                          overdispersion=TRUE)

## difference in the selected smoothing parameters
fitBIC$lambda;fitBICover$lambda
## plotting both situations
plot(fitBICover)
lines(years, fitBIC$logmortality, col=4, lwd=2, lty=2)

## interpolation, better predict.Mort1Dsmooth
exposure.int <- exposure
exposure.int[20:40] <- NA
## Wrong example (don't run)
## Mort1Dsmooth(x=years, y=death,
##             offset=log(exposure.int))
## using the argument w, with warning
w.int <- rep(1, length(years))
```

```

w.int[20:40] <- 0
fit1Dint <- Mort1Dsmooth(x=years, y=death,
                        offset=log(exposure.int), w=w.int)
plot(fit1Dint)

## extrapolation, better using predict.Mort1Dsmooth
years.ext <- seq(years[1], years[length(years)]+20)
exposure.ext <- c(exposure, rep(NA, 20))
death.ext <- c(death, rep(NA, 20))
## Wrong example (don't run)
## Mort1Dsmooth(x=years.ext, y=death.ext,
##             offset=log(exposure.ext))
## using the argument w, with warning
w.ext <- c(rep(1, length(years)), rep(0, 20))
## using the optimal lambda from fitBICover
fit1Dext <- Mort1Dsmooth(x=years.ext, y=death.ext,
                        offset=log(exposure.ext), w=w.ext,
                        method=3, lambda=fitBICover$lambda)

plot(fit1Dext)

```

---

Mort1Dsmooth\_checker    *Check Arguments for Function Mort1Dsmooth*

---

## Description

This is an internal function of package MortalitySmooth which checks whether the arguments given in the function Mort1Dsmooth have proper lengths and suitability.

## Usage

```

Mort1Dsmooth_checker(x, y, offset,
                    w, overdispersion,
                    ndx, deg, pord, lambda,
                    df, method, coefstart,
                    control)

```

## Arguments

x	values of the predictor variable.
y	set of counts response variable values.
offset	a vector with the offset.
w	an optional vector of weights to be used in the fitting process.
overdispersion	logical on the accounting for overdispersion.
ndx	number of internal knots -1.
deg	degree of the B-splines.
pord	order of differences.

lambda	smoothing parameter.
df	a number which specifies the degrees of freedom.
method	the method for controlling the amount of smoothing.
coefstart	a vector with the eventual starting coefficients.
control	a list of control parameters.

**Details**

Internal function used in [Mort1Dsmooth](#) for checking its arguments.

**Value**

A list with checked elements to be used in [Mort1Dsmooth](#)

**Author(s)**

Carlo G Camarda

**See Also**

[Mort1Dsmooth](#)

---

Mort1Dsmooth\_estimate *Estimate 1D P-splines for a given lambda*

---

**Description**

This is an internal function of package MortalitySmooth which estimates coefficients and computes diagnostics for penalized B-splines for a given smoothing parameter within the function Mort1Dsmooth.

**Usage**

```
Mort1Dsmooth_estimate(x, y, offset, wei, psi2,
                      B, lambda, DtD, a.init,
                      MON, TOL1, MAX.IT)
```

**Arguments**

x	Vector for the abscissa of data.
y	Vector of counts response.
offset	Vector with an a priori known component (optional).
wei	An optional vector of weights to be used in the fitting process.
psi2	Overdispersion parameter.
B	B-splines basis.

lambda	Smoothing parameter.
DtD	Inner product of the difference matrix.
a.init	Vector with the initial coefficients.
MON	Logical switch indicating if monitoring is required.
TOL1	The tolerance level in the IWLS algorithm.
MAX.IT	The maximum number of iterations.

### Details

Internal function used in [Mort1Dsmooth](#) for estimating coefficients and computing diagnostics.

### Value

A list with components:

a	fitted coefficients.
h	diagonal of the hat-matrix.
df	effective dimension of used degree of freedom.
aic	Akaike's Information Criterion.
bic	Bayesian Information Criterion.
dev	Poisson deviance.
tol	tolerance level.
BtWB	inner product of basis and weights.
P	penalty matrix.

### Author(s)

Carlo G Camarda

### See Also

[Mort1Dsmooth\\_update](#), [Mort1Dsmooth](#).

---

Mort1Dsmooth\_optimize *Optimize a 1D Penalized-Poisson IWLS over smoothing parameters*

---

### Description

This is an internal function of package MortalitySmooth which optimizes the smoothing parameter for penalized B-splines within the function Mort1Dsmooth.

**Usage**

```
Mort1Dsmooth_optimize(x, y, offset, wei,
                      psi2, B, DtD, a.init,
                      MON, TOL1, TOL2, RANGE, MAX.IT,
                      MET)
```

**Arguments**

x	vector for the abscissa of data.
y	vector of counts response.
offset	vector with an a priori known component (optional).
wei	an optional vector of weights to be used in the fitting process.
psi2	an overdispersion parameter used the quasi-likelihood approach.
B	B-splines basis.
DtD	inner product of the difference matrix
a.init	initial coefficients
MON	Logical switch indicating if monitoring is required.
TOL1	The tolerance level in the IWLS algorithm.
TOL2	difference between two adjacent smoothing parameters in the (pseudo) grid search, log-scale.
RANGE	range in which smoothing parameter should be searched.
MAX.IT	the maximum number of iterations
MET	the method for controlling the amount of smoothing

**Details**

The function aims to find the optimal smoothing parameter within the given RANGE in [Mort1Dsmooth](#) with method equal to 1 or 2 (BIC and AIC). It employs the function `cleversearch` from package `svcm` in two separate steps. First it searches using a rough grid (4 times TOL2) and the median of RANGE as starting lambda. Afterwards it searches in the restricted range around the sub-optimal smoothing parameter, using a finer grid defined by TOL2.

This procedure allows to find a precise smoothing parameter in an efficient way: we do not explore the full range of possible lambda values, moving at most one grid step up or down. Furthermore the two steps routine reduces the risk of finding sub-optimal smoothing parameter.

**Author(s)**

Carlo G Camarda

**See Also**

[Mort1Dsmooth\\_update](#), [Mort1Dsmooth\\_estimate](#), [Mort1Dsmooth](#).

---

Mort1Dsmooth\_update     *Update a 1D Penalized-Poisson Iteration*

---

**Description**

This is an internal function of package MortalitySmooth which update coefficients for penalized B-splines for a given smoothing parameter within the function Mort1Dsmooth.

**Usage**

```
Mort1Dsmooth_update(x, y, offset, wei, psi2, B, lambdaP, a)
```

**Arguments**

x	vector for the abscissa of data.
y	vector of counts response.
offset	vector with an a priori known component (optional).
wei	an optional vector of weights to be used in the fitting process.
psi2	an overdispersion parameter used the quasi-likelihood approach.
B	B-splines basis.
lambdaP	penalty factor, included the smoothing parameter.
a	old coefficients.

**Details**

Internal function used for updating coefficients of the B-splines in 1D Poisson penalized B-splines model.

**Value**

A vector of updated coefficients

**Author(s)**

Carlo G Camarda

**See Also**

[Mort1Dsmooth](#).

Mort2Dsmooth

*Fit Two-dimensional Poisson P-splines***Description**

Returns an object of class Mort2Dsmooth which is a two-dimensional P-splines smooth of the input data of degree and order fixed by the user. Specifically tailored to mortality data.

**Usage**

```
Mort2Dsmooth(x, y, Z, offset, W, overdispersion=FALSE,
             ndx = c(floor(length(x)/5), floor(length(y)/5)),
             deg = c(3, 3), pord = c(2, 2),
             lambdas = NULL, df = NULL, method = 1,
             coefstart = NULL,
             control = list())
```

**Arguments**

x	vector for the abscissa of data. These must be at least $2 \text{ ndx}[1] + 1$ of them.
y	vector for the ordinate of data. These must be at least $2 \text{ ndx}[2] + 1$ of them.
Z	matrix of counts response. Dimensions of Z should correspond to the length of x and y, respectively.
offset	matrix with an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric matrix of dimensions of Z or a single numeric value.
W	an optional matrix of weights to be used in the fitting process. This should be NULL or a numeric matrix of dimensions of Z or a single numeric value.
overdispersion	logical on the accounting for overdispersion in the smoothing parameters selection criterion. See Details. Default: FALSE.
ndx	vector with the number of internal knots -1 for each axis. Default: $[\text{floor}(\text{length}(x)/5), \text{floor}(\text{length}(y)/5)]$ .
deg	vector with the degree of the B-splines for each axis. Default: [3, 3].
pord	vector with the order of differences for each axis. Default: [2, 2].
lambdas	vector with smoothing parameters, possibly one for axis (optional).
df	a number which specifies the degrees of freedom (optional).
method	the method for controlling the amount of smoothing. method = 1 (default) adjusts the two smoothing parameters so that the BIC is minimized. method = 2 adjusts lambdas so that the AIC is minimized. method = 3 uses the values supplied for lambdas. Isotropic smoothing is allowed in this method. method = 4 adjusts lambdas so that the degrees of freedom is equal to the supplied df.
coefstart	an optional matrix of starting coefficients.
control	a list of control parameters. See Details.

## Details

The method fits a two-dimensional P-spline model with equally-spaced B-splines along  $x$  and  $y$ . The response variables must be a matrix of Poisson distributed counts. Offset can be provided, otherwise the default is that all weights are one.

The function is specifically tailored to smooth mortality data in one-dimensional setting. In such case the argument  $x$  would be the ages and the argument  $y$  the years under study. The matrix of death counts will be the argument  $Z$ . In a Poisson regression setting applied to actual death counts the `offset` will be the logarithm of the matrix of exposure population. See example below.

The function can obviously account for zero counts and definite offset. In a mortality context, the user can apply the function to data with zero deaths, but it has to take care that no exposures are equal to zero, i.e. `offset` equal to minus infinitive. In this last case, the argument  $W$  can help. The user would need to set weights equal to zero when exposures are equal to zero leading to interpolation of the data. See example below.

Regardless the presence of exposures equal to zero, the argument  $W$  can also be used for extrapolation and interpolation of the data. Nevertheless see the function [predict.Mort2Dsmooth](#) for a more comprehensive way to forecast mortality rates over ages and years.

The method produces results from a smoothing function which is the Kronecker product of B-spline basis over the two axes and include a discrete penalization directly on the differences of the B-splines coefficients. The user can set the order of difference, the degree of the B-splines and number of them for each of the axis. Nevertheless, the smoothing parameters `lambdas` are mainly used to tune the smoothness/model fidelity of the fitted values.

The ranges in which `lambda` is searched is given in `control` - `RANGEx` and `RANGEy`. Though they can be modified, the default values are suitable for most of the application.

There are [print.Mort2Dsmooth](#), [summary.Mort2Dsmooth](#), [plot.Mort2Dsmooth](#) [predict.Mort2Dsmooth](#) and [residuals.Mort2Dsmooth](#) methods available for this function.

Four methods for optimizing the smoothing parameters are available. The BIC is set as default. Minimization of the AIC is also possible. BIC will give always smoother outcomes with respect to AIC, especially for large sample size. Alternatively the user can directly provide the smoothing parameters (`method=3`) or the degrees of freedom to be used in the model (`method=4`). In this last case isotropic smoothing (same smoothing parameter over  $x$  and  $y$ ) is employed. If the user provides only a single value for the argument `lambdas`, isotropic smoothing is applied (with warning). Note that `Mort2Dsmooth` uses approximated degrees of freedom, therefore `method=4` will produce fitted values with degree of freedom only similar to the one provided in `df`. The tolerance level can be set via `control` - `TOL2`.

Note that the two-dimensional 'ultimate' smoothing with very large `lambda` will approach to a surface which is a product of two polynomial of degree `por1` and `por2`, respectively. In particular, when `por1=c(2,2)` the 'ultimate' smoothing is a bi-linear surface over  $x$  and  $y$ .

The argument `overdispersion` can be set to `TRUE` when smoothing parameters selection has to account for possible presence of over(under)dispersion. Mortality data often present overdispersion also known, in demography, as heterogeneity. Duplicates in insurance data can lead to overdispersed data, too. Smoothing parameters selection may be affected by this phenomenon. When `overdispersion=TRUE`, the function uses a penalized quasi-likelihood method for including an overdispersion parameter (`psi2`) in the fitting procedure. With this approach expected values are assumed equal to the variance multiplied by the parameter `psi2`. See reference. Note that with overdispersed data both BIC and AIC might select higher `lambdas`, leading to smoother outcomes.

When `overdispersion=FALSE` (default value) or `method=3` or `method=4`, `psi2` is estimated after the smoothing parameters have been employed. Overdispersion parameter larger (smaller) than 1 may be a sign of overdispersion (underdispersion).

The control argument is a list that can supply any of the following components:

`MON`: Logical. If `TRUE` tracing information on the progress of the fitting is produced. Default: `FALSE`.

`TOL1`: The absolute convergence tolerance for each completed scoring algorithm. Default: `1e-06`.

`TOL2`: Difference between two adjacent smoothing parameters in the (pseudo) grid search, log-scale. Useful only when `method` is equal to 1, 2 or 4. Default: `0.5`.

`RANGEx`: Range of smoothing parameters over `x` in which the grid-search is applied, commonly taken in log-scale. Default: `[10^-4 ; 10^6]`.

`RANGEy`: Range of smoothing parameters over `y` in which the grid-search is applied, commonly taken in log-scale. Default: `[10^-4 ; 10^6]`.

`MAX.IT`: The maximum number of iterations for each completed scoring algorithm. Default: `50`.

The arguments `MON`, `TOL1` and `MAX.IT` are kept during all the (pseudo) grid search when `method` is equal to 1, 2 or 4. Function `cleversearch` from package `svcm` is employed to speed the grid search. See [Mort2Dsmooth\\_optimize](#) for details.

The inner functions work using an arithmetic of arrays defined as Generalized Linear Array Model (GLAM) (see reference). In order to avoid construction of large Kronecker product basis from the large number of B-splines along the axes, the function profits of the special structure of both the data as rectangular array and the model matrix as tensor product. It uses sequence of nested matrix operations and this leads to low storage and high speed computation within the IWLS algorithm. Moreover, the function do not vectorize the whole system keeping the actual two-dimensional array structure within the scoring algorithm.

## Value

An object of the class `Mort2Dsmooth` with components:

<code>coefficients</code>	matrix of fitted (penalized) B-splines coefficients.
<code>residuals</code>	matrix of deviance residuals.
<code>fitted.values</code>	matrix of fitted counts.
<code>linear.predictor</code>	matrix of fitted linear predictor.
<code>logmortality</code>	fitted mortality rates in log-scale.
<code>df</code>	effective dimension.
<code>dev</code>	Poisson Deviance.
<code>aic</code>	Akaike's Information Criterion.
<code>bic</code>	Bayesian Information Criterion.
<code>psi2</code>	Overdispersion parameter.
<code>lambdas</code>	the selected (given) smoothing parameters.
<code>call</code>	the matched call.
<code>m</code>	length of argument <code>x</code> .

n	length of argument y.
tolerance	the used tolerance level.
lev	diagonal of the hat-matrix.
ndx	the number of internal knots -1 for each axis.
deg	degree of the B-splines for each axis.
pord	order of difference for each axis.
x	vector for the abscissa of data.
y	vector for the ordinate of data.
Z	matrix of counts response.
offset	matrix with an a priori known component.
W	matrix of weights.
Bx	the B-splines basis over x.
By	the B-splines basis over y.

**Author(s)**

Carlo G Camarda

**References**

Eilers, P. H. C., I. D. Currie, and M. Durban (2006). Fast and compact smoothing on large multidimensional grids. *Computational Statistics & Data Analysis* 50, 61-76.

Currie, I. D., M. Durban, and P. H. C. Eilers (2006). Generalized linear array models with applications to multidimensional smoothing. *Journal of the Royal Statistical Society. Series B.* 68, 259-280.

**See Also**

[predict.Mort2Dsmooth](#), [plot.Mort2Dsmooth](#).

**Examples**

```
## selected data
ages <- 50:100
years <- 1950:2006
death <- selectHMDdata("Sweden", "Deaths", "Females",
                      ages = ages, years = years)
exposure <- selectHMDdata("Sweden", "Exposures", "Females",
                        ages = ages, years = years)

## fit with BIC
fitBIC <- Mort2Dsmooth(x=ages, y=years, Z=death,
                     offset=log(exposure))

fitBIC
summary(fitBIC)

## plot age 50 log death rates (1st row)
```

```

plot(years, log(death[1,]/exposure[1,]),
      main="Mortality rates, log-scale.
          Swedish females, age 50, 1950:2006")
lines(years, fitBIC$logmortality[1,], col=2, lwd=2)

## plot over age and years
## fitted log death rates from fitBIC
grid. <- expand.grid(list(ages=ages, years=years))
grid.$lmx <- c(fitBIC$logmortality)
levelplot(lmx ~ years * ages , grid.,
          at=quantile(grid.$lmx, seq(0,1,length=10)),
          col.regions=rainbow(9))

## about Extra-Poisson variation (overdispersion)
## checking the presence of overdispersion
fitBIC$psi2
## fitting accounting for overdispersion
fitBICover <- Mort2Dsmooth(x=ages, y=years, Z=death,
                          offset=log(exposure),
                          overdispersion=TRUE)
## difference in the selected smoothing parameters
fitBIC$lambda;fitBICover$lambda

## interpolation when exposure equal to zero
new.exposure <- exposure
new.exposure[20:30, ] <- 0
## Wrong example (don't run)
## fit2Dint <- Mort2Dsmooth(x=ages, y=years, Z=death,
##                          offset=log(new.exposure))
## using the argument W, with warning
W <- matrix(1, length(ages), length(years))
W[20:30, ] <- 0
fit2Dint <- Mort2Dsmooth(x=ages, y=years, Z=death,
                        offset=log(new.exposure), W=W,
                        method=3, lambda=c(100, 100))

plot(fit2Dint)

## using only 90% of the data
W <- matrix(1, length(ages), length(years))
set.seed(3)
zeros <- sample(x=1:prod(dim(W)),
                size=round(prod(dim(W))/100*90),
                replace=FALSE)
W[zeros] <- 0
fit2Dint <- Mort2Dsmooth(x=ages, y=years, Z=death,
                        offset=log(exposure), W=W,
                        method=3, lambda=c(100, 100))

plot(fit2Dint)

## extrapolation over years, better predict.Mort2Dsmooth
years.new <- 1950:2025
M <- matrix(0, nrow=length(ages),

```

```

      ncol=length(years.new)-length(years))
death.new <- cbind(death, M)
exposure.new <- cbind(exposure, M)
W <- matrix(1, length(ages), length(years))
W.new <- cbind(W, M)

fit2Dext <- Mort2Dsmooth(x=ages, y=years.new, Z=death.new,
                        offset=log(exposure.new), W=W.new,
                        method=3, lambdas=c(1000, 316))

plot(fit2Dext)

```

---

Mort2Dsmooth\_checker    *Check Arguments for Function Mort2Dsmooth*


---

### Description

This is an internal function of package MortalitySmooth which checks whether the arguments given in the function Mort2Dsmooth have proper lengths and suitability.

### Usage

```

Mort2Dsmooth_checker(x, y, Z, offset, W,
                    overdispersion,
                    ndx, deg, pord, lambdas,
                    df, method, coefstart,
                    control)

```

### Arguments

x	vector for the abscissa of data.
y	vector for the ordinate of data.
Z	matrix of counts response.
offset	matrix with an a priori known component.
W	an optional matrix of weights to be used in the fitting process.
overdispersion	logical on the accounting for overdispersion.
ndx	vector with the number of internal knots -1 for each axis.
deg	vector with the degree of the B-splines for each axis.
pord	vector with the order of differences for each axis.
lambdas	vector with smoothing parameters, possibly one for axis.
df	a number which specifies the degrees of freedom.
method	the method for controlling the amount of smoothing.
coefstart	a matrix with the eventual starting coefficients.
control	a list of control parameters.

**Details**

Internal function used in [Mort2Dsmooth](#) for checking its arguments.

**Value**

A list with checked elements to be used in [Mort2Dsmooth](#).

**Author(s)**

Carlo G Camarda

**See Also**

[Mort2Dsmooth](#).

---

Mort2Dsmooth\_estimate *Estimate 2D P-splines for two given lambdas*

---

**Description**

This is an internal function of package MortalitySmooth which estimates coefficients and computes diagnostics for two-dimensional penalized B-splines for two given smoothing parameters within the function `Mort2Dsmooth`.

**Usage**

```
Mort2Dsmooth_estimate(x, y, Z, offset, psi2, wei,
                     Bx, By, nbx, nby, RTBx, RTBy,
                     lambdas, Px, Py, a.init,
                     MON, TOL1, MAX.IT)
```

**Arguments**

x	vector for the abscissa of data.
y	vector for the ordinate of data.
Z	matrix of counts response.
offset	matrix with an a priori known component (optional).
psi2	overdispersion parameter.
wei	an optional matrix of weights to be used in the fitting process.
Bx	B-splines basis for the x-axis.
By	B-splines basis for the y-axis.
nbx	number of B-splines for the x-axis.
nby	number of B-splines for the y-axis.
RTBx	tensors product of B-splines basis for the x-axis.

RTBy	tensors product of B-splines basis for the y-axis.
lambdas	vector with the two smoothing parameters.
Px	penalty factor for the x-axis.
Py	penalty factor for the y-axis.
a.init	matrix with the initial coefficients.
MON	logical switch indicating if monitoring is required.
TOL1	the tolerance level in the IWLS algorithm.
MAX.IT	the maximum number of iterations.

### Details

Internal function used in [Mort2Dsmooth](#) for estimating coefficients and computing diagnostics.

### Value

A list with components:

a	fitted coefficients (in a matrix).
h	diagonal of the hat-matrix.
df	effective dimension of used degree of freedom.
aic	Akaike's Information Criterion.
bic	Bayesian Information Criterion.
dev	Poisson deviance.
tol	tolerance level.
BWB	inner product of basis and weights.
P	penalty matrix.

### Author(s)

Carlo G Camarda

### See Also

[Mort2Dsmooth\\_update](#), [Mort2Dsmooth](#).

---

Mort2Dsmooth\_optimize *Optimize a 2D Penalized-Poisson IWLS over smoothing parameters*

---

### Description

This is an internal function of package MortalitySmooth which optimizes the smoothing parameter for penalized B-splines within the function Mort2Dsmooth.

### Usage

```
Mort2Dsmooth_optimize(x, y, Z, offset, wei,
                      psi2, Bx, By, nbx, nby,
                      RTBx, RTBy, Px, Py,
                      a.init,
                      MON, TOL1, TOL2,
                      RANGEx, RANGEy,
                      MAX.IT, MET)
```

### Arguments

x	vector for the abscissa of data.
y	vector for the ordinate of data.
Z	matrix of counts response.
offset	matrix with an a priori known component (optional).
wei	an optional matrix of weights to be used in the fitting process.
psi2	overdispersion parameter.
Bx	B-splines basis for the x-axis.
By	B-splines basis for the y-axis.
nbx	number of B-splines for the x-axis.
nby	number of B-splines for the y-axis.
RTBx	tensors product of B-splines basis for the x-axis.
RTBy	tensors product of B-splines basis for the y-axis.
Px	penalty factor for the x-axis.
Py	penalty factor for the y-axis.
a.init	matrix with the initial coefficients.
MON	Logical switch indicating if monitoring is required.
TOL1	The tolerance level in the IWLS algorithm.
TOL2	difference between two adjacent smoothing parameters in the grid search, log-scale.
RANGEx	range in which smoothing parameter for x should be searched.
RANGEy	range in which smoothing parameter for y should be searched.
MAX.IT	the maximum number of iterations
MET	the method for controlling the amount of smoothing

**Details**

The function aims to find the optimal smoothing parameters within the given RANGE<sub>x</sub> and RANGE<sub>y</sub> in [Mort2Dsmooth](#) with method equal to 1 or 2 (BIC and AIC). It employs the function `cleversearch` from package `svcm` in two separate steps. First it searches using a rough grid (4 times TOL2) and the median of RANGE<sub>x</sub> and RANGE<sub>y</sub> as starting lambdas. Afterwards it searches in the restricted areas around the sub-optimal smoothing parameters, using a finer grid defined by TOL2.

This procedure allows to find precise smoothing parameters in an efficient way: we do not explore the full ranges of possible lambda values, but we optimize each parameter in turn, moving at most one grid step up or down. Furthermore the two steps routine reduces the risk of finding sub-optimal smoothing parameters.

**Author(s)**

Carlo G Camarda

**See Also**

[Mort2Dsmooth\\_update](#), [Mort2Dsmooth\\_estimate](#), [Mort2Dsmooth](#).

---

Mort2Dsmooth\_se

*Compute a 2D standard errors*


---

**Description**

This is an internal function of package `MortalitySmooth` which calculates the inner product of a matrix (from a Kronecker product) and a sparse weight matrix in order to obtain standard errors. It uses the same idea employed in `MortSmooth.BWB` and the elements after the IWLS converged, including the penalty term.

**Usage**

```
Mort2Dsmooth_se(RTBx, RTBy, nbx, nby, BWB.P1)
```

**Arguments**

RTBx	tensors product of B-splines basis for the x-axis.
RTBy	tensors product of B-splines basis for the y-axis.
nbx	number of B-splines for the x-axis.
nby	number of B-splines for the y-axis.
BWB.P1	inverse of the LHS of the Poisson system of equations.

**Details**

This function is only used within `predict.Mort2Dsmooth` when standard errors are required. The arguments `BWB.P1` is the LHS after convergence is reached and smoothing parameter selected. The standard errors as given in the function are computed for the linear predictor term and simple computation is needed to obtain standard errors for the Poisson counts. Anyway `predict.Mort2Dsmooth` takes care of such differences.

The Generalized Linear Array Models setting is explained in the reference in `MortSmooth_BWB` and `Mort2Dsmooth`.

**Value**

A matrix of standard errors for the linear predictor term.

**Author(s)**

Carlo G Camarda

**See Also**

`Mort2Dsmooth`, `MortSmooth_BWB`, `predict.Mort2Dsmooth`.

---

Mort2Dsmooth\_update      *Update a 2D Penalized-Poisson Iteration*

---

**Description**

This is an internal function of package `MortalitySmooth` which update coefficients for penalized B-splines for two given smoothing parameters within the function `Mort2Dsmooth`.

**Usage**

```
Mort2Dsmooth_update(x, y, Z, offset, psi2, wei,
                   Bx, By, nbx, nby, RTBx, RTBy, P, a)
```

**Arguments**

<code>x</code>	vector for the abscissa of data.
<code>y</code>	vector for the ordinate of data.
<code>Z</code>	matrix of counts response.
<code>offset</code>	matrix with an a priori known component (optional).
<code>wei</code>	an optional matrix of weights to be used in the fitting process.
<code>psi2</code>	overdispersion parameter.
<code>Bx</code>	B-splines basis for the x-axis.
<code>By</code>	B-splines basis for the y-axis.

nbx	number of B-splines for the x-axis.
nby	number of B-splines for the y-axis.
RTBx	tensors product of B-splines basis for the x-axis.
RTBy	tensors product of B-splines basis for the y-axis.
P	penalty factor.
a	coefficients (in a matrix).

**Details**

Internal function used for updating a matrix coefficients of the B-splines in 2D Poisson penalized B-splines model.

**Value**

A matrix of updated coefficients.

**Author(s)**

Carlo G Camarda

**See Also**

[Mort2Dsmooth.](#)

---

MortSmooth_bbase	<i>Construct B-spline basis</i>
------------------	---------------------------------

---

**Description**

This is an internal function of package MortalitySmooth which creates equally-spaced B-splines basis over an abscissa of data within the function Mort1Dsmooth.

**Usage**

```
MortSmooth_bbase(x, x1, xr, ndx, deg)
```

**Arguments**

x	vector for the abscissa of data.
x1	left boundary.
xr	right boundary.
ndx	number of internal knots minus one or number of internal intervals.
deg	degree of the splines.

**Details**

The function reproduce an algorithm presented by Eilers and Marx (2010) using differences of truncated power functions (see [MortSmooth\\_tpower](#)). The final matrix has a single B-spline for each of the  $[ndx + deg]$  columns. The number of rows is equal to the length of  $x$ .

The function differs from `bs` in the package `splines` since it automatically constructed B-splines with identical shape. This would allow a simple interpretation of coefficients and application of simple differencing.

**Value**

A matrix containing equally-spaced B-splines of degree `deg` along  $x$  for each column.

**Author(s)**

Carlo G Camarda

**References**

Eilers and Marx (2010). Splines, Knots, and Penalties. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2, 637-653.

**See Also**

[MortSmooth\\_tpower](#).

**Examples**

```
x <- seq(0,1,length=500)
## B-splines basis of degree 1
B1 <- MortSmooth_bbase(x=x, xl=min(x), xr=max(x),
                      ndx=10, deg=1)
matplot(x, B1, t="l", main="B-splines basis of degree 1")
## B-splines basis of degree 3
B3 <- MortSmooth_bbase(x=x, xl=min(x), xr=max(x),
                      ndx=10, deg=3)
matplot(x, B3, t="l", main="B-splines basis of degree 3")
```

---

MortSmooth\_BcoefB

*Multiply a matrix (from a kronecker product) by a matrix*

---

**Description**

This is an internal function of package `MortalitySmooth` which compute the linear predictor and the RHS of the IWLS algorithm for 2D penalized B-splines within the function `Mort2Dsmooth`.

**Usage**

```
MortSmooth_BcoefB(X1, X2, mat)
```

**Arguments**

X1	first marginal matrix.
X2	second marginal matrix.
mat	matrix to be multiplied.

**Details**

Internal function used for computing elements within the [Mort2Dsmooth](#) function.

**Value**

A matrix that multiplies X1 by mat and the transpose of X2.

**Author(s)**

Carlo G Camarda

**See Also**

[Mort2Dsmooth](#), [Mort2Dsmooth\\_estimate](#).

---

MortSmooth_BWB	<i>Calculates the inner product of a matrix and a sparse weight matrix</i>
----------------	--

---

**Description**

This is an internal function of package MortalitySmooth which calculates the inner product of a matrix (from a Kronecker product) and a sparse weight matrix within the function `Mort2Dsmooth`. It constructs the LHD of the IWLS algorithm following the idea proposed in the Generalized Linear Array Models (see references).

**Usage**

```
MortSmooth_BWB(RTBx, RTBy, nbx, nby, W)
```

**Arguments**

RTBx	tensors product of B-splines basis for the x-axis.
RTBy	tensors product of B-splines basis for the y-axis.
nbx	number of B-splines for the x-axis.
nby	number of B-splines for the y-axis.
W	matrix of weights.

**Details**

The function employs an arithmetic of arrays which allows to define the LHS of the Poisson system of equations as a sequence of nested matrix operations. Such way of operating with arrays leads to low storage and high speed computation when data are structure as array.

**Value**

A matrix of inner product of a matrix and a sparse weight matrix.

**Author(s)**

Carlo G Camarda

**References**

Eilers, P. H. C., I. D. Currie, and M. Durban (2006). Fast and compact smoothing on large multidimensional grids. *Computational Statistics & Data Analysis* 50, 61-76.

Currie, I. D., M. Durban, and P. H. C. Eilers (2006). Generalized linear array models with applications to multidimensional smoothing. *Journal of the Royal Statistical Society. Series B.* 68, 259-280.

**See Also**

[Mort2Dsmooth](#), [Mort2Dsmooth\\_estimate](#), [Mort2Dsmooth\\_update](#).

---

MortSmooth_tpower	<i>Truncated p-th Power Function</i>
-------------------	--------------------------------------

---

**Description**

This is an internal function of package MortalitySmooth which constructs a truncated  $p$ -th power function along an abscissa within the function MortSmooth\_bbase

**Usage**

```
MortSmooth_tpower(x, t, p)
```

**Arguments**

x	vector for the abscissa of data.
t	vector of truncation points.
p	degree of the power.

**Details**

Internal function used in MortSmooth\_bbase. The vector t contains the knots. The simplest system of truncated power functions uses  $p = 0$  and it consists of step functions with jumps of size 1 at the truncation points t.

**Author(s)**

Carlo G Camarda

**References**

Eilers and Marx (2004). Splines, Knots, and Penalties. Unpublished manuscript.

**See Also**[MortSmooth\\_bbase](#).**Examples**

```
x <- seq(0,1,length=100)
f1 <- MortSmooth_tpower(x=x, t=0.1, p=3)
f2 <- MortSmooth_tpower(x=x, t=0.2, p=3)
f3 <- MortSmooth_tpower(x=x, t=0.3, p=3)
## Truncated 3rd power functions equally-spaced
plot(x, f1, t="1",
      main="Truncated 3rd power functions equally-spaced")
lines(x, f2, col=2)
lines(x, f3, col=3)
```

plot.HMDdata

*Plot age and year specific demographic data***Description**

Plot one or two-dimensional mortality data from a HMDdata object created by selectHMDdata.

**Usage**

```
## S3 method for class 'HMDdata'
plot(x, ...)
```

**Arguments**

x                    a HMDdata object.  
 ...                  other plotting parameters. Either passed to plot in a one-dimensional plot or passed to levelplot in a two-dimensional surface.

**Details**The function needs [HMDdata](#) to be uploaded.

A HMDdata object can be produced by selectHMDdata.

The function recognizes the HMDdata object, its dimensions and its attributes (country, type of data and sex). Given the dimensions of the object the function plot either a simple unidimensional plot or a more complex shaded contour map over age and years.

Note that rates are automatically plotted in log-scale.

**Value**

None. Function produces a plot.

**Author(s)**

Carlo G Camarda

**See Also**

[HMDdata](#), [selectHMDdata](#).

**Examples**

```
## 1-D plot
popDENf <- selectHMDdata("Denmark", "Population",
                        "Females", 50:100, 2000)
plot(popDENf, main="Danish female population, 2000",
     xlab="ages", pch=2, col=2)
ratDENf <- selectHMDdata("Denmark", "Rates",
                        "Females", 50:100, 2000)
plot(ratDENf, main="Danish female log death rates, 2000",
     xlab="ages")

## 2-D plot
years <- 1950:2000
ages <- 0:100
popJAPf <- selectHMDdata("Japan", "Population", "Females",
                        ages, years)
plot(popJAPf, main="Japanese female population")
ratSWEf <- selectHMDdata("Sweden", "Rates", "Females")
plot(ratSWEf, main="Swedish female log death rates")
```

---

plot.Mort1Dsmooth

*Plot of the Mort1Dsmooth outcomes*

---

**Description**

It produces a plot where the x-axis is the predictor of the given Mort1Dsmooth object and the y-axis is both actual and fitted log-rates, or actual and fitted counts.

**Usage**

```
## S3 method for class 'Mort1Dsmooth'
plot(x, Type = c("logrates", "deaths"), ...)
```

**Arguments**

x	an object of class "Mort1Dsmooth", usually, a result of a call to Mort1Dsmooth.
Type	the type of plot which should be returned. The alternatives are: logrates (default) and death.
...	other plotting parameters passed to plot. Not in used.

**Details**

The function provides a simple tool for checking the outcomes of a Mort1Dsmooth object. In a Poisson setting, one would commonly look at rates in log-scale, but actual and fitted counts can be plotted too.

Plotting parameters are given and cannot be changed.

**Value**

None. Function produces a plot

**Author(s)**

Carlo G Camarda

**See Also**

[Mort1Dsmooth](#) for computing Mort1Dsmooth.object.

**Examples**

```
## selected data
years <- 1930:2006
death <- selectHMDdata("Denmark", "Deaths", "Females",
                      ages = 60, years = years)
exposure <- selectHMDdata("Denmark", "Exposures", "Females",
                          ages = 60, years = years)
## fit
fit <- Mort1Dsmooth(x=years, y=death, offset=log(exposure),
                   method=3, lambda=100)
## plotting actual and fitted data
par(mfrow=c(1,2))
plot(fit)
plot(fit, "deaths")
par(mfrow=c(1,1))
```

---

plot.Mort2Dsmooth      *Plot of the Mort2Dsmooth outcomes*

---

### Description

It produces two level plots, side-by-side, where the x- and y-axis are the predictors (x and y) of the given Mort2Dsmooth object and the numeric response is either the matrices of actual and fitted log-rates, or the matrices of actual and fitted counts.

### Usage

```
## S3 method for class 'Mort2Dsmooth'
plot(x,
      type = c("logrates", "deaths"),
      palette = c("rainbow",
                  "heat.colors",
                  "terrain.colors",
                  "topo.colors",
                  "cm.colors"), ...)
```

### Arguments

x	an object of class "Mort2Dsmooth", usually, a result of a call to Mort2Dsmooth.
type	the type of plot which should be returned. The alternatives are: logrates (default) and death.
palette	a string with the color palette for creating a vector of contiguous colors. Default: rainbow.
...	other plotting parameters passed to levelplot. Not in used.

### Details

The function provides a simple tool for checking the outcomes of a Mort2Dsmooth object. In a Poisson setting, one would commonly look at rates in log-scale, but actual and fitted counts can be plotted too.

The function uses levelplot from package lattice to construct the two level plots side-by-side. Plotting parameters are given and cannot be changed. The only exception is the color palette: five different palletes are available and described in the associated documentation: the default rainbow as well as heat.colors, terrain.colors, topo.colors, and cm.colors.

A palette with 9 colors is chosen and the breaks for the level plot are chosen according to the deciles of distributions of actual and fitted log-rates (counts).

### Value

None. Function produces a plot.

**Author(s)**

Carlo G Camarda

**See Also**[Mort2Dsmooth](#) for computing Mort2Dsmooth.object.**Examples**

```
## selected data
ages <- 50:100
years <- 1970:2006
death <- selectHMDdata("Denmark", "Deaths", "Females",
                       ages = ages, years = years)
exposure <- selectHMDdata("Denmark", "Exposures", "Females",
                          ages = ages, years = years)

## fit
fit <- Mort2Dsmooth(x=ages, y=years, Z=death,
                   offset=log(exposure),
                   method=3, lambdas=c(100,500))

## plotting log-death rates
plot(fit, palette="terrain.colors")
## plotting death counts
plot(fit, type="death")
```

---

predict.Mort1Dsmooth    *Predict Method for 1D P-splines Fits*

---

**Description**

Obtains predictions, forecasts and optionally estimated standard errors of those predictions from a fitted Mort1Dsmooth object.

**Usage**

```
## S3 method for class 'Mort1Dsmooth'
predict(object, newdata = NULL,
        type = c("link", "response"),
        se.fit = FALSE, ...)
```

**Arguments**

object	an object of class "Mort1Dsmooth", usually, a result of a call to Mort1Dsmooth.
newdata	optionally, a vector in which to look for x with which to predict. If omitted, the fitted linear predictors are used.
type	the type of prediction required. The default ("link") is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable.
se.fit	logical switch indicating if standard errors are required. Default: FALSE.
...	other predict parameters to be passed to predict. Not in used.

**Details**

If `newdata` is omitted the predictions are based on the data used for the fit. Note that, in common with other prediction functions, any offset supplied as an argument is always ignored when predicting, unlike offsets specified in modelling.

Forecast is possible, therefore `newdata` can include values out of the range of the original `x`. See example below.

Interpolation is also feasible. See example below.

It is noteworthy to point out that when interpolating the B-spline coefficients form a polynomial sequence of degree  $2 * \text{pord} - 1$ , and for extrapolation/forecasting the degree is `pord-1`. For the default order of differences in `Mort1Dsmooth` which is `pord=2`, one has a cubic interpolation and a linear extrapolation.

**Value**

If `se.fit = FALSE`, a vector of predictions. If `se.fit = TRUE`, a list with components:

<code>fit</code>	predictions.
<code>se.fit</code>	estimated standard errors.

**Author(s)**

Carlo G Camarda

**See Also**

[Mort1Dsmooth](#) for computing `Mort1Dsmooth`.object.

**Examples**

```
## forecasting example
## selected data
years <- 1950:2006
death <- selectHMDdata("Denmark", "Deaths", "Females",
                      ages = 70, years = years)
exposure <- selectHMDdata("Denmark", "Exposures", "Females",
                        ages = 70, years = years)

## fit
fit <- Mort1Dsmooth(x=years, y=death,
                  offset=log(exposure))

## extrapolation
years.new <- 1950:2025
pred <- predict(fit, newdata=years.new,
              type="link", se.fit=TRUE)

## plotting actual and fitted log-rates
## and 95% confidence interval
plot(years, log(death/exposure),
     xlim=range(years.new), ylim=range(pred$fit))
lines(years.new, pred$fit, lwd=2, col=2)
lines(years.new, pred$fit + 2*pred$se.fit, lty=2, col=2)
```

```

lines(years.new, pred$fit - 2*pred$se.fit, lty=2, col=2)

## interpolation+extrapolating example over years
## selected+summed data
years0 <- 1860:2004
death0 <- selectHMDdata("Sweden", "Deaths",
                        "Females", 50, years0)
death <- c(rowsum(death0, gl(length(death0),
                             5, length(death0))))
exposure0 <- selectHMDdata("Sweden", "Exposures",
                           "Females", 50, years0)
exposure <- c(rowsum(exposure0, gl(length(exposure0),
                                    5, length(exposure0))))

years <- seq(1862, 2002, 5)
## fit
fit <- Mort1Dsmooth(x=years, y=death, offset=log(exposure))
## predict the model for each year
pre <- predict(fit, newdata=years0,
               se.fit=TRUE)

## plot log-rates
plot(years, log(death/exposure))
## add fitted log-rates from fit
## (every fifth year)
points(years, log(fit$fitted/exposure),
        pch=16, col=2)
## add to the plot fit and 95% confidence interval
## (every single year)
points(years0, pre$fit, col=4)
lines(years0, pre$fit + 2*pre$se, col=4)
lines(years0, pre$fit - 2*pre$se, col=4)

## interpolation+extrapolating example over ages
## selected+summed data
year <- 1965
death1.4 <- sum(selectHMDdata("Swi", "Deaths",
                              "Males", 1:4, year))
death5.99 <- colSums(matrix(selectHMDdata("Swi",
                                         "Deaths",
                                         "Males",
                                         5:99, year),
                             nrow=5))
death <- c(death1.4, death5.99)
exposure1.4 <- sum(selectHMDdata("Swi", "Exposures",
                                "Males", 1:4, year))
exposure5.99 <- colSums(matrix(selectHMDdata("Swi",
                                             "Exposures",
                                             "Males",
                                             5:99, year),
                                nrow=5))
exposure <- c(exposure1.4, exposure5.99)
ages <- c(2.5, seq(7, 97, 5))

## fit

```

```

fit <- Mort1Dsmooth(x=ages, y=death, offset=log(exposure),
                  ndx=10)
## since the aim is to interpolate FEW data-points
## we use a large number of B-splines allows a precise,
## but not parsimonius, description

## predict the model for each age
newages <- 1:100
pre <- predict(fit, newdata=newages,
              se.fit=TRUE)
## plot log-rates
plot(ages, log(death/exposure), pch=16)
## add fitted log-rates from fit
## (every fifth age)
points(ages, log(fit$fitted/exposure),
      pch=16, col=2)
## add to the plot fit and 95% confidence interval
## (every single year)
points(newages, pre$fit, col=4, t="o")
lines(newages, pre$fit + 2*pre$se, col=4)
lines(newages, pre$fit - 2*pre$se, col=4)

```

---

predict.Mort2Dsmooth    *Predict Method for 2D P-splines Fits*

---

### Description

Obtains predictions, forecasts and optionally estimated standard errors of those predictions from a fitted Mort2Dsmooth object.

### Usage

```

## S3 method for class 'Mort2Dsmooth'
predict(object, newdata = NULL,
       type = c("link", "response"),
       se.fit = FALSE, ...)

```

### Arguments

object	an object of class "Mort2Dsmooth", usually, a result of a call to Mort2Dsmooth.
newdata	optionally, a list in which to look for x and/or y with which to predict. If omitted, the fitted linear predictors are used.
type	the type of prediction required. The default ("link") is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable.
se.fit	logical switch indicating if standard errors are required. Default: FALSE.
...	other predict parameters to be passed to predict. Not in used.

**Details**

If `newdata` is omitted the predictions are based on the data used for the fit. Note that, in common with other prediction functions, any offset supplied as an argument is always ignored when predicting, unlike offsets specified in modelling.

The user can provide also a single predictor (either `x` or `y`) within the argument `newdata`. The name within the list `newdata` must be named `x` and `y`.

Forecast is possible, therefore `newdata` can include values out of the range of the original `x` and `y`. See example below.

Interpolation is also feasible. See example below.

**Value**

If `se.fit = FALSE`, a matrix of predictions. If `se.fit = TRUE`, a list with components:

<code>fit</code>	a matrix of predictions.
<code>se.fit</code>	a matrix of estimated standard errors.

**Author(s)**

Carlo G Camarda

**See Also**

[Mort2Dsmooth](#) for computing `Mort2Dsmooth`.object.

**Examples**

```
## computing confidence intervals
## selected data
years <- 1980:2006
ages <- 80:100
death <- selectHMDdata("Denmark", "Deaths", "Females",
  ages = ages, years = years)
exposure <- selectHMDdata("Denmark", "Exposures", "Females",
  ages = ages, years = years)

## fit
fit <- Mort2Dsmooth(x=ages, y=years, Z=death,
  offset=log(exposure),
  method=3, lambdas=c(100,500))

## predict and computing standard errors
pre <- predict(fit, se.fit=TRUE)

## plotting over ages and years 95% confidence intervals
## !hard to distinguish between upper
## and lower confidence bounds
grid. <- expand.grid(x = ages, y = years, gr = 1:2)
grid.$lmx <- c(c(pre$fit - 2*pre$se.fit),
  c(pre$fit + 2*pre$se.fit))
wireframe(lmx ~ x * y, data = grid., groups = gr,
```

```

        scales = list(arrows = FALSE),
        drape = TRUE, colorkey = TRUE)

## plotting age 80 (first row)
plot(years, log(death[1,] / exposure[1,]),
      main="Mortality rates, log-scale.
          Danish females, age 80, 1980:2006")
lines(years, pre$fit[1,], lwd=2, col=2)
lines(years, pre$fit[1,] + 2*pre$se.fit[1,],
      lwd=2, col=2, lty=2)
lines(years, pre$fit[1,] - 2*pre$se.fit[1,],
      lwd=2, col=2, lty=2)

## forecasting example
newyears <- 1980:2020
newdata <- list(x=ages, y=newyears)
pre.for <- predict(fit, newdata=newdata, se.fit=TRUE)

## plot fitted+forecast log-rates for all ages over years
matplot(years, t(log(death/exposure)), pch=1, cex=0.5,
        col=rainbow(length(ages)),
        xlim=range(newyears),
        ylim=range(pre.for$fit),
        ylab="log-mortality")
matlines(newyears, t(pre.for$fit), lty=1, lwd=2,
        col=rainbow(length(ages)))

## plot selected ages over years, with 95% confidence intervals
whiA <- c(1,6,11)
matplot(years, t(log(death[whiA,]/exposure[whiA,])),
        pch=1,
        xlim=range(newyears),
        ylim=c(-3.3, -1.5),
        ylab="log-mortality")
matlines(newyears, t(pre.for$fit[whiA,]), lty=1, lwd=2)
matlines(newyears, t(pre.for$fit[whiA,]+
        2*pre.for$se.fit[whiA,]), lty=2)
matlines(newyears, t(pre.for$fit[whiA,]-
        2*pre.for$se.fit[whiA,]), lty=2)

## interpolation example
## compute log-death rates for
## each calendar month and calendar ages
newyears12 <- seq(1990, 2000, length=11*11)
newages12 <- seq(90, 100, length=11*11)
newdata12 <- list(x=newages12, y=newyears12)
pre12 <- predict(fit, newdata=newdata12, se.fit=TRUE)

## death rates in June 1995 at age 95.5
which.age <- which(newages12==95.5)
which.year <- which(newyears12==1995.5)
exp(pre12$fit[which.age, which.year] +

```

```
c(-2*pre12$se.fit[which.age, which.year],
  0, 2*pre12$se.fit[which.age, which.year]))
```

---

residuals.Mort1Dsmooth

*Extract 1D P-splines Model Residuals*


---

## Description

Extracting different types of residuals from a Mort1Dsmooth object.

## Usage

```
## S3 method for class 'Mort1Dsmooth'
residuals(object,
           type = c("deviance", "pearson",
                  "anscombe", "working"), ...)
```

## Arguments

object	an object of class "Mort1Dsmooth", usually, a result of a call to Mort1Dsmooth.
type	the type of residuals which should be returned. The alternatives are: "deviance" (default), "anscombe", "pearson" and "working".
...	Further arguments passed to or from other methods. Not in used. Not in use.

## Details

The references define the types of residuals.

The way of computing the residuals are described in Section 2.4 of McCullagh and Nelder's book. The working residuals are merely the differences between fitted and actual counts.

## Value

A vector of the selected type of residuals for each of the predictor in the Mort1Dsmooth object.

## Author(s)

Carlo G Camarda

## References

Davison, A. C. and Snell, E. J. (1991). *Residuals and diagnostics*. In: Statistical Theory and Modelling. In Honour of Sir David Cox, FRS, eds. Hinkley, D. V., Reid, N. and Snell, E. J., Chapman & Hall.

McCullagh P. and Nelder, J. A. (1989). *Generalized Linear Models*. London: Chapman and Hall.

**See Also**

[Mort1Dsmooth](#) for computing Mort1Dsmooth.object.

**Examples**

```
## selected data
years <- 1970:2006
death <- selectHMDdata("Denmark", "Deaths", "Females",
  ages = 60, years = years)
exposure <- selectHMDdata("Denmark", "Exposures", "Females",
  ages = 60, years = years)

## fit
fit <- Mort1Dsmooth(x=years, y=death, offset=log(exposure),
  method=3, lambda=1000)

## extracting residuals
devR <- resid(fit, type="deviance")
ansR <- resid(fit, type="anscombe")
peaR <- resid(fit, type="pearson")
worR <- resid(fit, type="working")
## summaries
summary(devR)
summary(ansR)
summary(peaR)
summary(worR)
```

---

residuals.Mort2Dsmooth

*Extract 2D P-splines Model Residuals*

---

**Description**

Extracting different types of residuals from a Mort2Dsmooth object.

**Usage**

```
## S3 method for class 'Mort2Dsmooth'
residuals(object,
  type = c("deviance", "pearson",
    "anscombe", "working"), ...)
```

**Arguments**

object	An object of class "Mort2Dsmooth", usually, a result of a call to Mort2Dsmooth.
type	The type of residuals which should be returned. The alternatives are: "deviance" (default), "anscombe", "pearson" and "working".
...	Further arguments passed to or from other methods. Not in used. Not in use.

**Details**

The references define the types of residuals.

The way of computing the residuals are described in Section 2.4 of McCullagh and Nelder's book.

The working residuals are merely the differences between fitted and actual counts.

**Value**

A matrix of the selected type of residuals over both the x and the y axes in the Mort2Dsmooth object.

**Author(s)**

Carlo G Camarda

**References**

Davison, A. C. and Snell, E. J. (1991). *Residuals and diagnostics*. In: Statistical Theory and Modelling. In Honour of Sir David Cox, FRS, eds. Hinkley, D. V., Reid, N. and Snell, E. J., Chapman & Hall.

McCullagh P. and Nelder, J. A. (1989). *Generalized Linear Models*. London: Chapman and Hall.

**See Also**

[Mort2Dsmooth](#) for computing Mort2Dsmooth.object.

**Examples**

```
## selected data
ages <- 30:80
years <- 1970:2006
death <- selectHMDdata("Switzerland", "Deaths",
                      "Males",
                      ages = ages, years = years)
exposure <- selectHMDdata("Switzerland", "Exposures",
                          "Males",
                          ages = ages, years = years)

## fit
fit <- Mort2Dsmooth(x=ages, y=years, Z=death,
                   offset=log(exposure),
                   method=3, lambdas=c(300,10))

## extracting residuals
devR <- resid(fit, type="deviance")
ansR <- resid(fit, type="anscombe")
pear <- resid(fit, type="pearson")
worR <- resid(fit, type="working")

## plotting deviance residuals over age and years
res.list <- list(ages=ages, years=years)
res.grid <- expand.grid(res.list)
res.grid$dev <- c(devR)
```

```
levelplot(dev~years*ages, res.grid,
          at=c(min(devR), -2, -1, 1, 2, max(devR)))
```

---

selectHMDdata                      *Selecting Data from the HMDdata Object*

---

### Description

Creates subset of the HMDdata object.

### Usage

```
selectHMDdata(country,
              data = c("Population", "Deaths", "Exposures", "Rates"),
              sex = c("Females", "Males", "Total"),
              ages = NULL, years = NULL)
```

### Arguments

country	character string for the country name.
data	character string showing type of demographic series: either "Population", "Deaths", "Exposures" or "Rates". Default value "Population".
sex	character string showing sex of demographic series: either "Females", "Males" or "Total". Default value "Females".
ages	vector of ages to extract from data. If it is NULL, all ages will be selected.
years	vector of years to extract from data. If it is NULL, all available years will be selected.

### Details

HMDdata are loaded with the package.

Country names is not case-sensitive and initials can be used. However, distinguishable country name is necessary. Type of data and sex are case-sensitive and initials can be used.

Available ages are always from 0 to 110, whereas years depends on the country. Warning messages will be provided in case of selected default values. Attributes about country, data and sex will be given to the outcome.

### Value

Matrix object from [HMDdata](#) with a subset of country, type of data, sex, ages and years.

### Author(s)

Carlo G Camarda

**See Also**

[HMDdata](#).

**Examples**

```
## "Complete" example
x <- 1950:2000
den50 <- selectHMDdata("Denmark", "Death", "Females", 50, x)
plot(x, den50, main="Danish female deaths at age 50")
## "Incomplete" example with warning
jap50 <- selectHMDdata("jap", "Pop", "F", 50)
## Wrong example (don't run)
## selectHMDdata("Sw", "Pop", "F", 50, 2000)
## Sw can stand for both Sweden and Switzerland
```

---

summary.Mort1Dsmooth    *Summary for Mort1Dsmooth objects*

---

**Description**

Summarizes the Poisson P-spline model fitted to a unidimensional data. It returns various settings and measures.

**Usage**

```
## S3 method for class 'Mort1Dsmooth'
summary(object, ...)
```

**Arguments**

object            an object of class "Mort1Dsmooth", usually, a result of a call to Mort1Dsmooth.  
...                further arguments passed to or from other methods.

**Details**

print.summary.Mort1Dsmooth tries to be smart about formatting settings, outcomes, etc. After the matched call, the function presents several outcomes of the model, such as AIC, BIC, effective dimension, selected smoothing parameter, overdispersion parameter and a summary of the deviance residuals. The last lines show specifications and control parameters of the fitted model.

**Value**

It produces an object of class summary.Mort1Dsmooth which contains exactly the same components of the associated Mort1Dsmooth object.

**Author(s)**

Carlo G Camarda

**See Also**

[Mort1Dsmooth](#).

**Examples**

```
## selected data
years <- 1970:2006
death <- selectHMDdata("Sweden", "Deaths", "Females",
  ages = 0, years = years)
exposure <- selectHMDdata("Sweden", "Exposures", "Females",
  ages = 0, years = years)

## fit
fit <- Mort1Dsmooth(x=years, y=death, offset=log(exposure),
  method=3, lambda=30)

## summary
summary(fit)
```

---

summary.Mort2Dsmooth    *Summary for Mort2Dsmooth objects*

---

**Description**

Summarizes the Poisson P-spline model fitted to a two-dimensional data. It returns various settings and measures.

**Usage**

```
## S3 method for class 'Mort2Dsmooth'
summary(object, ...)
```

**Arguments**

object            an object of class "Mort2Dsmooth", usually, a result of a call to Mort2Dsmooth.  
 ...              further arguments passed to or from other methods.

**Details**

print.summary.Mort2Dsmooth tries to be smart about formatting settings, outcomes, etc. After the matched call, the function presents several outcomes of the model, such as AIC, BIC, effective dimension, selected smoothing parameters, overdispersion parameter and a summary of the deviance residuals. The last lines show specifications and control parameters of the fitted model on both axes.

**Value**

It produces an object of class summary.Mort2Dsmooth which contains exactly the same components of the associated Mort2Dsmooth object.

**Author(s)**

Carlo G Camarda

**See Also**

[Mort2Dsmooth.](#)

**Examples**

```
## selected data
ages <- 10:60
years <- 1950:2006
death <- selectHMDdata("Sweden", "Deaths", "Males",
                      ages = ages, years = years)
exposure <- selectHMDdata("Sweden", "Exposures", "Males",
                        ages = ages, years = years)

## fit
fit <- Mort2Dsmooth(x=ages, y=years, Z=death,
                  offset=log(exposure),
                  method=3, lambdas=c(0.1, 1000))

## summary
summary(fit)
```

# Index

- \*Topic **datasets**
    - HMDdata, [3](#)
    - selectHMDdata, [42](#)
  - \*Topic **hplot**
    - plot.HMDdata, [29](#)
    - plot.Mort1Dsmooth, [30](#)
    - plot.Mort2Dsmooth, [32](#)
  - \*Topic **models**
    - Mort1Dsmooth, [4](#)
    - Mort2Dsmooth, [14](#)
    - predict.Mort1Dsmooth, [33](#)
    - predict.Mort2Dsmooth, [36](#)
    - residuals.Mort1Dsmooth, [39](#)
    - residuals.Mort2Dsmooth, [40](#)
    - summary.Mort1Dsmooth, [43](#)
    - summary.Mort2Dsmooth, [44](#)
  - \*Topic **package**
    - MortalitySmooth-package, [2](#)
  - \*Topic **regression**
    - Mort1Dsmooth, [4](#)
    - Mort2Dsmooth, [14](#)
    - Mort2Dsmooth\_se, [23](#)
    - MortSmooth\_BWB, [27](#)
    - plot.Mort1Dsmooth, [30](#)
    - plot.Mort2Dsmooth, [32](#)
    - predict.Mort1Dsmooth, [33](#)
    - predict.Mort2Dsmooth, [36](#)
    - residuals.Mort1Dsmooth, [39](#)
    - residuals.Mort2Dsmooth, [40](#)
    - summary.Mort1Dsmooth, [43](#)
    - summary.Mort2Dsmooth, [44](#)
  - \*Topic **smooth**
    - Mort1Dsmooth, [4](#)
    - Mort1Dsmooth\_checker, [9](#)
    - Mort1Dsmooth\_estimate, [10](#)
    - Mort1Dsmooth\_optimize, [11](#)
    - Mort1Dsmooth\_update, [13](#)
    - Mort2Dsmooth, [14](#)
    - Mort2Dsmooth\_checker, [19](#)
    - Mort2Dsmooth\_estimate, [20](#)
    - Mort2Dsmooth\_optimize, [22](#)
    - Mort2Dsmooth\_se, [23](#)
    - Mort2Dsmooth\_update, [24](#)
    - MortalitySmooth
      - (MortalitySmooth-package), [2](#)
    - MortalitySmooth-package, [2](#)
    - MortSmooth\_bbase, [25](#)
    - MortSmooth\_BcoefB, [26](#)
    - MortSmooth\_BWB, [24](#)
    - MortSmooth\_tpower, [26](#)
  - Mort2Dsmooth\_checker, [19](#)
  - Mort2Dsmooth\_estimate, [20](#)
  - Mort2Dsmooth\_optimize, [22](#)
  - Mort2Dsmooth\_se, [23](#)
  - Mort2Dsmooth\_update, [24](#)
  - MortSmooth\_bbase, [25](#)
  - MortSmooth\_BcoefB, [26](#)
  - MortSmooth\_BWB, [27](#)
  - MortSmooth\_tpower, [28](#)
  - plot.Mort1Dsmooth, [30](#)
  - plot.Mort2Dsmooth, [32](#)
  - predict.Mort1Dsmooth, [33](#)
  - predict.Mort2Dsmooth, [36](#)
  - residuals.Mort1Dsmooth, [39](#)
  - residuals.Mort2Dsmooth, [40](#)
  - summary.Mort1Dsmooth, [43](#)
  - summary.Mort2Dsmooth, [44](#)
- HMDdata, [3](#), [29](#), [30](#), [42](#), [43](#)
- Mort1Dsmooth, [4](#), [10–13](#), [31](#), [34](#), [40](#), [44](#)
- Mort1Dsmooth\_checker, [9](#)
- Mort1Dsmooth\_estimate, [10](#), [12](#)
- Mort1Dsmooth\_optimize, [7](#), [8](#), [11](#)
- Mort1Dsmooth\_update, [11](#), [12](#), [13](#)
- Mort2Dsmooth, [14](#), [20](#), [21](#), [23–25](#), [27](#), [28](#), [33](#), [37](#), [41](#), [45](#)
- Mort2Dsmooth\_checker, [19](#)
- Mort2Dsmooth\_estimate, [20](#), [23](#), [27](#), [28](#)
- Mort2Dsmooth\_optimize, [16](#), [22](#)
- Mort2Dsmooth\_se, [23](#)
- Mort2Dsmooth\_update, [21](#), [23](#), [24](#), [28](#)
- plot.HMDdata, [29](#)

plot.Mort1Dsmooth, 6, 8, 30  
plot.Mort2Dsmooth, 15, 17, 32  
predict.Mort1Dsmooth, 6, 8, 33  
predict.Mort2Dsmooth, 15, 17, 24, 36  
print.Mort1Dsmooth, 6  
print.Mort1Dsmooth (Mort1Dsmooth), 4  
print.Mort2Dsmooth, 15  
print.Mort2Dsmooth (Mort2Dsmooth), 14  
print.summary.Mort1Dsmooth  
    (summary.Mort1Dsmooth), 43  
print.summary.Mort2Dsmooth  
    (summary.Mort2Dsmooth), 44

residuals.Mort1Dsmooth, 6, 39  
residuals.Mort2Dsmooth, 15, 40

selectHMDdata, 4, 30, 42  
summary.Mort1Dsmooth, 6, 43  
summary.Mort2Dsmooth, 15, 44