Package ‘JOPS’

April 29, 2025

Type Package

Title Practical Smoothing with P-Splines

Version 0.2.0

Maintainer Paul Eilers <`p.eilers@erasmusmc.nl`>

Description Functions and data to reproduce all plots in the book ``Practical Smoothing. The Joys of P-splines" by Paul H.C. Eilers and Brian D. Marx (2021, ISBN:978-1108482950).

License GPL-2 | GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Depends R (>= 3.1.0), SpATS (>= 1.0-13)

Imports colorspace, MASS, boot, fds, rpart, ggplot2, fields, spam, stats, graphics, grDevices

NeedsCompilation no

Author Paul Eilers [aut, cre],
Brian Marx [aut],
Bin Li [aut],
Jutta Gampe [aut],
Maria Xose Rodriguez-Alvarez [aut]

Repository CRAN

Date/Publication 2025-04-29 08:10:05 UTC

Contents

`bbase` *Compute a B-spline basis matrix*

Description

Compute a B-spline basis matrix using evenly spaced knots.

Usage

```
bbase(x, xl = min(x), xr = max(x), nseg = 10, bdeg = 3)
```

Arguments

`x`	a vector of argument values, at which the B-spline basis functions are to be evaluated.
`xl`	the lower limit of the domain of x; default is `min(x)`.
`xr`	the upper limit of the domain of x; default is `max(x)`.
`nseg`	the number of equally sized segments between xl and xr; default is 10.
`bdeg`	the degree of the splines, usually 1, 2, or 3 (default).

Details

If `xl` is larger than `min(x)`, it will be adjusted to `min(x)` and a warning wil be given. If `xr` is smaller than `max(x)`, it will be adjusted to `max(x)` and a warning wil be given. The values of the design parameters `x`, `xl`, `xr`, `ndeg`, `bdeg` and `type = 'bbase'` are added to the list of attributes of the matrix.

Value

A matrix with `length(x)` rows and `nseg` + `bdeg` columns.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

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

Eilers, P.H.C. and B.D. Marx (2010). Splines, knots and penalties. Wiley Interdisciplinary Reviews: Computational Statistics. Wiley: NY. DOI: 10.1002/wics.125

Examples

```
# Compute and plot a B-spline basis matrix
x = seq(0, 360, by = 2)
B = bbase(x, 0, 360, nseg = 8, bdeg = 3)
matplot(x, B, type = 'l', lty = 1, lwd = 2, xlab = 'x', ylab = '')
```

`binit` *Translated number vector to bin index.*

Description

Translates number vector to bin index, given lower and upper limits of the domain and number of bins. A support function for (smoothing) histograms.

Usage

```
binit(x, xmin = min(x), xmax = max(x), nbin = 100)
```

Arguments

`x`	a numerical vector.
`xmin`	the lower limit of the domain.
`xmax`	the upper limit of the domain.
`nbin`	the number of bins (default=100).

Value

A list with components:

`xbin`	a vector of `length(x)` with elements giving the bin index.
`xgrid`	a vector of `length(nbin)` with the midpoints of the bins.
`nbin`	the number of bins.

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

`bone_data` *Spinal bone relative mineral density*

Description

Relative spinal bone mineral density measurements on 261 North American adolescents. Each value is the difference in `spnbmd` taken on two consecutive visits, divided by the average. The age is the average age over the two visits.

Usage

`data(bone_data)`

Format

A dataframe with four columns:

`idnum`	ID of the child
`age`	age
`gender`	male or female
`spnbmd`	Relative Spinal bone mineral density.

Source

https://web.stanford.edu/~hastie/ElemStatLearn/datasets/bone.data

References

Bachrach, L.K., Hastie, T., Wang, M.-C., Narasimhan, B., Marcus, R. (1999). Bone Mineral Acquisition in Healthy Asian, Hispanic, Black and Caucasian Youth. A Longitudinal Study. *J Clin Endocrinol Metab* 84, 4702-12.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

`cbase` *Compute a circular B-spline basis matrix*

Description

Computes a circular B-spline basis matrix using evenly spaced knots.

Usage

```
cbase(x, xl = min(x), xr = max(x), nseg = 10, bdeg = 3)
```

Arguments

`x`	a vector of argument values, at which the B-spline basis functions are to be evaluated.
`xl`	the lower limit of the domain of x; default is `min(x)`.
`xr`	the upper limit of the domain of x; default is `max(x)`.
`nseg`	the number of B-spline segments (default 10) between xl and xr.
`bdeg`	the degree of the basis, usually 1, 2, or 3 (default).

Details

If `xl` is larger than `min(x)`, it wil be adjusted to `min(x)` and a warning wil be given. If `xr` is smaller than `max(x)`, it wil be adjusted to `max(x)` and a warning wil be given.

The design parameters `x`, `xl`, `xr`, `ndeg`, `bdeg` and `type = 'cbase'` are added to the list of attributes.

In a circular basis, the B-splines are wrapped around the boundaries of the domain. Use a circular basis for data like directions or angles. It should be combined with a circular penalty matrix, as computed by `cdiff()`.

Value

A matrix with `length(x)` rows and `nseg` columns.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Eilers, P.H.C., Marx, B.D., and Durban, M. (2015). Twenty years of P-splines, *SORT*, 39(2): 149-186.

Examples

```
# Compute and plot a circular B-spline basis matrix
x = seq(0, 360, by = 2)
B = cbase(x, 0, 360, nseg = 8, bdeg = 3)
matplot(x, B, type = 'l', lty = 1, lwd = 2, xlab = 'x', ylab = '')
title('Note how the ends connect smoothly meet at boundaries' )
```

`cdiff` *Compute a second order circular differencing matrix*

Description

Compute difference matrix used for circular penalities.

Usage

```
cdiff(n)
```

Arguments

`n`	number of rows (and columns) of the square differencing matrix.

Value

A square matrix with `n` rows and columns.

Author(s)

Paul Eilers

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Eilers, P.H.C., Marx, B.D., and Durban, M. (2015). Twenty years of P-splines, *SORT*, 39(2): 149-186.

Examples

```
# Compare standard and circular differencing matrix
n = 8
D1 = diff(diag(n), diff = 2)
D2 = cdiff(n)
oldpar = par(no.readonly = TRUE)
on.exit(par(oldpar))
par(mfrow = c(1, 2))
image(t(D1))
```

```
title('Linear differencing matrix')
image(t(D2))
title('Circular differencing matrix')
```

`CGHsim` *Simulation of CGH data*

Description

A crude simulation of comparative genomic hybridization (CGH) data.

Usage

```
data(CGHsim)
```

Format

A data frame with 400 rows and two columns:

`y` Log R ratio

`x` Genomic position (but in fact the row number).

Source

The simulation program could not be located anymore. But the data have a very simple structure.

`clone_base` *Clone a B-spline basis for new x*

Description

Extract basis parameters from an existing B-splines basis matrix, and use them for computing a new basis at new values of `x`.

Usage

```
clone_base(B, x)
```

Arguments

`B`	a B-splines basis matrix, computed with `bbase()` or `cbase()`.
`x`	a vector of new argument values.

Details

If values in `x` are outside the domain used for computing `B`, they will be discarded, with a warning.

Value

A matrix with number of rows=`length(xnew)`.

Author(s)

Paul Eilers

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Eilers, P.H.C., Marx, B.D., and Durban, M. (2015). Twenty years of P-splines, *SORT*, 39(2): 149-186.

Examples

```
x = seq(0, 10, length = 20)
n = length(x)
y = sin(x / 2) + rnorm(n) * 0.2
B = bbase(x)
nb = ncol(B)
D = diff(diag(nb), diff = 2)
lambda = 1
a = solve(t(B) %*% B + lambda * t(D)%*% D, t(B) %*% y)
# Clone basis on finer grid
xg = seq(0, 10, length = 200)
Bg = clone_base(B, xg)
yg = Bg %*% a
plot(x, y)
lines(xg, yg, col = 'blue')
```

`Complaints` *Environmental complaints from the Rijnomond area of The Netherlands*

Description

Environmental complaints about odors from the Rijnmond region (near Rotterdam in the Netherlands) in 1988.

Usage

```
data(Complaints)
```

Format

A dataframe with two columns:

`freq`	The daily number of complaints.
`count`	The number of days the specific complaint frequency occurred.

Details

In 1988, the Rijnmond Environmental Agency registered approximately 20,000 complaints about odors from regional inhabitants.

Source

Personal information from Paul Eilers.

Examples

```
plot(Complaints$freq, Complaints$count, type = 'h',
xlab = 'Number of complaints per day', ylab = 'Frequency')
```

`count2d` *Create a matrix of counts.*

Description

Count the number of occurrences of pairs of positive integers in two vectors, producing a matrix.

Usage

```
count2d(xb, yb, nb)
```

Arguments

`xb`	a vector of integers.
`yb`	a vector of integers.
`nb`	a vector of length 2 that provides the number of bins for the 2D histogram on `x` and `y`.

Details

This function builds a two-dimensional histogram, based on two two vectors of bin numbers (obtained with `binit`). Rows where `x[i]` > `nb[1]` or `y[i]` > `nb[2]` are discarded without a warning.

Value

A matrix with `nb[1]` rows and `nb[2]` columns with counts. It serves as the input for two-dimensional histogram smoothing.

`dev_calc` *Deviance calculation for GLM P-spline fitting.*

Description

Calculates the deviance and returns the ML estimated dispersion parameter for a variety of response distributions for P-spline fitting within the GLM framework.

Usage

```
dev_calc(
  family = "gaussian",
  y,
  mu,
  m_binomial = 0 * y + 1,
  r_gamma = 0 * y + 1
)
```

Arguments

`family`	the response distribution, e.g. `"gaussian"`, `"binomial"`, `"poisson"`, `"Gamma"` distribution. Quotes are needed; default `"family = gaussian"`.
`y`	the glm response vector of length `m`.
`mu`	the P-spline estimated mean for the glm response vector of length `m`.
`m_binomial`	a vector of binomial trials having `length(y)`, when `family = "binomial"`. Default is 1 vector.
`r_gamma`	a vector of gamma shape parameters, when `family = "Gamma"`. Default is 1 vector.

Value

A list with two fields:

`dev`	the estimated deviance.
`dispersion_parm`	the ML estimated dispersion parameter.

`Disks` *Prices of hard disk drives*

Description

Prices and capacities of hard disk drives, as advertised in a Dutch computer monthly in 1999. Prices are given in Dutch guilders; the Euro did not yet exist.

Usage

```
data(Disks)
```

Format

A dataframe with six columns:

`Year`	1999-2000
`Month`	month, 1-12
`Size`	capacity in Gb
`Buffer`	buffer size (Mb)
`RPM`	rotating speed (rpm)
`PriceDG`	in Dutch Guilders, divide by 2.2 for Euro.

Source

Personal information from Paul Eilers.

`ECG` *A section of an ECG (electrocardiogram)*

Description

The data set includes two signals, respiration and the ECG. Both signals are distorted by strong 60Hz interference from the mains power.

Usage

```
data(ECG)
```

Format

A data frame with three columns:

time	time in seconds
resp	respiration, arbitrary units
ecg	ECG, arbitrary units.

Source

https://physionet.org/content/fantasia/1.0.0/

References

Iyengar N, Peng C-K, Morin R, Goldberger AL, Lipsitz LA. Age-related alterations in the fractal scaling of cardiac interbeat interval dynamics. *Am J Physiol*, 1996; 271: 1078-1084.

Standard citation for PhysioNet: Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals (2003). Circulation. 101(23):e215-e220.

`ethanol` *Ethanol data*

Description

The ethanol data frame contains 88 sets of measurements for variables from an experiment in which ethanol was burned in a single cylinder automobile test engine.

This data set was part of the package SemiPar, which is no longer available at CRAN in compiled form. Copied to JOPS with slight modifications of the documentation.

Usage

```
data(ethanol)
```

Format

This data frame contains the following columns:

NOx the concentration of nitric oxide (NO) and nitrogen dioxide (NO2) in engine exhaust, normalized by the work done by the engine.

C the compression ratio of the engine

E the equivalence ratio at which the engine was run – a measure of the richness of the air/ethanol mix.

Source

Brinkman, N.D. (1981). Ethanol fuel – a single-cylinder engine study of efficiency and exhaust emissions. *SAE transactions* Vol. 90, No 810345, 1410–1424.

References

Ruppert, D., Wand, M.P. and Carroll, R.J. (2003)
Semiparametric Regression Cambridge University Press.
`https://drcarroll.wpengine.com/semiregbook/`

Examples

```
library(JOPS)
data(ethanol)
pairs(ethanol)
```

---

fitampl *Fit amplitude coeffcients in the bundle model for expectiles*

Description

There are two functions for fitting the expectile bundle model, one for estimating asymmetry parameters (`fitasy`), the other for estimating the amplitude function, `fitampl`, this function. See the details below.

Usage

```
fitampl(y, B, alpha, p, a, pord = 2, lambda)
```

Arguments

`y`	a response vector.
`B`	a proper B-spline basis matrix, see `bbase()`.
`alpha`	a vector of B-spline coefficients.
`p`	a vector of asymmetries.
`a`	a vector of asymmetry parameters.
`pord`	the order of the difference penalty, default is 2.
`lambda`	the positive tuning parameter for the penalty.

Details

The expectile bundle model determines a set of expectile curves for a point cloud with data vectors `x` and `y`, as $\psi_j x_i = a_j g(x_i)$. Here a_j is the asymmetry parameter corresponding to a given asymmetry p_j. A vector of asymmetries with all $0 < p_j < 1$ is specified by the user.

The asymmetric least squares objective function is

$$\sum_j \sum_i w_{ij}(y_i - \sum_j a_j g_j(x_i))^2.$$

The function $g(\cdot)$ is called the amplitude. The weights depend on the residuals:

$$w_{ij} = p_j$$

if $y_i > a_j g(x_i)$ and $w_{ij} = 1 - p_j$ otherwise.

The amplitude function is a sum of B-splines with coefficients `alpha`. There is no direct solution, so `alpha` and the asymmetry parameters `a` must be updated alternatingly. See the example.

Value

a vector of estimated B-spline coefficients.

Note

This is a simplification of the model described in the reference. There is no explict term for the trend.

Author(s)

Paul Eilers

References

Schnabel, S.K. and Eilers, P.H.C. (2013) A location-scale model for non-crossing expectile curves. *Stat* 2: 171–183.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
# Get the data
data(bone_data)
x = bone_data$age
y = bone_data$spnbmd
m <- length(x)

# Set asymmetry levels
p = c(0.005, 0.01, 0.02, 0.05, 0.2, 0.5, 0.8, 0.9, 0.95, 0.98, 0.99, 0.995)
np <- length(p)

# Set P-spline parameters
x0 <- 5
x1 <- 30
ndx <- 20
bdeg <- 3
pord <- 2

# Compute bases
B <- bbase(x, x0, x1, ndx, bdeg)
xg <- seq(from = min(x), to = max(x), length = 100)
Bg <- clone_base(B, xg)
n <- ncol(B)

lambda = 1
alpha <- rep(1,n)
a = p
for (it in 1:20){
  alpha <- fitampl(y, B, alpha, p, a, pord, lambda)
  alpha <- alpha / sqrt(mean(alpha ^ 2))
  anew <- fitasy(y, B, alpha, p, a)
```

```
  da = max(abs(a - anew))
  a = anew
  cat(it, da, '\n')
     if (da < 1e-6) break
}

# Compute bundle on grid
ampl <- Bg %*% alpha
Z <- ampl %*% a

# Plot data and bundle
plot(x, y, pch = 15, cex = 0.7, col = 'grey', xlab = 'Age', ylab = 'Density')
cols = colorspace::rainbow_hcl(np, start = 10, end = 350)
matlines(xg, Z, lty = 1, lwd = 2, col = cols)
```

`fitasy` *Fit asymmetry parameters in the expectile bundle model*

Description

There are two functions for fitting the expectile bundle model, the present one for estimating asymmetry parameters (`fitasy`), the other for estimating the amplitude function, `fitampl`. See the details below.

Usage

```
fitasy(y, B, b, p, c0)
```

Arguments

`y`	a response vector.
`B`	a proper B-spline basis matrix, see `bbase()`.
`b`	a vector of B-spline coefficients.
`p`	a vector of asymmetries with values between 0 and 1.
`c0`	a vector.

Details

The expectile bundle model determines a set of expectile curves for a point cloud with data vectors `x` and `y`, as $\psi_j x_i = a_j g(x_i)$. Here a_j is the asymmetry parameter corresponding to a given asymmetry p_j. A vector of asymmetries with all $0 < p_j < 1$ is specified by the user.

The asymmetric least squares objective function is

$$\sum_j \sum_i w_{ij}(y_i - \sum_j a_j g_j(x_i))^2.$$

The function $g(\cdot)$ is called the amplitude. The weights depend on the residuals:

$$w_{ij} = p_j$$

if $y_i > a_j g(x_i)$ and $w_{ij} = 1 - p_j$ otherwise.

The amplitude function is a sum of B-splines with coefficients `alpha`. There is no direct solution, so `alpha` and the asymmetry parameters `a` must be updated alternatingly. See the example.

Value

a vector of estimated asymmetry parameters .

Note

This is a simplification of the model described in the reference. There is no explict term for the trend.

Author(s)

Paul Eilers

References

Schnabel, S.K. and Eilers, P.H.C. (2013) A location-scale model for non-crossing expectile curves. *Stat* 2: 171–183.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
# Get the data
data(bone_data)
x = bone_data$age
y = bone_data$spnbmd
m <- length(x)

# Set asymmetry levels
p = c(0.005, 0.01, 0.02, 0.05, 0.2, 0.5, 0.8, 0.9, 0.95, 0.98, 0.99, 0.995)
np <- length(p)

# Set P-spline parameters
x0 <- 5
x1 <- 30
ndx <- 20
bdeg <- 3
pord <- 2

# Compute bases
B <- bbase(x, x0, x1, ndx, bdeg)
xg <- seq(from = min(x), to = max(x), length = 100)
Bg <- clone_base(B, xg)
```

```
n <- ncol(B)

lambda = 1
alpha <- rep(1,n)
a = p
for (it in 1:20){
  alpha <- fitampl(y, B, alpha, p, a, pord, lambda)
  alpha <- alpha / sqrt(mean(alpha ^ 2))
  anew <- fitasy(y, B, alpha, p, a)
  da = max(abs(a - anew))
  a = anew
  cat(it, da, '\n')
     if (da < 1e-6) break
}

# Compute bundle on grid
ampl <- Bg %*% alpha
Z <- ampl %*% a

# Plot data and bundle
plot(x, y, pch = 15, cex = 0.7, col = 'grey', xlab = 'Age', ylab = 'Density')
cols = colorspace::rainbow_hcl(np, start = 10, end = 350)
matlines(xg, Z, lty = 1, lwd = 2, col = cols)
```

`G519C18` *Chromosome G519C18 data*

Description

An extract of the data set `G519` in the Bioconductor package `Vega`, for chromosome 18.

Usage

```
data(G519C18)
```

Format

A dataframe with two columns:

- `y` Probe position
- `x` Log R Ratio.

References

https://www.bioconductor.org/packages/release/bioc/html/Vega.html

Examples

```
plot(G519C18$x, G519C18$y, type = 'l', ylab = 'LRR', xlab = 'Position', main = 'Chromosome 18')
```

`Greece_deaths` *Deaths in Greece in 1960.*

Description

Deaths in Greece in 1960.

Usage

```
data(Greece_deaths)
```

Format

A dataframe with three columns:

`Age`	0 - 85
`Male`	male deaths
`Female`	female deaths.

Details

All counts for ages above 84 have been grouped to one number for age 85.

Source

Personal information from Aris Perperoglou.

`Hepatitis` *Prevalence of Hepatitis among a sample of Bulgarian males.*

Description

Prevalence of Hepatitis among a sample of Bulgarian males.

Usage

```
data(Hepatitis)
```

Format

A data frame with three columns:

`Age`	years
`Infected`	number of infected persons
`Sampled`	number of sampled persons.

Source

Table 2 in Keiding (1991).

References

N. Keiding (1991) Age-Specific Incidence and Prevalence: A Statistical Perspective. *JRSS-A* 154, 371-396.

`hist2d` *Compute a 2D histogram*

Description

Compute a two-dimesnional histogram from two vectors (of the same length), `x` and `y`.

Usage

```
hist2d(x, y, nb = c(100, 100), xlim = range(x), ylim = range(y))
```

Arguments

`x`	a numeric vector.
`y`	a numeric vector of the same length as `x`.
`nb`	a vector `c(nbx, nby)`, or a scalar `nb`, providing the number of bins for `x`, and `y`; default is 100; see details.
`xlim`	a vector `c(xmin, xmax)` containing the limits of the domain of `x`; default `range(x)`.
`ylim`	a vector `c(ymin, ymax)` containing the limits of the domain of `y`; default `range(y)`.

Details

If `nb` is scalar, it is extended to `c(nb, nb)`, so that both dimensions will have the same number of bins.

Elements of `x` (`y`) that fall outside the range specified by `xlim` (`ylim`) are not counted.

Value

A list with components:

`H`	a matrix of dimension `nbx` by `nby` containing bin counts.
`xgrid`	a vector of length `nbx` representing centers of the bins for `x`.
`ygrid`	a vector of length `nby` representing centers of the bins for `y`.
`xbin`	a vector giving the bin number of each element of `x`.
`ybin`	a vector giving the bin number of each element of `y`.

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
data(faithful)
x = faithful$eruptions
y = faithful$waiting
C = hist2d(x, y, c(50,50))
image(C$xgrid, C$ygrid, C$H, xlab='Eruption length (min)', ylab='Waiting time (min)')
title('Old Faithful geyser')
```

`hist2dsm` *Smooth a 2D histogram*

Description

Fit a 2D smooth P-spline surface to a matrix of counts, assuming Poisson distributed observations.

Usage

```
hist2dsm(
  Y,
  nsegx = 10,
  nsegy = nsegx,
  bdeg = 3,
  lambdax = 10,
  lambday = lambdax,
  dx = 3,
  dy = dx,
  Mu = Y + 0.01,
  kappa = 1e-04,
  tol = 1e-05
)
```

Arguments

`Y`	a matrix of counts.
`nsegx`	the number of knots along `x` (default=10).
`nsegy`	the number of evenly spaced knots along `y` for Tensor product B-spline basis (default=10).
`bdeg`	the degree of the basis, default is 3.
`lambdax`	the positive number for the tuning parameter along `x`.
`lambday`	the positive number for the tuning parameter along `y`.

`dx`	the order of the difference penalty along `x`, default is 3.
`dy`	the order of the difference penalty along `y`, default is 3.
`Mu`	the initialization of the mean (default `Y + 0.01`).
`kappa`	a (small, positive) number for ridge tuning parameter to stabilize estimation (default `1e-4`).
`tol`	the convergence criterion (default `1e-5`).

Value

A list with elements:

`ed`	the effective dimension of the smooth 2D surface.
`Mu`	a matrix with the smooth estimates, with dimensions of `dim(Y)`
`pen`	the numerical value of the penalty.

Author(s)

Paul Eilers

References

Eilers, P.H.C., Marx, B.D., and Durban, M. (2015). Twenty years of P-splines, *SORT*, 39(2): 149-186.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
x = faithful$eruptions
y = faithful$waiting
h = hist2d(x, y, c(100, 100))
sm = hist2dsm(h$H, nsegx = 25, nsegy = 25, bdeg = 3, lambdax = 10, lambday = 10)
image(h$xgrid, h$ygrid, sm$Mu, xlab = 'Eruption length (min)',
      ylab = 'Waiting time (min)', main = 'Old Faithful')
```

`indiumoxide` *An X-ray diffractogram.*

Description

An X-ray diffractogram.

Usage

```
data(indiumoxide)
```

Format

A matrix with two columns:

`angle`	the angles (degrees) of diffraction
`count`	corresponding photon counts.

Details

An X-ray diffractogram of Indium-Tin oxide.

These data have been taken from the source of package `Diffractometry`, which is no longer available from CRAN in binary form.

Source

P.L. Davies, U. Gather, M. Meise, D. Mergel, T. Mildenberger (2008). Residual based localization and quantification of peaks in x-ray diffractograms, *Annals of Applied Statistics*, Vol. 2, No. 3, 861-886.

Examples

```
angle = indiumoxide[,1]
photon = indiumoxide[,2]
plot(angle, type = 'l', photon, xlab = 'Angle', ylab = 'Photon count')
```

`inverse_link` *Inverse link function, used for GLM fitting.*

Description

Inverse link function, used for GLM fitting.

Usage

```
inverse_link(x, link)
```

Arguments

`x`	scalar, vector, or matrix input.
`link`	the link function, one of `"identity"`, `"log"`, `"sqrt"`, `"logit"`, `"probit"`, `"cloglog"`, `"loglog"`, `"reciprocal"`; quotes are needed (default `"identity"`).

Value

The inverse link function applied to `x`. If `link` is not in the above list of allowed names, `NULL` will be returned.

`JOPS` *Joys of P-Splines*

Description

A package for working with and learning about P-splines. P-splines combine B-splines with discrete penalties to build a very flexible and effective smooth models. They can handle non-normal data in the style of generalized linear models.

This package provides functions for constructing B-spline bases and penalty matrices. It solves the penalized likelihood equations efficiently.

Several methods are provided to determine the values of penalty parameters automatically, using cross-validation, AIC, mixed models or fast Bayesian algorithms.

This package is a companion to the book by Eilers and Marx (2021). The book presents the underlying theory and contains many examples and the code `R` for each example is available on the website `https://psplines.bitbucket.io`

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

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

`JOPS_colors` *Custom color ramp.*

Description

Custom color ramp.

Usage

```
JOPS_colors(n)
```

Arguments

`n`	number of steps.

Value

custom color ramp.

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

`JOPS_point` *Themeing functions used to unify ggplot features*

Description

Custom size and color of points.

Usage

```
JOPS_point(s_size = 1.5)
```

Arguments

`s_size`	point size parameter for ggplot2 (default = 1.5).

Value

themeing function for ggplot2 features.

`JOPS_theme` *Custom theme for ggplot*

Description

Set a ggplot theme in black and white, with centered titles.

Set a ggplot theme in black and white, with centered titles.

Usage

```
JOPS_theme(h_just = 0.5)
```

```
JOPS_theme(h_just = 0.5)
```

Arguments

`h_just`	horizontal justification for ggplot2.

Value

custom theme for ggplot.

Custom theming function used to unify ggplot features.

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

`LAPS_dens` *Bayesian density estimation*

Description

Bayesian density estimation with P-splines and Laplace approximation.

Usage

```
LAPS_dens(B, P, y, loglambdas, tol = 1e-05, mon = FALSE)
```

Arguments

`B`	matrix (`m` by `n`) with B-spline basis, see `bbase()`.
`P`	penalty matrix (`n` by `n`).
`y`	vector (length `m`) of counts, usually a histogram.
`loglambdas`	a vector of values of logarithms of `lambda` to explore.
`tol`	convergence tolerance (relative change in coefficients), default `1e-5`.
`mon`	TRUE or FALSE to monitor the iteration history (default FALSE).

Details

The B-spline basis should be based on the midpoints of the histogram bins. See the example below. This function is based on the paper of Gressani and Lambert (2018) and code input by Oswaldo Gressani.

Value

A list with elements:

`alpha`	P-spline coefficients of length `n`.
`weights`	weights from the Laplace approximation, which sum to 1 and are the same length as `loglambdas`.
`mu`	a vector of length `m` of expected values.
`Cov`	covariance matrix (`m` by `m`) of `log(mu)`.
`lambda`	the penalty parameter.
`ed`	the effective model dimension.

Author(s)

Paul Eilers

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Gressani, O. and Lambert, P. (2018). Fast Bayesian inference using Laplace approximations in a flexible promotion time cure model based on P-splines. *Computational Statistics and Data Analysis* 124, 151-167.

Examples

```
# Smoothing a histogram of Old Faithful eruption durations
data(faithful)
durations = faithful[, 1]  # Eruption length

# Histogram with narrow bin widths
bw = 0.05
hst = hist(durations, breaks = seq(1, 6, by = bw), plot = TRUE)
x = hst$mids
y = hst$counts

# B-spline basis matrices, for fitting and plotting
nseg = 30
B = bbase(x, nseg = nseg)
xg = seq(min(x), max(x), by = 0.01)
Bg = bbase(xg, nseg = nseg)
n = ncol(B)

# Penalty matrix
D2 = diff(diag(n), diff = 2)
P2 = t(D2) %*% D2

# Fit the model
loglambs = seq(-1, 2, by = 0.05)
laps2 = LAPS_dens(B, P2, y, loglambs, mon = FALSE)
fhat2 = exp(Bg %*% laps2$alpha)
lines(xg, fhat2, col = "blue", lwd = 2)
```

`lidar` *LIDAR data*

Description

The `lidar` data frame has 221 observations from a light detection and ranging (LIDAR) experiment.

This data set was part of the package SemiPar, which is no longer available at CRAN in compiled form. Copied to JOPS with slight modifications of the documentation.

Usage

```
data(lidar)
```

Format

This data frame contains the following columns:

range distance travelled before the light is reflected back to its source.

logratio logarithm of the ratio of received light from two laser sources.

Source

Sigrist, M. (Ed.) (1994). *Air Monitoring by Spectroscopic Techniques* (Chemical Analysis Series, vol. 197). New York: Wiley.

References

Ruppert, D., Wand, M.P. and Carroll, R.J. (2003)
Semiparametric Regression Cambridge University Press.
https://drcarroll.wpengine.com/semiregbook/

Examples

```
library(JOPS)
data(lidar)
attach(lidar)
plot(range,logratio)
```

`Mixture` *Mixture Data*

Description

The mixture data were obtained in an unpublished experiment in 2001 by Zhenyu Wang at University of Amsterdam, under the supervision of Age Smilde. We are grateful for the permission to use the data.

Usage

```
data(Mixture)
```

Format

A list consisting of the following:

`fractions` a 34 x 3 matrix of mixure fractions (rows sum to unity): `Water` (subboiled demi water (self made)), `1,2ethanediol` (99.8% Sigma-Aldrich Germany), `3amino1propanol` (99% Merk Schuchardt Germany)

`xspectra` spectra array, 34 (observations) x 401 (wavelenths channels) x 12 (temperatures (C): 30, 35, 37.5, 40, 45, 47.5, 50, 55, 60, 62.5, 65, 70)

`wl` wavelengths for the spectra, 700 to 1100 (nm), by 1nm.

Details

The following instruments and chemicals were used in the experiment: HP 8453 spectrophotometer (Hewlett-Packard, Palo Alto, CA); 2cm closed quartz cuvette with glass thermostatable jacket; Pt-100 temperature sensor; Neslab microprocessor EX-111 circulator bath; UV-visible Chemstation software (Rev A.02.04) on a Hewlett-Packard Vectra XM2 PC.

References

Eilers, P. H. C., and Marx, B. D. (2003). Multivariate calibration with temperature interaction using two-dimensional penalized signal regression. *Chemometrics and Intellegent Laboratory Systems*, 66, 159–174.

Marx, B. D., Eilers, P. H. C., and Li, B. (2011). Multidimensional single-index signal regression. *Chemometrics and Intelligent Laboratory Systems*, 109(2), 120–130. [see the Appendix within]

Zhenyou Wang and Age Smilde, Univeristy of Amsterdam, The Netherlands. Personal communication.

`ova` *Ovarian cancer data*

Description

Ovarian cancer data

Usage

`data(ova)`

Format

A dataframe with five columns:

`Diameter`

`FIGO`

`Karnofsky`

`time`

`death` death

Source

tba

`pclm` *Fit a composite link model*

Description

Fit a smooth latent distribution using the penalized composite link model (PCLM).

Usage

```
pclm(y, C, B, lambda = 1, pord = 2, itmax = 50, show = FALSE)
```

Arguments

`y`	a vector of counts, length `m`.
`C`	a composition matrix, `m` by `q`.
`B`	a B-spline basis matrix, `q` by `n`.
`lambda`	the penalty parameter.
`pord`	the the order of the difference penalty (default = 2).
`itmax`	the maximum number of iterations (default = 50).
`show`	Set to TRUE or FALSE to display iteration history (default = FALSE).

Details

The composite link model assumes that $E(y) = \mu = C\exp(B\alpha)$, where $\exp(B\alpha)$ is a latent discrete distribution, usually on a finer grid than that for y.

Note that `sum(gamma) == sum(mu)`.

Value

A list with the following items:

`alpha`	the estimated B-spline coefficients, length `n`.
`gamma`	the estimated latent distribution, length `q`.
`mu`	estimated values of `y`, length `m`.
`dev`	the deviance of the model.
`ed`	the effective model dimension.
`aic`	Akaike's Information Criterion.

Author(s)

Paul Eilers and Jutta Gampe

References

Eilers, P. H. C. (2007). III-posed problems with counts, the composite link model and penalized likelihood. *Statistical Modelling*, 7(3), 239–254.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
# Left and right boundaries, and counts, of wide intervals of the data
cb <- c( 0, 20, 30, 40, 50, 60)
ce <- c(20, 30, 40, 50, 60, 70)
y <- c(79, 54, 19, 1, 1, 0)

# Construct the composition matrix
m <- length(y)
n <- max(ce)
C <- matrix(0, m, n)
for (i in 1:m) C[i, cb[i]:ce[i]] <- 1

mids = (cb + ce) / 2 - 0.5
widths = ce - cb + 1
dens = y / widths / sum(y)
x = (1:n) - 0.5
B = bbase(x)
fit = pclm(y, C, B, lambda = 2, pord = 2, show = TRUE)
gamma = fit$gamma / sum(fit$gamma)
# Plot density estimate and data
plot(x, gamma, type = 'l', lwd = 2, xlab = "Lead Concentration", ylab = "Density")
rect(cb, 0, ce, dens, density = rep(10, 6), angle = rep(45, 6))
```

`plot.ps2dglm` *Plotting function for* `ps2DGLM`

Description

Plotting function for 2D P-spline (GLM) smooothing (using `ps2DGLM` with `class` `ps2dglm`).

Usage

```
## S3 method for class 'ps2dglm'
plot(x, ..., xlab = " ", ylab = " ", Resol = 100, se = 2)
```

Arguments

`x`	the P-spline object, usually from `ps2DGLM`.
`...`	other parameters.
`xlab`	label for the x-axis, e.g. "my x" (quotes required).

`ylab`	label for the y-axis, e.g. "my y" (quotes required).
`Resol`	resolution for plotting, default `Resol = 100`.
`se`	a scalar, e.g. `se = 2` to produce twice se surfaces, set `se` > 0 (or set `se = 0` to supress).

Value

`Plot`	a plot of the mean (inverse link) 2D P-spline (GLM) smooth surface.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Eilers, P.H.C., Marx, B.D., and Durban, M. (2015). Twenty years of P-splines, *SORT*, 39(2): 149-186.

Examples

```
library(fields)
library(JOPS)
# Extract data
library(rpart)
Kyphosis <- kyphosis$Kyphosis
Age <- kyphosis$Age
Start <- kyphosis$Start
y <- 1 * (Kyphosis == "present") # make y 0/1
fit <- ps2DGLM(
  Data = cbind(Start, Age, y),
  Pars = rbind(c(1, 18, 10, 3, .1, 2), c(1, 206, 10, 3, .1, 2)),
  family = "binomial"
)
plot(fit, xlab = "Start", ylab = "Age")
#title(main = "Probability of Kyphosis")
```

`plot.ps2dnormal` *Plotting function for* `ps2DNormal`

Description

Plotting function for 2D P-spline smooothing (using `ps2DNormal` with `class` `ps2dnormal`).

Usage

```
## S3 method for class 'ps2dnormal'
plot(x, ..., xlab = " ", ylab = " ", Resol = 100)
```

Arguments

`x`	the P-spline object, usually from `ps2DNormal`.
`...`	other parameters.
`xlab`	label for the x-axis, e.g. "my x" (quotes required).
`ylab`	label for the y-axis, e.g. "my y" (quotes required).
`Resol`	resolution for plotting, default `Resol = 100`.

Value

`Plot`	a plot of the smooth 2D P-spline smooth surface.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Eilers, P.H.C., Marx, B.D., and Durban, M. (2015). Twenty years of P-splines, *SORT*, 39(2): 149-186.

Examples

```
library(fields)
library(spam)
library(JOPS)

# Get the data
data(ethanol)
x <- ethanol$C
y <- ethanol$E
z <- ethanol$NOx

# Set parameters for domain
xlo <- 7
xhi <- 19
ylo <- 0.5
yhi <- 1.25

# Set P-spline parameters, fit and compute surface
xpars <- c(xlo, xhi, 10, 3, 3, 1)
ypars <- c(ylo, yhi, 10, 3, 3, 1)
Pars1 <- rbind(xpars, ypars)
fit <- ps2DNormal(cbind(x, y, z), Pars = Pars1)
plot(fit, xlab = "C", ylab = "E")
```

`plot.ps2dsignal` *Plotting function for* `ps2DSignal`

Description

Plotting function for 2D P-spline signal regression coefficients (using `ps2DSignal` with `class` `ps2dsignal`). Although standard error surface bands can be comuputed they are intentially left out as they are not interpretable, and there is generally little data to steer such a high-dimensional parameterization.

Usage

```
## S3 method for class 'ps2dsignal'
plot(x, ..., xlab = " ", ylab = " ", Resol = 200)
```

Arguments

`x`	the P-spline object, usually from `ps2DSignal`.
`...`	other parameters.
`xlab`	label for the x-axis, e.g. "my x" (quotes required).
`ylab`	label for the y-axis, e.g. "my y" (quotes required).
`Resol`	Resolution of bgrid (default `Resol = 200`).

Value

`Plot`	a plot of the 2D P-spline signal coefficent surface.

Author(s)

Paul Eilers and Brian Marx

References

Marx, B.D. and Eilers, P.H.C. (2005). Multidimensional penalized signal regression, *Technometrics*, 47: 13-22.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(fields)
library(JOPS)

# Get the data
x0 <- Sugar$X
x0 <- x0 - apply(x0, 1, mean) # center Signal
y <- as.vector(Sugar$y[, 3]) # Response is Ash
```

```
# Inputs for two-dimensional signal regression
nseg <- c(7, 37)
pord <- c(3, 3)
min_ <- c(230, 275)
max_ <- c(340, 560)
M1_index <- rev(c(340, 325, 305, 290, 255, 240, 230))
M2_index <- seq(from = 275, to = 560, by = .5)
p1 <- length(M1_index)
p2 <- length(M2_index)

# Fit optimal model based on LOOCV
opt_lam <- c(8858.6679, 428.1332) # Found via svcm
Pars_opt <- rbind(
 c(min_[1], max_[1], nseg[1], 3, opt_lam[1], pord[1]),
 c(min_[2], max_[2], nseg[2], 3, opt_lam[2], pord[2]))

fit <- ps2DSignal(y, x0, p1, p2, "unfolded", M1_index, M2_index,
       Pars_opt, int = FALSE, ridge_adj = 1e-4 )

# Plotting coefficient image
plot(fit)
```

`plot.pspfit` *Plotting function for* `psNormal`, `psPoisson`, `psBinomial`

Description

Plotting function for P-spline smooth with normal, Poisson, or binomial responses (`class pspfit`), with or without standard error bands.

Usage

```
## S3 method for class 'pspfit'
plot(x, ..., se = 2, xlab = "", ylab = "", col = "black", pch = 1)
```

Arguments

`x`	the P-spline object, usually from psNormal, psPoisson, psBinomial.
`...`	other parameters.
`se`	a scalar, e.g. `se = 2` to produce twice se bands, set se > 0 (or set se=0 to supress).
`xlab`	label for the x-axis.
`ylab`	label for the y-axis.
`col`	color for points.
`pch`	point character.

Value

`Plot`	a plot of the mean (inverse link) smoothed normal, Poisson, or binomial responses, with or without se bands.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Eilers, P.H.C., Marx, B.D., and Durban, M. (2015). Twenty years of P-splines, *SORT*, 39(2): 149-186.

Examples

```
library(JOPS)
#Extract data
library(MASS)
# Get the data
data(mcycle)
x = mcycle$times
y = mcycle$accel
fit1 = psNormal(x, y, nseg = 20, bdeg = 3, pord = 2, lambda = .8)
plot(fit1, se = 2, xlab = "time (ms)", ylab = "accel")

library(JOPS)
library(boot)
# Extract the data
Count = hist(boot::coal$date, breaks=c(1851:1963), plot = FALSE)$counts
Year = c(1851:1962)
xl = min(Year)
xr = max(Year)

# Poisson smoothing
nseg = 20
bdeg = 3
fit1=psPoisson(Year, Count, xl, xr, nseg, bdeg, pord = 2,
lambda = 1)
names(fit1)
plot(fit1, xlab = "Year", ylab = "Count", se = 2)

library(JOPS)
#Extract data
library(rpart)
Kyphosis = kyphosis$Kyphosis
Age  =kyphosis$Age
y = 1 * (Kyphosis == "present")  # make y 0/1
# Binomial smoothing
fit1 = psBinomial(Age, y, xl = min(Age), xr = max(Age), nseg = 20,
```

```
                bdeg = 3, pord = 2, lambda = 1)
names(fit1)
plot(fit1, xlab = "Age", ylab = '0/1', se = 2)
```

`plot.pssignal` *Plotting function for* `psSignal`

Description

Plotting function for signal regression P-spline smooth coefficients (using `psSignal` with class `pssignal`), with or without standard error bands.

Usage

```
## S3 method for class 'pssignal'
plot(x, ..., se = 2, xlab = "", ylab = "", col = "black", lty = 1)
```

Arguments

`x`	the P-spline x, usually from `psSignal`.
`...`	other parameters.
`se`	a scalar, e.g. `se = 2` to produce twice se bands, set `se` > 0 (or set `se = 0` to supress).
`xlab`	label for the x-axis, e.g. "my x" (quotes required).
`ylab`	label for the y-axis, e.g. "my y" (quotes required).
`col`	color.
`lty`	line type for plotting e.g. `lty = 2`.

Value

`Plot`	a plot of the smooth P-spline signal coefficent vector, with or without standard error bands.

Author(s)

Paul Eilers and Brian Marx

References

Marx, B.D. and Eilers, P.H.C. (1999). Generalized linear regression for sampled signals and curves: A P-spline approach. *Technometrics*, 41(1): 1-13.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(JOPS)
# Get the data
library(fds)
data(nirc)
iindex=nirc$x
X=nirc$y
sel= 50:650 #1200 <= x & x<= 2400
X=X[sel, ]
iindex=iindex[sel]
dX=diff(X)
diindex=iindex[-1]
y=as.vector(labc[1,1:40])
oout = 23
dX=t(dX[,-oout])
y=y[-oout]
fit2 = psSignal(y, dX, diindex, nseg = 25,lambda = 0.0001)
plot(fit2, se = 2, xlab = 'Coefficient Index', ylab= "ps Smooth Coeff")
title(main='25 B-spline segments with tuning=0.0001')
names(fit2)
```

`plot.psvcsignal` *Plotting function for* `psVCSignal`

Description

Plotting function for varying-coefficent signal regression P-spline smooth coefficients (using `psVCSignal` with `class psvcsignal`). Although se surface bands can be comuputed they are intentially left out as they are not interpretable, and there is generally little data to steer such a high-dimensional parameterization.

Usage

```
## S3 method for class 'psvcsignal'
plot(x, ..., xlab = " ", ylab = " ", Resol = 100)
```

Arguments

`x`	the P-spline object, usually from `psVCSignal`.
`...`	other parameters.
`xlab`	label for the x-axis, e.g. "my x" (quotes required).
`ylab`	label for the y-axis, e.g. "my y" (quotes required).
`Resol`	resolution for plotting, default `Resol = 100`.

Value

`Plot`	a two panel plot, one of the 2D P-spline signal coefficient surface and another that displays several slices of the smooth coefficient vectors at fixed levels of the varying index.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P. H. C. and Marx, B. D. (2003). Multivariate calibration with temperature interaction using two-dimensional penalized signal regression. *Chemometrics and Intellegent Laboratory Systems*, 66, 159–174.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(fds)
data(nirc)
iindex <- nirc$x
X <- nirc$y
sel <- 50:650 # 1200 <= x & x<= 2400
X <- X[sel, ]
iindex <- iindex[sel]
dX <- diff(X)
diindex <- iindex[-1]
y <- as.vector(labc[1, 1:40]) # percent fat
t_var <- as.vector(labc[4, 1:40]) # percent flour
oout <- 23
dX <- t(dX[, -oout])
y <- y[-oout]
t_var = t_var[-oout]
Pars = rbind(c(min(diindex), max(diindex), 25, 3, 1e-7, 2),
c(min(t_var), max(t_var), 20, 3, 0.0001, 2))
fit1 <- psVCSignal(y, dX, diindex, t_var, Pars = Pars,
family = "gaussian", link = "identity", int = TRUE)
plot(fit1, xlab = "Coefficient Index", ylab = "VC: % Flour")
names(fit1)
```

`plot.simpsr` *Plotting function for* `sim_psr`

Description

Plotting function for single-index signal regression with tensor product P-splines (using `sim_psr` with class `simpsr`).

Usage

```
## S3 method for class 'simpsr'
plot(x, ..., xlab = " ", ylab = " ", Resol = 100)
```

Arguments

`x`	the P-spline object, usually from `sim_psr`.
`...`	other parameters.
`xlab`	label for the x-axis, e.g. "my x" (quotes required).
`ylab`	label for the y-axis, e.g. "my y" (quotes required).
`Resol`	resolution for plotting, default `Resol = 100`.

Value

`Plot`	a two panel plot, one for the estimated P-spline signal coefficent vector, and another for the estimated (unkown) P-spline smooth link function.

Author(s)

Paul Eilers, Brian Marx, and Bin Li

References

Eilers, P.H.C., B. Li, B.D. Marx (2009). Multivariate calibration with single-index signal regression, *Chemometrics and Intellegent Laboratory Systems*, 96(2), 196-202.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(JOPS)
# Get the data
library(fds)
data(nirc)
iindex <- nirc$x
X <- nirc$y
sel <- 50:650 # 1200 <= x & x<= 2400
X <- X[sel, ]
iindex <- iindex[sel]
dX <- diff(X)
diindex <- iindex[-1]
y <- as.vector(labc[1, 1:40])
oout <- 23
dX <- t(dX[, -oout])
y <- y[-oout]

pords <- c(2, 2)
nsegs <- c(27, 7)
bdegs = c(3, 3)
```

```
lambdas <- c(1e-6, .1)
max_iter <- 100

# Single-index model
fit <- sim_psr(y, dX, diindex, nsegs, bdegs, lambdas, pords,
             max_iter)
plot(fit, xlab = "Wavelength (nm)", ylab = " ")
```

`plot.simvcpsr` *Plotting function for* `sim_vcpsr`

Description

Plotting function for varying-coefficient single-index signal regression using tensor product P-splines (using `sim_vcpsr` with class `simvcpsr`).

Usage

```
## S3 method for class 'simvcpsr'
plot(x, ..., xlab = " ", ylab = " ", Resol = 100)
```

Arguments

`x`	the P-spline object, usually from `sim_vcpsr`.
`...`	other parameters.
`xlab`	label for the x-axis, e.g. "my x" (quotes required).
`ylab`	label for the y-axis, e.g. "my y" (quotes required).
`Resol`	resolution for plotting, default `Resol = 100`.

Value

`Plot`	a plot of the estimated 2D P-spline signal coefficient surface along with the companion plot of the estimated 2D P-spline varying link function surface. Slices of these plots, at fixed levels of the indexing covariate, are also provided.

Author(s)

Paul Eilers and Brian Marx

References

Marx, B. D. (2015). Varying-coefficient single-index signal regression. *Chemometrics and Intellegent Laboratory Systems*, 143, 111–121.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
# Load libraries
library(fields) # Needed for plotting

# Get the data
Dat <- Mixture

# Dimensions: observations, temperature index, signal
m <- 34
p1 <- 401
p2 <- 12

# Stacking mixture data, each mixture has 12 signals stacked
# The first differenced spectra are also computed.
mixture_data <- matrix(0, nrow = p2 * m, ncol = p1)
for (ii in 1:m)
{
  mixture_data[((ii - 1) * p2 + 1):(ii * p2), 1:p1] <-
    t(as.matrix(Dat$xspectra[ii, , ]))
  d_mixture_data <- t(diff(t(mixture_data)))
}

# Response (typo fixed) and index for signal
y_mixture <- Dat$fractions
y_mixture[17, 3] <- 0.1501
index_mixture <- Dat$wl

# Select response and replicated for the 12 temps
# Column 1: water; 2: ethanediol; 3: amino-1-propanol
y <- as.vector(y_mixture[, 2])
y <- rep(y, each = p2)

bdegs = c(3, 3, 3, 3)
pords <- c(2, 2, 2, 2)
nsegs <- c(12, 5, 5, 5) # Set to c(27, 7, 7 ,7) for given lambdas
mins <- c(700, 30)
maxs <- c(1100, 70)
lambdas <- c(1e-11, 100, 0.5, 1) # based on svcm search
x_index <- seq(from = 701, to = 1100, by = 1) # for dX
t_var_sub <- c(30, 35, 37.5, 40, 45, 47.5, 50, 55, 60, 62.5, 65, 70)
t_var <- rep(t_var_sub, m)
max_iter <- 2 # Set higher in practice, e.g. 100
int <- TRUE

# Defining x as first differenced spectra, number of channels.
x <- d_mixture_data


# Single-index VC model using optimal tuning
fit <- sim_vcpsr(y, x, t_var, x_index, nsegs, bdegs, lambdas, pords,
             max_iter = max_iter, mins = mins, maxs = maxs)
```

```
plot(fit, xlab = "Wavelength (nm)", ylab = "Temp C")
```

`predict.ps2dglm` *Predict function for* `ps2DGLM`

Description

Prediction function which returns both linear predictor and inverse link predictions at arbitrary (x, y) data locations (using `ps2DGLM` with `class` `ps2dglm`).

Usage

```
## S3 method for class 'ps2dglm'
predict(object, ..., XY, type = "mu")
```

Arguments

`object`	an object using `ps2DGLM`.
`...`	other parameters.
`XY`	a matrix of arbitrary (x, y) locations for desired prediction.
`type`	the mean value `type = "mu"` (default) or linear predictor `type = "eta"`.

Value

`pred`	the estimated mean (inverse link function) (default) or the linear predictor prediction with `type = "eta"`, for arbitary (x, y) locations in `XY`.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P.H.C., Marx, B.D., and Durban, M. (2015). Twenty years of P-splines, *SORT*, 39(2): 149-186.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(fields)
library(JOPS)
# Extract data
library(rpart)
Kyphosis <- kyphosis$Kyphosis
Age <- kyphosis$Age
Start <- kyphosis$Start
y <- 1 * (Kyphosis == "present") # make y 0/1
fit <- ps2DGLM(
  Data = cbind(Start, Age, y),
  Pars = rbind(c(1, 18, 10, 3, .1, 2), c(1, 206, 10, 3, .1, 2)),
  family = "binomial", link = "logit")
predict(fit, XY = cbind(Start, Age)[1:5,])
```

`predict.ps2dnormal` *Predict function for* `ps2DNormal`

Description

Prediction function which returns linear predictions at arbitrary (x, y) data locations (using `ps2DNormal` with `class` `ps2dnormal`).

Usage

```
## S3 method for class 'ps2dnormal'
predict(object, ..., XY)
```

Arguments

`object`	an object using ps2DNormal.
`...`	other parameters.
`XY`	a matrix of arbitrary (x, y) locations for desired prediction.

Value

`pred`	the estimated mean at (`x`, `y`) locations, in `XY`.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P.H.C., Marx, B.D., and Durban, M. (2015). Twenty years of P-splines, *SORT*, 39(2): 149-186.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(fields)
library(spam)
library(JOPS)

# Get the data
data(ethanol)
x <- ethanol$C
y <- ethanol$E
z <- ethanol$NOx

# Set parameters for domain
xlo <- 7
xhi <- 19
ylo <- 0.5
yhi <- 1.25

# Set P-spline parameters, fit and compute surface
xpars <- c(xlo, xhi, 10, 3, 0.01, 1)
ypars <- c(ylo, yhi, 10, 3, 0.1, 1)
Pars1 <- rbind(xpars, ypars)
fit <- ps2DNormal(cbind(x, y, z), Pars = Pars1)
predict(fit, XY = cbind(x, y)[1:5, ])
```

`predict.ps2dsignal` *Predict function for* `ps2DSignal`

Description

Prediction function which returns both linear predictor and inverse link predictions for arbitrary 2D signals (using `ps2DSignal` with `class` `ps2dsignal`).

Usage

```
## S3 method for class 'ps2dsignal'
predict(object, ..., M_pred, M_type = "unfolded", type = "mu")
```

Arguments

`object`	an object using `ps2DSignal`.
`...`	other parameters.
`M_pred`	a matrix of `q` arbitrary "stacked" or "unfolded" signal matrices of dimension (`q` by `p1`) by `p2` or `q` by (`p1` by `p2`, respectively, for desired prediction (default "unfolded").
`M_type`	"stacked" or "unfolded" (default).
`type`	the mean value `type = "mu"` (default) or linear predictor `type = "eta"`.

Value

`pred`	the estimated mean (inverse link function) or the linear predictor prediction with `type = "eta"`, for arbitary 2D signals in `M_pred`.

Author(s)

Paul Eilers and Brian Marx

References

Marx, B.D. and Eilers, P.H.C. (2005). Multidimensional penalized signal regression, *Technometrics*, 47: 13-22.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(fields)
library(JOPS)

# Get the data
x0 <- Sugar$X
x0 <- x0 - apply(x0, 1, mean) # center Signal
y <- as.vector(Sugar$y[, 3]) # Response is Ash

# Inputs for two-dimensional signal regression
nseg <- c(7, 37)
pord <- c(3, 3)
min_ <- c(230, 275)
max_ <- c(340, 560)
M1_index <- rev(c(340, 325, 305, 290, 255, 240, 230))
M2_index <- seq(from = 275, to = 560, by = .5)
p1 <- length(M1_index)
p2 <- length(M2_index)

# Fit optimal model based on LOOCV
opt_lam <- c(8858.6679, 428.1332) # Found via svcm
Pars_opt <- rbind(
  c(min_[1], max_[1], nseg[1], 3, opt_lam[1], pord[1]),
  c(min_[2], max_[2], nseg[2], 3, opt_lam[2], pord[2])
)
fit <- ps2DSignal(y, x0, p1, p2, "unfolded", M1_index, M2_index,
  Pars_opt,int = TRUE, ridge_adj = 0.0001,
  M_pred = x0 )


predict(fit, M_pred= x0, type = "mu", M_type = "unfolded")
```

`predict.pspfit` *Predict function for* `psNormal`, `psBinomial`, `psPoisson`

Description

Prediction function which returns both linear predictor and inverse link predictions at arbitrary data locations (using `psNormal`, `psBinomial`, `psPoisson` with `class` `pspfit`).

Usage

```
## S3 method for class 'pspfit'
predict(object, ..., x, type = "mu")
```

Arguments

`object`	an object using `psNormal`, `psBinomial`, or `psPoisson` .
`...`	other parameters.
`x`	a scalar or vector of arbitrary `x` locations for desired prediction.
`type`	the mean value `type = "mu"` (default) or linear predictor `type = "eta"`.

Value

`pred`	the estimated mean (inverse link function) (default) or the linear predictor prediction with `type = "eta"`, at arbitary `x` locations.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P.H.C., Marx, B.D., and Durban, M. (2015). Twenty years of P-splines, *SORT*, 39(2): 149-186.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(JOPS)
library(boot)

# Extract the data
Count <- hist(boot::coal$date, breaks = c(1851:1963), plot = FALSE)$counts
Year <- c(1851:1962)
xl <- min(Year)
xr <- max(Year)
```

```
# Poisson smoothing
nseg <- 20
bdeg <- 3
fit1 <- psPoisson(Year, Count, xl, xr, nseg, bdeg, pord = 2, lambda = 1)
names(fit1)
plot(fit1, xlab = "Year", ylab = "Count", se = 2)
predict(fit1, x = fit1$x[1:5])
predict(fit1, x = fit1$x[1:5], type = "eta")
```

`predict.pssignal` *Predict function for* `psSignal`

Description

Prediction function which returns both linear predictor and inverse link predictions, for an arbitrary matrix of signals (using `psSignal` with `class` `pssignal`).

Usage

```
## S3 method for class 'pssignal'
predict(object, ..., X_pred, type = "mu")
```

Arguments

`object`	an object using `psSignal`.
`...`	other parameters.
`X_pred`	a matrix of arbitrary signals with `ncol(X)` == `length(x_index)` locations for desired prediction.
`type`	the mean value `type` = `"mu"` (default) or linear predictor `type` = `"eta"`.

Value

`pred`	the estimated mean (inverse link function) (default) or the linear predictor prediction with `type` = `"eta"`, for a matrix of signals in `X_pred`.

Author(s)

Paul Eilers and Brian Marx

References

Marx, B.D. and Eilers, P.H.C. (1999). Generalized linear regression for sampled signals and curves: A P-spline approach. *Technometrics*, 41(1): 1-13.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(JOPS)
# Get the data
library(fds)
data(nirc)
iindex=nirc$x
X=nirc$y
sel= 50:650 #1200 <= x & x<= 2400
X=X[sel,]
iindex=iindex[sel]
dX=diff(X)
diindex=iindex[-1]
y=as.vector(labc[1,1:40])
oout=23
dX=t(dX[,-oout])
y=y[-oout]
fit1 = psSignal(y, dX, diindex, nseg = 25,lambda = 0.0001)
predict(fit1, X_pred = dX[1:5, ])
predict(fit1, X_pred = dX[1:5, ], type = 'eta')
```

`predict.psvcsignal` *Predict function for* `psVCSignal`

Description

Prediction function which returns both linear predictor and inverse link predictions for an arbitrary matrix of signals with their vector of companion indexing covariates (using `psVCSignal` with `class psvcsignal`).

Usage

```
## S3 method for class 'psvcsignal'
predict(object, ..., X_pred, t_pred, type = "mu")
```

Arguments

`object`	an object using `psVCSignal`.
`...`	other parameters.
`X_pred`	a matrix of `q` arbitrary signal vectors of dimension `q` by `p1` for desired prediction.
`t_pred`	a `q` vector for the varying index variable associated with `X_pred`.
`type`	the mean value `type = "mu"` (default) or linear predictor `type = "eta"`.

Value

`pred`	the estimated mean (inverse link function) (default) or the linear predictor prediction with `type = "eta"`, at signals in matrix `X_pred` and covariates in vector `t_pred`.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P. H. C. and Marx, B. D. (2003). Multivariate calibration with temperature interaction using two-dimensional penalized signal regression. *Chemometrics and Intellegent Laboratory Systems*, 66, 159–174.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(fds)
data(nirc)
iindex <- nirc$x
X <- nirc$y
sel <- 50:650 # 1200 <= x & x<= 2400
X <- X[sel, ]
iindex <- iindex[sel]
dX <- diff(X)
diindex <- iindex[-1]
y <- as.vector(labc[1, 1:40]) # percent fat
t_var <- as.vector(labc[4, 1:40]) # percent flour
oout <- 23
dX <- t(dX[, -oout])
y <- y[-oout]
t_var = t_var[-oout]
Pars = rbind(c(min(diindex), max(diindex), 25, 3, 1e-7, 2),
c(min(t_var), max(t_var), 20, 3, 0.0001, 2))
fit1 <- psVCSignal(y, dX, diindex, t_var, Pars = Pars,
family = "gaussian", link = "identity", int = TRUE)
predict(fit1, X_pred = dX[1:5,], t_pred = t_var[1:5])
```

`predict.simpsr` *Predict function for* `sim_psr`

Description

Prediction function which returns single-index inverse link linear predictions at arbitrary data locations (using `sim_psr` with `class simpsr`).

Usage

```
## S3 method for class 'simpsr'
predict(object, ..., X_pred)
```

Arguments

`object`	an object using `sim_psr`.
`...`	other parameters.
`X_pred`	a matrix of arbitrary signals with `ncol(X_pred) = length(x_index)` locations for desired prediction.

Value

`pred`	the estimated (inverse single-index) mean for the signals in `X_pred`.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P.H.C., B. Li, B.D. Marx (2009). Multivariate calibration with single-index signal regression, *Chemometrics and Intellegent Laboratory Systems*, 96(2), 196-202.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(JOPS)
# Get the data
library(fds)
data(nirc)
iindex <- nirc$x
X <- nirc$y
sel <- 50:650 # 1200 <= x & x<= 2400
X <- X[sel, ]
iindex <- iindex[sel]
dX <- diff(X)
diindex <- iindex[-1]
y <- as.vector(labc[1, 1:40])
oout <- 23
dX <- t(dX[, -oout])
y <- y[-oout]

pords <- c(2, 2)
nsegs <- c(27, 7)
bdegs = c(3, 3)
lambdas <- c(1e-6, .1)
max_iter <- 100

# Single-index model
fit <- sim_psr(y, dX, diindex, nsegs, bdegs, lambdas, pords,
             max_iter)
predict(fit, X_pred = dX)
```

`predict.simvcpsr` *Predict function for* `sim_vcpsr`

Description

Prediction function which returns varying-coefficient single-index inverse link linear predictions at arbitrary data locations (using `sim_vcpsr` with `class` `simvcpsr`).

Usage

```
## S3 method for class 'simvcpsr'
predict(object, ..., X_pred, t_pred)
```

Arguments

`object`	an object using `sim_vcpsr`.
`...`	other parameters.
`X_pred`	a matrix of arbitrary signals with `ncol(X_pred) = length(x_index)` locations for desired prediction.
`t_pred`	a `q` vector for the VC index variable associated with `X_pred`.

Value

`pred`	the estimated (inverse single-index) mean for the signals in the matrix `X_pred`, with the companion vector of indexing covariates in `t_pred`.

Author(s)

Paul Eilers and Brian Marx

References

Marx, B. D. (2015). Varying-coefficient single-index signal regression. *Chemometrics and Intellegent Laboratory Systems*, 143, 111–121.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
# Load libraries
library(fields) # Needed for plotting

# Get the data
Dat <- Mixture

# Dimensions: observations, temperature index, signal
m <- 34
```

```
p1 <- 401
p2 <- 12

# Stacking mixture data, each mixture has 12 signals stacked
# The first differenced spectra are also computed.
mixture_data <- matrix(0, nrow = p2 * m, ncol = p1)
for (ii in 1:m)
{
  mixture_data[((ii - 1) * p2 + 1):(ii * p2), 1:p1] <-
    t(as.matrix(Dat$xspectra[ii, , ]))
  d_mixture_data <- t(diff(t(mixture_data)))
}

# Response (typo fixed) and index for signal
y_mixture <- Dat$fractions
y_mixture[17, 3] <- 0.1501
index_mixture <- Dat$wl

# Select response and replicated for the 12 temps
# Column 1: water; 2: ethanediol; 3: amino-1-propanol
y <- as.vector(y_mixture[, 2])
y <- rep(y, each = p2)

bdegs = c(3, 3, 3, 3)
pords <- c(2, 2, 2, 2)
nsegs <- c(12, 5, 5, 5) # Set to c(27, 7, 7 ,7) for given lambdas
mins <- c(700, 30)
maxs <- c(1100, 70)
lambdas <- c(1e-11, 100, 0.5, 1) # based on svcm search
x_index <- seq(from = 701, to = 1100, by = 1) # for dX
t_var_sub <- c(30, 35, 37.5, 40, 45, 47.5, 50, 55, 60, 62.5, 65, 70)
t_var <- rep(t_var_sub, m)
max_iter <- 2 # Set higher in practice, e.g. 100
int <- TRUE

# Defining x as first differenced spectra, number of channels.
x <- d_mixture_data


# Single-index VC model using optimal tuning
fit <- sim_vcpsr(y, x, t_var, x_index, nsegs, bdegs, lambdas, pords,
             max_iter = max_iter, mins = mins, maxs = maxs)

predict(fit, X_pred = x, t_pred = t_var)
```

`ps2DGLM` *Two-dimensional smoothing of scattered normal or non-normal (GLM) responses using tensor product P-splines.*

Description

`ps2DGLM` is used to smooth scattered normal or non-normal responses, with aniosotripic penalization of tensor product P-splines.

Usage

```
ps2DGLM(
  Data,
  Pars = rbind(c(min(Data[, 1]), max(Data[, 1]), 10, 3, 1, 2), c(min(Data[, 2]),
    max(Data[, 2]), 10, 3, 1, 2)),
  ridge_adj = 0,
  XYpred = Data[, 1:2],
  z_predicted = NULL,
  se_pred = 2,
  family = "gaussian",
  link = "default",
  m_binomial = rep(1, nrow(Data)),
  wts = rep(1, nrow(Data)),
  r_gamma = rep(1, nrow(Data))
)
```

Arguments

`Data`	a matrix of 3 columns `x, y, z` of equal length; the response is `z`.
`Pars`	a matrix of 2 rows, where the first and second row sets the P-spline paramters for `x` and `y`, respectively. Each row consists of: `min max nseg bdeg lambda pord`. The `min` and `max` set the ranges, `nseg` (default 10) is the number of evenly spaced segments between `min` and `max`, `bdeg` is the degree of the basis (default 3 for cubic), `lambda` is the (positive) tuning parameter for the penalty (default 1), `pord` is the number for the order of the difference penalty (default 2).
`ridge_adj`	a ridge penalty tuning parameter, usually set to small value, e.g. `1e-8` to stabilize estimation (default 0).
`XYpred`	a matrix with two columns `(x, y)` that give the coordinates of (future) prediction; the default is the data locations.
`z_predicted`	a vector of responses associated with `XYpred`, useful for external validation with `family = "gaussian"`.
`se_pred`	a scalar, default `se_pred = 2` to produce se surfaces, set `se_pred` > 0. Used for CIs for `XYpred` locations.
`family`	`"gaussian"`, `"binomial"`, `"poisson"`, `"Gamma"` (quotes needed). Default is "gaussian".
`link`	the link function, one of `"identity"`, `"log"`, `"sqrt"`, `"logit"`, `"probit"`, `"cloglog"`, `"loglog"`, `"reciprocal"`; quotes are needed (default `"identity"`).
`m_binomial`	vector of binomial trials, default is vector of ones with `family = "binomial"`, NULL otherwise.
`wts`	non-negative weights, which can be zero (default ones).
`r_gamma`	gamma scale parameter, default is vector ones with `family = "Gamma"`, NULL otherwise.

Details

Support functions needed: `pspline_fitter`, `bbase`, and `pspline_2dchecker`.

Value

`pcoef`	a vector of length `(Pars[1,3]+Pars[1,4])*(Pars[2,3]+Pars[2,4])` of (unfolded) estimated P-spline coefficients.
`mu`	a vector of `length(z)` of smooth estimated means (at the `x,y` locations).
`dev`	the deviance of fit.
`eff_df`	the approximate effective dimension of fit.
`aic`	AIC.
`df_resid`	approximate df residual.
`cv`	leave-one-out standard error prediction, when `family = 'gaussian'`.
`cv_predicted`	standard error prediction for `y_predict`, when `family = 'gaussian'`.
`avediff_pred`	mean absolute difference prediction, when `family = 'gaussian'`.
`Pars`	the design and tuning parameters (see arguments above).
`dispersion_parm`	estimate of dispersion, `dev/df_resid`.
`summary_predicted`	inverse link prediction vectors, and `se_pred` bands.
`eta_predicted`	estimated linear predictor of `length(z)`.
`press_mu`	leave-one-out prediction of mean, when `family = 'gaussian'`.
`bin_percent_correct`	percent correct classification based on 0.5 cut-off (when `family = "binomial"`).
`Data`	a matrix of 3 columns `x, y, z` of equal length; the response is `z`.
`Q`	the tensor product B-spline basis.
`qr`	the Q-R of the model.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Eilers, P.H.C., Marx, B.D., and Durban, M. (2015). Twenty years of P-splines, *SORT*, 39(2): 149-186.

See Also

ps2DNormal

Examples

```
library(fields)
library(JOPS)
# Extract data
library(rpart)
Kyphosis <- kyphosis$Kyphosis
Age <- kyphosis$Age
Start <- kyphosis$Start
y <- 1 * (Kyphosis == "present") # make y 0/1
fit <- ps2DGLM(
  Data = cbind(Start, Age, y),
  Pars = rbind(c(1, 18, 10, 3, .1, 2), c(1, 206, 10, 3, .1, 2)),
  family = "binomial", link = "logit")
plot(fit, xlab = "Start", ylab = "Age")
#title(main = "Probability of Kyphosis")
```

`ps2DNormal` *Two-dimensional smoothing scattered (normal) data using P-splines.*

Description

ps2DNormal is used to smooth scattered (normal) data, with anisotropic penalization of tensor product P-splines.

Usage

```
ps2DNormal(
  Data,
  Pars = rbind(c(min(Data[, 1]), max(Data[, 1]), 10, 3, 1, 2), c(min(Data[, 2]),
    max(Data[, 2]), 10, 3, 1, 2)),
  XYpred = expand.grid(Data[, 1], Data[, 2])
)
```

Arguments

`Data`	a matrix of 3 columns `x`, `y`, `z` of equal length; the response is `z`.
`Pars`	a matrix of 2 rows, where the first and second row sets the P-spline paramters for `x` and `y`, respectively. Each row consists of: `min` `max` `nseg` `bdeg` `lambda` `pord`. The `min` and `max` set the ranges, `nseg` (default 10) is the number of evenly spaced segments between `min` and `max`, `bdeg` is the degree of the basis (default 3 for cubic), `lambda` is the (positive) tuning parameter for the penalty (default 1), `pord` is the number for the order of the difference penalty (default 2),
`XYpred`	a matrix with two columns `(x,y)` that give the coordinates of (future) prediction; the default is the data locations.

Details

Support functions needed: `pspline_fitter`, `bbase`, and `pspline_2dchecker`.

Value

`coef`	a vector of length `(Pars[1,3]+Pars[1,4])*(Pars[2,3]+Pars[2,4])` of (unfolded) estimated P-spline coefficients.
`fit`	a vector of `length(y)` of smooth estimated means (at the `x,y` locations).
`pred`	a vector of length `nrow(XYpred)` of (future) predictions.
`Pars`	the design and tuning parameters (see arguments above).
`cv`	leave-one-out standard error of prediction or root average PRESS.
`h`	"hat" diagonals of tensor P-spline fit.
`B`	tensor product B-spline basis used for fitting.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Eilers, P.H.C., Marx, B.D., and Durban, M. (2015). Twenty years of P-splines, *SORT*, 39(2): 149-186.

See Also

ps2DGLM

Examples

```
library(fields)
library(spam)
library(JOPS)

# Get the data
data(ethanol)
x <- ethanol$C
y <- ethanol$E
z <- ethanol$NOx

# Set parameters for domain
xlo <- 7
xhi <- 19
ylo <- 0.5
yhi <- 1.25

# Set P-spline parameters, fit and compute surface
xpars <- c(xlo, xhi, 10, 3, 3, 1)
ypars <- c(ylo, yhi, 10, 3, 3, 1)
Pars1 <- rbind(xpars, ypars)
fit <- ps2DNormal(cbind(x, y, z), Pars = Pars1)
plot(fit, xlab = "C", ylab = "E")
```

`ps2DSignal` *Two-dimensional penalized signal regression using P-splines.*

Description

`ps2DSignal` is a function used to regress a (glm) response onto a two-dimensional signal or image, with aniosotripic penalization of tensor product P-splines.

Usage

```
ps2DSignal(
  y,
  M,
  p1,
  p2,
  M_type = "stacked",
  M1_index = c(1:p1),
  M2_index = c(1:p2),
  Pars = rbind(c(1, p1, 10, 3, 1, 2), c(1, p2, 10, 3, 1, 2)),
  ridge_adj = 1e-06,
  M_pred = M,
  y_predicted = NULL,
  family = "gaussian",
  link = "default",
  m_binomial = 1 + 0 * y,
  wts = 1 + 0 * y,
  r_gamma = 1 + 0 * y,
  int = TRUE,
  se_pred = 2
)
```

Arguments

`y`	a response vector of length `m`, usually continuous, binary/bimomial or counts.
`M`	The signal/image regressors, which are either "stacked" or "unfolded", with dimensions (`m * p1`) by `p2` (i.e. `m` stacked matrices each of `p1` by `p2`) or with dimensions `m` by (`p1 * p2`) (i.e. regressor matrix with `m` regressor rows, each with column length `p1 * p2`), respectively.
`p1`	the row dimension of the image.
`p2`	the column dimension of the image.
`M_type`	"stacked" (signal as matrix) or "unfolded" (signal as vector).
`M1_index`	an index of length `p1` for rows of regressor matrix (default is a simple sequence).
`M2_index`	an index of length `p2` for columns of regressor matrix (default is a simple sequence).

`Pars`	a matrix of 2 rows, where the first and second row sets the P-spline paramters for `x` and `y`, respectively. Each row consists of: `min` `max` `nseg` `bdeg` `lambda` `pord`. The `min` and `max` set the ranges, `nseg` (default 10) is the number of evenly spaced segments between `min` and `max`, `bdeg` is the degree of the basis (default 3 for cubic), `lambda` is the (positive) tuning parameter for the penalty (default 1), `pord` is the number for the order of the difference penalty (default 2).
`ridge_adj`	A ridge penalty tuning parameter (usually set to small value, default `1e-6`, to stabilize estimation).
`M_pred`	(e.g. stacked (`q` * `p1`) by `p2` signal inputs or (unfolded) `q` by (`p1` * `p2`) signal inputs for `q` new predictions.
`y_predicted`	a vector of responses from a cv data set (assoc. with `M_pred`), when `family = "gaussian"`.
`family`	the response distribution, e.g. `"gaussian"`, `"binomial"`, `"poisson"`, `"Gamma"` distribution. Quotes are needed. Default is "gaussian".
`link`	the link function, one of `"identity"`, `"log"`, `"sqrt"`, `"logit"`, `"probit"`, `"cloglog"`, `"loglog"`, `"reciprocal"`; quotes are needed (default `"identity"`).
`m_binomial`	a vector of binomial trials having `length(y)`. Default is 1 vector for `family = "binomial"`, NULL otherwise.
`wts`	the weight vector of `length(y)`. Default is 1.
`r_gamma`	a vector of gamma shape parameters. Default is 1 vector for for `family = "Gamma"`, NULL otherwise.
`int`	set to TRUE or FALSE to include intercept term in linear predictor (default `TRUE`).
`se_pred`	a scalar, e.g. `se = 2` (default) to produce twice se surfaces, set `se` > 0. Used for CIs at `XYpred` locations.

Details

Support functions needed: `pspline_fitter`, `bbase`, and `pspline_2dchecker`.

Value

`pcoef`	a vector of length `(Pars[1,3]+Pars[1,4])*(Pars[2,3]+Pars[2,4])` of (unfolded) estimated P-spline coefficients for tensor surface.
`summary_predicted`	inverse link prediction vectors, and standard error surfaces.
`dev`	deviance of fit.
`eff_df`	the approximate effective dimension of fit.
`aic`	AIC.
`df_resid`	approximate df residual.
`cv`	leave-one-out standard error prediction, when `family = "gaussian"`.
`cv_predicted`	standard error prediction for `y_predict`, when `family = "gaussian"`.
`avediff_pred`	mean absolute difference prediction, when `family = 'gaussian'`.

`Pars`	design and tuning parameters (see above arguments).
`Dispersion_parm`	estimate of dispersion, `dev/df_resid`.
`summary_predicted`	inverse link prediction vectors at `M_pred`, and standard error bands.
`eta_predicted`	estimated linear predictor of `length(y)`.
`press_mu`	leave-one-out prediction of mean, when `family = "gaussian"`.
`bin_percent_correct`	percent correct classification based on 0.5 cut-off, when `family = "binomial"`, NULL otherwise.
`B`	Tensor basis (`p1` x `p2`) by (`n1` x `n2`) for 2D signal regression.
`Q`	Effective regressors (`m` by `n1` * `n2`) for 2D signal regression.
`Ahat`	smooth P-spline coefficient vector of length `p1` x `p2`, constructed by `B` %*% `pcoef`.
`M`	the signal/image regressors.
`y`	the response vector.
`M1index`	index of length `p1` for rows of regressor matrix.
`M2index`	index of length `p2` for columns of regressor matrix.
`M_type`	"stacked" or "unfolded".
`w`	GLM weight vector of length `m`.
`h`	"hat" diagonals.
`ridge_adj`	additional ridge tuning parameter to stabilize estimation.

Author(s)

Paul Eilers and Brian Marx

References

Marx, B.D. and Eilers, P.H.C. (2005). Multidimensional penalized signal regression, *Technometrics*, 47: 13-22.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(fields)
library(JOPS)

# Get the data
x0 <- Sugar$X
x0 <- x0 - apply(x0, 1, mean) # center Signal
y <- as.vector(Sugar$y[, 3]) # Response is Ash

# Inputs for two-dimensional signal regression
```

```
nseg <- c(7, 37)
pord <- c(3, 3)
min_ <- c(230, 275)
max_ <- c(340, 560)
M1_index <- rev(c(340, 325, 305, 290, 255, 240, 230))
M2_index <- seq(from = 275, to = 560, by = .5)
p1 <- length(M1_index)
p2 <- length(M2_index)

# Fit optimal model based on LOOCV
opt_lam <- c(8858.6679, 428.1332) # Found via svcm
Pars_opt <- rbind(
  c(min_[1], max_[1], nseg[1], 3, opt_lam[1], pord[1]),
  c(min_[2], max_[2], nseg[2], 3, opt_lam[2], pord[2])
)
fit <- ps2DSignal(y, x0, p1, p2, "unfolded", M1_index, M2_index,
  Pars_opt,int = TRUE, ridge_adj = 0.0001,
  M_pred = x0 )

# Plotting coefficient image
 plot(fit)
```

`ps2D_PartialDeriv` *Partial derivative two-dimensional smoothing scattered (normal) data using P-splines.*

Description

`ps2D_PartialDeriv` provides the partial derivative P-spline surface along `x`, with aniosotripic penalization of tensor product B-splines.

Usage

```
ps2D_PartialDeriv(
  Data,
  Pars = rbind(c(min(Data[, 1]), max(Data[, 1]), 10, 3, 1, 2), c(min(Data[, 2]),
    max(Data[, 2]), 10, 3, 1, 2)),
  XYpred = cbind(Data[, 1], Data[, 2])
)
```

Arguments

`Data`	a matrix of 3 columns `x`, `y`, `z` of equal length; the response is `z`.
`Pars`	a matrix of 2 rows, where the first and second row sets the P-spline paramters for `x` and `y`, respectively. Each row consists of: `min` `max` `nseg` `bdeg` `lambda` `pord`. The `min` and `max` set the ranges, `nseg` (default 10) is the number of evenly spaced segments between `min` and `max`, `bdeg` is the degree of the basis (default 3 for cubic), `lambda` is the (positive) tuning parameter for the penalty (default 1), `pord` is the number for the order of the difference penalty (default 2).

`XYpred`	a matrix with two columns `(x, y)` that give the coordinates of (future) prediction; the default is the data locations.

Details

This is support function for `sim_vcpsr`.

Value

`coef`	a vector of length `(Pars[1, 3] + Pars[1, 4]) * (Pars[1, 3] + Pars[1, 4]).` of (unfolded) estimated P-spline coefficients.
`B`	the tensor product B-spline matrix of dimensions `m` by `length(coef)`.
`fit`	a vector of `length(y)` of smooth estimated means (at the `x, y` locations).
`pred`	a vector of length `nrow(XYpred)` of (future) predictions.
`d_coef`	a vector of length `(Pars[1, 3] + Pars[1,4] - 1) * (Pars[1,3]+Pars[1,4]).` of (unfolded) partial derivative estimated P-spline coefficients.
`B_d`	the tensor product B-spline matrix of dimensions `m` by `lengh(d_coef)`, associated with the partial derivative of the tensor basis.
`d_fit`	a vector of `length(y)` of partial derivative (along `x`) of the smooth estimated means (at the `x, y` locations).
`d_pred`	a vector of length `nrow(XYpred)` of partial derivative (future) predictions.
`Pars`	a matrix of 2 rows, where each the first (second) row sets the P-spline paramters for `x (y)`: `min max nseg bdeg lambda pord`. See the argument above.
`cv`	root leave-one-out CV or root average PRESS.
`XYpred`	a matrix with two columns `(x, y)` that give the coordinates of (future) prediction; the default is the data locations.

Author(s)

Brian Marx

References

Marx, B. D. (2015). Varying-coefficient single-index signal regression. *Chemometrics and Intelligent Laboratory Systems*, 143, 111–121.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

`psBinomial` *Smoothing scattered binomial data using P-splines.*

Description

`psBinomial` is used to smooth scattered binomial data using P-splines using a logit link function.

Usage

```
psBinomial(
  x,
  y,
  xl = min(x),
  xr = max(x),
  nseg = 10,
  bdeg = 3,
  pord = 2,
  lambda = 1,
  ntrials = 0 * y + 1,
  wts = NULL,
  show = FALSE,
  iter = 100,
  xgrid = 100
)
```

Arguments

`x`	the vector for the continuous regressor of `length(y)` and the abcissae, on which the B-spline basis is constructed.
`y`	the response vector, usually 0/1 or binomial counts.
`xl`	the lower limit for the domain of `x` (default is min(x)).
`xr`	the upper limit for the domain of `x` (default is max(x)).
`nseg`	the number of evenly spaced segments between xl and xr.
`bdeg`	the number of the degree of the basis, usually 1, 2 (default), or 3.
`pord`	the number of the order of the difference penalty, usually 1, 2, or 3 (defalult).
`lambda`	the (positive) number for the tuning parameter for the penalty.
`ntrials`	the vector for the number of binomial trials (default = 1).
`wts`	the vector of weights, default is 1, zeros allowed.
`show`	Set to TRUE or FALSE to display iteration history.
`iter`	a scalar to set the maximum number of iterations, default `iter = 100`.
`xgrid`	a scalar or a vector that gives the `x` locations for prediction, useful for plotting. If a scalar (default 100) is used then a uniform grid of this size along (`xl`, `xr`).

Value

`pcoef`	a vector of length `n` of estimated P-spline coefficients.
`p`	a vector of length `m` of estimated probabilities.
`muhat`	a vector of length `m` of estimated means (`ntrials*p`).
`dev`	deviance
`effdim`	effective dimension of the smooth.
`aic`	AIC
`wts`	a vector of preset weights (default = 1).
`nseg`	the number of B-spline segments.
`bdeg`	the degree of the B-spline basis.
`pord`	the order of the difference penalty.
`family`	the GLM family (repsonse distribution).
`link`	the link function.
`y`	the binomial response.
`x`	the regressor on which the basis is constructed.
`P`	"half" of the penalty matrix, `P'P = lambda*D'D`.
`B`	the B-spline basis.
`lambda`	the positive tuning parameter.
`dispersion`	dispersion parameter estimated `dev/(m-effdim)`.
`xgrid`	gridded `x` values,useful for plotting.
`ygrid`	gridded fitted linear predictor values, useful for plotting.
`pgrid`	gridded (inverse link) fitted probability values, useful for plotting.
`se_eta`	gridded standard errors for the linear predictor.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Eilers, P.H.C., Marx, B.D., and Durban, M. (2015). Twenty years of P-splines, *SORT*, 39(2): 149-186.

Examples

```
library(JOPS)
# Extract data
library(rpart)
Kyphosis <- kyphosis$Kyphosis
Age <- kyphosis$Age
```

```
y <- 1 * (Kyphosis == "present") # make y 0/1
fit1 <- psBinomial(Age, y,
  xl = min(Age), xr = max(Age), nseg = 20,
  bdeg = 3, pord = 2, lambda = 10
)
names(fit1)
plot(fit1, xlab = "Age", ylab = "0/1", se = 2)
```

`psNormal` *Smoothing scattered (normal) data using P-splines.*

Description

`psNormal` is used to smooth scattered (normal) data using P-splines (with identity link function).

Usage

```
psNormal(
  x,
  y,
  xl = min(x),
  xr = max(x),
  nseg = 10,
  bdeg = 3,
  pord = 2,
  lambda = 1,
  wts = NULL,
  xgrid = 100
)
```

Arguments

`x`	the vector for the continuous regressor of `length(y)` and the abcissae used to build the B-spline basis.
`y`	the response vector, usually continuous data.
`xl`	the number for the min along `x` (default is min(x)) .
`xr`	the number for the max along `x` (default is max(x)).
`nseg`	the number of evenly spaced segments between `xl` and `xr`.
`bdeg`	the number of the degree of the basis, usually 1, 2 (default), or 3.
`pord`	the number of the order of the difference penalty, usually 1, 2, or 3 (defalult).
`lambda`	the (positive) number for the tuning parameter for the penalty (default 1).
`wts`	the vector of general weights, default is 1; zero allowed.
`xgrid`	a scalar or a vector that gives the `x` locations for prediction, useful for plotting. If a scalar (default 100) is used then a uniform grid of this size along (`xl`, `xr`).

Value

`pcoeff`	a vector of length `n` of estimated P-spline coefficients.
`muhat`	a vector of length `m` of smooth estimated means.
`B`	a matrix of dimension `m` by `n` for the B-spline basis matrix.
`wts`	a vector of length `m` of weights.
`effdim`	estimated effective dimension.
`ed_resid`	approximate df residual.
`sigma`	square root of MSE.
`cv`	standard error of leave-one-out prediction or root average PRESS.
`nseg`	the number of B-spline segments.
`bdeg`	the degree of the B-spline basis.
`pord`	the order of the difference penalty.
`lambda`	the positive tuning parameter.
`xgrid`	gridded x values, useful for plotting.
`ygrid`	gridded fitted mean values, useful for plotting.
`se_eta`	gridded standard errors for the fitted mean values, useful for plotting.
`P`	"half" of the penalty, such that `P'P=` `lambda` `D'D`.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Eilers, P.H.C., Marx, B.D., and Durban, M. (2015). Twenty years of P-splines, *SORT*, 39(2): 149-186.

Examples

```
library(JOPS)
library(MASS)
data(mcycle)
x <- mcycle$times
y <- mcycle$accel
fit1 <- psNormal(x, y, nseg = 20, bdeg = 3, pord = 2, lambda = .8)
plot(fit1, se = 2, xlab = "Time (ms)", ylab = "Acceleration")
```

`psNormal_Deriv` *Derivative for a P-spline fit of scattered (normal) data.*

Description

`psNormal_Deriv` provides the derivative P-spline fit along `x`.

Usage

```
psNormal_Deriv(
  x,
  y,
  xl = min(x),
  xr = max(x),
  nseg = 10,
  bdeg = 3,
  pord = 2,
  lambda = 1,
  wts = rep(1, length(y)),
  xgrid = x
)
```

Arguments

`x`	the vector for the continuous regressor of `length(y)` and the abcissae of fit.
`y`	the response vector, usually continuous data.
`xl`	the number for the min along `x` (default is min(x)) .
`xr`	the number for the max along `x` (default is max(x)).
`nseg`	the number of evenly spaced segments between `xl` and `xr`.
`bdeg`	the number of the degree of the basis, usually 1, 2, or 3 (defalult).
`pord`	the number of the order of the difference penalty, usually 1, 2 (defalult), or 3.
`lambda`	the positive tuning parameter (default 1).
`wts`	the vector of weights, default is 1; 0/1 allowed.
`xgrid`	a scalar or a vector that gives the `x` locations for prediction, useful for plotting. If a scalar (default 100) is used then a uniform grid of this size along (`xl`, `xr`).

Details

This is also a support function needed for `sim_psr` and `sim_vcpsr`. SISR (Eilers, Li, Marx, 2009).

Value

`coef`	a vector of `length(nsegs + bdeg)` of estimated P-spline coefficients.
`B`	The B-spline matrix of dimensions `m` by `length(coef)`.
`fit`	a vector of `length(y)` of smooth estimated means (at the `x` locations).
`pred`	a vector of `length(xgrid)` of (future) predictions.
`d_coef`	a vector of `length(nsegs + bdeg - 1)` of differenced (derivative) estimated P-spline coefficients.
`B_d`	The first derivative B-spline matrix of dimensions `m` by `lengh(d_coef)`.
`d_fit`	a vector of `length(y)` of partial derivative (along `x`) of the smooth estimated means (at the `x` locations).
`d_pred`	a vector of length `lenght(xgrid)` of partial derivative (future) predictions.
`xl`	the number for the min along `x` (default is min(`x`)).
`xr`	the number for the max along `x` (default is max(`x`)).
`nseg`	the number of evenly spaced segments between `xl` and `xr`.
`bdeg`	the number of the degree of the basis, usually 1, 2, or 3 (default).
`pord`	the number of the order of the difference penalty, usually 1, 2 (default), or 3.
`lambda`	the positive tuning parameter (default 1).

Author(s)

Paul Eilers and Brian Marx

References

Marx, B. D. (2015). Varying-coefficient single-index signal regression. *Chemometrics and Intelligent Laboratory Systems*, 143, 111–121.

Eilers, P.H.C., B. Li, B.D. Marx (2009). Multivariate calibration with single-index signal regression, *Chemometrics and Intellegent Laboratory Systems*, 96(2), 196-202.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

See Also

sim_psr sim_vcpsr

`pspline2d_checker` *P-spline 2D tensor product checking algorithm for the GLM.*

Description

`pspline_2dchecker` checks to see if all the 2D tensor inputs associated for P-spines are properly defined.

Usage

```
pspline2d_checker(
  family,
  link,
  bdeg1,
  bdeg2,
  pord1,
  pord2,
  nseg1,
  nseg2,
  lambda1,
  lambda2,
  ridge_adj,
  wts
)
```

Arguments

`family`	the response distribution, e.g. `"gaussian"`, `"binomial"`, `"poisson"`, `"Gamma"` distribution. Quotes are needed.
`link`	the link function, one of `"identity"`, `"log"`, `"sqrt"`, `"logit"`, `"probit"`, `"cloglog"`, `"loglog"`, `"reciprocal"`; quotes are needed.
`bdeg1`	the degree of B-splines.
`bdeg2`	the degree of B-splines.
`pord1`	the order of the penalty.
`pord2`	the order of the penalty.
`nseg1`	the number of evenly spaced B-spline segmements.
`nseg2`	the number of evenly spaced B-spline segmements.
`lambda1`	the positive tuning parameter for the difference penalty.
`lambda2`	the positive tuning parameter for the difference penalty.
`ridge_adj`	the positive tuning parameter for the ridge penalty.
`wts`	the weight vector, separate from GLM weights.

Value

`list`	same as inputs, with warnings if required.

`pspline_checker` *P-spline checking algorithm for the GLM.*

Description

`pspline_checker` checks to see if all the inputs associated for P-spines are properly defined.

Usage

```
pspline_checker(family, link, bdeg, pord, nseg, lambda, ridge_adj, wts)
```

Arguments

`family`	the response distribution, e.g. `"gaussian"`, `"binomial"`, `"poisson"`, `"Gamma"` distribution. Quotes are needed.
`link`	the link function, one of `"identity"`, `"log"`, `"sqrt"`, `"logit"`, `"probit"`, `"cloglog"`, `"loglog"`, `"reciprocal"`;
`bdeg`	the degree of B-splines.
`pord`	the order of the penalty.
`nseg`	the number of evenly-spaced B-spline segmements.
`lambda`	the positive tuning parameter for the difference penalty.
`ridge_adj`	the positive tuning parameter for the ridge penalty.
`wts`	the weight vector, separate from GLM weights.

Value

`list`	same as inputs, with warnings if required.

`pspline_fitter` *P-spline fitting algorithm for the GLM.*

Description

`pspline_fitter` appies the method of scoring to a variety of response distributions and link functions within for P-spline fitting within the GLM framework.

Usage

```
pspline_fitter(
  y,
  B,
  family = "gaussian",
  link = "identity",
  P,
  P_ridge = 0 * diag(ncol(B)),
  wts = 0 * y + 1,
  m_binomial = 0 * y + 1,
  r_gamma = 0 * y + 1
)
```

Arguments

`y`	the glm response vector of length `m`.
`B`	The effective P-spline regressors, e.g. `B` for B-splines, `Q=X %*% B` for PSR.
`family`	the response distribution, e.g. `"gaussian"`, `"binomial"`, `"poisson"`, `"Gamma"` distribution; quotes are needed (default `family = "gaussian"`.)
`link`	the link function, one of `"identity"`, `"log"`, `"sqrt"`, `"logit"`, `"probit"`, `"cloglog"`, `"loglog"`, `"reciprocal"`; quotes are needed (default `link = "identity"`).
`P`	P-spline ("half") penalty matrix for data augmentation, such that `P'P = lambda D'D`.
`P_ridge`	ridge ("half") penalty for data augmentation, usually `sqrt(lambda_r)*I` (default 0).
`wts`	the weight vector of `length(y)`, separate from GLM weights.
`m_binomial`	a vector of binomial trials having `length(y)`, when `family = "binomial"`. Default is 1 vector.
`r_gamma`	a vector of gamma shape parameters, when `family = "Gamma"`. Default is 1 vector.

Value

`coef`	the estimated P-spline coefficient regressor, using the effective regressors.
`w`	`wts*w`, GLM weight vector times input weights of length m.
`f`	the `lsfit` object using data augmentation to get P-spline coefficient estimates.
`eta`	the linear predictor from `f`.

`psPoisson` *Smoothing scattered Poisson data using P-splines.*

Description

`psPoisson` is used to smooth scattered Poisson data using P-splines with a log link function.

Usage

```
psPoisson(
  x,
  y,
  xl = min(x),
  xr = max(x),
  nseg = 10,
  bdeg = 3,
  pord = 2,
  lambda = 1,
  wts = NULL,
  show = FALSE,
  iter = 100,
  xgrid = 100
)
```

Arguments

`x`	the vector for the continuous regressor of `length(y)` and the abcissae used to build the B-spline basis.
`y`	the response vector, usually count data.
`xl`	the number for the min along `x` (default is min(x)).
`xr`	the number for the max along `x` (default is max(x)).
`nseg`	the number of evenly spaced segments between `xl` and `xr` (default 10).
`bdeg`	the number of the degree of the basis, usually 1, 2, or 3 (defalult).
`pord`	the number of the order of the difference penalty, usually 1, 2 (default), or 3.
`lambda`	the (positive) number for the tuning parameter for the penalty (default 1).
`wts`	the vector of general weights, zeros are allowed (default 1).
`show`	Set to TRUE or FALSE to display iteration history (default FALSE).
`iter`	a scalar to set the maximum number of iterations, default `iter=100`.
`xgrid`	a scalar or a vector that gives the `x` locations for prediction, useful for plotting. If a scalar (default 100) is used then a uniform grid of this size along (`xl`, `xr`).

Value

`pcoef`	a vector of length `n` of estimated P-spline coefficients.
`muhat`	a vector of length `m` of estimated means.
`B`	the `m` by `n` B-spline basis.
`dev`	deviance of fit.
`effdim`	effective dimension of fit.
`aic`	AIC.
`wts`	the vector of given prior weights.
`nseg`	the number of B-spline segments.
`bdeg`	the degree of the B-spline basis.
`pord`	the order of the difference penalty.
`lambda`	the positive tuning parameter.
`family`	the family of the response (`"Poisson"`).
`link`	the link function used (`"log"`).
`xgrid`	gridded `x` values, useful for plotting.
`ygrid`	gridded fitted linear predictor values, useful for plotting.
`mugrid`	gridded (inverse link) fitted mean values, useful for plotting.
`se_eta`	gridded standard errors for the linear predictor.
`dispersion`	Dispersion parameter estimated `dev/(m-effdim)`.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Eilers, P.H.C., Marx, B.D., and Durban, M. (2015). Twenty years of P-splines, *SORT*, 39(2): 149-186.

Examples

```
library(JOPS)
library(boot)

# Extract the data
Count <- hist(boot::coal$date, breaks = c(1851:1963), plot = FALSE)$counts
Year <- c(1851:1962)
xl <- min(Year)
xr <- max(Year)

# Poisson smoothing
nseg <- 20
```

```
bdeg <- 3
fit1 <- psPoisson(Year, Count, xl, xr, nseg, bdeg, pord = 2, lambda = 1)
plot(fit1, xlab = "Year", ylab = "Count", se = 2)
```

`psSignal` *Smooth signal (multivariate calibration) regression using P-splines.*

Description

Smooth signal (multivariate calibration) regression using P-splines.

Usage

```
psSignal(
  y,
  x_signal,
  x_index = c(1:ncol(x_signal)),
  nseg = 10,
  bdeg = 3,
  pord = 3,
  lambda = 1,
  wts = 1 + 0 * y,
  family = "gaussian",
  link = "default",
  m_binomial = 1 + 0 * y,
  r_gamma = wts,
  y_predicted = NULL,
  x_predicted = x_signal,
  ridge_adj = 0,
  int = TRUE
)
```

Arguments

`y`	a (glm) response vector, usually continuous, binomial or count data.
`x_signal`	a matrix of continuous regressor with `nrow(x_signal)` == `length(y)`, often a discrete digitization of a signal or histogram or time series.
`x_index`	a vector to of length `ncol(x_signal)` == `p`, associated with the ordering index of the signal. Default is `1:ncol(x_signal)`.
`nseg`	the number of evenly spaced segments between `xl` and `xr` (default 10).
`bdeg`	the degree of the basis, usually 1, 2, or 3 (defalult).
`pord`	the order of the difference penalty, usually 1, 2, or 3 (defalult).
`lambda`	the (positive) tuning parameter for the penalty (default 1).
`wts`	the weight vector of `length(y)`; default is 1.

`family`	the response distribution, e.g. `"gaussian"`, `"binomial"`, `"poisson"`, `"Gamma"` distribution; quotes are needed. Default is `"gaussian"`.
`link`	the link function, one of `"identity"`, `"log"`, `"sqrt"`, `"logit"`, `"probit"`, `"cloglog"`, `"loglog"`, `"reciprocal"`; quotes are needed (default `"identity"`).
`m_binomial`	a vector of binomial trials having length(y); default is 1 vector for `family = "binomial"`, NULL otherwise.
`r_gamma`	a vector of gamma shape parameters. Default is 1 vector for `family = "Gamma"`, NULL otherwise.
`y_predicted`	a vector of responses associated with `x_predicted` which are used to calculate standard error of external prediction. Default is NULL.
`x_predicted`	a matrix of external signals to yield external prediction.
`ridge_adj`	A ridge penalty tuning parameter, which can be set to small value, e.g. `1e-8` to stabilize estimation, (default 0).
`int`	set to TRUE or FALSE to include intercept term in linear predictor (default TRUE).

Details

Support functions needed: `pspline_fitter`, `bbase` and `pspline_checker`.

Value

`coef`	a vector with `length(n)` of estimated P-spline coefficients.
`mu`	a vector with `length(m)` of estimated means.
`eta`	a vector of `length(m)` of estimated linear predictors.
`B`	the B-spline basis (for the coefficients), with dimension `p` by `n`.
`deviance`	the deviance of fit.
`eff_df`	the approximate effective dimension of fit.
`aic`	AIC.
`df_resid`	approximate df residual.
`beta`	a vector of length `p`, containing estimated smooth signal coefficients.
`std_beta`	a vector of length `p`, containing standard errors of smooth signal coefficients.
`cv`	leave-one-out standard error prediction, when `family = "gaussian"`.
`cv_predicted`	standard error prediction for `y_predict`, when `family = "gaussian"`, NULL otherwise.
`nseg`	the number of evenly spaced B-spline segments.
`bdeg`	the degree of B-splines.
`pord`	the order of the difference penalty.
`lambda`	the positive tuning parameter.
`family`	the family of the response.
`link`	the link function.

`y_intercept`	the estimated y-intercept (when `int = TRUE`.)
`int`	a logical variable related to use of y-intercept in model.
`dispersion_param`	estimate of dispersion, `Dev/df_resid`.
`summary_predicted`	inverse link prediction vectors, and twice se bands.
`eta_predicted`	estimated linear predictor of `length(y)`.
`press_mu`	leave-one-out prediction of mean, when `family = "gaussian"`, NULL otherwise.
`bin_percent_correct`	percent correct classification based on 0.5 cut-off, when `family = binomial`, NULL otherwise.
`x_index`	a vector to of length `ncol(x_signal) == p`, associated with the ordering of the signal.

Author(s)

Brian Marx

References

Marx, B.D. and Eilers, P.H.C. (1999). Generalized linear regression for sampled signals and curves: A P-spline approach. *Technometrics*, 41(1): 1-13.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(JOPS)
# Get the data
library(fds)
data(nirc)
iindex <- nirc$x
X <- nirc$y
sel <- 50:650 # 1200 <= x & x<= 2400
X <- X[sel, ]
iindex <- iindex[sel]
dX <- diff(X)
diindex <- iindex[-1]
y <- as.vector(labc[1, 1:40]) # percent fat
oout <- 23
dX <- t(dX[, -oout])
y <- y[-oout]
fit1 <- psSignal(y, dX, diindex, nseg = 25, bdeg = 3, lambda = 0.0001,
pord = 2, family = "gaussian", link = "identity", x_predicted = dX, int = TRUE)
plot(fit1, xlab = "Coefficient Index", ylab = "ps Smooth Coeff")
title(main = "25 B-spline segments with tuning = 0.0001")
names(fit1)
```

`psVCSignal` *Varying-coefficient penalized signal regression using P-splines.*

Description

`psVCSignal` is used to regress a (glm) response onto a signal such that the signal coefficients can vary over another covariate `t`. Anisotripic penalization of tensor product B-splines produces a 2D coefficient surface that can be sliced at `t`.

@details Support functions needed: `pspline_fitter`, `pspline_2dchecker`, and `bbase`.

@import stats

Usage

```
psVCSignal(
  y,
  X,
  x_index,
  t_var,
 Pars = rbind(c(min(x_index), max(x_index), 10, 3, 1, 2), c(min(t_var), max(t_var), 10,
    3, 1, 2)),
  family = "gaussian",
  link = "default",
  m_binomial = 1 + 0 * y,
  wts = 1 + 0 * y,
  r_gamma = 1 + 0 * y,
  X_pred = X,
  t_pred = t_var,
  y_predicted = NULL,
  ridge_adj = 1e-08,
  int = TRUE
)
```

Arguments

`y`	a glm response vector of length `m`, usually continuous, binary/bimomial or counts.
`X`	a `m` by `p1` Signal matrix of regressors.
`x_index`	`p1`-vector for index of Signal (e.g. wavelength).
`t_var`	`p2`-vector with other (indexing) variable in coefficient surface (e.g. temperature, depth, time).
`Pars`	a matrix with 2 rows, each with P-spline parameters: `min` `max` `nseg` `bdeg` `lambda` `pord`, for row and columns of tensor product surface; defaults are min and max for `x_index` and `t_var` (resp.), `nseg` = 10, `bdeg` =3, `lambda` = 1, `pord` = 2.
`family`	the response distribution, e.g. `"gaussian"`, `"binomial"`, `"poisson"`, `"Gamma"` distribution; quotes are needed (default `"gaussian"`.

`link`	the link function, one of `"identity"`, `"log"`, `"sqrt"`, `"logit"`, `"probit"`, `"cloglog"`, `"loglog"`, `"reciprocal"`); quotes are needed (default `"identity"`.
`m_binomial`	a vector of binomial trials having `length(y)`. Default is 1 vector for `family = "binomial"`, NULL otherwise.
`wts`	a `m` vector of weights (default 1).
`r_gamma`	a vector of gamma shape parameters. Default is 1 vector for `family = "Gamma"`, NULL otherwise.
`X_pred`	a matrix of signals with `ncol(X)` columns for prediction, default is `X`.
`t_pred`	a vector for the VC indexing variable with length `nrow(X_pred)`, default is `t_var`.
`y_predicted`	a vector for the responses associated with `X_pred` with length `nrow(X_pred)` useful for CV when `family = "binomial"`, default is `NULL`.
`ridge_adj`	a small ridge penalty tuning parameter to regularize estimation (default `1e-8`).
`int`	intercept set to TRUE or FALSE for intercept term.

Value

`pcoef`	a vector of length `(Pars[1,3]+Pars[1,4])*(Pars[2,3]+Pars[2,4])` of estimated P-spline coefficients for tensor surface.
`summary_predicted`	inverse link prediction vectors, and twice se bands.
`dev`	the deviance of fit.
`eff_dim`	the approximate effective dimension of fit.
`family`	the family of the response.
`link`	the link function.
`aic`	AIC.
`df_resid`	approximate df residual.
`cv`	leave-one-out standard error prediction when `family = "gaussian"`, NULL otherwise.
`cv_predicted`	standard error prediction for `y_predict` when `family = "gaussian"`, NULL otherwise.
`Pars`	design and tuning parameters; see arguments above.
`dispersion_parm`	estimate of dispersion, `Dev/df_resid`.
`summary_predicted`	inverse link prediction vectors, and twice se bands.
`eta_predicted`	estimated linear predictor of `length(y)`.
`press_mu`	leave-one-out prediction of mean when `family = "gaussian"`, NULL otherwise.
`bin_percent_correct`	percent correct classification based on 0.5 cut-off when `family = "binomial"`, NULL otherwise.

`Bx`	B-spline basis matrix of dimension `p1` by `n1`, along `x_index`.
`By`	B-spline basis matrix of dimension `p2` by `n2`, along `t_var`.
`Q`	Modified tensor basis (`m` by `(n1*n2)`) for VC signal regression.
`yint`	the estimated y-intercept (when `int = TRUE`.)
`int`	a logical variable related to use of y-intercept in model.

Author(s)

Paul Eilers and Brian Marx

References

Eilers, P.H.C. and Marx, B.D. (2003). Multivariate calibration with temperature interaction using two-dimensional penalized signal regression. *Chemometrics and Intellegent Laboratory Systems*, 66, 159–174.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(fds)
data(nirc)
iindex <- nirc$x
X <- nirc$y
sel <- 50:650 # 1200 <= x & x<= 2400
X <- X[sel, ]
iindex <- iindex[sel]
dX <- diff(X)
diindex <- iindex[-1]
y <- as.vector(labc[1, 1:40]) # percent fat
t_var <- as.vector(labc[4, 1:40]) # percent flour
oout <- 23
dX <- t(dX[, -oout])
y <- y[-oout]
t_var = t_var[-oout]
Pars = rbind(c(min(diindex), max(diindex), 25, 3, 1e-7, 2),
c(min(t_var), max(t_var), 20, 3, 0.0001, 2))
fit1 <- psVCSignal(y, dX, diindex, t_var, Pars = Pars,
family = "gaussian", link = "identity", int = TRUE)
plot(fit1, xlab = "Coefficient Index", ylab = "VC: % Flour")
names(fit1)
```

`rdw` *Observations on the widths of red blood cell distributions (RDW).*

Description

Observations on the widths of red blood cell distributions (RDW).

Usage

```
data(rdw)
```

Format

A vector.

Source

Erasmus University Medical Centre, Rotterdam, The Netherlands

Examples

```
data(rdw)
hist(rdw, breaks = 20)
```

`rowtens` *Compute the row tensor product of two matrices*

Description

Compute the row tensor product of two matrices with identical numbers of rows.

Usage

```
rowtens(X, Y = X)
```

Arguments

`X`	a numeric matrix.
`Y`	a numeric matrix (if missing, `Y = x`).

Details

The input matrices must have the same number of rows, say `m`. If their numbers of columns are `n1` and `n2`, the result is a matrix with `m` rows and `n1 * n2` columns. Each row of the result is the Kronecker product of the corresponding rows of `X` and `Y`.

Value

The row-wise tensor product of the two matrices.

Author(s)

Paul Eilers

References

Eilers, P. H. C. and Currie, I. D. and Durban, M. (2006) Fast and compact smoothing on large multidimensional grids *CSDA* 50, 61–76.

`save_PDF` *Save a plot as a PDF file.*

Description

Save a plot as a PDF file in a (default) folder. The present default is determined by the folder structure for the production of the book.

Usage

```
save_PDF(
  fname = "scratch",
  folder = "../../Graphs",
  show = T,
  width = 6,
  height = 4.5
)
```

Arguments

`fname`	the file name without the extension PDF (default: `scratch`).
`folder`	the folder for saving PDF plots (default `../../Graphs`).
`show`	a logical parameter; if TRUE the full file name will be displayed.
`width`	figure width in inches (default = 6).
`height`	figure height in inches (default = 4.5).

Value

save a plot as a PDF file.

Author(s)

Paul Eilers

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

`set_panels` *Prepare graphics layout for multiple panels*

Description

Adapt margins and axes layout for multiple panels.

Usage

```
set_panels(rows = 1, cols = 1)
```

Arguments

`rows`	number of rows.
`cols`	number of columns.

Value

Prepare graphics layout for multiple panels

Author(s)

Paul Eilers

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

`set_window` *Open a graphics window.*

Description

Open a a window for graphics, with specified width and height.

Usage

```
set_window(width = 6, height = 4.5, kill = TRUE, noRStudioGD = TRUE)
```

Arguments

`width`	figure width in inches (default = 6).
`height`	figure height in inches (default = 4.5).
`kill`	if TRUE (default) closes all graphics windows. Works only for Windows.
`noRStudioGD`	if TRUE: do not use the RStudio device (which does not accept width and height).

Value

open a graphics window.

Note

Currently only works for Windows!

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

`sim_psr` *Single-Index signal regression using P-splines*

Description

`sim_psr` is a single-index signal regression model that estimates both the signal coefficients vector and the unknown link function using P-splines.

Usage

```
sim_psr(
  y,
  X,
  x_index = c(1:ncol(X)),
  nsegs = rep(10, 2),
  bdegs = rep(3, 3),
  lambdas = rep(1, 2),
  pords = rep(2, 2),
  max_iter = 100
)
```

Arguments

`y`	a response vector of length `m`, usually continuous.
`X`	The signal regressors with dimension `m` by `p`.
`x_index`	an index of length `p` for columns of signal matrix; default is simple sequence, `c(1: ncol(X))`.
`nsegs`	a vector of length 2 containing the number of evenly spaced segments between min and max, for each the coefficient vector and the (unknown) link function, resp. (default `c(10, 10)`).
`bdegs`	a vector of length 2 containing the degree of B-splines, for the coefficient vector and the (unknown) link function, resp. (default cubic or `c(3, 3)`).
`lambdas`	a vector of length 2 containing the positive tuning parameters, for each the coefficient vector and the (unknown) link function, resp. (default `c(1, 1)`).

`pords`	a vector of length 2 containing the difference penalty order, for each the coefficient vector and the (unknown) link function, resp. (default`c(2, 2)`).
`max_iter`	a scalar for the maximum number of iterations (default 100).

Value

`y`	the response vector of length `m`.
`alpha`	the P-spline coefficient vector of length `(nsegs[1]+bdeg[1])`.
`iter`	the number of iterations used for the single-index fit.
`yint`	the estimated y-intercept for the single-index model.
`B`	the B-spline matrix built along the signal index, using `nsegs[1]`, used for the coefficient vector.
`Q`	the effective regressors from the `psVCSignal` portion of the single-index fit with dimension `m` by `length(alpha)`.
`nsegs`	a vector of length 2 containing the number of evenly spaced segments between min and max, for each the coefficient vector and the link function, resp.
`bdegs`	a vector of length 2 containing the degree of B-splines, for each the coefficient vector and the link function, resp.
`lambdas`	a vector of length 2 containing the positive tuning parameters, for each the coefficient vector and the link function, resp.
`pords`	a vector of length 2 containing the difference penalty order, for each the coefficient vector and the link function, resp.
`eta`	the estimated linear predictor for the single-index fit.
`cv`	the leave-one-out cross-validation statistic or the standard error of prediction for the single-index fit.
`delta_alpha`	change measure in signal-coefficent parameters at convervence.
`x_index`	the index of length `p` for columns of signal matrix.
`f_fit`	the `psNormal` object, fitting link function f(`eta`).
`f_eta`	the predicted values of the link function estimated with `f_fit` or estimated f(`eta`), at `x` = `eta`.

Author(s)

Paul Eilers, Brian Marx, and Bin Li

References

Eilers, P.H.C., B. Li, B.D. Marx (2009). Multivariate calibration with single-index signal regression, *Chemometrics and Intellegent Laboratory Systems*, 96(2), 196-202.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(JOPS)
# Get the data
library(fds)
data(nirc)
iindex <- nirc$x
X <- nirc$y
sel <- 50:650 # 1200 <= x & x<= 2400
X <- X[sel, ]
iindex <- iindex[sel]
dX <- diff(X)
diindex <- iindex[-1]
y <- as.vector(labc[1, 1:40])
oout <- 23
dX <- t(dX[, -oout])
y <- y[-oout]

pords <- c(2, 2)
nsegs <- c(27, 7)
bdegs = c(3, 3)
lambdas <- c(1e-6, .1)
max_iter <- 100

# Single-index model
fit <- sim_psr(y, dX, diindex, nsegs, bdegs, lambdas, pords,
             max_iter)

plot(fit, xlab = "Wavelength (nm)", ylab = " ")
```

`sim_vcpsr` *Varying-coefficient single-index signal regression using tensor P-splines.*

Description

`sim_vcpsr` is a varying-coefficient single-index signal regression approach that allows both the signal coefficients and the unknown link function to vary with an indexing variable `t`, e.g. temperature. Two surfaces are estimated (coefficent and link) that can be sliced at arbitary `t`. Anisotripic penalization with P-splines is used on both.

Usage

```
sim_vcpsr(
  y,
  X,
  t_var,
  x_index = c(1:ncol(X)),
  nsegs = rep(10, 4),
```

```
  bdegs = rep(3, 4),
  lambdas = rep(1, 4),
  pords = rep(2, 4),
  max_iter = 100,
  mins = c(min(x_index), min(t_var)),
  maxs = c(max(x_index), max(t_var))
)
```

Arguments

`y`	a response vector of length `m`, usually continuous.
`X`	the signal regressors with dimension `m` by `p1`.
`t_var`	the varying coeffient indexing variable of length `m`.
`x_index`	an index of length `p` for columns of signal matrix; default is simple sequence.
`nsegs`	a vector of length 4 containing the number of evenly spaced segments between min and max, for each the coefficient surface (row and col) and link surface (row and col), resp. (default `rep(10, 4)`.
`bdegs`	a vector of length 4 containing the degree of B-splines, for each the coefficient surface (row and col) and link surface (row and col), resp. (default cubic `rep(3, 4)`).
`lambdas`	a vector of length 4 containing the positive tuning parameters, for each the coefficient surface (row and col) and link surface (row and col), resp. (default `rep(1, 4)`).
`pords`	a vector of length 4 containing the difference penalty order, for each the coefficient surface (row and col) and link surface (row and col), resp. (default `rep(2, 4)`).
`max_iter`	a scalar for the maximum number of iterations (default 100)
`mins`	A vector length 2, containing min for signal index and `t_var`, default associated with `x_index` and `t_var` minimums; default is respective minimums.
`maxs`	A vector length 2, containing max for signal index and `t_var`, default associated with `x_index` and `t_var` maximums; default is respective maximums.

Value

`y`	the response vector of length `m`.
`alpha`	the P-spline coefficient vector (unfolded) of length `(nsegs[1]+bdeg[1])*(negs[2]+bdeg[2])`.
`iter`	the number of iterations used for the single-index fit.
`yint`	the estimated y-intercept for the single-index model.
`Bx`	the B-spline matrix built along the signal index, using `nsegs[1]`, used for the coefficient surface.
`By`	the B-spline matrix built along the `t_var` index, using `nsegs[2]`, used for the coefficient surface.
`Q`	the effective regressors from the `psVCSignal` portion of the single-index fit with dimension `m` by `length(alpha)`.

`t_var`	the VC indexing variable of length `m`.
`nsegs`	a vector of length 4 containing the number of evenly spaced segments between min and max, for each the coefficient surface (row and col) and link surface (row and col).
`bdegs`	a vector of length 4 containing the degree of B-splines, for each the coefficient surface (row and col) and link surface (row and col).
`lambdas`	a vector of length 4 containing the positive tuning parameters, for each the coefficient surface (row and col) and link surface (row and col).
`pords`	a vector of length 4 containing the difference penalty order, for each the coefficient surface (row and col) and link surface (row and col).
`mins`	a vector length 2, containing min for signal index and `t_var`.
`maxs`	a vector length 2, containing max for signal index and `t_var`.
`eta`	the estimated linear predictor for the single-index fit.
`Pars`	a matrix of 2 rows associated with the signal coefficient surface design parameters, each row: `c(min, max, nseg, bdeg, lambda, pord)` for linear predictor `x_index` and `t_var`, resp.
`pPars`	a matrix of 2 rows associated with the link function design parameters, each row: `c(min, max, nseg, bdeg, lambda, pord)` for linear predictor `eta` and `t_var`, resp.
`cv`	the leave-one-out cross-validation statistic or the standard error of prediction for the single-index fit.
`delta_alpha`	change measure in signal-coefficent parameters at convergence.
`fit2D`	`ps2DNormal` object, fitting f(`eta`, `t_var`).

Author(s)

Paul Eilers and Brian Marx

References

Marx, B. D. (2015). Varying-coefficient single-index signal regression. *Chemometrics and Intelligent Laboratory Systems*, 143, 111–121.

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
# Load libraries
library(fields) # Needed for plotting

# Get the data
Dat <- Mixture

# Dimensions: observations, temperature index, signal
m <- 34
p1 <- 401
```

```
p2 <- 12

# Stacking mixture data, each mixture has 12 signals stacked
# The first differenced spectra are also computed.
mixture_data <- matrix(0, nrow = p2 * m, ncol = p1)
for (ii in 1:m)
{
  mixture_data[((ii - 1) * p2 + 1):(ii * p2), 1:p1] <-
    t(as.matrix(Dat$xspectra[ii, , ]))
  d_mixture_data <- t(diff(t(mixture_data)))
}

# Response (typo fixed) and index for signal
y_mixture <- Dat$fractions
y_mixture[17, 3] <- 0.1501
index_mixture <- Dat$wl

# Select response and replicated for the 12 temps
# Column 1: water; 2: ethanediol; 3: amino-1-propanol
y <- as.vector(y_mixture[, 2])
y <- rep(y, each = p2)

bdegs = c(3, 3, 3, 3)
pords <- c(2, 2, 2, 2)
nsegs <- c(12, 5, 5, 5) # Set to c(27, 7, 7 ,7) for given lambdas
mins <- c(700, 30)
maxs <- c(1100, 70)
lambdas <- c(1e-11, 100, 0.5, 1) # based on svcm search
x_index <- seq(from = 701, to = 1100, by = 1) # for dX
t_var_sub <- c(30, 35, 37.5, 40, 45, 47.5, 50, 55, 60, 62.5, 65, 70)
t_var <- rep(t_var_sub, m)
max_iter <- 2 # Set higher in practice, e.g. 100
int <- TRUE

# Defining x as first differenced spectra, number of channels.
x <- d_mixture_data


# Single-index VC model using optimal tuning
fit <- sim_vcpsr(y, x, t_var, x_index, nsegs, bdegs, lambdas, pords,
             max_iter = max_iter, mins = mins, maxs = maxs)

plot(fit, xlab = "Wavelength (nm)", ylab = "Temp C")
```

`SpATS.nogeno` *Two-dimensional P-spline smoothing*

Description

Two-dimensional smoothing of scattered data points with tensor product P-splines.

Usage

```
SpATS.nogeno(
  response,
  spatial,
  fixed = NULL,
  random = NULL,
  data,
  family = gaussian(),
  offset = 0,
  weights = NULL,
  control = list(maxit = 100)
)
```

Arguments

`response`	a character string with the name of the variable that contains the response variable of interest.
`spatial`	a right hand `formula` object specifying the spatial P-Spline model. See `SAP` and `PSANOVA` for more details about how to specify the spatial trend.
`fixed`	an optional right hand `formula` object specifying the fixed effects.
`random`	an optional right hand `formula` object specifying the random effects. Currently, only sets of independent and identically distributed random effects can be incorporated.
`data`	a data frame containing the variables.
`family`	object of class `family` specifying the distribution and link function.
`offset`	an optional numerical vector containing an a priori known component to be included in the linear predictor during fitting.
`weights`	an optional numerical vector of weights to be used in the fitting process. By default, the weights are considered to be one.
`control`	a list of control values.

Details

This function is a modified version of the function `SpATS` in the package `SpATS`. The difference is that genotypes have been removed.

Value

A list with the following components:

`call`	the matched call.
`data`	the original supplied data argument with a new column with the weights used during the fitting process.
`model`	a list with the model components: response, spatial, fixed and/or random.
`fitted`	a numeric vector with the fitted values.

`residuals`	a numeric vector with deviance residuals.
`psi`	a two-length vector with the values of the dispersion parameters at convergence. For Gaussian responses both elements coincide, being the (REML) estimate of dispersion parameter. For non-Gaussian responses, the result depends on the argument `update.psi` of the `controlSpATS` function. If this argument was specified to `FALSE` (the default), the first component of the vector corresponds to the default value used for the dispersion parameter (usually 1). The second element, correspond to the (REML) estimate of the dispersion parameter at convergence. If the argument `update.psi` was specified to `TRUE`, both components coincide (as in the Gaussian case).
`var.comp`	a numeric vector with the (REML) variance component estimates. This vector contains the variance components associated with the spatial trend, as well as those related with the random model terms.
`eff.dim`	a numeric vector with the estimated effective dimension (or effective degrees of freedom) for each model component (spatial, fixed and/or random).
`dim`	a numeric vector with the (model) dimension of each model component (spatial, fixed and/or random). This value corresponds to the number of parameters to be estimated.
`dim.nom`	a numeric vector with the (nominal) dimension of each component (spatial, fixed and/or random). For the random terms of the model, this value corresponds to upper bound for the effective dimension (i.e., the maximum effective dimension a random term can achive). This nominal dimension is $rank[X, Z_k] - rank[X]$, where Z_k is the design matrix of the kth random factor and X is the design matrix of the fixed part of the model. In most cases (but not always), the nominal dimension corresponds to the model dimension minus one, “lost” due to the implicit constraint that ensures the mean of the random effects to be zero.
`nobs`	number of observations used to fit the model.
`niterations`	number of iterations EM-algorithm.
`deviance`	the (REML) deviance at convergence (i.e., -2 times the restricted log-likelihood).
`coeff`	a numeric vector with the estimated fixed and random effect coefficients.
`terms`	a list with the model terms: response, spatial, fixed and/or random. The information provided here is useful for printing and prediction purposes.
`vcov`	inverse of the coefficient matrix of the mixed models equations. The inverse is needed for the computation of standard errors. For computational issues, the inverse is returned as a list: C22_inv corresponds to the coefficient matrix associated with the spatial, the fixed and the random components.

Author(s)

Maria-Xose Rodriguez-Alvarez and Paul Eilers

References

Rodriguez-Alvarez, M.X, Boer, M.P., van Eeuwijk, F.A., and Eilers, P.H.C. (2018). Correcting for spatial heterogeneity in plant breeding experiments with P-splines. Spatial Statistics, 23, 52 - 71. https://doi.org/10.1016/j.spasta.2017.10.003.

Examples

```
# Get the data
data(ethanol)

# Fit the PS-ANOVA model
ps2d <- SpATS.nogeno(response = "NOx",
                     spatial = ~PSANOVA(E, C,  nseg = c(20, 20), nest.div = c(2, 2)),
                     data = ethanol,
                     control = list(maxit = 100, tolerance = 1e-05,
                                    monitoring = 0, update.psi = FALSE))

# Report effective dimensions, if desired
# print(summary(ps2d))

# Compute component surface and their sum on a fine grid
Tr = obtain.spatialtrend(ps2d, grid = c(100, 100))

# Plot surface and contours
image(Tr$row.p, Tr$col.p, Tr$fit, col = terrain.colors(100), xlab = 'C', ylab = 'E')
contour(Tr$row.p, Tr$col.p, Tr$fit, add = TRUE, col = 'blue')
points(ethanol$C, ethanol$E, pch = '+')
```

`spbase` *Compute a sparse B-spline basis on evenly spaced knots*

Description

Constructs a sparse B-spline basis on evenly spaced knots.

Usage

```
spbase(x, xl = min(x), xr = max(x), nseg = 10, bdeg = 3)
```

Arguments

`x`	a vector of argument values, at which the B-spline basis functions are to be evaluated.
`xl`	the lower limit of the domain of `x` (default `min(x)`).
`xr`	the upper limit of the domain of `x` (default `max(x)`) .
`nseg`	the number of evenly spaced segments between `xl` and `xr` (default 10).
`bdeg`	the degree of the basis, usually 1, 2, or 3 (default).

Value

A sparse matrix (in `spam` format) with `length(x)` of rows= and `nseg` + `bdeg` columns.

Author(s)

Paul Eilers

References

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

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(JOPS)
# Basis  on grid
x = seq(0, 4, length = 1000)
B = spbase(x, 0, 4, nseg = 50, bdeg = 3)
nb1 = ncol(B)
matplot(x, B, type = 'l', lty = 1, lwd = 1, xlab = 'x', ylab = '')
cat('Dimensions of B:', nrow(B), 'by', ncol(B), 'with', length(B@entries), 'non-zero elements' )
```

`Sugar` *Sugar Processing Data*

Description

Sugar was sampled continuously during eight hours to make a mean sample representative for one "shift" (eight hour period). Samples were taken during the three months of operation (the so-called campaign) in late autumn from a sugar plant in Scandinavia giving a total of 268 samples. The sugar was sampled directly from the final unit operation (centrifuge) of the process.

Usage

```
data(Sugar)
```

Format

A list consisting of the following:

`y`	a 268 x 3 matrix of quality parameters: `date`, `color`, `ash*1000`
`X`	fluoresence array, 268 (observations) x [571 (emission channels) x 7 (excitation channels)]
`Lab`	Lab information
`DimX`	array dimension for `X`
`Yidx`	names (id) for `y`
`EmAx`	Emmission levels for axis (nm)
`ExAx`	Excitation levels for axis (nm)

`time`

`readmetime`

`Lname`

`LabNumber`

`ProcNumber`

`Proc`

`DimLab`

`DimProc`

Source

https://ucphchemometrics.com/sugar-process-data/

References

R. Bro, Exploratory study of sugar production using fluorescence spectroscopy and multi-way analysis, *Chemom. Intell. Lab. Syst.*, 1999, (46), 133-147.

`Suicide` *Suicide Data Set*

Description

The dataset comprises lengths (in days) of psychiatric treatment spells for patients used as controls in a study of suicide risks.

Usage

```
data(Suicide)
```

Format

A dataframe with one column: `y`.

Source

Silverman, B. (1986). Density Estimation for Statistics and Data Analysis. Chapman & Hall.

References

Silverman, B. (1986). Density Estimation for Statistics and Data Analysis. Chapman & Hall.

`tpower` *Compute a truncated power function.*

Description

Compute a truncated power function.

Usage

```
tpower(x, knot, p)
```

Arguments

`x`	a vector on which the basis is calculated.
`knot`	a scalar giving the truncation point.
`p`	a scalar power for the basis, e.g. `p = 3` for cubic TPF.

Value

a vector with the truncated power function.

Author(s)

Paul Eilers

References

Eilers, P.H.C. and Marx, B.D. (2021). *Practical Smoothing, The Joys of P-splines.* Cambridge University Press.

Examples

```
library(JOPS)
# Basis  on grid
x = seq(0, 4, length = 500)
knots = 0:3
Y = outer(x, knots, tpower, 1)
matplot(x, Y, type ='l', lwd = 2, xlab = 'x', ylab = '',
main ='Linear TPF basis')
```

`Varstar` *Brightness of a variable star.*

Description

Brightness of a variable star.

Usage

```
data(Varstar)
```

Format

A dataframe with eleven columns (`V1-V11`):

`V1`	day index
`V2`	brightness
`V3-V11`	Paul Eilers, personal communication.

References

Paul Eilers (personal communication).

`Woodsurf` *Profile of a sanded piece of wood.*

Description

Profile of a sanded piece of wood.

Usage

```
data(Woodsurf)
```

Format

A data frame with one column: `y`.

Source

Pandit, S.M. and Wu, S.M. (1993). *Time Series and System Analysis with Applications*. Krieger Publishing Company.

Index