Package ‘survPen’

March 20, 2026

Title Multidimensional Penalized Splines for (Excess) Hazard Models,
Relative Mortality Ratio Models and Marginal Intensity Models

Version 2.0.3

Description Fits (excess) hazard, relative mortality ratio or marginal intensity models with multidimensional penalized splines allowing for
time-dependent effects, non-linear effects and interactions between several continuous covariates. In survival and net survival analysis, in addition to modelling the effect of time (via the baseline hazard), one has often to deal with several continuous covariates and model their functional forms, their time-dependent effects, and their interactions. Model specification becomes therefore a complex problem and penalized regression splines represent an appealing solution to that problem as splines offer the required flexibility while penalization limits overfitting issues. Current implementations of penalized survival models can be slow or unstable and sometimes lack some key features like taking into account expected mortality to provide net survival and excess hazard estimates. In contrast, survPen provides an automated, fast, and stable implementation (thanks to explicit calculation of the derivatives of the likelihood) and offers a unified framework for
multidimensional penalized hazard and excess hazard models. Later versions (>2.0.0) include penalized models for relative mortality ratio, and marginal intensity in recurrent event setting.
survPen may be of interest to those who 1) analyse any kind of time-to-event data: mortality, disease relapse, machinery breakdown, unemployment, etc 2) wish to describe the associated hazard and to understand which predictors impact its dynamics,
3) wish to model the relative mortality ratio between a cohort and a reference population, 4) wish to describe the marginal intensity for recurrent event data.
See Fauvernier et al. (2019a) <doi:10.21105/joss.01434> for an overview of the package and Fauvernier et al. (2019b) <doi:10.1111/rssc.12368> for the method.

Depends R (>= 4.0.0)

License GPL-3 | file LICENSE

Imports statmod, stats, Rcpp (>= 1.0.2)

LinkingTo Rcpp, RcppEigen

URL https://github.com/fauvernierma/survPen

BugReports https://github.com/fauvernierma/survPen/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-03-20 14:30:02 UTC

Author Mathieu Fauvernier [aut, cre],
Laurent Roche [aut],
Laurent Remontet [aut],
Zoe Uhry [ctb],
Nadine Bossard [ctb],
Elsa Coz [ctb]

Maintainer Mathieu Fauvernier <`mathieu.fauvernier@gmail.com`>

Contents

`colSums2` *colSums of a matrix*

Description

colSums of a matrix

Usage

```
colSums2(Mat)
```

Arguments

`Mat`	a matrix.

Value

colSums(Mat)

`constraint` *Sum-to-zero constraint*

Description

Applies the sum-to-zero constraints to design and penalty matrices.

Usage

```
constraint(X, S, Z = NULL)
```

Arguments

`X`	A design matrix
`S`	A penalty matrix or a list of penalty matrices
`Z`	A list of sum-to-zero constraint matrices; default is NULL

Value

List of objects with the following items:

`X`	Design matrix
`S`	Penalty matrix or list of penalty matrices
`Z`	List of sum-to-zero constraint matrices

Examples

```
library(survPen)

set.seed(15)

X <- matrix(rnorm(10*3),nrow=10,ncol=3)
S <- matrix(rnorm(3*3),nrow=3,ncol=3) ; S <- 0.5*( S + t(S))

# applying sum-to-zero constraint to a desgin matrix and a penalty matrix
constr <- constraint(X,S)
```

`cor.var` *Implementation of the corrected variance Vc*

Description

Takes the model at convergence and calculates the variance matrix corrected for smoothing parameter uncertainty

Usage

```
cor.var(model)
```

Arguments

`model`	survPen object, see `survPen.fit` for details

Value

survPen object with corrected variance Vc

`crs` *Bases for cubic regression splines (equivalent to "cr" in* `mgcv`*)*

Description

Builds the design matrix and the penalty matrix for cubic regression splines.

Usage

```
crs(x, knots = NULL, df = 10, intercept = TRUE)
```

Arguments

`x`	Numeric vector
`knots`	Numeric vectors that specifies the knots of the splines (including boundaries); default is NULL
`df`	numeric value that indicates the number of knots desired (or degrees of freedom) if knots=NULL; default is 10
`intercept`	if FALSE, the intercept is excluded from the basis; default is TRUE

Details

See package `mgcv` and section 4.1.2 of Wood (2006) for more details about this basis

Value

List of three elements

`bs`	design matrix
`pen`	penalty matrix
`knots`	vector of knots (specified or calculated from `df`)

References

Wood, S. N. (2006), Generalized additive models: an introduction with R. London: Chapman & Hall/CRC.

Examples

```
x <- seq(1,10,length=100)
# natural cubic spline with 3 knots
crs(x,knots=c(1,5,10))
```

`crs.FP` *Penalty matrix constructor for cubic regression splines*

Description

constructs the penalty matrix associated with cubic regression splines basis. This function is called inside `crs`.

Usage

```
crs.FP(knots, h)
```

Arguments

`knots`	Numeric vectors that specifies the knots of the splines (including boundaries)
`h`	vector of knots differences (corresponds to `diff(sort(knots))`)

Value

List of two elements:

`F.mat`	matrix used in function `crs` for basis construction
`P.mat`	penalty matrix

Examples

```
library(survPen)

# construction of the penalty matrix using a sequence of knots
knots <- c(0,0.25,0.5,0.75,1)
diff.knots <- diff(knots)

crs.FP(knots,diff.knots)
```

`CumulHazard` *Cumulative hazard (integral of hazard) only*

Description

Cumulative hazard (integral of hazard) only

Usage

```
CumulHazard(X_GL, weights, tm, n_legendre, n, beta, is_pwcst, pwcst_weights)
```

Arguments

`X_GL`	list of matrices (`length(X.GL)=n.legendre`) for Gauss-Legendre quadrature
`weights`	vector of weights for Gauss-Legendre integration on [-1;1]
`tm`	vector of midpoints times for Gauss-Legendre integration; tm = 0.5*(t1 - t0)
`n_legendre`	number of nodes for Gauss-Legendre quadrature
`n`	number of individuals in the dataset
`beta`	vector of estimated regression parameters
`is_pwcst`	True if there is a piecewise constant baseline specified. False otherwise
`pwcst_weights`	if is_pwcst is TRUE, matrix of weights giving the time contribution of each individual on each sub-interval. Otherwise NULL

Value

cumulative hazard (integral of hazard)

`datCancer` *Patients diagnosed with cervical cancer*

Description

A simulated dataset containing the follow-up times of 2000 patients diagnosed with cervical cancer between 1990 and 2010. End of follow-up is June 30th 2013. The variables are as follows:

- begin. beginning of follow-up. For illustration purposes about left truncation only (0–1)
- fu. follow-up time in years (0–5)
- age. age at diagnosis in years, from 21.39 to 99.33
- yod. decimal year of diagnosis, from 1990.023 to 2010.999
- dead. censoring indicator (1 for dead, 0 for censored)
- rate. expected mortality rate (from overall mortality of the general population) (0–0.38)

Usage

```
data(datCancer)
```

Format

A data frame with 2000 rows and 6 variables

`DerivCumulHazard` *Cumulative hazard (integral of hazard) and its first and second derivatives wrt regression parameters beta*

Description

Cumulative hazard (integral of hazard) and its first and second derivatives wrt regression parameters beta

Usage

```
DerivCumulHazard(
  X_GL,
  weights,
  tm,
  n_legendre,
  n,
  p,
  beta,
  expected,
  type,
  is_pwcst,
  pwcst_weights
)
```

Arguments

`X_GL`	list of matrices (`length(X.GL)=n.legendre`) for Gauss-Legendre quadrature
`weights`	vector of weights for Gauss-Legendre integration on [-1;1]
`tm`	vector of midpoints times for Gauss-Legendre integration; tm = 0.5*(t1 - t0)
`n_legendre`	number of nodes for Gauss-Legendre quadrature
`n`	number of individuals in the dataset
`p`	number of regression parameters
`beta`	vector of estimated regression parameters
`expected`	vector of expected hazard rates
`type`	"net", "overall" or "mult"
`is_pwcst`	True if there is a piecewise constant baseline specified. False otherwise
`pwcst_weights`	if is.pwcst is TRUE, matrix of weights giving the time contribution of each individual on each sub-interval. Otherwise NULL

Value

List of objects with the following items:

`integral`	cumulative hazard (integral of hazard)
`f.first`	first derivative of cumulative hazard wrt beta
`f.second`	second derivative of cumulative hazard wrt beta

`deriv_R` *Derivative of a Choleski factor*

Description

Derivative of a Choleski factor

Usage

```
deriv_R(deriv_Vp, p, R1)
```

Arguments

`deriv_Vp`	derivatives of the Bayesian covariance matrix wrt rho (log smoothing parameters).
`p`	number of regression parameters
`R1`	Choleski factor of Vp

Value

a list containing the derivatives of R1 wrt rho (log smoothing parameters)

`design.matrix` *Design matrix for the model needed in Gauss-Legendre quadrature*

Description

Builds the design matrix for the whole model when the sum-to-zero constraints are specified. The function is called inside `model.cons` for Gauss-Legendre quadrature.

Usage

```
design.matrix(
  formula,
  data.spec,
  t1.name,
  Z.smf,
  Z.tensor,
  Z.tint,
  list.smf,
  list.tensor,
  list.tint,
  list.rd
)
```

Arguments

`formula`	formula object identifying the model
`data.spec`	data frame that represents the environment from which the covariate values and knots are to be calculated
`t1.name`	name of the vector of follow-up times
`Z.smf`	List of matrices that represents the sum-to-zero constraint to apply for `smf` splines
`Z.tensor`	List of matrices that represents the sum-to-zero constraint to apply for `tensor` splines
`Z.tint`	List of matrices that represents the sum-to-zero constraint to apply for `tint` splines
`list.smf`	List of all smf.smooth.spec objects contained in the model
`list.tensor`	List of all tensor.smooth.spec objects contained in the model
`list.tint`	List of all tint.smooth.spec objects contained in the model
`list.rd`	List of all rd.smooth.spec objects contained in the model

Value

design matrix for the model

Examples

```
library(survPen)

# standard spline of time with 4 knots

data <- data.frame(time=seq(0,5,length=100),event=1,t0=0)

form <- ~ smf(time,knots=c(0,1,3,5))

t1 <- eval(substitute(time), data)
t0 <- eval(substitute(t0), data)
event <- eval(substitute(event), data)

# Setting up the model
model.c <- model.cons(form,lambda=0,data.spec=data,t1=t1,t1.name="time",
t0=rep(0,100),t0.name="t0",event=event,event.name="event",
expected=rep(0,100),expected.name=NULL,type="overall",n.legendre=20,
cl="survPen(form,data,t1=time,event=event)",beta.ini=NULL)

# Retrieving the sum-to-zero constraint matrices and the list of knots
Z.smf <- model.c$Z.smf ; list.smf <- model.c$list.smf

# Calculating the design matrix
design.M <- design.matrix(form,data.spec=data,t1.name="time",Z.smf=Z.smf,list.smf=list.smf,
Z.tensor=NULL,Z.tint=NULL,list.tensor=NULL,list.tint=NULL,list.rd=NULL)
```

`expected.table` *French women mortality table*

Description

French women mortality table to serve as example of reference/expected mortality in excess hazard and relative mortality ratio models The data come from the human mortality databse website: https://www.mortality.org/Country/Country?cntr=FRATNP

- Age. Age group for 1-year interval from exact age x to just before exact age x+1 (0-110+)
- Year. Calendar Year (1816-2021)
- mx. Central death rate between ages x and x+1

Usage

```
data(expected.table)
```

Format

A data frame with 22866 rows and 3 variables

`grad_rho` *Gradient vector of LCV and LAML wrt rho (log smoothing parameters)*

Description

Gradient vector of LCV and LAML wrt rho (log smoothing parameters)

Usage

```
grad_rho(
  X_GL,
  GL_temp,
  haz_GL,
  deriv_rho_beta,
  weights,
  tm,
  nb_smooth,
  p,
  n_legendre,
  S_list,
  temp_LAML,
  Vp,
  S_beta,
  beta,
  inverse_new_S,
  X,
  temp_deriv3,
  event,
  expected,
  type,
  Ve,
  mat_temp,
  method
)
```

Arguments

`X_GL`	list of matrices (`length(X.GL)=n.legendre`) for Gauss-Legendre quadrature
`GL_temp`	list of vectors used to make intermediate calculations and save computation time
`haz_GL`	list of all the matrix-vector multiplications X.GL[[i]]%*%beta for Gauss Legendre integration in order to save computation time
`deriv_rho_beta`	firt derivative of beta wrt rho (implicit differentiation)
`weights`	vector of weights for Gauss-Legendre integration on [-1;1]
`tm`	vector of midpoints times for Gauss-Legendre integration; tm = 0.5*(t1 - t0)

`nb_smooth`	number of smoothing parameters
`p`	number of regression parameters
`n_legendre`	number of nodes for Gauss-Legendre quadrature
`S_list`	List of all the rescaled penalty matrices multiplied by their associated smoothing parameters
`temp_LAML`	temporary matrix used when method="LAML" to save computation time
`Vp`	Bayesian covariance matrix
`S_beta`	List such that S_beta[[i]]=S_list[[i]]%*%beta
`beta`	vector of estimated regression parameters
`inverse_new_S`	inverse of the penalty matrix
`X`	design matrix for the model
`temp_deriv3`	temporary matrix for third derivatives calculation when type="net" to save computation time
`event`	vector of right-censoring indicators
`expected`	vector of expected hazard rates
`type`	"net" or "overall"
`Ve`	frequentist covariance matrix
`mat_temp`	temporary matrix used when method="LCV" to save computation time
`method`	criterion used to select the smoothing parameters. Should be "LAML" or "LCV"; default is "LAML"

Value

List of objects with the following items:

`grad_rho`	gradient vector of LCV or LAML
`deriv_rho_inv_Hess_beta`	List of first derivatives of Vp wrt rho
`deriv_rho_Hess_unpen_beta`	List of first derivatives of the Hessian of the unpenalized log-likelihood wrt rho

---

`grad_rho_mult` *Gradient vector of LCV and LAML wrt rho (log smoothing parameters). Version for multiplicative decomposition : relative mortality ratio model*

Description

Gradient vector of LCV and LAML wrt rho (log smoothing parameters). Version for multiplicative decomposition : relative mortality ratio model

Usage

```
grad_rho_mult(
  X_GL,
  GL_temp,
  haz_GL,
  deriv_rho_beta,
  weights,
  tm,
  nb_smooth,
  p,
  n_legendre,
  S_list,
  temp_LAML,
  Vp,
  S_beta,
  beta,
  inverse_new_S,
  X,
  event,
  expected,
  Ve,
  mat_temp,
  method
)
```

Arguments

`X_GL`	list of matrices (`length(X.GL)=n.legendre`) for Gauss-Legendre quadrature
`GL_temp`	list of vectors used to make intermediate calculations and save computation time
`haz_GL`	list of all the matrix-vector multiplications X.GL[[i]]%*%beta for Gauss Legendre integration in order to save computation time
`deriv_rho_beta`	firt derivative of beta wrt rho (implicit differentiation)
`weights`	vector of weights for Gauss-Legendre integration on [-1;1]
`tm`	vector of midpoints times for Gauss-Legendre integration; tm = 0.5*(t1 - t0)
`nb_smooth`	number of smoothing parameters
`p`	number of regression parameters
`n_legendre`	number of nodes for Gauss-Legendre quadrature
`S_list`	List of all the rescaled penalty matrices multiplied by their associated smoothing parameters
`temp_LAML`	temporary matrix used when method="LAML" to save computation time
`Vp`	Bayesian covariance matrix
`S_beta`	List such that S_beta[[i]]=S_list[[i]]%*%beta
`beta`	vector of estimated regression parameters
`inverse_new_S`	inverse of the penalty matrix

`X`	design matrix for the model
`event`	vector of right-censoring indicators
`expected`	vector of expected hazard rates
`Ve`	frequentist covariance matrix
`mat_temp`	temporary matrix used when method="LCV" to save computation time
`method`	criterion used to select the smoothing parameters. Should be "LAML" or "LCV"; default is "LAML"

Value

List of objects with the following items:

`grad_rho`	gradient vector of LCV or LAML
`deriv_rho_inv_Hess_beta`	List of first derivatives of Vp wrt rho
`deriv_rho_Hess_unpen_beta`	List of first derivatives of the Hessian of the unpenalized log-likelihood wrt rho

---

`HazGL` *Gauss-Legendre evaluations*

Description

Gauss-Legendre evaluations

Usage

```
HazGL(X_GL, n_legendre, beta)
```

Arguments

`X_GL`	list of matrices (`length(X.GL)=n.legendre`) for Gauss-Legendre quadrature
`n_legendre`	number of nodes for Gauss-Legendre quadrature
`beta`	vector of estimated regression parameters

Value

list of all the matrix-vector multiplications X.GL[[i]]%*%beta for Gauss Legendre integration in order to save computation time

---

`HeartFailure` *Patients with heart failure at risk of recurrent hospitalization events*

Description

A simulated dataset containing 3 068 observations (2 268 events) in 800 patients with heart failure. The dataset is based on hfaction_cpx12 dataset from package WA. The variables are as follows:

- id. patient identifcation number
- treatment. treatment=0 for control and treatment=1 for exercise training
- t0. beginning of follow-up for a given event
- t1. end of follow-up for a given event (up to 3.27 years)
- enum. event identification number for a given patient (between 1 and 6 events per patient)
- event. event indicator (1 for hospitalization, 0 for censored)

Usage

```
data(HeartFailure)
```

Format

A data frame with 3 068 rows and 6 variables

`Hess_rho` *Hessian matrix of LCV and LAML wrt rho (log smoothing parameters)*

Description

Hessian matrix of LCV and LAML wrt rho (log smoothing parameters)

Usage

```
Hess_rho(
  X_GL,
  X_GL_Q,
  GL_temp,
  haz_GL,
  deriv2_rho_beta,
  deriv_rho_beta,
  weights,
  tm,
  nb_smooth,
  p,
  n_legendre,
```

```
  deriv_rho_inv_Hess_beta,
  deriv_rho_Hess_unpen_beta,
  S_list,
  minus_eigen_inv_Hess_beta,
  temp_LAML,
  temp_LAML2,
  Vp,
  S_beta,
  beta,
  inverse_new_S,
  X,
  X_Q,
  temp_deriv3,
  temp_deriv4,
  event,
  expected,
  type,
  Ve,
  deriv_rho_Ve,
  mat_temp,
  deriv_mat_temp,
  eigen_mat_temp,
  method
)
```

Arguments

`X_GL`	list of matrices (`length(X.GL)=n.legendre`) for Gauss-Legendre quadrature
`X_GL_Q`	list of transformed matrices from X_GL in order to calculate only the diagonal of the fourth derivative of the likelihood
`GL_temp`	list of vectors used to make intermediate calculations and save computation time
`haz_GL`	list of all the matrix-vector multiplications X.GL[[i]]%*%beta for Gauss Legendre integration in order to save computation time
`deriv2_rho_beta`	second derivatives of beta wrt rho (implicit differentiation)
`deriv_rho_beta`	firt derivatives of beta wrt rho (implicit differentiation)
`weights`	vector of weights for Gauss-Legendre integration on [-1;1]
`tm`	vector of midpoints times for Gauss-Legendre integration; tm = 0.5*(t1 - t0)
`nb_smooth`	number of smoothing parameters
`p`	number of regression parameters
`n_legendre`	number of nodes for Gauss-Legendre quadrature
`deriv_rho_inv_Hess_beta`	list of first derivatives of Vp wrt rho
`deriv_rho_Hess_unpen_beta`	list of first derivatives of Hessian of unpenalized log likelihood wrt rho

`S_list`	List of all the rescaled penalty matrices multiplied by their associated smoothing parameters
`minus_eigen_inv_Hess_beta`	vector of eigenvalues of Vp
`temp_LAML`	temporary matrix used when method="LAML" to save computation time
`temp_LAML2`	temporary matrix used when method="LAML" to save computation time
`Vp`	Bayesian covariance matrix
`S_beta`	List such that S_beta[[i]]=S_list[[i]]%*%beta
`beta`	vector of estimated regression parameters
`inverse_new_S`	inverse of the penalty matrix
`X`	design matrix for the model
`X_Q`	transformed design matrix in order to calculate only the diagonal of the fourth derivative of the likelihood
`temp_deriv3`	temporary matrix for third derivatives calculation when type="net" to save computation time
`temp_deriv4`	temporary matrix for fourth derivatives calculation when type="net" to save computation time
`event`	vector of right-censoring indicators
`expected`	vector of expected hazard rates
`type`	"net" or "overall"
`Ve`	frequentist covariance matrix
`deriv_rho_Ve`	list of derivatives of Ve wrt rho
`mat_temp`	temporary matrix used when method="LCV" to save computation time
`deriv_mat_temp`	list of derivatives of mat_temp wrt rho
`eigen_mat_temp`	vector of eigenvalues of mat_temp
`method`	criterion used to select the smoothing parameters. Should be "LAML" or "LCV"; default is "LAML"

Value

Hessian matrix of LCV or LAML wrt rho

`Hess_rho_mult` *Hessian matrix of LCV and LAML wrt rho (log smoothing parameters). Version for multiplicative decomposition : relative mortality ratio model*

Description

Hessian matrix of LCV and LAML wrt rho (log smoothing parameters). Version for multiplicative decomposition : relative mortality ratio model

Usage

```
Hess_rho_mult(
  X_GL,
  X_GL_Q,
  GL_temp,
  haz_GL,
  deriv2_rho_beta,
  deriv_rho_beta,
  weights,
  tm,
  nb_smooth,
  p,
  n_legendre,
  deriv_rho_inv_Hess_beta,
  deriv_rho_Hess_unpen_beta,
  S_list,
  minus_eigen_inv_Hess_beta,
  temp_LAML,
  temp_LAML2,
  Vp,
  S_beta,
  beta,
  inverse_new_S,
  X,
  X_Q,
  event,
  expected,
  Ve,
  deriv_rho_Ve,
  mat_temp,
  deriv_mat_temp,
  eigen_mat_temp,
  method
)
```

Arguments

`X_GL`	list of matrices (`length(X.GL)=n.legendre`) for Gauss-Legendre quadrature
`X_GL_Q`	list of transformed matrices from X_GL in order to calculate only the diagonal of the fourth derivative of the likelihood
`GL_temp`	list of vectors used to make intermediate calculations and save computation time
`haz_GL`	list of all the matrix-vector multiplications X.GL[[i]]%*%beta for Gauss Legendre integration in order to save computation time
`deriv2_rho_beta`	second derivatives of beta wrt rho (implicit differentiation)
`deriv_rho_beta`	firt derivatives of beta wrt rho (implicit differentiation)
`weights`	vector of weights for Gauss-Legendre integration on [-1;1]

`tm`	vector of midpoints times for Gauss-Legendre integration; tm = 0.5*(t1 - t0)
`nb_smooth`	number of smoothing parameters
`p`	number of regression parameters
`n_legendre`	number of nodes for Gauss-Legendre quadrature
`deriv_rho_inv_Hess_beta`	list of first derivatives of Vp wrt rho
`deriv_rho_Hess_unpen_beta`	list of first derivatives of Hessian of unpenalized log likelihood wrt rho
`S_list`	List of all the rescaled penalty matrices multiplied by their associated smoothing parameters
`minus_eigen_inv_Hess_beta`	vector of eigenvalues of Vp
`temp_LAML`	temporary matrix used when method="LAML" to save computation time
`temp_LAML2`	temporary matrix used when method="LAML" to save computation time
`Vp`	Bayesian covariance matrix
`S_beta`	List such that S_beta[[i]]=S_list[[i]]%*%beta
`beta`	vector of estimated regression parameters
`inverse_new_S`	inverse of the penalty matrix
`X`	design matrix for the model
`X_Q`	transformed design matrix in order to calculate only the diagonal of the fourth derivative of the likelihood
`event`	vector of right-censoring indicators
`expected`	vector of expected hazard rates
`Ve`	frequentist covariance matrix
`deriv_rho_Ve`	list of derivatives of Ve wrt rho
`mat_temp`	temporary matrix used when method="LCV" to save computation time
`deriv_mat_temp`	list of derivatives of mat_temp wrt rho
`eigen_mat_temp`	vector of eigenvalues of mat_temp
`method`	criterion used to select the smoothing parameters. Should be "LAML" or "LCV"; default is "LAML"

Value

Hessian matrix of LCV or LAML wrt rho

`instr` *Position of the nth occurrence of a string in another one*

Description

Returns the position of the nth occurrence of str2 in str1. Returns 0 if str2 is not found. This code was first suggested by Abdelmonem Mahmoud Amer in https://stackoverflow.com/a/33005653/5421090

Usage

```
instr(str1, str2, startpos = 1, n = 1)
```

Arguments

`str1`	main string in which str2 is to be found
`str2`	substring contained in str1
`startpos`	starting position in str1; default is 1
`n`	which occurrence is to be found; default is 1

Value

number representing the nth position of str2 in str1

Examples

```
library(survPen)

instr("character test to find the position of the third letter r","r",n=3)
```

`inv.repam` *Reverses the initial reparameterization for stable evaluation of the log determinant of the penalty matrix*

Description

Transforms the final model by reversing the initial reparameterization performed by `repam`. Derives the corrected version of the Bayesian covariance matrix

Usage

```
inv.repam(model, X.ini, S.pen.ini)
```

Arguments

`model`	survPen object, see `survPen.fit` for details
`X.ini`	initial design matrix (before reparameterization)
`S.pen.ini`	initial penalty matrices

Value

survPen object with standard parameterization

`list.wicss` *List of ICSS standards for age-standardization of cancer (net) survival*

Description

Four data frames are available in the list : 1, 2, 3 and "prostate". Each one corresponds to certain types of cancer. Details can be found in Corazzieri et al. (2004) (10.1016/j.ejca.2004.07.002) or at (in French) : https://www.santepubliquefrance.fr/docs/survie-des-personnes-atteintes-de-cancer-en-france-metropolitaine-1989-2018-materiel-et-methodes For each data frame, the variables are as follows:

- AgeClass. Age classes considered. Closed on the left and open on the right.
- AgeWeights. Weights associated with each age class

Usage

```
data(list.wicss)
```

Format

A list containing four data frames of 5 rows and 2 variables each

---

`model.cons` *Design and penalty matrices for the model*

Description

Sets up the model before optimization. Builds the design matrix, the penalty matrix and all the design matrices needed for Gauss-Legendre quadrature.

Usage

```
model.cons(
  formula,
  lambda,
  data.spec,
  t1,
  t1.name,
  t0,
  t0.name,
  event,
  event.name,
  expected,
  expected.name,
  type,
  n.legendre,
  cl,
  beta.ini
)
```

Arguments

`formula`	formula object identifying the model
`lambda`	vector of smoothing parameters
`data.spec`	data frame that represents the environment from which the covariate values and knots are to be calculated
`t1`	vector of follow-up times
`t1.name`	name of `t1` in `data.spec`
`t0`	vector of origin times (usually filled with zeros)
`t0.name`	name of `t0` in `data.spec`
`event`	vector of censoring indicators
`event.name`	name of event in `data.spec`
`expected`	vector of expected hazard
`expected.name`	name of expected in `data.spec`
`type`	"net" or "overall"
`n.legendre`	number of nodes for Gauss-Legendre quadrature
`cl`	original `survPen` call
`beta.ini`	initial set of regression parameters

Value

List of objects with the following items:

`cl`	original `survPen` call
`type`	"net", "overall", or "mult"

`n.legendre`	number of nodes for Gauss-Legendre quadrature. If is.pwcst is TRUE, for simplicity of implementation, n.legendre actually corresponds to the number of sub-intervals
`n`	number of individuals
`p`	number of parameters
`X.para`	design matrix associated with fully parametric parameters (unpenalized)
`X.smooth`	design matrix associated with the penalized parameters
`X`	design matrix for the model
`is.pwcst`	TRUE if there is a piecewise constant (excess) hazard specification. In that case the cumulative hazard can be derived without Gauss-Legendre quadrature
`pwcst.breaks`	if is.pwcst is TRUE, vector of breaks defining the sub-intervals on which the hazard is constant. Otherwise NULL.
`pwcst.weights`	if is.pwcst is TRUE, matrix of weights giving the time contribution of each individual on each sub-interval. Otherwise NULL.
`leg`	list of nodes and weights for Gauss-Legendre integration on [-1;1] as returned by `gauss.quad`
`X.GL`	list of matrices (`length(X.GL)=n.legendre`) for Gauss-Legendre quadrature
`S`	penalty matrix for the model. Sum of the elements of `S.list`
`S.scale`	vector of rescaling factors for the penalty matrices
`rank.S`	rank of the penalty matrix
`S.F`	balanced penalty matrix as described in section 3.1.2 of (Wood,2016). Sum of the elements of `S.F.list`
`U.F`	Eigen vectors of S.F, useful for the initial reparameterization to separate penalized ad unpenalized subvectors. Allows stable evaluation of the log determinant of S and its derivatives
`S.smf`	List of penalty matrices associated with all "smf" calls
`S.tensor`	List of penalty matrices associated with all "tensor" calls
`S.tint`	List of penalty matrices associated with all "tint" calls
`S.rd`	List of penalty matrices associated with all "rd" calls
`smooth.name.smf`	List of names for the "smf" calls associated with S.smf
`smooth.name.tensor`	List of names for the "tensor" calls associated with S.tensor
`smooth.name.tint`	List of names for the "tint" calls associated with S.tint
`smooth.name.rd`	List of names for the "rd" calls associated with S.rd
`S.pen`	List of all the rescaled penalty matrices redimensioned to df.tot size. Every element of `pen` noted `pen[[i]]` is made from a penalty matrix returned by `smooth.cons` and is multiplied by the factor S.scale=norm(X,type="I")^2/norm(pen[[i]],type="I")
`S.list`	Equivalent to S.pen but with every element multiplied by its associated smoothing parameter

`S.F.list`	Equivalent to S.pen but with every element divided by its Frobenius norm
`lambda`	vector of smoothing parameters
`df.para`	degrees of freedom associated with fully parametric terms (unpenalized)
`df.smooth`	degrees of freedom associated with penalized terms
`df.tot`	`df.para` + `df.smooth`
`list.smf`	List of all `smf.smooth.spec` objects contained in the model
`list.tensor`	List of all `tensor.smooth.spec` objects contained in the model
`list.tint`	List of all `tint.smooth.spec` objects contained in the model
`nb.smooth`	number of smoothing parameters
`Z.smf`	List of matrices that represents the sum-to-zero constraints to apply for `smf` splines
`Z.tensor`	List of matrices that represents the sum-to-zero constraints to apply for `tensor` splines
`Z.tint`	List of matrices that represents the sum-to-zero constraints to apply for `tint` splines
`beta.ini`	initial set of regression parameters

Examples

```
library(survPen)

# standard spline of time with 4 knots

data <- data.frame(time=seq(0,5,length=100),event=1,t0=0)

form <- ~ smf(time,knots=c(0,1,3,5))

t1 <- eval(substitute(time), data)
t0 <- eval(substitute(t0), data)
event <- eval(substitute(event), data)

# The following code sets up everything we need in order to fit the model
model.c <- model.cons(form,lambda=0,data.spec=data,t1=t1,t1.name="time",
t0=rep(0,100),t0.name="t0",event=event,event.name="event",
expected=rep(0,100),expected.name=NULL,type="overall",n.legendre=20,
cl="survPen(form,data,t1=time,event=event)",beta.ini=NULL)
```

`NR.beta` *Inner Newton-Raphson algorithm for regression parameters estimation*

Description

Applies Newton-Raphson algorithm for beta estimation. Two specific modifications aims at guaranteeing convergence : first the hessian is perturbed whenever it is not positive definite and second, at each step, if the penalized log-likelihood is not maximized, the step is halved until it is.

Usage

```
NR.beta(build, beta.ini, detail.beta, max.it.beta = 200, tol.beta = 1e-04)
```

Arguments

`build`	list of objects returned by `model.cons`
`beta.ini`	vector of initial regression parameters; default is NULL, in which case the first beta will be `log(sum(event)/sum(t1))` and the others will be zero (except if there are "by" variables or if there is a piecewise constant hazard specification in which cases all betas are set to zero)
`detail.beta`	if TRUE, details concerning the optimization process in the regression parameters are displayed; default is FALSE
`max.it.beta`	maximum number of iterations to reach convergence in the regression parameters; default is 200
`tol.beta`	convergence tolerance for regression parameters; default is `1e-04`

Details

If we note `ll.pen` and `beta` respectively the current penalized log-likelihood and estimated parameters and `ll.pen.old` and `betaold` the previous ones, the algorithm goes on while (abs(ll.pen-ll.pen.old)>tol.beta) or any(abs((beta-betaold)/betaold)>tol.beta)

Value

List of objects:

`beta`	estimated regression parameters
`ll.unpen`	log-likelihood at convergence
`ll.pen`	penalized log-likelihood at convergence
`iter.beta`	number of iterations needed to converge

Examples

```
library(survPen)

# standard spline of time with 4 knots

data <- data.frame(time=seq(0,5,length=100),event=1,t0=0)

form <- ~ smf(time,knots=c(0,1,3,5))

t1 <- eval(substitute(time), data)
t0 <- eval(substitute(t0), data)
event <- eval(substitute(event), data)

# Setting up the model before fitting
model.c <- model.cons(form,lambda=0,data.spec=data,t1=t1,t1.name="time",
t0=rep(0,100),t0.name="t0",event=event,event.name="event",
```

```
expected=rep(0,100),expected.name=NULL,type="overall",n.legendre=20,
cl="survPen(form,data,t1=time,event=event)",beta.ini=NULL)

# Estimating the regression parameters at given smoothing parameter (here lambda=0)
Newton1 <- NR.beta(model.c,beta.ini=rep(0,4),detail.beta=TRUE)
```

`NR.rho` *Outer Newton-Raphson algorithm for smoothing parameters estimation via LCV or LAML optimization*

Description

Applies Newton-Raphson algorithm for smoothing parameters estimation. Two specific modifications aims at guaranteeing convergence : first the hessian is perturbed whenever it is not positive definite and second, at each step, if LCV or -LAML is not minimized, the step is halved until it is.

Usage

```
NR.rho(
  build,
  rho.ini,
  data,
  formula,
  max.it.beta = 200,
  max.it.rho = 30,
  beta.ini = NULL,
  detail.rho = FALSE,
  detail.beta = FALSE,
  nb.smooth,
  tol.beta = 1e-04,
  tol.rho = 1e-04,
  step.max = 5,
  method = "LAML"
)
```

Arguments

`build`	list of objects returned by `model.cons`
`rho.ini`	vector of initial log smoothing parameters; if it is NULL, all log lambda are set to -1
`data`	an optional data frame containing the variables in the model
`formula`	formula object specifying the model
`max.it.beta`	maximum number of iterations to reach convergence in the regression parameters; default is 200

`max.it.rho`	maximum number of iterations to reach convergence in the smoothing parameters; default is 30
`beta.ini`	vector of initial regression parameters; default is NULL, in which case the first beta will be `log(sum(event)/sum(t1))` and the others will be zero (except if there are "by" variables or if there is a piecewise constant hazard specification in which cases all betas are set to zero)
`detail.rho`	if TRUE, details concerning the optimization process in the smoothing parameters are displayed; default is FALSE
`detail.beta`	if TRUE, details concerning the optimization process in the regression parameters are displayed; default is FALSE
`nb.smooth`	number of smoothing parameters
`tol.beta`	convergence tolerance for regression parameters; default is `1e-04`
`tol.rho`	convergence tolerance for smoothing parameters; default is `1e-04`
`step.max`	maximum absolute value possible for any component of the step vector (on the log smoothing parameter scale); default is 5
`method`	LCV or LAML; default is LAML

Details

If we note `val` the current LCV or LAML value, `val.old` the previous one and `grad` the gradient vector of LCV or LAML with respect to the log smoothing parameters, the algorithm goes on `while(abs(val-val.old)>tol.rho|any(abs(grad)>tol.rho))`

Value

object of class survPen (see `survPen.fit` for details)

Examples

```
library(survPen)

# standard spline of time with 4 knots

data <- data.frame(time=seq(0,5,length=100),event=1,t0=0)

form <- ~ smf(time,knots=c(0,1,3,5))

t1 <- eval(substitute(time), data)
t0 <- eval(substitute(t0), data)
event <- eval(substitute(event), data)

# Setting up the model before fitting
model.c <- model.cons(form,lambda=0,data.spec=data,t1=t1,t1.name="time",
t0=rep(0,100),t0.name="t0",event=event,event.name="event",
expected=0,expected.name=NULL,type="overall",n.legendre=20,
cl="survPen(form,data,t1=time,event=event)",beta.ini=NULL)

# Estimating the smoothing parameter and the regression parameters
```

```
# we need to apply a reparameterization to model.c before fitting
constructor <- repam(model.c)$build # model constructor
constructor$optim.rho <- 1 # we tell it we want to estimate the log smoothing parameters (rho)
Newton2 <- NR.rho(constructor,rho.ini=-1,data,form,nb.smooth=1,detail.rho=TRUE)
```

`predict.survPen` *Hazard and Survival prediction from fitted* `survPen` *model*

Description

Takes a fitted `survPen` object and produces hazard and survival predictions given a new set of values for the model covariates.

Usage

```
## S3 method for class 'survPen'
predict(
  object,
  newdata,
  newdata.ref = NULL,
  n.legendre = 50,
  conf.int = 0.95,
  do.surv = TRUE,
  type = "standard",
  exclude.random = FALSE,
  get.deriv.H = FALSE,
  ...
)
```

Arguments

`object`	a fitted `survPen` object as produced by `survPen.fit`
`newdata`	data frame giving the new covariates value
`newdata.ref`	data frame giving the new covariates value for the reference population (used only when type="HR")
`n.legendre`	number of nodes to approximate the cumulative hazard by Gauss-Legendre quadrature; default is 50
`conf.int`	numeric value giving the precision of the confidence intervals; default is 0.95
`do.surv`	If TRUE (the default), the survival (or cumulative ratio for type='mult') and its lower and upper confidence values are computed. Survival computation requires numerical integration and can be time-consuming so if you only want the hazard use do.surv=FALSE; default is TRUE
`type`	if type="lpmatrix" returns the design matrix (or linear predictor matrix) corresponding to the new values of the covariates; if equals "HR", returns the predicted HR and survival difference (with CIs) between newdata and newdata.ref; default is "standard" for classical hazard and survival estimation

`exclude.random`	if TRUE all random effects are set to zero; default is FALSE
`get.deriv.H`	if TRUE, the derivatives wrt to the regression parameters of the cumulative hazard are returned; default is FALSE
`...`	other arguments

Details

The confidence intervals noted CI.U are built on the log cumulative hazard scale U=log(H) (efficient scale in terms of respect towards the normality assumption) using Delta method. The confidence intervals on the survival scale are then `CI.surv = exp(-exp(CI.U))`

Value

List of objects:

`haz`	hazard predicted by the model
`haz.inf`	lower value for the confidence interval of the hazard based on the Bayesian covariance matrix Vp (Wood et al. 2016)
`haz.sup`	Upper value for the confidence interval of the hazard based on the Bayesian covariance matrix Vp
`surv`	survival predicted by the model
`surv.inf`	lower value for the confidence interval of the survival based on the Bayesian covariance matrix Vp
`surv.sup`	Upper value for the confidence interval of the survival based on the Bayesian covariance matrix Vp
`deriv.H`	derivatives wrt to the regression parameters of the cumulative hazard. Useful to calculate standardized survival
`HR`	predicted hazard ratio ; only when type = "HR"
`HR.inf`	lower value for the confidence interval of the hazard ratio based on the Bayesian covariance matrix Vp ; only when type = "HR"
`HR.sup`	Upper value for the confidence interval of the hazard ratio based on the Bayesian covariance matrix Vp ; only when type = "HR"
`surv.diff`	predicted relative difference ; only when type = "HR"
`surv.diff.inf`	lower value for the confidence interval of the survival difference based on the Bayesian covariance matrix Vp ; only when type = "HR"
`surv.diff.sup`	Upper value for the confidence interval of the survival difference based on the Bayesian covariance matrix Vp ; only when type = "HR"
`ratio`	relative mortality ratio predicted by the model ; only for relative mortality ratio model (type="mult")
`ratio.inf`	lower value for the confidence interval of the relative mortality ratio based on the Bayesian covariance matrix Vp (Wood et al. 2016); only for relative mortality ratio model (type="mult")
`ratio.sup`	Upper value for the confidence interval of the relative mortality ratio on the Bayesian covariance matrix Vp; only for relative mortality ratio model (type="mult")

`cumul.ratio`	cumulative relative mortality ratio predicted by the model ; only for relative mortality ratio model (type="mult")
`cumul.ratio.inf`	lower value for the confidence interval of the cumulative relative mortality ratio based on the Bayesian covariance matrix Vp (Wood et al. 2016); only for relative mortality ratio model (type="mult")
`cumul.ratio.sup`	Upper value for the confidence interval of the cumulative relative mortality ratio on the Bayesian covariance matrix Vp; only for relative mortality ratio model (type="mult")
`RR`	predicted ratio of relative mortality ratios ; only for relative mortality ratio model when type = "HR"
`RR.inf`	lower value for the confidence interval of the ratio of relative mortality ratios based on the Bayesian covariance matrix Vp ; only for relative mortality ratio model when type = "HR"
`RR.sup`	Upper value for the confidence interval of the ratio of relative mortality ratios based on the Bayesian covariance matrix Vp ; only for relative mortality ratio model when type = "HR"

References

Wood, S.N., Pya, N. and Saefken, B. (2016), Smoothing parameter and model selection for general smooth models (with discussion). Journal of the American Statistical Association 111, 1548-1575

Examples

```
library(survPen)
data(datCancer) # simulated dataset with 2000 individuals diagnosed with cervical cancer

f1 <- ~tensor(fu,age,df=c(5,5))

# hazard model
mod1 <- survPen(f1,data=datCancer,t1=fu,event=dead,expected=NULL,method="LAML")

# predicting hazard and survival curves for age 60
nt <- seq(0,5,le=50)
pred <- predict(mod1,data.frame(fu=nt,age=60))
pred$haz
pred$surv

# predicting hazard ratio at 1 year according to age (with reference age of 50)
newdata1 <- data.frame(fu=1,age=seq(30,90,by=1))
newdata.ref1 <- data.frame(fu=1,age=rep(50,times=61))
predHR_1 <- predict(mod1,newdata=newdata1,newdata.ref=newdata.ref1,type="HR")
predHR_1$HR
predHR_1$HR.inf
predHR_1$HR.sup

# predicting hazard ratio at 3 years according to age (with reference age of 50)
# and difference of survival at 3 years
```

```
newdata3 <- data.frame(fu=3,age=seq(30,90,by=1))
newdata.ref3 <- data.frame(fu=3,age=rep(50,times=61))
predHR_3 <- predict(mod1,newdata=newdata3,newdata.ref=newdata.ref3,type="HR")

# Hazard ratio
predHR_3$HR
predHR_3$HR.inf
predHR_3$HR.sup


# Difference of survival
predHR_3$diff.surv
predHR_3$diff.surv.inf
predHR_3$diff.surv.sup
```

`predSNS` *Prediction of grouped indicators : population (net) survival (PNS) and age-standardized (net) survival (SNS)*

Description

Allows the prediction of population and age-standardized (net) survival as well as associated confidence intervals

Usage

```
predSNS(
  model,
  time.points,
  newdata,
  weight.table,
  var.name,
  var.model,
  conf.int = 0.95,
  method = "exact",
  n.legendre = 50
)
```

Arguments

`model`	a fitted `survPen` model
`time.points`	vector of follow-up values
`newdata`	dataset containing the original age values used for fitting
`weight.table`	dataset containing the age classes used for standardization, must be in the same format as the elements of the following list `list.wicss`

`var.name`	list containing one element : the column name in newdata that reports age values. This element should be named after the age variable present in the model formula. Typically, if newdata contains an 'age' column while the model uses a centered age 'agec', the list should be: list(agec="age")
`var.model`	list containing one element : the function that allows retrieving the age variable used in model formula from original age. Typically for age centered on 50, list(agec=function(age) age - 50)
`conf.int`	numeric value giving the precision of the confidence intervals; default is 0.95
`method`	should be either 'exact' or 'approx'. The 'exact' method uses all age values in newdata for predictions. The 'approx' method uses either newdata$age (if age values are whole numbers) or floor(newdata$age) + 0.5 (if age values are not whole numbers) and then removes duplicates to reduce computational cost.
`n.legendre`	number of nodes to approximate the cumulative hazard by Gauss-Legendre quadrature; default is 50

Details

The weight table used should always be in the same format as elements of `list.wicss`. Only age-standardization is possible for now. All other variables necessary for model predictions should be fixed to a single value. For simplicity, in what follows we will consider that survival only depends on time and age.

Value

List of nine elements

`class.table`	Number of individuals in each age class
`SNS`	Vector of predicted age-standardized (net) survival
`SNS.inf`	Lower bound of confidence intervals associated with predicted age-standardized (net) survival
`SNS.sup`	Upper bound of confidence intervals associated with predicted age-standardized (net) survival
`PNS`	Vector of predicted population (net) survival
`PNS.inf`	Lower bound of confidence intervals associated with predicted population (net) survival
`PNS.sup`	Upper bound of confidence intervals associated with predicted population (net) survival
`PNS_per_class`	matrix of predicted population (net) survival in each age class
`PNS_per_class.inf`	Lower bound of confidence intervals associated with predicted population (net) survival in each age class
`PNS_per_class.sup`	Upper bound of confidence intervals associated with predicted population (net) survival in each age class

Population Net Survival (PNS)

For a given group of individuals, PNS at time t is defined as

$$PNS(t) = \sum_i 1/n * S_i(t, a_i)$$

where a_i is the age of individual i

Standardized Net Survival (SNS)

SNS at time t is defined as

$$SNS(t) = \sum_i w_i * S_i(t, a_i)$$

where a_i is the age of individual i and $w_i = w_{refj(i)}/n_{j(i)}$. $w_{refj(i)}$ is the weigth of age class j in the reference population (it corresponds to weight.table$AgeWeights). Where $n_{j(i)}$ is the total number of individuals present in age class $j(i)$: the age class of individual i.

Standardized Net Survival (SNS) with method="approx"

For large datasets, SNS calculation is quite heavy. To reduce computational cost, the idea is to regroup individuals who have similar age values. By using floor(age) + 0.5 instead of age, the gain will be substantial while the prediction error will be minimal (method="approx" will give slightly different predictions compared to method="exact"). Of course, if the provided age values are whole numbers then said provided age values will be used directly for grouping and there will be no prediction error (method="approx" and method="exact" will give the exact same predictions).

$$SNS(t) = \sum_a \tilde{w}_a * S(t, a)$$

The sum is here calculated over all possible values of age instead of all individuals. We have $\tilde{w}_a = n_a * w_{refj(a)}/n_{j(a)}$. Where $j(a)$ is the age class of age a while n_a is the number of individuals with age a.

Variance and Confidence Intervals

Confidence intervals for SNS are derived assuming normality of log(log(-SNS)) Lower and upper bound are given by

$$IC_{95\%}(SNS) = [SNS^{1.96*\sqrt{(}Var(Log(Delta_{SNS})))}; SNS^{-1.96*\sqrt{(}Var(Log(Delta_{SNS})))}]$$

with

$$Delta_{SNS} = -log(SNS)$$

$Var(Log(Delta_{SNS}))$ is derived by Delta method.

Confidence intervals for PNS are derived in the exact same way.

References

Corazziari, I., Quinn, M., & Capocaccia, R. (2004). Standard cancer patient population for age standardising survival ratios. European journal of cancer (Oxford, England : 1990), 40(15), 2307–2316. https://doi.org/10.1016/j.ejca.2004.07.002.

Examples

```
data(datCancer)
data(list.wicss)

don <- datCancer
don$agec <- don$age - 50 # using centered age for modelling

#-------------------- model with time and age

knots.t<-quantile(don$fu[don$dead==1],probs=seq(0,1,length=6)) # knots for time
knots.agec<-quantile(don$agec[don$dead==1],probs=seq(0,1,length=5))   # knots for age

formula <- as.formula(~tensor(fu,agec,df=c(length(knots.t),length(knots.agec)),
knots=list(fu=knots.t,age=knots.agec)))

mod <- survPen(formula,data=don,t1=fu,event=dead,n.legendre=20, expected=rate)


#-------------------- Age classes and associated weights for age-standardized
# net survival prediction

# weights of type 1
wicss <- list.wicss[["1"]]

# to estimate population net survival, prediction dataframe
# is needed. It should contain original data for age

pred.pop <- data.frame(age=don$age)

#-------------------- prediction : age-standardized net survival and population net survival

pred <- predSNS(mod,time.points=seq(0,5,by=0.1),newdata=pred.pop,
weight.table=wicss,var.name=list(agec="age"),
var.model=list(agec=function(age) age - 50),method="approx")
```

`print.summary.survPen` *print summary for a* `survPen` *fit*

Description

print summary for a `survPen` fit

Usage

```
## S3 method for class 'summary.survPen'
print(
```

```
    x,
    digits = max(3, getOption("digits") - 2),
    signif.stars = getOption("show.signif.stars"),
    ...
  )
```

Arguments

`x`	an object of class `summary.survPen`
`digits`	controls number of digits printed in output.
`signif.stars`	Should significance stars be printed alongside output.
`...`	other arguments

Value

print of summary

`pwcst` *Defining piecewise constant (excess) hazard in survPen formulae*

Description

Used inside a formula object to define a piecewise constant (excess) hazard. This is useful since it triggers an explicit calculation of cumulative hazard calculation (much more efficient and more precise than Gauss-Legendre quadrature when hazard is constant). The breaks given are used to defined sub-intervals that are left-open (except the first interval which is always left-closed) and right-closed. Internally, this constructor uses the cut function on the follow-up time with options include.lowest=TRUE and right=TRUE Important : this function must not be used with other time-dependent effect functions because the Gauss-Legendre quadrature will not operate correctly. If you really want to fit such a model, please use the cut function with the time variable as an argument to fit a piecewise constant hazard (and do not forget to use a huge number of Gauss-Legendre quadrature nodes, typically n.legendre=500)

Usage

```
pwcst(breaks)
```

Arguments

`breaks`	numeric vector that specifies the boundaries of each sub-interval on which the hazard is constant

Value

object of class `pwcst.spec`

`pwcst.breaks`	numeric vector that specifies the boundaries of each sub-interval on which the hazard is constant

Examples

```
library(survPen)

data(datCancer)

# piece constant hazard on 6 sub-intervals : [0;0.5]; ]0.5;1]; ]1;2]; ]2;3]; ]3;4]; ]4;5]
formula <- ~pwcst(breaks=c(0,0.5,1,2,3,4,5))
mod <- survPen(formula,t1=fu,event=dead,data=datCancer)

# The same but in an inefficient way
formula2 <- ~cut(fu,breaks=c(0,0.5,1,2,3,4,5),include.lowest=TRUE,right=TRUE)
mod.inefficient <- survPen(formula2,t1=fu,event=dead,data=datCancer,n.legendre=500)
```

`rd` *Defining random effects in survPen formulae*

Description

Used inside a formula object to define a random effect.

Usage

```
rd(...)
```

Arguments

`...`	Any number of covariates separated by ","

Value

object of class `rd.smooth.spec`

Examples

```
# cubic regression spline of time with 10 unspecified knots + random effect at the cluster level
formula.test <- ~smf(time,df=10) + rd(cluster)
```

`repam` *Applies initial reparameterization for stable evaluation of the log determinant of the penalty matrix*

Description

Transforms the object from `model.cons` by applying the matrix reparameterization (matrix U.F). The reparameterization is reversed at convergence by `inv.repam`.

Usage

```
repam(build)
```

Arguments

`build`	object as returned by `model.cons`

Value

`build`	an object as returned by `model.cons`
`X.ini`	initial design matrix (before reparameterization)
`S.pen.ini`	initial penalty matrices

Examples

```
library(survPen)

# standard spline of time with 4 knots

data <- data.frame(time=seq(0,5,length=100),event=1,t0=0)

form <- ~ smf(time,knots=c(0,1,3,5))

t1 <- eval(substitute(time), data)
t0 <- eval(substitute(t0), data)
event <- eval(substitute(event), data)

# Setting up the model before fitting
model.c <- model.cons(form,lambda=0,data.spec=data,t1=t1,t1.name="time",
t0=rep(0,100),t0.name="t0",event=event,event.name="event",
expected=rep(0,100),expected.name=NULL,type="overall",n.legendre=20,
cl="survPen(form,data,t1=time,event=event)",beta.ini=NULL)

# Reparameterization allows separating the parameters into unpenalized and
# penalized ones for maximum numerical stability
re.model.c <- repam(model.c)
```

`robust.var` *Implementation of the robust variance Vr*

Description

Takes the model at convergence and calculates the robust variance matrix accounting for correlated survival times

Usage

```
robust.var(model, data, cluster.name, n.legendre = 50)
```

Arguments

`model`	survPen object, see `survPen.fit` for details
`data`	original dataset
`cluster.name`	name of cluster variable in data
`n.legendre`	number of nodes for Gauss-Legendre quadrature; default is 50

Value

survPen object with robust variance Vr

`smf` *Defining smooths in survPen formulae*

Description

Used inside a formula object to define a smooth, a tensor product smooth or a tensor product interaction. Natural cubic regression splines (linear beyond the knots, equivalent to `ns` from package `splines`) are used as marginal bases. While `tensor` builds a tensor product of marginal bases including the intercepts, `tint` applies a tensor product of the marginal bases without their intercepts. Unlike `tensor`, the marginal effects of the covariates should also be present in the formula when using `tint`. For a conceptual difference between tensor products and tensor product interactions see Section 5.6.3 from Wood (2017)

Usage

```
smf(..., knots = NULL, df = NULL, by = NULL, same.rho = FALSE)

tensor(..., knots = NULL, df = NULL, by = NULL, same.rho = FALSE)

tint(..., knots = NULL, df = NULL, by = NULL, same.rho = FALSE)
```

Arguments

`...`	Any number of covariates separated by ","
`knots`	numeric vector that specifies the knots of the splines (including boundaries); default is NULL, in which case the knots are spread through the covariate values using quantiles. Precisely, for the term "smf(x,df=df1)", the vector of knots will be: quantile(unique(x),seq(0,1,length=df1))
`df`	numeric value that indicates the number of knots (or degrees of freedom) desired; default is NULL. If knots and df are NULL, df will be set to 10
`by`	numeric or factor variable in order to define a varying coefficient smooth
`same.rho`	if the specified by variable is a factor, specifies whether the smoothing parameters should be the same for all levels; default is FALSE.

Value

object of class `smf.smooth.spec`, `tensor.smooth.spec` or `tint.smooth.spec` (see `smooth.spec` for details)

References

Wood, S. N. (2017), Generalized additive models: an introduction with R. Second Edition. London: Chapman & Hall/CRC.

Examples

```
# penalized cubic regression spline of time with 5 unspecified knots
formula.test <- ~smf(time,df=5)

# suppose that we want to fit a model from formula.test
library(survPen)
data(datCancer)

mod.test <- survPen(~smf(fu,df=5) ,data=datCancer,t1=fu,event=dead)

# then the knots can be retrieved like this:
mod.test$list.smf[[1]]$knots
# or calculated like this
quantile(unique(datCancer$fu),seq(0,1,length=5))


# penalized cubic regression splines of time and age with respectively 5 and 7 unspecified knots
formula.test2 <- ~smf(time,df=5)+smf(age,df=7)

# penalized cubic regression splines of time and age with respectively 3 and 4 specified knots
formula.test3 <- ~smf(time,knots=c(0,3,5))+smf(age,knots=c(30,50,70,90))

# penalized tensor product for time and age with respectively 5 and 4 unspecified knots leading
# to 5*4 = 20 regression parameters
formula.test <- ~tensor(time,age,df=c(5,4))

# penalized tensor product for time and age with respectively 3 and 4 specified knots
```

```
formula.test3 <- ~tensor(time,agec,knots=list(c(0,3,5),c(30,50,70,90)))

# penalized tensor product for time, age and year with respectively 6, 5 and 4 unspecified knots
formula.test <- ~tensor(time,age,year,df=c(6,5,4))

# penalized tensor product interaction for time and age with respectively 5 and 4 unspecified knots
# main effects are specified as penalized cubic regression splines
formula.test <- ~smf(time,df=5)+smf(age,df=4)+tint(time,age,df=c(5,4))
```

`smooth.cons` *Design and penalty matrices of penalized splines in a smooth.spec object*

Description

Builds the design and penalty matrices from the result of `smooth.spec`.

Usage

```
smooth.cons(
  term,
  knots,
  df,
  by = NULL,
  option,
  data.spec,
  same.rho = FALSE,
  name
)
```

Arguments

`term`	Vector of strings that generally comes from the value "term" of a `smooth.spec` object.
`knots`	List of numeric vectors that specifies the knots of the splines (including boundaries).
`df`	Degrees of freedom: numeric vector that indicates the number of knots desired for each covariate.
`by`	numeric or factor variable in order to define a varying coefficient smooth; default is NULL.
`option`	"smf", "tensor" or "tint".
`data.spec`	data frame that represents the environment from which the covariate values and knots are to be calculated; default is NULL.
`same.rho`	if there is a factor by variable, should the smoothing parameters be the same for all levels; default is FALSE.
`name`	simplified name of the smooth.spec call.

Value

List of objects with the following items:

`X`	Design matrix
`pen`	List of penalty matrices
`term`	Vector of strings giving the names of each covariate
`knots`	list of numeric vectors that specifies the knots for each covariate
`dim`	Number of covariates
`all.df`	Numeric vector giving the number of knots associated with each covariate
`sum.df`	Sum of all.df
`Z.smf`	List of matrices that represents the sum-to-zero constraint to apply for "smf" splines
`Z.tensor`	List of matrices that represents the sum-to-zero constraint to apply for "tensor" splines
`Z.tint`	List of matrices that represents the sum-to-zero constraint to apply for "tint" splines
`lambda.name`	name of the smoothing parameters

Examples

```
library(survPen)

# standard spline of time with 4 knots (so we get a design matrix with 3 columns
# because of centering constraint)

data <- data.frame(time=seq(0,5,length=100))
smooth.c <- smooth.cons("time",knots=list(c(0,1,3,5)),df=4,option="smf",
data.spec=data,name="smf(time)")
```

`smooth.cons.integral` *Design matrix of penalized splines in a smooth.spec object for Gauss-Legendre quadrature*

Description

Almost identical to `smooth.cons`. This version is dedicated to Gauss-Legendre quadrature. Here, the sum-to-zero constraints must be specified so that they correspond to the ones that were calculated with the initial dataset.

Usage

```
smooth.cons.integral(
  term,
  knots,
  df,
  by = NULL,
  option,
  data.spec,
  Z.smf,
  Z.tensor,
  Z.tint,
  name
)
```

Arguments

`term`	Vector of strings that generally comes from the value "term" of a smooth.spec object
`knots`	List of numeric vectors that specifies the knots of the splines (including boundaries).
`df`	Degrees of freedom : numeric vector that indicates the number of knots desired for each covariate.
`by`	numeric or factor variable in order to define a varying coefficient smooth; default is NULL.
`option`	"smf", "tensor" or "tint".
`data.spec`	data frame that represents the environment from which the covariate values and knots are to be calculated; default is NULL.
`Z.smf`	List of matrices that represents the sum-to-zero constraint to apply for `smf` splines.
`Z.tensor`	List of matrices that represents the sum-to-zero constraint to apply for `tensor` splines.
`Z.tint`	List of matrices that represents the sum-to-zero constraint to apply for `tint` splines.
`name`	simplified name of the smooth.spec call.

Value

design matrix

Examples

```
library(survPen)

# standard spline of time with 4 knots (so we get a design matrix with 3 columns
# because of centering constraint)
```

```
data <- data.frame(time=seq(0,5,length=100))

# retrieving sum-to-zero constraint matrices
Z.smf <- smooth.cons("time",knots=list(c(0,1,3,5)),df=4,option="smf",
data.spec=data,name="smf(time)")$Z.smf

# constructing the design matrices for Gauss-Legendre quadrature
smooth.c.int <- smooth.cons.integral("time",knots=list(c(0,1,3,5)),df=4,option="smf",data.spec=data,
name="smf(time)",Z.smf=Z.smf,Z.tensor=NULL,Z.tint=NULL)
```

`smooth.spec` *Covariates specified as penalized splines*

Description

Specifies the covariates to be considered as penalized splines.

Usage

```
smooth.spec(
  ...,
  knots = NULL,
  df = NULL,
  by = NULL,
  option = NULL,
  same.rho = FALSE
)
```

Arguments

`...`	Numeric vectors specified in `smf`, `tensor` or `tint`
`knots`	List of numeric vectors that specifies the knots of the splines (including boundaries); default is NULL
`df`	Degrees of freedom: numeric vector that indicates the number of knots desired for each covariate; default is NULL
`by`	numeric or factor variable in order to define a varying coefficient smooth; default is NULL
`option`	"smf", "tensor" or "tint". Depends on the wrapper function; default is "smf"
`same.rho`	if there is a factor by variable, should the smoothing parameters be the same for all levels; default is FALSE.

Value

object of class smooth.spec

`term`	Vector of strings giving the names of each covariate specified in ...
`dim`	Numeric value giving the number of covariates associated with this spline
`knots`	list of numeric vectors that specifies the knots for each covariate
`df`	Numeric vector giving the number of knots associated with each covariate
`by`	numeric or factor variable in order to define a varying coefficient smooth
`same.rho`	if there is a factor by variable, should the smoothing parameters be the same for all levels; default is FALSE
`name`	simplified name of the call to function smooth.spec

Examples

```
library(survPen)

# standard spline of time with 10 unspecified knots
smooth.spec(time)

# tensor of time and age with 5*5 specified knots
smooth.s <- smooth.spec(time,age,knots=list(time=seq(0,5,length=5),age=seq(20,80,length=5)),
option="tensor")
```

`splitmult` *Split original dataset at specified times to fit a multiplicative model*

Description

This function allows splitting the original dataset in order to retrieve all the expected mortality rates available according to each individual's follow-up time. Typically, the expected mortality rates come from national mortality tables and values are available for every combination of age and year (often with 1-year increment).

Usage

```
splitmult(data, cut, start = NULL, end, event)
```

Arguments

`data`	orginal datset
`cut`	vector of timepoints to cut at (usually every year of follow-up)
`start`	character string with name of start variable (will be created and set to zero if it does not exist)
`end`	character string with name of event time variable
`event`	character string with name of censoring indicator

Details

This function is close to the survsplit function proposed in relsurv package, but it is simpler since fewer features are needed.

Value

split dataset with follow-up time split at specified times. An 'id_row' column is added to identify original row numbers

Examples

```
library(survPen)
data(datCancer)
data(expected.table)

#-------------------- creating split dataset for multiplicative model

splitdat <- splitmult(datCancer, cut = (1:5), end = "fu",
event = "dead")

#-------------------- merging with expected mortality table

# deriving current age and year (closest whole number)
splitdat$age_current <- floor(splitdat$age + splitdat$fu + 0.5)

splitdat$year_current <- floor(splitdat$yod + splitdat$fu + 0.5)


splitdat <- merge(splitdat, expected.table,
                by.x=c("age_current","year_current"), by.y=c("Age","Year"),all.x=TRUE)
```

`summary.survPen` *Summary for a* `survPen` *fit*

Description

Takes a fitted `survPen` object and produces various useful summaries from it.

Usage

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

Arguments

`object`	a fitted `survPen` object as produced by `survPen.fit`
`...`	other arguments

Value

List of objects:

`call`	the original survPen call
`formula`	the original survPen formula
`coefficients`	reports the regression parameters estimates for unpenalized terms with the associated standard errors
`HR_TAB`	reports the exponential of the regression parameters estimates for unpenalized terms with the associated CI
`edf.per.smooth`	reports the edf associated with each smooth term
`random`	TRUE if there are random effects in the model
`random.effects`	reports the estimates of the log standard deviation (log(sd)) of every random effects plus the estimated standard error (also on the log(sd) scale)
`likelihood`	unpenalized likelihood of the model
`penalized.likelihood`	penalized likelihood of the model
`nb.smooth`	number of smoothing parameters
`smoothing.parameter`	smoothing parameters estimates
`parameters`	number of regression parameters
`edf`	effective degrees of freedom
`method`	smoothing selection criterion used (LAML or LCV)
`val.criterion`	minimized value of criterion. For LAML, what is reported is the negative log marginal likelihood
`converged`	convergence indicator, TRUE or FALSE. TRUE if Hess.beta.modif=FALSE and Hess.rho.modif=FALSE (or NULL)

Examples

```
library(survPen)

data(datCancer) # simulated dataset with 2000 individuals diagnosed with cervical cancer

# model : unidimensional penalized spline for time since diagnosis with 5 knots
f1 <- ~smf(fu,df=5)

# fitting hazard model
mod1 <- survPen(f1,data=datCancer,t1=fu,event=dead,expected=NULL,method="LAML")

# summary
summary(mod1)
```

---

survPen *(Excess) hazard model with (multidimensional) penalized splines and integrated smoothness estimation*

Description

Please have a look to survival_analysis_with_survPen vignette for a thorough description.

Fits an (excess) hazard model with (multidimensional) penalized splines allowing for time-dependent effects, non-linear effects and interactions between several continuous covariates. The linear predictor is specified on the logarithm of the (excess) hazard. Smooth terms are represented using cubic regression splines with associated quadratic penalties. For multidimensional smooths, tensor product splines or tensor product interactions are available. Smoothness is estimated automatically by optimizing one of two criteria: Laplace approximate marginal likelihood (LAML) or likelihood cross-validation (LCV). When specifying the model's formula, no distinction is made between the part relative to the form of the baseline hazard and the one relative to the effects of the covariates. Thus, time-dependent effects are naturally specified as interactions with some function of time via "*" or ":". See the examples below for more details. The main functions of the survPen package are `survPen`, `smf`, `tensor`, `tint` and `rd`. The first one fits the model while the other four are constructors for penalized splines.

The user must be aware that the `survPen` package does not depend on `mgcv`. Thus, all the functionalities available in `mgcv` in terms of types of splines (such as thin plate regression splines or P-splines) are not available in `survPen` (yet).

Usage

```
survPen(
  formula,
  data,
  t1,
  t0 = NULL,
  event,
  expected = NULL,
  lambda = NULL,
  rho.ini = NULL,
  max.it.beta = 200,
  max.it.rho = 30,
  beta.ini = NULL,
  detail.rho = FALSE,
  detail.beta = FALSE,
  n.legendre = NULL,
  method = "LAML",
  tol.beta = 1e-04,
  tol.rho = 1e-04,
  step.max = 5,
  type = "overall",
```

```
  cluster = NULL
)
```

Arguments

`formula`	formula object specifying the model. Penalized terms are specified using `smf` (comparable to `s(...,bs="cr")` in `mgcv`), `tensor` (comparable to `te(...,bs="cr")` in `mgcv`), `tint` (comparable to `ti(...,bs="cr")` in `mgcv`), or `rd` (comparable to `s(...,bs="re")` in `mgcv`).
`data`	an optional data frame containing the variables in the model
`t1`	vector of follow-up times or name of the column in `data` containing follow-up times
`t0`	vector of origin times or name of the column in `data` containing origin times; allows to take into account left truncation; default is NULL, in which case it will be a vector of zeroes
`event`	vector of right-censoring indicators or name of the column in `data` containing right-censoring indicators; 1 if the event occurred and 0 otherwise
`expected`	(for net survival only) vector of expected hazard or name of the column in `data` containing expected hazard; default is NULL, in which case overall survival will be estimated
`lambda`	vector of smoothing parameters; default is NULL when it is to be estimated by LAML or LCV
`rho.ini`	vector of initial log smoothing parameters; default is NULL, in which case every initial log lambda will be -1
`max.it.beta`	maximum number of iterations to reach convergence in the regression parameters; default is 200
`max.it.rho`	maximum number of iterations to reach convergence in the smoothing parameters; default is 30
`beta.ini`	vector of initial regression parameters; default is NULL, in which case the first beta will be `log(sum(event)/sum(t1))` and the others will be zero (except if there are "by" variables or if there is a piecewise constant hazard specification in which cases all betas are set to zero)
`detail.rho`	if TRUE, details concerning the optimization process in the smoothing parameters are displayed; default is FALSE
`detail.beta`	if TRUE, details concerning the optimization process in the regression parameters are displayed; default is FALSE
`n.legendre`	number of Gauss-Legendre quadrature nodes to be used to compute the cumulative hazard; default is NULL. If not supplied the value is set to 20 for (excess) hazard models and 10 for relative mortality ratio models
`method`	criterion used to select the smoothing parameters. Should be "LAML" or "LCV"; default is "LAML"
`tol.beta`	convergence tolerance for regression parameters; default is `1e-04`. See `NR.beta` for details

`tol.rho`	convergence tolerance for smoothing parameters; default is `1e-04`. See `NR.rho` for details
`step.max`	maximum absolute value possible for any component of the step vector (on the log smoothing parameter scale) in LCV or LAML optimization; default is 5. If necessary, consider lowering this value to achieve convergence
`type`	should be either 'overall' for hazard regression, 'net' for excess hazard regression, or 'mult' for relative mortality ratio regression
`cluster`	cluster variable for marginal hazard (intensity) models

Details

In time-to-event analysis, we may deal with one or several continuous covariates whose functional forms, time-dependent effects and interaction structure are challenging. One possible way to deal with these effects and interactions is to use the classical approximation of the survival likelihood by a Poisson likelihood. Thus, by artificially splitting the data, the package `mgcv` can then be used to fit penalized hazard models (Remontet et al. 2018). The problem with this option is that the setup is rather complex and the method can fail with huge datasets (before splitting). Wood et al. (2016) provided a general penalized framework that made available smooth function estimation to a wide variety of models. They proposed to estimate smoothing parameters by maximizing a Laplace approximate marginal likelihood (LAML) criterion and demonstrate how statistical consistency is maintained by doing so. The `survPen` function implements the framework described by Wood et al. (2016) for modelling time-to-event data without requiring data splitting and Poisson likelihood approximation. The effects of continuous covariates are represented using low rank spline bases with associated quadratic penalties. The `survPen` function allows to account simultaneously for time-dependent effects, non-linear effects and interactions between several continuous covariates without the need to build a possibly demanding model-selection procedure. Besides LAML, a likelihood cross-validation (LCV) criterion (O Sullivan 1988) can be used for smoothing parameter estimation. First and second derivatives of LCV with respect to the smoothing parameters are implemented so that LCV optimization is computationally equivalent to the LAML optimization proposed by Wood et al. (2016). In practice, LAML optimization is generally both a bit faster and a bit more stable so it is used as default. For m covariates $(x_1, \ldots, x_m)$, if we note $h(t, x_1, \ldots, x_m)$ the hazard at time t, the hazard model is the following :

$$log[h(t, x_1, \ldots, x_m)] = \sum_j g_j(t, x_1, \ldots, x_m)$$

where each g_j is either the marginal basis of a specific covariate or a tensor product smooth of any number of covariates. The marginal bases of the covariates are represented as natural (or restricted) cubic splines (as in function `ns` from library `splines`) with associated quadratic penalties. Full parametric (unpenalized) terms for the effects of covariates are also possible (see the examples below). Each g_j is then associated with zero, one or several smoothing parameters. The estimation procedure is based on outer Newton-Raphson iterations for the smoothing parameters and on inner Newton-Raphson iterations for the regression parameters (see Wood et al. 2016). Estimation of the regression parameters in the inner algorithm is by direct maximization of the penalized likelihood of the survival model, therefore avoiding data augmentation and Poisson likelihood approximation. The cumulative hazard included in the log-likelihood is approximated by Gauss-Legendre quadrature for numerical stability.

Value

Object of class "survPen" (see `survPenObject` for details)

by variables

The `smf`, `tensor` and `tint` terms used to specify smooths accept an argument `by`. This `by` argument allows for building varying-coefficient models i.e. for letting smooths interact with factors or parametric terms. If a `by` variable is numeric, then its ith element multiples the ith row of the model matrix corresponding to the smooth term concerned. If a `by` variable is a factor then it generates an indicator vector for each level of the factor, unless it is an ordered factor. In the non-ordered case, the model matrix for the smooth term is then replicated for each factor level, and each copy has its rows multiplied by the corresponding rows of its indicator variable. The smoothness penalties are also duplicated for each factor level. In short a different smooth is generated for each factor level. The main interest of by variables over separated models is the `same.rho` argument (for `smf`, `tensor` and `tint`) which allows forcing all smooths to have the same smoothing parameter(s). Ordered `by` variables are handled in the same way, except that no smooth is generated for the first level of the ordered factor. This is useful if you are interested in differences from a reference level.

See the survival_analysis_with_survPen vignette for more details.

Random effects

i.i.d random effects can be specified using penalization. Indeed, the ridge penalty is equivalent to an assumption that the regression parameters are i.i.d. normal random effects. Thus, it is easy to fit a frailty hazard model. For example, consider the model term `rd(clust)` which will result in a model matrix component corresponding to `model.matrix(~clust-1)` being added to the model matrix for the whole model. The associated regression parameters are assumed i.i.d. normal, with unknown variance (to be estimated). This assumption is equivalent to an identity penalty matrix (i.e. a ridge penalty) on the regression parameters. The unknown smoothing parameter λ associated with the term `rd(clust)` is directly linked to the unknown variance σ^2: $\sigma^2 = \frac{1}{\lambda * S.scale}$. Then, the estimated log standard deviation is: $log(\hat{\sigma}) = -0.5 * log(\hat{\lambda}) - 0.5 * log(S.scale)$. And the estimated variance of the log standard deviation is: $Var[log(\hat{\sigma})] = 0.25 * Var[log(\hat{\lambda})] = 0.25 * inv.Hess.rho$. See the survival_analysis_with_survPen vignette for more details.

This approach allows implementing commonly used random effect structures. For example if `g` is a factor then `rd(g)` produces a random parameter for each level of `g`, the random parameters being i.i.d. normal. If `g` is a factor and `x` is numeric, then `rd(g,x)` produces an i.i.d. normal random slope relating the response to `x` for each level of `g`. Thus, random effects treated as penalized splines allow specifying frailty (excess) hazard models (Charvat et al. 2016). For each individual i from cluster (usually geographical unit) j, a possible model would be:

$$log[h(t_{ij}, x_{ij1}, \ldots, x_{ijm})] = \sum_k g_k(t_{ij}, x_{ij1}, \ldots, x_{ijm}) + w_j$$

where `w_j` follows a normal distribution with mean 0. The random effect associated with the cluster variable is specified with the model term `rd(cluster)`. We could also specify a random effect depending on age for example with the model term `rd(cluster,age)`. `u_j = exp(w_j)` is known as the shared frailty.

See the survival_analysis_with_survPen vignette for more details.

Excess hazard model

When studying the survival of patients who suffer from a common pathology we may be interested in the concept of excess mortality that represents the mortality due to that pathology. For example, in cancer epidemiology, individuals may die from cancer or from another cause. The problem is that the cause of death is often either unavailable or unreliable. Supposing that the mortality due to other causes may be obtained from the total mortality of the general population (called expected mortality for cancer patients), we can define the concept of excess mortality. The excess mortality is directly linked to the concept of net survival, which would be the observed survival if patients could not die from other causes. Therefore, when such competing events are present, one may choose to fit an excess hazard model instead of a classical hazard model. Flexible excess hazard models have already been proposed (for examples see Remontet et al. 2007, Charvat et al. 2016) but none of them deals with a penalized framework (in a non-fully Bayesian setting). Excess mortality can be estimated supposing that, in patients suffering from a common pathology, mortality due to others causes than the pathology can be obtained from the (all cause) mortality of the general population; the latter is referred to as the expected mortality h_P. The mortality observed in the patients (h_O) is actually decomposed as the sum of h_P and the excess mortality due to the pathology (h_E). This may be written as:

$$h_O(t,x) = h_E(t,x) + h_P(a+t,z)$$

In that equation, t is the time since cancer diagnosis, a is the age at diagnosis, h_P is the mortality of the general population at age $a+t$ given demographical characteristics z (h_P is considered known and available from national statistics), and x a vector of variables that may have an effect on h_E. Including the age in the model is necessary in order to deal with the informative censoring due to other causes of death. Thus, for m covariates $(x_1,\dots,x_m)$, if we note $h_E(t,x_1,\dots,x_m)$ the excess hazard at time t, the excess hazard model is the following:

$$log[h_E(t,x_1,\dots,x_m)] = \sum_j g_j(t,x_1,\dots,x_m)$$

Relative mortality ratio model

Another important feature of the `survPen` package is that it allows fitting penalized relative mortality ratio models.

As we discussed above, the excess mortality setting considers that the mortality (all causes) observed in the patients (h_O) is actually decomposed as the sum of the expected mortality h_P and the excess mortality due to the pathology (h_E).

This may be written as:

$$h_O(t,x) = h_E(t,x) + h_P(a+t,z)$$

One limitation of such a decomposition is that h_E is considered positive. Indeed, sometimes this assumption is not met. For example, in prostate cancer patients with low stages at diagnosis, we observe an 'undermortality' due to selection effects and better overall medical care. In that case, the excess mortality is actually neagtive and the net survival setting fails to describe the reality of those patients. Besides, the excess mortality setting considers the studied disease as an independent cause of death (conditionally on the covariates) compared to the other causes. This point of view is not usely considered in multiple sclerosis epidemiology for example, where the disease is seen as a comorbidity impacting all pre- existing causes of death. In that case, the observed hazard is decomposed as product of population hazard and a relative mortality ratio r

This may be written as:

$$h_O(t,x) = r(t,x) * h_P(a+t,z)$$

This decomposition was first proposed in a modelling framework by Andersen et al. (1985). However Andersen's model was a non-flexible semi-parametric model.

The `survPen` package allows modelling the relative mortality ratio r as a multidimensional function of time and covariates. For m covariates $(x_1,\ldots,x_m)$, if we note $r(t,x_1,\ldots,x_m)$ the relative mortality ratio at time t, the model is as follows:

$$log[r(t,x_1,\ldots,x_m)] = \sum_j g_j(t,x_1,\ldots,x_m)$$

Where the g_j functions may be penalized unidimensional or penalized tensor product splines. All features described for the (excess) hazard setting still apply when fitting a relative mortality ratio model. One difference lies in the predictions. With a fitted relative mortality ratio model, you can only retrieve the relative mortality ratio and cumulative relative mortality ratio predictions (with CIs), as well as the ratios of realtive mortality ratio (with type='HR'). No survival prediction (let alone survival difference) will be directly available because its calculation depends on expected mortality rates.

Finally, one important difference between an excess hazard model and relative mortality ratio model is data preparation. For an excess hazard model we only need individual data with expected mortality rate at the time of death. Whereas in a relative mortality ratio model, the contribution to an individual to the likelihood requires all possible expected mortality rate values during the entire follow-up. Therefore, since the expected mortality rates come from national mortality tables usually available in 1-year intervals, we need to split the original dataset as many times as there are 1-year intervals during each individual's follow-up. The function `splitmult` will help you getting the splitdataset from the original one.

See the survival_analysis_with_survPen vignette for more details and an example of analysis.

Marginal hazard (intensity) models with robust standard errors

In presence of correlated time-to-event data (for example recurrent event data), robust standard errors accounting for said correlation need to be derived. The 'survPen' package allows deriving such robust standard errors based on sandwich estimators (often called Huber sandwich estimator, see also Coz et al. submitted to Biostatistics, for an example in the recurrent event setting).

The user only needs to specify the 'cluster' variable defining the statistical units for which repeated observations are available. This specification is performed via the 'cluster' argument.

See the survival_analysis_with_survPen vignette for more details and an example of analysis.

Convergence

No convergence indicator is given. If the function returns an object of class `survPen`, it means that the algorithm has converged. If convergence issues occur, an error message is displayed. If convergence issues occur, do not refrain to use detail.rho and/or detail.beta to see exactly what is going on in the optimization process. To achieve convergence, consider lowering step.max and/or changing rho.ini and beta.ini. If your excess hazard model fails to converge, consider fitting a hazard model and use its estimated parameters as initial values for the excess hazard model. Finally, do not refrain to change the "method" argument (LCV or LAML) if convergence issues occur.

Other

Be aware that all character variables are transformed to factors before fitting.

References

Andersen, P. K., Borch-Johnsen, K., Deckert, T., Green, A., Hougaard, P., Keiding, N., and Kreiner, S. (1985). A Cox regression model for the relative mortality and its application to diabetes mellitus survival data. Biometrics, 921-932.

Charvat, H., Remontet, L., Bossard, N., Roche, L., Dejardin, O., Rachet, B., ... and Belot, A. (2016), A multilevel excess hazard model to estimate net survival on hierarchical data allowing for non linear and non proportional effects of covariates. Statistics in medicine, 35(18), 3066-3084.

Coz, E., Charvat, H., Maucort-Boulch, D., and Fauvernier, M. (submitted to Biostatistics). Flexible penalized marginal intensity models for recurrent event data. Fauvernier, M., Roche, L., Uhry, Z., Tron, L., Bossard, N., Remontet, L. and the CENSUR Working Survival Group. Multidimensional penalized hazard model with continuous covariates: applications for studying trends and social inequalities in cancer survival, in revision in the Journal of the Royal Statistical Society, series C.

O Sullivan, F. (1988), Fast computation of fully automated log-density and log-hazard estimators. SIAM Journal on scientific and statistical computing, 9(2), 363-379.

Remontet, L., Bossard, N., Belot, A., & Esteve, J. (2007), An overall strategy based on regression models to estimate relative survival and model the effects of prognostic factors in cancer survival studies. Statistics in medicine, 26(10), 2214-2228.

Remontet, L., Uhry, Z., Bossard, N., Iwaz, J., Belot, A., Danieli, C., Charvat, H., Roche, L. and CENSUR Working Survival Group (2018) Flexible and structured survival model for a simultaneous estimation of non-linear and non-proportional effects and complex interactions between continuous variables: Performance of this multidimensional penalized spline approach in net survival trend analysis. Stat Methods Med Res. 2018 Jan 1:962280218779408. doi: 10.1177/0962280218779408. [Epub ahead of print].

Wood, S.N., Pya, N. and Saefken, B. (2016), Smoothing parameter and model selection for general smooth models (with discussion). Journal of the American Statistical Association 111, 1548-1575

Examples

```
library(survPen)
data(datCancer) # simulated dataset with 2000 individuals diagnosed with cervical cancer

#------------------------------------------------------------ example 0
# Comparison between restricted cubic splines and penalized restricted cubic splines

library(splines)

# unpenalized
```

```
f <- ~ns(fu,knots=c(0.25, 0.5, 1, 2, 4),Boundary.knots=c(0,5))

mod <- survPen(f,data=datCancer,t1=fu,event=dead)

# penalized
f.pen <- ~ smf(fu,knots=c(0,0.25, 0.5, 1, 2, 4,5)) # careful here: the boundary knots are included

mod.pen <- survPen(f.pen,data=datCancer,t1=fu,event=dead)

# predictions

new.time <- seq(0,5,length=100)
pred <- predict(mod,data.frame(fu=new.time))
pred.pen <- predict(mod.pen,data.frame(fu=new.time))

par(mfrow=c(1,1))
plot(new.time,pred$haz,type="l",ylim=c(0,0.2),main="hazard vs time",
xlab="time since diagnosis (years)",ylab="hazard",col="red")
lines(new.time,pred.pen$haz,col="blue3")
legend("topright",legend=c("unpenalized","penalized"),
col=c("red","blue3"),lty=rep(1,2))



#-------------------------------------------------------- example 1
# hazard models with unpenalized formulas compared to a penalized tensor product smooth

library(survPen)
data(datCancer) # simulated dataset with 2000 individuals diagnosed with cervical cancer

# constant hazard model
f.cst <- ~1
mod.cst <- survPen(f.cst,data=datCancer,t1=fu,event=dead)

# piecewise constant hazard model
f.pwcst <- ~pwcst(breaks=seq(0,5,by=0.5))
mod.pwcst <- survPen(f.pwcst,data=datCancer,t1=fu,event=dead)

# linear effect of time
f.lin <- ~fu
mod.lin <- survPen(f.lin,data=datCancer,t1=fu,event=dead)

# linear effect of time and age with proportional effect of age
f.lin.age <- ~fu+age
mod.lin.age <- survPen(f.lin.age,data=datCancer,t1=fu,event=dead)

# linear effect of time and age with time-dependent effect of age (linear)
f.lin.inter.age <- ~fu*age
mod.lin.inter.age <- survPen(f.lin.inter.age,data=datCancer,t1=fu,event=dead)

# cubic B-spline of time with a knot at 1 year, linear effect of age and time-dependent effect
# of age with a quadratic B-spline of time with a knot at 1 year
library(splines)
```

```
f.spline.inter.age <- ~bs(fu,knots=c(1),Boundary.knots=c(0,5))+age+
age:bs(fu,knots=c(1),Boundary.knots=c(0,5),degree=2)
# here, bs indicates an unpenalized cubic spline

mod.spline.inter.age <- survPen(f.spline.inter.age,data=datCancer,t1=fu,event=dead)


# tensor of time and age
f.tensor <- ~tensor(fu,age)
mod.tensor <- survPen(f.tensor,data=datCancer,t1=fu,event=dead)


# predictions of the models at age 60

new.time <- seq(0,5,length=100)
pred.cst <- predict(mod.cst,data.frame(fu=new.time))
pred.pwcst <- predict(mod.pwcst,data.frame(fu=new.time))
pred.lin <- predict(mod.lin,data.frame(fu=new.time))
pred.lin.age <- predict(mod.lin.age,data.frame(fu=new.time,age=60))
pred.lin.inter.age <- predict(mod.lin.inter.age,data.frame(fu=new.time,age=60))
pred.spline.inter.age <- predict(mod.spline.inter.age,data.frame(fu=new.time,age=60))
pred.tensor <- predict(mod.tensor,data.frame(fu=new.time,age=60))

lwd1 <- 2

par(mfrow=c(1,1))
plot(new.time,pred.cst$haz,type="l",ylim=c(0,0.2),main="hazard vs time",
xlab="time since diagnosis (years)",ylab="hazard",col="blue3",lwd=lwd1)
segments(x0=new.time[1:99],x1=new.time[2:100],y0=pred.pwcst$haz[1:99],col="lightblue2",lwd=lwd1)
lines(new.time,pred.lin$haz,col="green3",lwd=lwd1)
lines(new.time,pred.lin.age$haz,col="yellow",lwd=lwd1)
lines(new.time,pred.lin.inter.age$haz,col="orange",lwd=lwd1)
lines(new.time,pred.spline.inter.age$haz,col="red",lwd=lwd1)
lines(new.time,pred.tensor$haz,col="black",lwd=lwd1)
legend("topright",
legend=c("cst","pwcst","lin","lin.age","lin.inter.age","spline.inter.age","tensor"),
col=c("blue3","lightblue2","green3","yellow","orange","red","black"),
lty=rep(1,7),lwd=rep(lwd1,7))


# you can also calculate the hazard yourself with the lpmatrix option.
# For example, compare the following predictions:
haz.tensor <- pred.tensor$haz

X.tensor <- predict(mod.tensor,data.frame(fu=new.time,age=60),type="lpmatrix")
haz.tensor.lpmatrix <- exp(X.tensor%mult%mod.tensor$coefficients)

summary(haz.tensor.lpmatrix - haz.tensor)

#---------------- The 95% confidence intervals can be calculated like this:

# standard errors from the Bayesian covariance matrix Vp
std <- sqrt(rowSums((X.tensor%mult%mod.tensor$Vp)*X.tensor))
```

```
qt.norm <- stats::qnorm(1-(1-0.95)/2)
haz.inf <- as.vector(exp(X.tensor%mult%mod.tensor$coefficients-qt.norm*std))
haz.sup <- as.vector(exp(X.tensor%mult%mod.tensor$coefficients+qt.norm*std))

# checking that they are similar to the ones given by the predict function
summary(haz.inf - pred.tensor$haz.inf)
summary(haz.sup - pred.tensor$haz.sup)


#---------------------------------------------------------- example 2

library(survPen)
data(datCancer) # simulated dataset with 2000 individuals diagnosed with cervical cancer

# model : unidimensional penalized spline for time since diagnosis with 5 knots
f1 <- ~smf(fu,df=5)
# when knots are not specified, quantiles are used. For example, for the term "smf(x,df=df1)",
# the vector of knots will be: quantile(unique(x),seq(0,1,length=df1))

# you can specify your own knots if you want
# f1 <- ~smf(fu,knots=c(0,1,3,6,8))

# hazard model
mod1 <- survPen(f1,data=datCancer,t1=fu,event=dead,expected=NULL,method="LAML")
summary(mod1)

# to see where the knots were placed
mod1$list.smf

# with LCV instead of LAML
mod1bis <- survPen(f1,data=datCancer,t1=fu,event=dead,expected=NULL,method="LCV")
summary(mod1bis)

# hazard model taking into account left truncation (not representative of cancer data,
# the begin variable was simulated for illustration purposes only)
mod2 <- survPen(f1,data=datCancer,t0=begin,t1=fu,event=dead,expected=NULL,method="LAML")
summary(mod2)

# excess hazard model
mod3 <- survPen(f1,data=datCancer,t1=fu,event=dead,expected=rate,method="LAML")
summary(mod3)

# compare the predictions of the models
new.time <- seq(0,5,length=50)
pred1 <- predict(mod1,data.frame(fu=new.time))
pred1bis <- predict(mod1bis,data.frame(fu=new.time))
pred2 <- predict(mod2,data.frame(fu=new.time))
pred3 <- predict(mod3,data.frame(fu=new.time))

# LAML vs LCV
par(mfrow=c(1,2))
plot(new.time,pred1$haz,type="l",ylim=c(0,0.2),main="LCV vs LAML",
```

```
xlab="time since diagnosis (years)",ylab="hazard")
lines(new.time,pred1bis$haz,col="blue3")
legend("topright",legend=c("LAML","LCV"),col=c("black","blue3"),lty=c(1,1))

plot(new.time,pred1$surv,type="l",ylim=c(0,1),main="LCV vs LAML",
xlab="time since diagnosis (years)",ylab="survival")
lines(new.time,pred1bis$surv,col="blue3")



# hazard vs excess hazard
par(mfrow=c(1,2))
plot(new.time,pred1$haz,type="l",ylim=c(0,0.2),main="hazard vs excess hazard",
xlab="time since diagnosis (years)",ylab="hazard")
lines(new.time,pred3$haz,col="green3")
legend("topright",legend=c("overall","excess"),col=c("black","green3"),lty=c(1,1))

plot(new.time,pred1$surv,type="l",ylim=c(0,1),main="survival vs net survival",
xlab="time",ylab="survival")
lines(new.time,pred3$surv,col="green3")
legend("topright",legend=c("overall survival","net survival"), col=c("black","green3"), lty=c(1,1))

# hazard vs excess hazard with 95% Bayesian confidence intervals (based on Vp matrix,
# see predict.survPen)
par(mfrow=c(1,1))
plot(new.time,pred1$haz,type="l",ylim=c(0,0.2),main="hazard vs excess hazard",
xlab="time since diagnosis (years)",ylab="hazard")
lines(new.time,pred3$haz,col="green3")
legend("topright",legend=c("overall","excess"),col=c("black","green3"),lty=c(1,1))

lines(new.time,pred1$haz.inf,lty=2)
lines(new.time,pred1$haz.sup,lty=2)

lines(new.time,pred3$haz.inf,lty=2,col="green3")
lines(new.time,pred3$haz.sup,lty=2,col="green3")



#---------------------------------------------------------- example 3

library(survPen)
data(datCancer) # simulated dataset with 2000 individuals diagnosed with cervical cancer

# models: tensor product smooth vs tensor product interaction of time since diagnosis and
# age at diagnosis. Smoothing parameters are estimated via LAML maximization
f2 <- ~tensor(fu,age,df=c(5,5))

f3 <- ~tint(fu,df=5)+tint(age,df=5)+tint(fu,age,df=c(5,5))

# hazard model
mod4 <- survPen(f2,data=datCancer,t1=fu,event=dead)
summary(mod4)
```

```
mod5 <- survPen(f3,data=datCancer,t1=fu,event=dead)
summary(mod5)

# predictions
new.age <- seq(50,90,length=50)
new.time <- seq(0,7,length=50)

Z4 <- outer(new.time,new.age,function(t,a) predict(mod4,data.frame(fu=t,age=a))$haz)
Z5 <- outer(new.time,new.age,function(t,a) predict(mod5,data.frame(fu=t,age=a))$haz)

# color settings
col.pal <- colorRampPalette(c("white", "red"))
colors <- col.pal(100)

facet <- function(z){

facet.center <- (z[-1, -1] + z[-1, -ncol(z)] + z[-nrow(z), -1] + z[-nrow(z), -ncol(z)])/4
cut(facet.center, 100)

}

# plot the hazard surfaces for both models
par(mfrow=c(1,2))
persp(new.time,new.age,Z4,col=colors[facet(Z4)],main="tensor",theta=30,
xlab="time since diagnosis",ylab="age at diagnosis",zlab="excess hazard",ticktype="detailed")
persp(new.time,new.age,Z5,col=colors[facet(Z5)],main="tint",theta=30,
xlab="time since diagnosis",ylab="age at diagnosis",zlab="excess hazard",ticktype="detailed")

#---------------------------------------------------------- example 4

library(survPen)
data(datCancer) # simulated dataset with 2000 individuals diagnosed with cervical cancer

# model : tensor product spline for time, age and yod (year of diagnosis)
# yod is not centered here since it does not create unstability but be careful in practice
# and consider centering your covariates if you encounter convergence issues
f4 <- ~tensor(fu,age,yod,df=c(5,5,5))

# excess hazard model
mod6 <- survPen(f4,data=datCancer,t1=fu,event=dead,expected=rate)
summary(mod6)


# predictions of the surfaces for ages 50, 60, 70 and 80
new.year <- seq(1990,2010,length=30)
new.time <- seq(0,5,length=50)

Z_50 <- outer(new.time,new.year,function(t,y) predict(mod6,data.frame(fu=t,yod=y,age=50))$haz)
Z_60 <- outer(new.time,new.year,function(t,y) predict(mod6,data.frame(fu=t,yod=y,age=60))$haz)
Z_70 <- outer(new.time,new.year,function(t,y) predict(mod6,data.frame(fu=t,yod=y,age=70))$haz)
Z_80 <- outer(new.time,new.year,function(t,y) predict(mod6,data.frame(fu=t,yod=y,age=80))$haz)
```

```
# plot the hazard surfaces for a given age
par(mfrow=c(2,2))
persp(new.time,new.year,Z_50,col=colors[facet(Z_50)],main="age 50",theta=20,
xlab="time since diagnosis",ylab="yod",zlab="excess hazard",ticktype="detailed")
persp(new.time,new.year,Z_60,col=colors[facet(Z_60)],main="age 60",theta=20,
xlab="time since diagnosis",ylab="yod",zlab="excess hazard",ticktype="detailed")
persp(new.time,new.year,Z_70,col=colors[facet(Z_70)],main="age 70",theta=20,
xlab="time since diagnosis",ylab="yod",zlab="excess hazard",ticktype="detailed")
persp(new.time,new.year,Z_80,col=colors[facet(Z_80)],main="age 80",theta=20,
xlab="time since diagnosis",ylab="yod",zlab="excess hazard",ticktype="detailed")

########################################
```

`survPen.fit` *(Excess) hazard model with multidimensional penalized splines for given smoothing parameters*

Description

Fits an (excess) hazard model. If penalized splines are present, the smoothing parameters are specified.

Usage

```
survPen.fit(
  build,
  data,
  formula,
  max.it.beta = 200,
  beta.ini = NULL,
  detail.beta = FALSE,
  method = "LAML",
  tol.beta = 1e-04
)
```

Arguments

`build`	list of objects returned by `model.cons`
`data`	an optional data frame containing the variables in the model
`formula`	formula object specifying the model
`max.it.beta`	maximum number of iterations to reach convergence in the regression parameters; default is 200

`beta.ini`	vector of initial regression parameters; default is NULL, in which case the first beta will be `log(sum(event)/sum(t1))` and the others will be zero (except if there are "by" variables or if there is a piecewise constant hazard specification in which cases all betas are set to zero)
`detail.beta`	if TRUE, details concerning the optimization process in the regression parameters are displayed; default is FALSE
`method`	criterion used to select the smoothing parameters. Should be "LAML" or "LCV"; default is "LAML"
`tol.beta`	convergence tolerance for regression parameters; default is `1e-04`. See `NR.beta` for details

Value

Object of class "survPen" (see `survPenObject` for details)

Examples

```
library(survPen)

# standard spline of time with 4 knots

data <- data.frame(time=seq(0,5,length=100),event=1,t0=0)

form <- ~ smf(time,knots=c(0,1,3,5))

t1 <- eval(substitute(time), data)
t0 <- eval(substitute(t0), data)
event <- eval(substitute(event), data)

# Setting up the model before fitting
model.c <- model.cons(form,lambda=0,data.spec=data,t1=t1,t1.name="time",
t0=rep(0,100),t0.name="t0",event=event,event.name="event",
expected=rep(0,100),expected.name=NULL,type="overall",n.legendre=20,
cl="survPen(form,data,t1=time,event=event)",beta.ini=NULL)

# fitting
mod <- survPen.fit(model.c,data,form)
```

`survPenObject` *Fitted survPen object*

Description

A fitted survPen object returned by function `survPen` and of class "survPen". Method functions predict and summary are available for this class.

Value

A `survPen` object has the following elements:

`call`	original `survPen` call
`formula`	formula object specifying the model
`t0.name`	name of the vector of origin times
`t1.name`	name of the vector of follow-up times
`event.name`	name of the vector of right-censoring indicators
`expected.name`	name of the vector of expected hazard
`haz`	fitted hazard
`coefficients`	estimated regression parameters. Unpenalized parameters are first, followed by the penalized ones
`type`	"net" for net survival estimation with penalized excess hazard model, "overall" for overall survival with penalized hazard model, or "mult" for penalized relative mortality ratio model
`df.para`	degrees of freedom associated with fully parametric terms (unpenalized)
`df.smooth`	degrees of freedom associated with penalized terms
`p`	number of regression parameters
`edf`	effective degrees of freedom
`edf1`	alternative effective degrees of freedom ; used as an upper bound for edf2
`edf2`	effective degrees of freedom corrected for smoothing parameter uncertainty
`aic`	Akaike information criterion with number of parameters replaced by edf when there are penalized terms. Corresponds to 2*edf - 2*ll.unpen
`aic2`	Akaike information criterion corrected for smoothing parameter uncertainty. Be careful though, this is still a work in progress, especially when one of the smoothing parameters tends to infinity.
`iter.beta`	vector of numbers of iterations needed to estimate the regression parameters for each smoothing parameters trial. It thus contains `iter.rho+1` elements.
`X`	design matrix of the model
`S`	penalty matrix of the model
`S.scale`	vector of rescaling factors for the penalty matrices
`S.list`	Equivalent to pen but with every element multiplied by its associated smoothing parameter
`S.smf`	List of penalty matrices associated with all "smf" calls
`S.tensor`	List of penalty matrices associated with all "tensor" calls
`S.tint`	List of penalty matrices associated with all "tint" calls
`S.rd`	List of penalty matrices associated with all "rd" calls
`smooth.name.smf`	List of names for the "smf" calls associated with S.smf
`smooth.name.tensor`	List of names for the "tensor" calls associated with S.tensor

`smooth.name.tint`	List of names for the "tint" calls associated with S.tint
`smooth.name.rd`	List of names for the "rd" calls associated with S.rd
`S.pen`	List of all the rescaled penalty matrices redimensioned to df.tot size. Every element of `S.pen` noted `S.pen[[i]]` is made from a penalty matrix `pen[[i]]` returned by `smooth.cons` and is multiplied by S.scale
`grad.unpen.beta`	gradient vector of the log-likelihood with respect to the regression parameters
`grad.beta`	gradient vector of the penalized log-likelihood with respect to the regression parameters
`Hess.unpen.beta`	hessian of the log-likelihood with respect to the regression parameters
`Hess.beta`	hessian of the penalized log-likelihood with respect to the regression parameters
`Hess.beta.modif`	if TRUE, the hessian of the penalized log-likelihood has been perturbed at convergence
`ll.unpen`	log-likelihood at convergence
`ll.pen`	penalized log-likelihood at convergence
`deriv.rho.beta`	transpose of the Jacobian of beta with respect to the log smoothing parameters
`deriv.rho.inv.Hess.beta`	list containing the derivatives of the inverse of `Hess` with respect to the log smoothing parameters
`deriv.rho.Hess.unpen.beta`	list containing the derivatives of `Hess.unpen` with respect to the log smoothing parameters
`lambda`	estimated or given smoothing parameters
`nb.smooth`	number of smoothing parameters
`iter.rho`	number of iterations needed to estimate the smoothing parameters
`optim.rho`	identify whether the smoothing parameters were estimated or not; 1 when exiting the function `NR.rho`; default is NULL
`method`	criterion used for smoothing parameter estimation
`criterion.val`	value of the criterion used for smoothing parameter estimation at convergence
`LCV`	Likelihood cross-validation criterion at convergence
`LAML`	negative Laplace approximate marginal likelihood at convergence
`grad.rho`	gradient vector of criterion with respect to the log smoothing parameters
`Hess.rho`	hessian matrix of criterion with respect to the log smoothing parameters
`inv.Hess.rho`	inverse of `Hess.rho`
`Hess.rho.modif`	if TRUE, the hessian of LCV or LAML has been perturbed at convergence
`Ve`	Frequentist covariance matrix
`Vr`	Robust frequentist covariance matrix accounting for correlated survival times
`Vp`	Bayesian covariance matrix

`Vc`	Bayesian covariance matrix corrected for smoothing parameter uncertainty (see Wood et al. 2016)
`Vc.approx`	Kass and Steffey approximation of `Vc` (see Wood et al. 2016)
`Z.smf`	List of matrices that represents the sum-to-zero constraint to apply for `smf` splines
`Z.tensor`	List of matrices that represents the sum-to-zero constraint to apply for `tensor` splines
`Z.tint`	List of matrices that represents the sum-to-zero constraint to apply for `tint` splines
`list.smf`	List of all `smf.smooth.spec` objects contained in the model
`list.tensor`	List of all `tensor.smooth.spec` objects contained in the model
`list.tint`	List of all `tint.smooth.spec` objects contained in the model
`list.rd`	List of all `rd.smooth.spec` objects contained in the model
`U.F`	Eigen vectors of S.F, useful for the initial reparameterization to separate penalized ad unpenalized subvectors. Allows stable evaluation of the log determinant of S and its derivatives
`is.pwcst`	TRUE if there is a piecewise constant (excess) hazard specification. In that case the cumulative hazard can be derived without Gauss-Legendre quadrature
`pwcst.breaks`	if is.pwcst is TRUE, vector of breaks defining the sub-intervals on which the hazard is constant. Otherwise NULL.
`factor.structure`	List containing the levels and classes of all factor variables present in the data frame used for fitting
`converged`	convergence indicator, TRUE or FALSE. TRUE if Hess.beta.modif=FALSE and Hess.rho.modif=FALSE (or NULL)

References

Wood, S.N., Pya, N. and Saefken, B. (2016), Smoothing parameter and model selection for general smooth models (with discussion). Journal of the American Statistical Association 111, 1548-1575

`tensor.in` *tensor model matrix for two marginal bases*

Description

Function called recursively inside `tensor.prod.X`.

Usage

```
tensor.in(X1, X2)
```

Arguments

`X1`	first marginal design matrix with n rows and p1 columns
`X2`	first marginal design matrix with n rows and p2 columns

Value

Matrix of dimensions n*(p1*p2) representing the row tensor product of the matrices X1 and X2

Examples

```
library(survPen)

# row-wise tensor product between two design matrices
set.seed(15)

X1 <- matrix(rnorm(10*3),nrow=10,ncol=3)
X2 <- matrix(rnorm(10*2),nrow=10,ncol=2)
tensor.in(X1,X2)
```

`tensor.prod.S` *Tensor product for penalty matrices*

Description

Computes the penalty matrices of a tensor product smooth from the marginal penalty matrices. The code is from function `tensor.prod.penalties` in `mgcv` package.

Usage

```
tensor.prod.S(S)
```

Arguments

`S`	list of m marginal penalty matrices

Value

`TS`	List of the penalty matrices associated with the tensor product smooth

Examples

```
library(survPen)

# tensor product between three penalty matrices
set.seed(15)

S1 <- matrix(rnorm(3*3),nrow=3,ncol=3)
```

```
S2 <- matrix(rnorm(2*2),nrow=2,ncol=2)

S1 <- 0.5*(S1 + t(S1) ) ; S2 <- 0.5*(S2 + t(S2) )

tensor.prod.S(list(S1,S2))
```

`tensor.prod.X` *tensor model matrix*

Description

Computes the model matrix of tensor product smooth from the marginal bases.

Usage

```
tensor.prod.X(X)
```

Arguments

`X`	list of m design matrices with n rows and p1, p2, ... pm columns respectively

Value

`T`	Matrix of dimensions n*(p1*p2*...*pm) representing the row tensor product of the matrices in X

Examples

```
library(survPen)

# row-wise tensor product between three design matrices
set.seed(15)

X1 <- matrix(rnorm(10*3),nrow=10,ncol=3)
X2 <- matrix(rnorm(10*2),nrow=10,ncol=2)
X3 <- matrix(rnorm(10*2),nrow=10,ncol=2)
tensor.prod.X(list(X1,X2,X3))
```

`%cross%` *Matrix cross-multiplication between two matrices*

Description

Matrix cross-multiplication between two matrices

Usage

```
Mat1 %cross% Mat2
```

Arguments

`Mat1`	a matrix.
`Mat2`	another matrix.

Value

prod the product t(Mat1)

`%mult%` *Matrix multiplication between two matrices*

Description

Matrix multiplication between two matrices

Usage

```
Mat1 %mult% Mat2
```

Arguments

`Mat1`	a matrix.
`Mat2`	another matrix.

Value

prod the product Mat1

`%vec%` *Matrix multiplication between a matrix and a vector*

Description

Matrix multiplication between a matrix and a vector

Usage

```
Mat %vec% vec
```

Arguments

`Mat`	a matrix.
`vec`	a vector.

Value

prod the product Mat

Index