

Package ‘rstpm2’

May 29, 2018

Type Package

Title Generalized Survival Models

Version 1.4.2

Date 2018-05-29

Depends R (>= 3.0.2), methods, survival, splines

Imports graphics, Rcpp (>= 0.10.2), numDeriv, stats, mgcv, bbmle (>= 1.0.20), fastGHQuad

Suggests RUnit, eha

LinkingTo Rcpp,RcppArmadillo

Author Mark Clements [aut, cre],
Xing-Rong Liu [aut],
Paul Lambert [ctb]

Maintainer Mark Clements <mark.clements@ki.se>

Description R implementation of generalized survival models, where $g(S(t|x))=\eta(t,x)$ for a link function g , survival S at time t with covariates x and a linear predictor $\eta(t,x)$. The main assumption is that the time effect(s) are smooth. For fully parametric models with natural splines, this re-implements Stata's 'stpm2' function, which are flexible parametric survival models developed by Royston and colleagues. We have extended the parametric models to include any smooth parametric smoothers for time. We have also extended the model to include any smooth penalized smoothers from the 'mgcv' package, using penalized likelihood. These models include left truncation, right censoring, interval censoring, gamma frailties and normal random effects. This also includes a smooth implementation of accelerated failure time models.

URL <http://github.com/mclements/rstpm2>

BugReports <http://github.com/mclements/rstpm2/issues>

License GPL-2 | GPL-3

LazyData yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-05-29 13:45:06 UTC

R topics documented:

Rstpm2-package	2
aft	3
aft-class	5
brcancer	6
coef<-	7
colon	7
cox.tvc	8
eform.stpm2	9
grad	10
incrVar	10
legendre.quadrature.rule.200	11
lines.stpm2	12
nsx	13
nsxD	15
numDeltaMethod	17
plot-methods	18
popmort	19
predict-methods	20
predict.nsx	22
predictnl	23
predictnl-methods	25
pstpm2	25
pstpm2-class	29
residuals-methods	31
rstpm2-internal	32
stpm2	32
stpm2-class	36
tvcCoxph-class	37
Index	39

Rstpm2-package	<i>Flexible parametric survival models.</i>
----------------	---

Description

The package implements the stpm2 models from Stata. Such models use a flexible parametric formulation for survival models, using natural splines to model the log-cumulative hazard. Model predictions are rich, allowing for direct estimation of the hazard, survival, hazard ratios, hazard differences and survival differences. The models allow for time-varying effects, left truncation and relative survival.

The R implementation departs from the Stata implementation, using the ns() function, which is based on a projection of B-splines, rather than using truncated power splines as per Stata.

Details

```

Package: Rstpm2
Type: Package
Version: 1.0
Date: 2011-07-06
License: GPL-2
LazyLoad: yes
Depends: methods, bbmle
Imports: splines, survival, stats, graphics

```

The package exports the `stpm2` object, which inherits from the `mle2` object from the `bbmle` package. Methods are specified for the `stpm2` object, including `predict` and `plot` methods.

Author(s)

Mark Clements and Paul Lambert.
 Maintainer: <mark.clements@ki.se>

See Also

[stpm2](#)

Examples

```

data(brcancer)
summary(fit <- stpm2(Surv(rectime, censrec==1)~hormon, data=brcancer, df=3))
summary(fit.tvc <- stpm2(Surv(rectime, censrec==1)~hormon, data=brcancer, df=3,
  tvc=list(hormon=3)))
anova(fit, fit.tvc)
plot(fit.tvc, newdata=data.frame(hormon=0), type="hr", var="hormon")

```

aft

Parametric accelerated failure time model with smooth time functions

Description

This implements the accelerated failure time models $S_0(t \exp(\beta x))$ and $S_0(\int_0^t \exp(\beta x(u)) du)$. The baseline function $S_0(t^*)$ is modelled as $\exp(-\exp(\eta_0(\log(t^*))))$, where $\eta_0(\log(t^*))$ is a linear predictor using natural splines.

Usage

```

aft(formula, data, smooth.formula = NULL, df = 3,
  control = list(parscale = 1, maxit = 1000),
  init = NULL, weights = NULL, timeVar = "", time0Var = "",
  log.time.transform = TRUE,
  reltol = 1e-08, trace = 0, contrasts = NULL, subset = NULL,
  use.gr = TRUE, ...)

```

Arguments

<code>formula</code>	a formula object, with the response on the left of a <code>~</code> operator, and the regression terms (excluding time) on the right. The response should be a survival object as returned by the <code>Surv</code> function. The terms can include linear effects for any time-varying coefficients. [required]
<code>data</code>	a data-frame in which to interpret the variables named in the <code>formula</code> argument. [at present: required]
<code>smooth.formula</code>	a formula for describing the time effects for the linear predictor, excluding the baseline $S_0(t^*)$, but including time-dependent acceleration factors. The time-dependent acceleration factors can be modelled with any smooth functions.
<code>df</code>	an integer that describes the degrees of freedom for the <code>ns</code> function for modelling the baseline log-cumulative hazards function (default=3).
<code>control</code>	control argument passed to <code>optim</code> .
<code>init</code>	<code>init</code> should either be <code>FALSE</code> , such that initial values will be determined using Cox regression, or a numeric vector of initial values.
<code>weights</code>	an optional vector of 'prior weights' to be used in the fitting process. Should be <code>NULL</code> or a numeric vector.
<code>timeVar</code>	string variable defining the time variable. By default, this is determined from the survival object, however this may be ambiguous if two variables define the time.
<code>time0Var</code>	string variable to determine the entry variable; useful for when more than one data variable is used in the entry time.
<code>log.time.transform</code>	logical for whether to log-transform time when calculating the design matrix for the derivative of S_0 with respect to time.
<code>reltol</code>	relative tolerance for the model convergence
<code>trace</code>	integer for whether to provide trace information from the <code>optim</code> procedure
<code>contrasts</code>	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process.
<code>use.gr</code>	logical indicating whether to use gradients in the calculation
<code>...</code>	additional arguments to be passed to the <code>mle2</code> .

Details

The implementation extends the `mle2` object from the `bbmle` package. The model inherits all of the methods from the `mle2` class.

Value

An `stpm2`-class object that inherits from `mle2`-class.

Author(s)

Mark Clements.

See Also

[survreg](#), [coxph](#)

Examples

```
summary(aft(Surv(rectime,censrec==1)~hormon,data=brcancer,df=4))
```

aft-class	<i>Class "stpm2" ~~~</i>
-----------	--------------------------

Description

Regression object for aft.

Objects from the Class

Objects can be created by calls of the form `new("aft", ...)` and `aft(...)`.

Slots

args: Object of class "list" ~~

Extends

Class "[mle2](#)", directly.

Methods

plot signature(x = "aft", y = "missing"): ...

predict signature(object = "aft"): ...

predictnl signature(object = "aft", ...): ...

Examples

```
showClass("aft")
```

brcancer

German breast cancer data from Stata.

Description

See <http://www.stata-press.com/data/r11/brcancer.dta>.

Usage

```
data(brcancer)
```

Format

A data frame with 686 observations on the following 15 variables.

id a numeric vector

hormon hormonal therapy

x1 age, years

x2 menopausal status

x3 tumour size, mm

x4 tumour grade

x5 number of positive nodes

x6 progesterone receptor, fmol

x7 estrogen receptor, fmol

rectime recurrence free survival time, days

censrec censoring indicator

x4a tumour grade \geq 2

x4b tumour grade $=$ 3

x5e $\exp(-0.12*x5)$

Examples

```
data(brcancer)
## maybe str(brcancer) ; plot(brcancer) ...
```

coef<- *Generic method to update the coef in an object.*

Description

Generic method to update the coef in an object.

Usage

```
coef(x) <- value
```

Arguments

x	object to be updated
value	value of the coefficient to be updated.

Details

This simple generic method is used for the numerical delta method.

Value

The updated object is returned.

Examples

```
##---- Should be DIRECTLY executable !! ----  
##-- ==> Define data, use random,  
##--or do help(data=index) for the standard data sets.  
  
## The function is currently defined as  
function (x, value)  
  UseMethod("coef<-")
```

colon *Colon cancer.*

Description

Diagnoses of colon cancer.

Usage

```
data(colon)
```

Format

A data frame with 15564 observations on the following 13 variables.

sex Sex (1=male, 2=female)
 age Age at diagnosis
 stage Clinical stage at diagnosis (1=Unknown, 2=Localised, 3=Regional, 4=Distant)
 mmdx Month of diagnosis
 yydx Year of diagnosis
 surv_mm Survival time in months
 surv_yy Survival time in years
 status Vital status at last contact (1=Alive, 2=Dead: cancer, 3=Dead; other, 4=Lost to follow-up)
 subsite Anatomical subsite of tumour (1=Coecum and ascending, 2=Transverse, 3=Descending and sigmoid, 4=Other and NOS)
 year8594 Year of diagnosis (1=Diagnosed 75-84, 2=Diagnosed 85-94)
 agegrp Age in 4 categories (1=0-44, 2=45-59, 3=60-74, 4=75+)
 dx Date of diagnosis
 exit Date of exit

Details

Caution: there is a colon dataset in the survival package. We recommend using `data(colon, package="rstm2")` to ensure the correct dataset is used.

Examples

```
data(colon, package="rstm2") # avoids name conflict with survival::colon
## maybe str(colon) ; ...
```

cox.tvc

Test for a time-varying effect in the coxph model

Description

Test for a time-varying effect in the coxph model by re-fitting the partial likelihood including a time-varying effect, plot the effect size, and return the re-fitted model. The main advantage of this function over the `tt()` special is that it scales well for moderate sized datasets (cf. `tt` which expands the dataset and scales very poorly).

Usage

```
cox.tvc(obj, var=NULL, method="logt")
```


Arguments

obj	A coxph object. Currently restricted to right censoring with Breslow ties and without stratification, etc.
var	String for the effect name. Currently assumes simple continuous effects.
method	A string representing the possible time transformations. Currently only "logt".

Value

Returns a tvcCoxph object (which inherits from the mle2 class) of the re-fitted model.

See Also

[coxph](#), [cox.zph](#)

Examples

```
## As per the example for cox.zph:
fit <- coxph(Surv(futime, fustat) ~ age + ecog.ps,
             data=ovarian)
temp <- rstpm2:::cox.tvc(fit, "age")
print(temp)           # display the results
plot(temp)           # plot curves
```

eform.stpm2	<i>S3 method for to provide exponentiated coefficients with confidence intervals.</i>
-------------	---

Description

S3 method for to provide exponentiated coefficients with confidence intervals.

Usage

```
eform(object, ...)
## S3 method for class 'stpm2'
eform(object, parm, level = 0.95, method = c("Profile"), name = "exp(beta)")
```

Arguments

object	regression object
parm	not currently used
level	significance level for the confidence interval
method	Currently only the profile method is available.
name	name for the fitted value
...	other arguments

grad	<i>gradient function (internal function)</i>
------	--

Description

Numerical gradient for a function at a given value (internal).

Usage

```
grad(func, x, ...)
```

Arguments

func	Function taking a vector argument x (returns a vector of length>=1)
x	vector of arguments for where the gradient is wanted.
...	other arguments to the function

Details

$(\text{func}(x+\text{delta}, \dots) - \text{func}(x-\text{delta}, \dots)) / (2 \text{ delta})$ where delta is the third root of the machine precision times $\text{pmax}(1, \text{abs}(x))$.

Value

A vector if $\text{func}(x)$ has length 1, otherwise a matrix with rows for x and columns for $\text{func}(x)$.

Author(s)

Mark Clements.

See Also

numDelta()

incrVar	<i>Utility that returns a function to increment a variable in a data-frame.</i>
---------	---

Description

A functional approach to defining an increment in one or more variables in a data-frame. Given a variable name and an increment value, return a function that takes any data-frame to return a data-frame with incremented values.

Usage

```
incrVar(var, increment = 1)
```

Arguments

var String for the name(s) of the variable(s) to be incremented
 increment Value that the variable should be incremented.

Details

Useful for defining transformations for calculating rate ratios.

Value

A function with a single data argument that increments the variables in the data list/data-frame.

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (var, increment = 1)
{
  n <- length(var)
  if (n > 1 && length(increment)==1)
    increment <- rep(increment, n)
  function(data) {
    for (i in 1:n) {
      data[[var[i]]] <- data[[var[i]]] + increment[i]
    }
    data
  }
}
```

legendre.quadrature.rule.200

Legendre quadrature rule for n=200.

Description

Legendre quadrature rule for n=200.

Usage

```
data(legendre.quadrature.rule.200)
```

Format

A data frame with 200 observations on the following 2 variables.

x x values between -1 and 1
 w weights

Examples

```
data(legendre.quadrature.rule.200)
## maybe str(legendre.quadrature.rule.200) ; ...
```

lines.stpm2

S3 methods for lines

Description

S3 methods for lines

Usage

```
## S3 method for class 'stpm2'
lines(x, newdata = NULL, type = "surv", col = 1, ci.col= "grey",
lty = par("lty"), ci = FALSE, rug = FALSE, var = NULL,
exposed = incrVar(var), times = NULL, ...)
## S3 method for class 'pstpm2'
lines(x, newdata = NULL, type = "surv", col = 1,
ci.col= "grey",
lty = par("lty"), ci = FALSE, rug = FALSE, var = NULL,
exposed = incrVar(var), times = NULL, ...)
```

Arguments

x	an stpm2 object
newdata	required list of new data. This defines the unexposed newdata (<i>excluding</i> the event times).
type	specify the type of prediction
col	line colour
lty	line type
ci.col	confidence interval colour
ci	whether to plot the confidence interval band (default=TRUE)
rug	whether to add a rug plot of the event times to the current plot (default=TRUE)
var	specify the variable name or names for the exposed/unexposed (names are given as characters)
exposed	function that takes newdata and returns the exposed dataset. By default, this increments var
times	specifies the times. By default, this uses a span of the observed times.
...	additional arguments (add to the plot command)

nsx *Generate a Basis Matrix for Natural Cubic Splines (with eXtensions)*

Description

Generate the B-spline basis matrix for a natural cubic spline (with eXtensions).

Usage

```
nsx(x, df = NULL, knots = NULL, intercept = FALSE,
    Boundary.knots = range(x), derivs = if (cure) c(2, 1) else c(2, 2),
    log = FALSE, centre = FALSE,
    cure = FALSE, stata.stpm2.compatible = FALSE)
```

Arguments

x	the predictor variable. Missing values are allowed.
df	degrees of freedom. One can supply df rather than knots; ns() then chooses $df - 1 - intercept + 4 - \text{sum}(\text{derivs})$ knots at suitably chosen quantiles of x (which will ignore missing values).
knots	breakpoints that define the spline. The default is no knots; together with the natural boundary conditions this results in a basis for linear regression on x. Typical values are the mean or median for one knot, quantiles for more knots. See also Boundary.knots.
intercept	if TRUE, an intercept is included in the basis; default is FALSE.
Boundary.knots	boundary points at which to impose the natural boundary conditions and anchor the B-spline basis (default the range of the data). If both knots and Boundary.knots are supplied, the basis parameters do not depend on x. Data can extend beyond Boundary.knots
derivs	an integer vector of length 2 with values between 0 and 2 giving the derivative constraint order at the left and right boundary knots; an order of 2 constrains the second derivative to zero ($f''(x)=0$); an order of 1 constrains the first and second derivatives to zero ($f'(x)=f''(x)=0$); an order of 0 constrains the zero, first and second derivatives to zero ($f(x)=f'(x)=f''(x)=0$)
log	a Boolean indicating whether the underlying values have been log transformed; (deprecated: only used to calculate derivatives in rstpm2:::stpm2Old)
centre	if specified, then centre the splines at this value (i.e. $f(\text{centre})=0$) (default=FALSE)
cure	a Boolean indicated whether to estimate cure; changes the default derivs argument, such that the right boundary has the first and second derivatives constrained to zero; defaults to FALSE
stata.stpm2.compatible	a Boolean to determine whether to use Stata stpm's default knot placement; defaults to FALSE

Value

A matrix of dimension $\text{length}(x) * \text{df}$ where either `df` was supplied or if `knots` were supplied, $\text{df} = \text{length}(\text{knots}) + 1 + \text{intercept}$. Attributes are returned that correspond to the arguments to `ns`, and explicitly give the `knots`, `Boundary.knots` etc for use by `predict.nsx()`.

`nsx()` is based on the functions `ns` and `spline.des`. It generates a basis matrix for representing the family of piecewise-cubic splines with the specified sequence of interior knots, and the natural boundary conditions. These enforce the constraint that the function is linear beyond the boundary knots, which can either be supplied, else default to the extremes of the data. A primary use is in modeling formula to directly specify a natural spline term in a model.

The extensions from `ns` are: specification of the derivative constraints at the boundary knots; whether to centre the knots; incorporation of cure using derivatives; compatible knots with Stata's `stpm2`; and an indicator for a log-transformation of `x` for calculating derivatives.

References

Hastie, T. J. (1992) Generalized additive models. Chapter 7 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

See Also

`ns`, `bs`, `predict.nsx`, `SafePrediction`

Examples

```
require(stats); require(graphics); require(splines)
nsx(women$height, df = 5)
summary(fm1 <- lm(weight ~ ns(height, df = 5), data = women))

## example of safe prediction
plot(women, xlab = "Height (in)", ylab = "Weight (lb)")
ht <- seq(57, 73, length.out = 200)
lines(ht, predict(fm1, data.frame(height=ht)))
```

nsxD	<i>Generate a Basis Matrix for the first derivative of Natural Cubic Splines (with eXtensions)</i>
------	--

Description

Generate the B-spline basis matrix for the first derivative of a natural cubic spline (with eXtensions).

Usage

```
nsxD(x, df = NULL, knots = NULL, intercept = FALSE,
     Boundary.knots = range(x), derivs = if (cure) c(2, 1) else c(2, 2),
     log = FALSE, centre = FALSE,
     cure = FALSE, stata.stpm2.compatible = FALSE)
```

Arguments

x	the predictor variable. Missing values are allowed.
df	degrees of freedom. One can supply df rather than knots; ns() then chooses $df - 1 - intercept + 4 - sum(derivs)$ knots at suitably chosen quantiles of x (which will ignore missing values).
knots	breakpoints that define the spline. The default is no knots; together with the natural boundary conditions this results in a basis for linear regression on x. Typical values are the mean or median for one knot, quantiles for more knots. See also Boundary.knots.
intercept	if TRUE, an intercept is included in the basis; default is FALSE.
Boundary.knots	boundary points at which to impose the natural boundary conditions and anchor the B-spline basis (default the range of the data). If both knots and Boundary.knots are supplied, the basis parameters do not depend on x. Data can extend beyond Boundary.knots
derivs	an integer vector of length 2 with values between 0 and 2 giving the derivative constraint order at the left and right boundary knots; an order of 2 constrains the second derivative to zero ($f''(x)=0$); an order of 1 constrains the first and second derivatives to zero ($f'(x)=f''(x)=0$); an order of 0 constrains the zero, first and second derivatives to zero ($f(x)=f'(x)=f''(x)=0$)
log	a Boolean indicating whether the underlying values have been log transformed; (deprecated: only used to calculate derivatives in rstpm2:::stpm2Old)
centre	if specified, then centre the splines at this value (i.e. $f(centre)=0$) (default=FALSE)
cure	a Boolean indicated whether to estimate cure; changes the default derivs argument, such that the right boundary has the first and second derivatives constrained to zero; defaults to FALSE
stata.stpm2.compatible	a Boolean to determine whether to use Stata stpm's default knot placement; defaults to FALSE

Value

A matrix of dimension $\text{length}(x) * \text{df}$ where either `df` was supplied or if `knots` were supplied, $\text{df} = \text{length}(\text{knots}) + 1 + \text{intercept}$. Attributes are returned that correspond to the arguments to `ns`, and explicitly give the `knots`, `Boundary.knots` etc for use by `predict.nxD()`.

`nxD()` is based on the functions `ns` and `spline.des`. It generates a basis matrix for representing the family of piecewise-cubic splines with the specified sequence of interior knots, and the natural boundary conditions. These enforce the constraint that the function is linear beyond the boundary knots, which can either be supplied, else default to the extremes of the data. A primary use is in modeling formula to directly specify a natural spline term in a model.

The extensions from `ns` are: specification of the derivative constraints at the boundary knots; whether to centre the knots; incorporation of cure using derivatives; compatible knots with Stata's `stpm2`; and an indicator for a log-transformation of `x` for calculating derivatives.

References

Hastie, T. J. (1992) Generalized additive models. Chapter 7 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

See Also

`ns`, `bs`, `predict.nsx`, `SafePrediction`

Examples

```
require(stats); require(graphics); require(splines)
nsx(women$height, df = 5)
summary(fm1 <- lm(weight ~ ns(height, df = 5), data = women))

## example of safe prediction
plot(women, xlab = "Height (in)", ylab = "Weight (lb)")
ht <- seq(57, 73, length.out = 200)
lines(ht, predict(fm1, data.frame(height=ht)))
```

numDeltaMethod	<i>Calculate numerical delta method for non-linear predictions.</i>
----------------	---

Description

Given a regression object and an independent prediction function (as a function of the coefficients), calculate the point estimate and standard errors

Usage

```
numDeltaMethod(object, fun, gd=NULL, ...)
```

Arguments

object	A regression object with methods <code>coef</code> and <code>vcov</code> .
fun	An independent prediction function with signature <code>function(coef, ...)</code> .
gd	Specified gradients
...	Other arguments passed to <code>fun</code> .

Details

A more user-friendly interface is provided by `predictnl`.

Value

Estimate	Point estimates
SE	Standard errors

See Also

See Also [predictnl](#).

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (object, fun, ...)
{
  coef <- coef(object)
  est <- fun(coef, ...)
  Sigma <- vcov(object)
  gd <- grad(fun, coef, ...)
  se.est <- as.vector(sqrt(colSums(gd * (Sigma %*% gd))))
  data.frame(Estimate = est, SE = se.est)
}
```

plot-methods

*plots for an stpm2 fit***Description**

Given an stpm2 fit, return a plot

Usage

```
## S4 method for signature 'stpm2'
plot(x,y,newdata,type="surv",
      xlab="Time",line.col=1,ci.col="grey",
      add=FALSE,ci=TRUE,rug=TRUE,
      var=NULL,exposed=incrVar(var),times=NULL,...)

## S4 method for signature 'pstpm2'
plot(x,y,newdata,type="surv",
      xlab="Time",line.col=1,ci.col="grey",
      add=FALSE,ci=TRUE,rug=TRUE,
      var=NULL,exposed=incrVar(var),times=NULL,...)
```

Arguments

x	an stpm2 object
y	not used (for generic compatibility)
newdata	required list of new data. This defines the unexposed newdata (<i>excluding</i> the event times).
type	specify the type of prediction
xlab	x-axis label
line.col	line colour
ci.col	confidence interval colour
ci	whether to plot the confidence interval band (default=TRUE)
add	whether to add to the current plot (add=TRUE) or make a new plot (add=FALSE) (default=FALSE)
rug	whether to add a rug plot of the event times to the current plot (default=TRUE)
var	specify the variable name or names for the exposed/unexposed (names are given as characters)
exposed	function that takes newdata and returns the exposed dataset. By default, this increments var
times	specifies the times. By default, this uses a span of the observed times.
...	additional arguments (add to the plot command)

Methods

`x = "stpm2", y = "missing"` an stpm2 fit

See Also

[stpm2](#)

popmort

Background mortality rates for the colon dataset.

Description

Background mortality rates for the colon dataset.

Usage

```
data(popmort)
```

Format

A data frame with 10600 observations on the following 5 variables.

`sex` Sex (1=male, 2=female)

`prob` One year probability of survival

`rate` All cause mortality rate

`age` Age by single year of age through to age 105 years

`year` Calendar period

Examples

```
data(popmort)
## maybe str(popmort) ; ...
```

predict-methods

*Predicted values for an stpm2 or pstpm2 fit***Description**

Given an stpm2 fit and an optional list of new data, return predictions

Usage

```
## S4 method for signature 'stpm2'
predict(object, newdata=NULL,
        type=c("surv", "cumhaz", "hazard", "density", "hr", "sdiff",
              "hdiff", "loghazard", "link", "meansurv", "meansurvdiff", "meanhr",
              "odds", "or", "margsurv", "marghaz", "marghr", "meanhaz", "af",
              "fail", "margfail", "meanmargsurv", "uncured", "rmst", "probcure"),
        grid=FALSE, seqLength=300,
        type.relsurv=c("excess", "total", "other"), scale=365.24,
        rmap=list(), ratetable=survival::survexp.us,
        se.fit=FALSE, link=NULL, exposed=incrVar(var), var=NULL,
        keep.attributes=FALSE, use.gr=TRUE, level=0.95, ...)

## S4 method for signature 'pstpm2'
predict(object, newdata=NULL,
        type=c("surv", "cumhaz", "hazard", "density", "hr", "sdiff",
              "hdiff", "loghazard", "link", "meansurv", "meansurvdiff", "meanhr",
              "odds", "or", "margsurv", "marghaz", "marghr", "meanhaz", "af",
              "fail", "margfail", "meanmargsurv", "rmst"),
        grid=FALSE, seqLength=300,
        se.fit=FALSE, link=NULL, exposed=incrVar(var), var=NULL,
        keep.attributes=FALSE, use.gr=TRUE, level=0.95, ...)
```

Arguments

object	an stpm2 or pstpm2 object
newdata	optional list of new data (required if type in ("hr", "sdiff", "hdiff", "meansurvdiff", "or", "uncured")). For type in ("hr", "sdiff", "hdiff", "meansurvdiff", "or", "af", "uncured"), this defines the unexposed newdata. This can be combined with grid to get a regular set of event times (i.e. newdata would <i>not</i> include the event times).
type	specify the type of prediction: <ul style="list-style-type: none"> • "surv" survival probabilities • "cumhaz" cumulative hazard • "hazard" hazard • "density" density • "hr" hazard ratio • "sdiff" survival difference

	<ul style="list-style-type: none"> • "hdiff" hazard difference • "loghazard" log hazards • "meansurv" mean survival • "meansurvdiff" mean survival difference • "odds" odds • "or" odds ratio • "margsurv" marginal (population) survival • "marghaz" marginal (population) hazard • "marghr" marginal (population) hazard ratio • "meanhaz" mean hazard • "meanhr" mean hazard ratio • "af" attributable fraction • "fail" failure (=1-survival) • "margfail" marginal failure (=1-marginal survival) • "meanmargsurv" mean marginal survival, averaged over the frailty distribution • "uncured" distribution for the uncured • "rmst" restricted mean survival time • "probcure" probability of cure
grid	whether to merge newdata with a regular sequence of event times (default=FALSE)
seqLength	length of the sequence used when grid=TRUE
type.relsurv	type of predictions for relative survival models: either "excess", "total" or "other"
scale	scale to go from the days in the ratetable object to the analysis time used in the analysis
rmap	an optional list that maps data set names to the ratetable names. See survexp
ratetable	a table of event rates used in relative survival when type.relsurv is "total" or "other"
se.fit	whether to calculate confidence intervals (default=FALSE)
link	allows a different link for the confidence interval calculation (default=NULL, such that switch(type,surv="cloglog",cumhaz="log",hazard="log",hr="log",sdiff="I",hdiff="I",loghazard="I",link="I",odds="log",or="log",margsurv="cloglog",marghaz="log",marghr="log"))
exposed	a function that takes newdata and returns a transformed data-frame for those exposed or the counterfactual (defaults to incrementing "var")
var	specify the variable name or names for the exposed/unexposed (names are given as characters)
level	significance level for the confidence interval
keep.attributes	Boolean to determine whether the output should include the newdata as an attribute (default=TRUE)
use.gr	Boolean to determine whether to use gradients in the variance calculations when they are available (default=TRUE)
...	additional arguments (for generic compatibility)

Details

The confidence interval estimation is based on the delta method using numerical differentiation.

Value

A data-frame with components Estimate, lower and upper, with an attribute "newdata" for the newdata data-frame.

Methods

object= "stpm2" an stpm2 fit

See Also

[stpm2](#)

predict.nsx

Evaluate a Spline Basis

Description

Evaluate a predefined spline basis at given values.

Usage

```
## S3 method for class 'nsx'
predict(object, newx, ...)
```

Arguments

object	the result of a call to nsx having attributes describing knots, degree, etc.
newx	the x values at which evaluations are required.
...	Optional additional arguments. At present no additional arguments are used.

Value

An object just like object, except evaluated at the new values of x.

These are methods for the generic function [predict](#) for objects inheriting from classes "nsx". See [predict](#) for the general behavior of this function.

See Also

[nsx](#).

Examples

```
basis <- nsx(women$height, df = 5)
newX <- seq(58, 72, length.out = 51)
# evaluate the basis at the new data
predict(basis, newX)
```

predictnl

Estimation of standard errors using the numerical delta method.

Description

A simple, yet exceedingly useful, approach to estimate the variance of a function using the numerical delta method. A number of packages provide functions that analytically calculate the gradients; we use numerical derivatives, which generalises to models that do not offer analytical derivatives (e.g. ordinary differential equations, integration), or to examples that are tedious or error-prone to calculate (e.g. sums of predictions from GLMs).

Usage

```
## Default S3 method:
predictnl(object, fun, newdata=NULL, gd=NULL, ...)
## S3 method for class 'lm'
predictnl(object, fun, newdata=NULL, ...)
## S3 method for class 'predictnl'
print(x, ...)
## S3 method for class 'formula'
predict(object, data, newdata, na.action, type="model.matrix", ...)
## S3 method for class 'predictnl'
confint(object, parm, level=0.95, ...)
```

Arguments

object	An object with coef, vcov and `coef<-` methods (required).
fun	A function that takes object as the first argument, possibly with newdata and other arguments (required). See notes for why it is often useful to include newdata as an argument to the function.
newdata	An optional argument that defines newdata to be passed to fun.
gd	An optional matrix of gradients. If this is not specified, then the gradients are calculated using finite differences.
parm	currently ignored
level	significance level for 2-sided confidence intervals
x	a predictnl object to be printed.
data	object used to define the model frame
na.action	passed to model.frame
type	currently restricted to "model.matrix"
...	Other arguments that are passed to fun.

Details

The signature for fun is either `fun(object, ...)` or `fun(object, newdata=NULL, ...)`.

The different `predictnl` methods call the utility function `numDeltaMethod`, which in turn calls the `grad` function for numerical differentiation. The `numDeltaMethod` function calls the standard `coef` and `vcov` methods, and the non-standard ``coef<-`` method for changing the coefficients in a regression object. This non-standard method has been provided for several regression objects and essentially mirrors the `coef` method.

One potential issue is that some `predict` methods do not re-calculate their predictions for the fitted dataset (i.e. when `newdata=NULL`). As the `predictnl` function changes the fitted coefficients, it is required that the predictions are re-calculated. One solution is to pass `newdata` as an argument to both `predictnl` and `fun`; alternatively, `newdata` can be specified in `fun`. These approaches are described in the examples below. The `numDeltaMethod` method called by `predictnl` provides a warning when the variance estimates are zero, which may be due to this cause.

For completeness, it is worth discussing why the example `predictnl(fit,predict)` does not work for when `fit` is a `glm` object. First, `predict.glm` does not update the predictions for the fitted data. Second, the default `predict` method has a signature `predict(object, ...)`, which does not include a `newdata` argument. We could then either (i) require that a `newdata` argument be passed to the `fun` function for all examples, which would make this corner case work, or (ii) only pass the `newdata` argument if it is non-null or in the formals for the `fun` function, which would fail for this corner case. The current API defaults to the latter case (ii). To support this approach, the `predictnl.lm` method replaces a null `newdata` with `object$data`. We also provide a revised `numdelta:::predict.lm` method that performs the same operation, although its use is not encouraged due to its clumsiness.

Value

Returns an object of class an object with class `c("predictnl", "data.frame")` elements `c("fit", "se.fit", "Estimate"`, and with methods `print` and `confint`. Note that the `Estimate` and `SE` fields are deprecated and their use is discouraged, as we would like to remove them from future releases.

Author(s)

Mark Clements

Examples

```
df <- data.frame(x=0:1, y=c(10, 20))
fit <- glm(y ~ x, df, family=poisson)

predictnl(fit,
          function(obj)
            diff(predict(obj, type="response")))
```

predictnl-methods *~~ Methods for Function predictnl ~~*

Description

~~ Methods for function predictnl ~~

Methods

predictnl signature(object = "mle2", ...): Similar to predictnl.default, using S4 methods.

pstpm2 *Penalised generalised survival model*

Description

This implements the generalised survival model $g(S(t|x)) = \eta$, where g is a link function, S is survival, t is time, x are covariates and η is a linear predictor. The linear predictor can include penalised smoothers for the time effects, for time:covariate interactions and for covariate effects using the mgcv smoothers. The main model assumption is that the time effects in the linear predictor are smooth. This extends the class of flexible parametric survival models developed by Royston and colleagues. The model has been extended to include relative survival, Gamma frailties and normal random effects.

Usage

```
pstpm2(formula, data, smooth.formula = NULL, smooth.args = NULL,
        logH.args = NULL,
        tvc = NULL,
        control = list(parscale = 1, maxit = 300), init = NULL,
        coxph.strata = NULL, coxph.formula = NULL,
        weights = NULL, robust = FALSE,
        bhazard = NULL, bhazinit = 0.1, timeVar = "", time0Var = "",
        sp=NULL, use.gr = TRUE,
        criterion=c("GCV","BIC"), penalty = c("logH","h"),
        smoother.parameters = NULL,
        alpha=if (is.null(sp)) switch(criterion,GCV=1,BIC=1) else 1,
        sp.init=1, trace = 0,
        link.type=c("PH","PO","probit","AH","AO"), theta.A0=0,
        optimiser = c("BFGS", "NelderMead", "Nlm"), log.time.transform=TRUE,
        recurrent = FALSE, frailty=!is.null(cluster) & !robust,cluster = NULL,
        logtheta=-6, nodes=9,
        RandDist=c("Gamma","LogN"), adaptive = TRUE, maxkappa=1000, Z = ~1,
        reltol = list(search = 1.0e-10, final = 1.0e-10, outer=1.0e-5),outer_optim=1,
        contrasts = NULL, subset = NULL, robust_initial = FALSE, ...)
```

Arguments

formula	a formula object, with the response on the left of a \sim operator, and the parametric terms on the right. The response must be a survival object as returned by the Surv function. [required]
data	a data.frame in which to interpret the variables named in the formula argument.
smooth.formula	a <code>mgcv::gam</code> formula for describing the time effects and time-dependent effects and smoothed covariate effects on the linear predictor scale (default=NULL). The default model is equal to $\sim s(\log(\text{time}), k=-1)$ where <code>time</code> is the time variable.
smooth.args	a list describing the arguments for the <code>s</code> function for modelling the baseline time effect on the linear predictor scale (default=NULL).
logH.args	as per <code>smooth.args</code> . Deprecated.
tvc	a list with the names of the time-varying coefficients (e.g. <code>tvc=list(hormon)</code> , which is equivalent to <code>smooth.formula=$\sim \dots + s(\log(\text{time}), \text{by}=\text{hormon})$</code>).
control	control argument passed to <code>optim</code> .
init	<code>init</code> should either be <code>FALSE</code> , such that initial values will be determined using Cox regression, or a numeric vector of initial values.
coxph.strata	variable in the data argument for stratification of the coxph model fit for estimating initial values.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be <code>NULL</code> or a numeric vector.
robust	Boolean used to determine whether to use a robust variance estimator.
bhazard	variable for the baseline hazard for relative survival
bhazinit	scalar used to adjust the background cumulative hazards for calculating initial values. Default=0.1.
timeVar	variable defining the time variable. By default, this is determined from the survival object, however this may be ambiguous if two variables define the time
sp	fix the value of the smoothing parameters.
use.gr	in R, a Boolean to determine whether to use the gradient in the optimisation
criterion	in Rcpp, determine whether to use "GCV" or "BIC" for for the smoothing parameter selection.
penalty	use either the "logH" penalty, which is the default penalty from <code>mgcv</code> , or the "h" hazard penalty.
smoother.parameters	for the hazard penalty, a list with components which are lists with components <code>var</code> , <code>transform</code> and <code>inverse</code> .
alpha	an ad hoc tuning parameter for the smoothing parameter.
sp.init	initial values for the smoothing parameters.
trace	integer for trace reporting; 0 represents no additional reporting.
contrasts	an optional list. See the <code>contrasts.arg</code> of model.matrix.default .

subset	an optional vector specifying a subset of observations to be used in the fitting process.
coxph.formula	additional formula used to improve the fitting of initial values [optional and rarely used].
time0Var	string variable to determine the entry variable; useful for when more than one data variable is used in the entry time.
link.type	type of link function. For "PH" (generalised proportional hazards), $g(S)=\log(-\log(S))$; for "PO" (generalised proportional odds), $g(S)=-\text{logit}(S)$; for "probit" (generalised probit), $g(S)=-\text{probit}(S)$; for "AH" (generalised additive hazards), $g(S)=-\log(S)$; for "AO" (generalised Aranda-Ordaz), $g(S)=\log((S^{-(\text{theta.AO})}-1)/\text{theta.AO})$.
theta.AO	theta parameter for the Aranda-Ordaz link type.
optimiser	select which optimiser is used
log.time.transform	should a log-transformation be used for calculating the derivative of the design matrix with respect to time? (default=TRUE)
recurrent	logical for whether clustered, left truncated data are recurrent or for first event (where the latter requires an adjustment for the frailties or random effects)
frailty	logical for whether to fit a shared frailty model
cluster	string for the data variable that determines the cluster for the frailty
logtheta	initial value for log-theta used in the gamma shared frailty model
nodes	number of integration points for Gaussian quadrature
RandDist	type of distribution for the random effect or frailty
adaptive	logical for whether to use adaptive or non-adaptive quadrature
maxkappa	double float value for the maximum value of the weight used in the constraint
Z	formula for the design matrix for the random effects
reltol	list with components for search and final relative tolerances.
outer_optim	Integer to indicate the algorithm for outer optimisation. If outer_optim=1, then use Nelder-Mead, otherwise use Nlm.
robust_initial	logical for whether to use Nelder-Mead to find initial values (max 50 iterations). This is useful for ill-posed initial values.
...	additional arguments to be passed to the mle2 .

Details

The implementation extends the `mle2` object from the `bbmle` package.

The default smoother for time on the linear predictor scale is `s(log(time))`.

Value

A `pstpm2`-class object.

Author(s)

Mark Clements, Xing-Rong Liu.

Examples

```
## Not run:
data(brcancer)
## standard Kaplan-Meier curves by hormon
plot(survfit(Surv(rectime/365,censrec==1)~1,data=brcancer,subset=hormon==1),
     xlab="Recurrence free survival time (years)",
     ylab="Survival")
lines(survfit(Surv(rectime/365,censrec==1)~1,data=brcancer,subset=hormon==0),col=2,
      conf.int=TRUE)
legend("topright", legend=c("Hormonal therapy","No hormonal therapy"),lty=1,col=1:2,bty="n")

## now fit a penalised stpm2 model
fit <- pstpm2(Surv(rectime/365,censrec==1)~hormon,data=brcancer)
## no S4 generic lines() method: instead, use plot(..., add=TRUE)
plot(fit,newdata=data.frame(hormon=1),type="surv",add=TRUE,ci=FALSE,line.col="blue",lwd=2,
     rug=FALSE)
plot(fit,newdata=data.frame(hormon=0),type="surv",add=TRUE,ci=FALSE,line.col="green",lwd=2,
     rug=FALSE)

## plot showing proportional hazards
plot(fit,newdata=data.frame(hormon=1),type="hazard",line.col="blue",lwd=2,
     rug=FALSE,ylim=c(0,1e-3))
plot(fit,newdata=data.frame(hormon=0),type="hazard",add=TRUE,ci=FALSE,line.col="green",lwd=2,
     rug=FALSE)

## time-varying hazard ratios
fit.tvc <- pstpm2(Surv(rectime,censrec==1)~1,
                 data=brcancer,
                 smooth.formula=~s(log(rectime))+s(log(rectime),by=hormon))
plot(fit.tvc,newdata=data.frame(hormon=1),type="hazard",line.col="blue",lwd=2,
     rug=FALSE)
plot(fit.tvc,newdata=data.frame(hormon=0),type="hazard",line.col="red",lwd=2,
     add=TRUE)

## Smooth covariate effects
fit.smoothx <- pstpm2(Surv(rectime,censrec==1)~1,
                    data=brcancer,
                    smooth.formula=~s(log(rectime))+s(x1))
ages <- seq(21,80,length=301)
haz <- predict(fit.smoothx,newdata=data.frame(hormon=1,rectime=365,x1=ages),
              type="hazard",se.fit=TRUE)
matplot(ages,haz/haz[150,1],type="l",log="y",ylab="Hazard ratio")

## compare with df=5 from stpm2
fit.stpm2 <- stpm2(Surv(rectime/365,censrec==1)~hormon,data=brcancer,df=7)
plot(fit,newdata=data.frame(hormon=1),type="hazard",line.col="blue",lwd=2,
     rug=FALSE,ylim=c(0,1e-3))
plot(fit.stpm2,newdata=data.frame(hormon=1),type="hazard",line.col="orange",lwd=2,
```

```

    rug=FALSE,add=TRUE,ci=FALSE)

## time-varying coefficient
##summary(fit.tvc <- pstpm2(Surv(rectime,censrec==1)~hormon,data=brcancer,
##                          tvc=list(hormon=3)))
##anova(fit,fit.tvc) # compare with and without tvc (unclear whether this is valid)

## some more plots
## plot(fit.tvc,newdata=data.frame(hormon=0),type="hr",var="hormon")
##                               # no lines method: use add=TRUE
## plot(fit.tvc,newdata=data.frame(hormon=1),type="hr",var="hormon",
##       add=TRUE,ci=FALSE,line.col=2)

## plot(fit.tvc,newdata=data.frame(hormon=0),type="sdiff",var="hormon")

## plot(fit.tvc,newdata=data.frame(hormon=0),type="hdiff",var="hormon")

## plot(fit.tvc,newdata=data.frame(hormon=0),type="hazard")
## plot(fit.tvc,newdata=data.frame(hormon=1),type="hazard",line.col=2,ci=FALSE,add=TRUE)

## End(Not run)

```

pstpm2-class

Class "pstpm2"

Description

Regression object for pstpm2.

Objects from the Class

Objects can be created by calls of the form `new("pstpm2", ...)` and `pstpm2(...)`.

Slots

xlevels: Object of class "list" ~~
contrasts: Object of class "listOrNULL" ~~
terms: Object of class "terms" ~~
gam: Object of class "gam" ~~
logli: Object of class "function" ~~
timeVar: Object of class "character" ~~
time0Var: Object of class "character" ~~
time0Expr: Object of class "nameOrcall" ~~
timeExpr: Object of class "nameOrcall" ~~
like: Object of class "function" ~~

```

model.frame: Object of class "list" ~~
fullformula: Object of class "formula" ~~
delayed: Object of class "logical" ~~
frailty: Object of class "logical" ~~
x: Object of class "matrix" ~~
xd: Object of class "matrix" ~~
termsd: Object of class "terms" ~~
Call: Object of class "character" ~~
y: Object of class "Surv" ~~
sp: Object of class "numeric" ~~
nevent: Object of class "numeric" ~~
link: Object of class "list" ~~
edf: Object of class "numeric" ~~
edf_var: Object of class "numeric" ~~
df: Object of class "numeric" ~~
call: Object of class "language" ~~
call.orig: Object of class "language" ~~
coef: Object of class "numeric" ~~
fullcoef: Object of class "numeric" ~~
vcov: Object of class "matrix" ~~
min: Object of class "numeric" ~~
details: Object of class "list" ~~
minuslogl: Object of class "function" ~~
method: Object of class "character" ~~
data: Object of class "list" ~~
formula: Object of class "character" ~~
optimizer: Object of class "character" ~~
args: Object of class "list" ~~

```

Extends

Class "[mle2](#)", directly.

Methods

```

plot signature(x = "pstpm2", y = "missing"): ...
lines signature(x = "pstpm2", ...): ...
anova signature(object = "pstpm2",...): ...
AIC signature(object = "pstpm2",...,k=2): ...

```

```

AICc signature(object = "pstpm2",...,nobs=NULL, k=2): ...
BIC signature(object = "pstpm2",..., nobs = NULL): ...
qAICc signature(object = "pstpm2",..., nobs = NULL, dispersion = 1, k = 2): ...
qAIC signature(object = "pstpm2",..., dispersion = 1, k = 2): ...
summary signature(object = "pstpm2",...): ...
eform signature(object = "pstpm2",...): ...
predictnl signature(object = "pstpm2",...): ...

```

Examples

```
showClass("pstpm2")
```

```
residuals-methods      Residual values for an stpm2 or pstpm2 fit
```

Description

Given an stpm2 or pstpm2 fit, return residuals

Usage

```

## S4 method for signature 'stpm2'
residuals(object, type=c("li","gradli"))
## S4 method for signature 'pstpm2'
residuals(object, type=c("li","gradli"))

```

Arguments

object	an stpm2 or pstpm2 object
type	specify the type of residuals: <ul style="list-style-type: none"> • "li" log-likelihood components (not strictly residuals) • "gradli" gradient of the log-likelihood components (not strictly residuals)

Details

The gradients are analytical.

Value

A vector or matrix.

Methods

object= "stpm2" an stpm2 fit

See Also[stpm2](#)

rstm2-internal	<i>Internal functions for the rstm2 package.</i>
----------------	--

Description

Various utility functions used internally to the rstm2 package.

Usage

```
lhs(formula)
rhs(formula)
lhs(formula) <- value
rhs(formula) <- value
```

Arguments

formula	A formula
value	A symbolic value to replace the current value.

stpm2	<i>Fully parametric generalised survival model</i>
-------	--

Description

This implements the generalised survival model $g(S(t|x)) = \eta$, where g is a link function, S is survival, t is time, x are covariates and η is a linear predictor. The main model assumption is that the time effects in the linear predictor are smooth. This extends the class of flexible parametric survival models developed by Royston and colleagues. The model has been extended to include relative survival, Gamma frailties and normal random effects.

Usage

```
stpm2(formula, data,
      smooth.formula = NULL, smooth.args = NULL,
      df = 3, cure = FALSE, logH.args = NULL,
      logH.formula = NULL, tvc = NULL, tvc.formula =
      NULL, control = list(parscale = 1, maxit = 300),
      init = NULL, coxph.strata = NULL, weights = NULL,
      robust = FALSE, baseoff = FALSE, bhazard = NULL,
      bhazinit = 0.1,
      timeVar = "", time0Var = "", use.gr = TRUE,
      optimiser=c("BFGS","NelderMead"), log.time.transform=TRUE,
```



```

reltol=1.0e-8, trace = 0,
link.type=c("PH","PO","probit","AH","A0"), theta.A0=0,
frailty = !is.null(cluster) & !robust, cluster = NULL, logtheta=-6, nodes=9,
RandDist=c("Gamma","LogN"), recurrent = FALSE, adaptive=TRUE,
maxkappa=1000, Z=~1,
contrasts = NULL,
subset = NULL, robust_initial = FALSE, ...)

```

Arguments

formula	a formula object, with the response on the left of a ~ operator, and the regression terms (excluding time) on the right. The response must be a survival object as returned by the Surv function. The terms should include linear terms for any time-varying coefficients. [required]
data	a data.frame in which to interpret the variables named in the formula argument. [at present: required]
smooth.formula	a formula for describing the time effects for the linear predictor, including the baseline and the time-dependent effects (default=NULL). Only one of df, smooth.formula, smooth.args, logH.args or logH.formula is required. The default model is equal to nsx(log(time),df=3).
smooth.args	a list describing the arguments for the nsx function for modelling the baseline time effect on the linear predictor scale (default=NULL). Use this or smooth.formula for changing the knot placement and specifying cure models.
df	an integer that describes the degrees of freedom for the ns function for modelling the baseline log-cumulative hazard (default=3).
logH.args	as per smooth.args. Deprecated.
logH.formula	as per smooth.formula. Deprecated.
tvc	a list with the names of the time-varying coefficients and the degrees of freedom (e.g. tvc=list(x=3) specifies x as a time-varying coefficient with 3 degrees of freedom).
tvc.formula	a formula for describing the time-varying coefficients. If a time-varying coefficient is being model, then only one of tvc and tvc.formula is required.
bhazard	a vector for the background hazard for relative survival estimation. At present, this does not use data and it is required for all individuals - although it is only used at the event times.
bhazinit	scalar used to adjust the background cumulative hazards for calculating initial values. Default=0.1.
control	control argument passed to optim.
init	init should either be FALSE, such that initial values will be determined using Cox regression, or a numeric vector of initial values.
coxph.strata	variable in the data argument for stratification of the coxph model fit for estimating initial values.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.

<code>robust</code>	Boolean used to determine whether to use a robust variance estimator.
<code>baseoff</code>	Boolean used to determine whether fully define the model using <code>tvc.formula</code> rather than combining <code>logH.formula</code> and <code>tvc.formula</code>
<code>timeVar</code>	variable defining the time variable. By default, this is determined from the survival object, however this may be ambiguous if two variables define the time
<code>contrasts</code>	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process.
<code>cure</code>	logical for whether to estimate a cure model.
<code>time0Var</code>	string variable to determine the entry variable; useful for when more than one data variable is used in the entry time.
<code>use.gr</code>	logical indicating whether to use gradients in the calculation
<code>optimiser</code>	select which optimiser is used
<code>log.time.transform</code>	should a log-transformation be used for calculating the derivative of the design matrix with respect to time? (default=TRUE)
<code>link.type</code>	type of link function. For "PH" (generalised proportional hazards), $g(S)=\log(-\log(S))$; for "PO" (generalised proportional odds), $g(S)=-\text{logit}(S)$; for "probit" (generalised probit), $g(S)=-\text{probit}(S)$; for "AH" (generalised additive hazards), $g(S)=-\log(S)$; for "AO" (generalised Aranda-Ordaz), $g(S)=\log((S^{(-\text{theta.AO})}-1)/\text{theta.AO})$.
<code>theta.AO</code>	theta parameter for the Aranda-Ordaz link type.
<code>reltol</code>	relative tolerance for the model convergence
<code>trace</code>	logical for whether to provide trace information
<code>frailty</code>	logical for whether to fit a shared frailty model
<code>cluster</code>	string for the data variable that determines the cluster for the frailty
<code>nodes</code>	number of integration points for Gaussian quadrature
<code>RandDist</code>	type of distribution for the random effect or frailty
<code>recurrent</code>	logical for whether clustered, left truncated data are recurrent or for first event (where the latter requires an adjustment for the frailties or random effects)
<code>logtheta</code>	initial value for log-theta used in the gamma shared frailty model
<code>adaptive</code>	logical for whether to use adaptive or non-adaptive quadrature
<code>maxkappa</code>	double float value for the maximum value of the weight used in the constraint
<code>Z</code>	formula for the design matrix for the random effects
<code>robust_initial</code>	logical for whether to use Nelder-Mead to find initial values (max 50 iterations). This is useful for ill-posed initial values.
<code>...</code>	additional arguments to be passed to the <code>mle2</code> .

Details

The implementation extends the `mle2` object from the `bbmle` package. The model inherits all of the methods from the `mle2` class.

The default linear predictor includes a time effect modelled using natural splines for `log(time)` with three degrees of freedom.

Value

An stpm2-class object that inherits from mle2-class.

Author(s)

Mark Clements, Xing-Rong Liu.

Examples

```

data(brcancer)
summary(fit <- stpm2(Surv(rectime,censrec==1)~hormon,data=brcancer,df=3))

## some predictions
head(predict(fit,se.fit=TRUE,type="surv"))
head(predict(fit,se.fit=TRUE,type="hazard"))

## some plots
plot(fit,newdata=data.frame(hormon=0),type="hazard")
plot(fit,newdata=data.frame(hormon=0),type="surv")

## the same model using logH.formula
summary(stpm2(Surv(rectime,censrec==1)~hormon,data=brcancer,logH.formula=~ns(log(rectime),df=3)))

## time-varying coefficient
summary(fit.tvc <- stpm2(Surv(rectime,censrec==1)~hormon,data=brcancer,df=3,
                        tvc=list(hormon=3)))
anova(fit,fit.tvc) # compare with and without tvc

## some more plots
plot(fit.tvc,newdata=data.frame(hormon=0),type="hr",var="hormon",ylim=c(0,2))
# no lines method: use add=TRUE
plot(fit.tvc,newdata=data.frame(hormon=1),type="hr",var="hormon",
     add=TRUE,ci=FALSE,line.col=2)

plot(fit.tvc,newdata=data.frame(hormon=0),type="sdiff",var="hormon")

plot(fit.tvc,newdata=data.frame(hormon=0),type="hdiff",var="hormon")

plot(fit.tvc,newdata=data.frame(hormon=0),type="hazard")
plot(fit.tvc,newdata=data.frame(hormon=1),type="hazard",line.col=2,ci=FALSE,add=TRUE)

## compare number of knots
hormon0 <- data.frame(hormon=0)
plot(fit,type="hazard",newdata=hormon0)
AIC(fit)
for (df in 4:6) {
  fit.new <- stpm2(Surv(rectime,censrec==1)~hormon,data=brcancer,df=df)
  plot(fit.new,type="hazard",newdata=hormon0,add=TRUE,ci=FALSE,line.col=df)
  print(AIC(fit.new))
}

```

```
stpm2-class          Class "stpm2" ~~~
```

Description

Regression object for stpm2.

Objects from the Class

Objects can be created by calls of the form `new("stpm2", ...)` and `stpm2(...)`.

Slots

```
xlevels: Object of class "list" ~~
contrasts: Object of class "listOrNULL" ~~
terms: Object of class "terms" ~~
logli: Object of class "function" ~~
lm: Object of class "lm" ~~
timeVar: Object of class "character" ~~
time0Var: Object of class "character" ~~
timeExpr: Object of class "nameOrcall" ~~
time0Expr: Object of class "nameOrcall" ~~
delayed: Object of class "logical" ~~
frailty: Object of class "logical" ~~
interval: Object of class "logical" ~~
model.frame: Object of class "list" ~~
call.formula: Object of class "formula" ~~
x: Object of class "matrix" ~~
xd: Object of class "matrix" ~~
termsd: Object of class "terms" ~~
Call: Object of class "character" ~~
y: Object of class "Surv" ~~
link: Object of class "list" ~~
call: Object of class "language" ~~
call.orig: Object of class "language" ~~
coef: Object of class "numeric" ~~
fullcoef: Object of class "numeric" ~~
vcov: Object of class "matrix" ~~
```

```

min: Object of class "numeric" ~~
details: Object of class "list" ~~
minuslog1: Object of class "function" ~~
method: Object of class "character" ~~
data: Object of class "list" ~~
formula: Object of class "character" ~~
optimizer: Object of class "character" ~~
args: Object of class "list" ~~

```

Extends

Class ["mle2"](#), directly.

Methods

```

plot signature(x = "stpm2", y = "missing"): ...
lines signature(x = "stpm2", ...): ...
predictnl signature(object = "stpm2", ...): ...
summary signature(object = "stpm2", ...): ...
eform signature(object = "stpm2", ...): ...

```

Examples

```
showClass("stpm2")
```

tvcCoxph-class	Class "tvcCoxph"
----------------	------------------

Description

Experimental approach to modelling time-dependent effects in Cox regression.

Objects from the Class

Objects can be created by calls of the form `new("tvcCoxph", ...)` or `cox.tvc(...)`. See the ["mle2"](#) documentation.

Slots

call: Object of class "language" ~~
call.orig: Object of class "language" ~~
coef: Object of class "numeric" ~~
fullcoef: Object of class "numeric" ~~
vcov: Object of class "matrix" ~~
min: Object of class "numeric" ~~
details: Object of class "list" ~~
minuslogl: Object of class "function" ~~
method: Object of class "character" ~~
data: Object of class "list" ~~
formula: Object of class "character" ~~
optimizer: Object of class "character" ~~

Extends

Class "[mle2](#)", directly.

Methods

plot signature(x = "tvcCoxph", y = "missing"): ...

Examples

```
showClass("tvcCoxph")
```

Index

- *Topic **\textasciitildekwd1**
 - coef<-, 7
 - grad, 10
 - incrVar, 10
 - *Topic **\textasciitildekwd2**
 - coef<-, 7
 - grad, 10
 - incrVar, 10
 - *Topic **classes**
 - aft-class, 5
 - pstpm2-class, 29
 - stpm2-class, 36
 - tvcCoxph-class, 37
 - *Topic **datasets**
 - brcancer, 6
 - colon, 7
 - legendre.quadrature.rule.200, 11
 - popmort, 19
 - *Topic **methods**
 - plot-methods, 18
 - predict-methods, 20
 - predictnl-methods, 25
 - residuals-methods, 31
 - *Topic **package**
 - Rstpm2-package, 2
 - *Topic **smooth**
 - aft, 3
 - nsx, 13
 - nsxD, 15
 - predict.nsx, 22
 - *Topic **survival**
 - aft, 3
 - *Topic **time-varying,Cox**
 - cox.tvc, 8
- aft, 3
- aft-class, 5
- AIC, pstpm2-method (pstpm2-class), 29
- AICc, pstpm2-method (pstpm2-class), 29
- anova, pstpm2-method (pstpm2-class), 29
- BIC, pstpm2-method (pstpm2-class), 29
- brcancer, 6
- bs, 14, 16
- coef<-, 7
- colon, 7
- confint.predictnl (predictnl), 23
- cox.tvc, 8
- cox.zph, 9
- coxph, 5, 9
- eform (eform.stpm2), 9
- eform, pstpm2-method (pstpm2-class), 29
- eform, stpm2-method (stpm2-class), 36
- eform.stpm2, 9
- grad, 10
- incrVar, 10
- legendre.quadrature.rule.200, 11
- lhs (rstpm2-internal), 32
- lhs<- (rstpm2-internal), 32
- lines, pstpm2-method (pstpm2-class), 29
- lines, stpm2-method (stpm2-class), 36
- lines.pstpm2 (lines.stpm2), 12
- lines.stpm2, 12
- mle2, 3–5, 27, 30, 34, 37, 38
- model.matrix.default, 4, 26, 34
- ns, 14, 16
- nsx, 13, 22
- nsxD, 15
- numDeltaMethod, 17
- plot, aft, missing-method (aft-class), 5
- plot, pstpm2, missing-method (pstpm2-class), 29
- plot, pstpm2-method (plot-methods), 18

plot, stpm2, missing-method
 (stpm2-class), 36
 plot, stpm2-method (plot-methods), 18
 plot, tvCoxph, missing-method
 (tvCoxph-class), 37
 plot-methods, 18
 popmort, 19
 predict, 22
 predict, aft-method (aft-class), 5
 predict, pstpm2-method
 (predict-methods), 20
 predict, stpm2-method (predict-methods),
 20
 predict-methods, 20
 predict.formula (predictnl), 23
 predict.nsx, 22
 predictnl, 17, 23
 predictnl, aft-method (aft-class), 5
 predictnl, mle2-method
 (predictnl-methods), 25
 predictnl, pstpm2-method (pstpm2-class),
 29
 predictnl, stpm2-method (stpm2-class), 36
 predictnl-methods, 25
 print.predictnl (predictnl), 23
 pstpm2, 25
 pstpm2-class, 29

 qAICc, pstpm2-method (pstpm2-class), 29

 residuals, pstpm2-method
 (residuals-methods), 31
 residuals, stpm2-method
 (residuals-methods), 31
 residuals-methods, 31
 rhs (rstpm2-internal), 32
 rhs<- (rstpm2-internal), 32
 Rstpm2 (Rstpm2-package), 2
 rstpm2-internal, 32
 Rstpm2-package, 2

 SafePrediction, 14, 16
 spline.des, 14, 16
 stpm2, 3, 19, 22, 32, 32
 stpm2-class, 36
 summary, pstpm2-method (pstpm2-class), 29
 summary, stpm2-method (stpm2-class), 36
 Surv, 4, 26, 33
 survreg, 5

 tvCoxph-class, 37