Introduction to the rstpm2 package

Mark Clements
Karolinska Institutet

Abstract

This vignette outlines the methods and provides some examples for generalised survival models as implemented in the R **rstpm2** package.

Keywords: survival, splines.

1. Background and theory

Generalised survival models provide a flexible and general approach to modelling survival or time-to-event data. The survival function $S(t|x)$ to time t for covariates x is defined in terms of a inverse link function G and a linear prediction $\eta(t,x)$, such that

$$S(t|x;\,\theta) = G(\eta(t,x;\,\theta))$$

where η is a function of both time t and covariates x, with regression parameters θ. We can calculate the hazard from this function, where

$$\begin{aligned} h(t|x;\,\theta) &= \frac{d}{dt}\left(-\log(S(t|x;\,\theta))\right) \\ &= \frac{-G'(\eta(t,x;\,\theta))}{G(\eta(t,x;\,\theta))}\frac{\partial\eta(t,x;\,\theta)}{\partial t} \end{aligned}$$

We model using a linear predictor $\eta(t,x;\,\theta) = X(t,x)\theta$ for a design matrix $X(t,x)$. The linear predictor can be constructed in a flexible manner, with the main constraint being that the time effects be smooth and twice differentiable. We calculate the derivative for the linear predictor using finite differences, such that

$$\frac{\partial\eta(t,x;\,\theta)}{\partial t} = \frac{\partial X(t,x)\theta}{\partial t} = \frac{X(t+\epsilon,x) - X(t-\epsilon,x)}{2\epsilon}\theta = X_D(t,x)\theta$$

for a derivative design matrix $X_D(t,x)$. This formulation allows for considerable flexibility in the construction of the linear predictor, with possible interactions between time and covariates.

The default smoother for time using natural splines for log(time), which is the flexible parametric survival model developed by Royston and Parmar (2003) and implemented by the Stata command `stpm2` [1]

The models are estimated using maximum likelihood estimation (MLE) for fully parametric models, penalised MLE for penalised smoothers, maximum marginal likelihood estimation

[1] As a technical aside, the Stata implementation uses natural splines using a truncated power basis with orthogonalisation, while the `ns()` function in `R` uses a matrix projection of B-splines. Note that we have implemented an extended `nsx()` function for natural splines that includes cure splines, centering, and a compatibility argument to use Stata `stpm2`'s unusual specification of quantiles.

Link description	Inverse link function $G(\eta(t,x;\theta))$	Interpretation	`link.type`
log−log	$\exp(-\exp(\eta(t,x;\theta)))$	Proportional hazards	`"PH"`
logit	$\text{expit}(-\eta(t,x;\theta))$	Proportional odds	`"PO"`
probit	$\Phi(-\eta(t,x;\theta))$	Probit	`"probit"`
log	$\exp(-\eta(t,x;\theta))$	Additive hazards	`"AH"`
Aranda-Ordaz	$\exp(-\log(\psi * \exp(\eta(t,x;\theta)) + 1)/\psi)$	Aranda-Ordaz	`"AO"`

Table 1: Implemented link functions

(MMLE) for parametric models with clustered data, or penalised MMLE for penalised models with clustered data. The likelihoods include left truncation, right censoring and interval censoring. For clustered data, we include Gamma frailties and normal random effects. Details on these models are available from https://doi.org/10.1177/0962280216664760 and https://doi.org/10.1002/sim.7451.

2. Syntax

The main functions for fitting the models are `stpm2` for parametric models, possibly with clustered data, and `pstpm2` for penalised models, possibly with clustered data. A subset of the syntax for `stpm2` is:

```
stpm2(formula, data, smooth.formula = NULL,
      df = 3, tvc = NULL,
      link.type=c("PH","PO","probit","AH","AO"), theta.AO=0,
      bhazard = NULL,
      robust = FALSE, cluster = NULL, frailty = !is.null(cluster) & !robust,
      RandDist=c("Gamma","LogN"),
      ...)
```

The `formula` has a `Surv` object on the left-hand-side and a linear predictor on the right-hand-side that does *not* include time (for `pstpm2`, it also does not include penalised functions). The time effects can be specified in several ways: the most general is using `smooth.formula`, where the right-hand-side of the formula specifies functions for time that are smooth with respect to time. This specification can include interactions between time and covariates. As an example, `smooth.formula=~nsx(log(time),df=3)+x:nsx(log(time),df=2)` specifies a baseline natural spline smoother of the log of the variable `time` used in the `Surv` object with three degrees of freedom, with an interaction between a covariate `x` and a natural spline smoother of log(`time`) with two degrees of freedom. Other specifications of time effects have equivalent formulations: for example, `df=3` is equivalent to `smooth.formula=~nsx(log(time),df=3)` for the variable `time`. Similarly, `tvc=list(x=2)` is equivalent to `smooth.formula=~x:nsx(log(time),df=2)`. Moreover, for a log-linear interaction between a covariate and time, use `smooth.formula=~x:log(time)`

A current limitation of the implementation is that the dataset `data` needs to be specified.

Type of link is specified with the `link.type` argument; this defaults to a log−log link for proportional hazards (see Table 1). For the Aranda-Ordaz link, the fixed value of the scale term ψ is specified using the `theta.AO` argument. For relative survival, a vector for the baseline hazard can be specified using the `bhazard` argument. A vector for the clusters can be specified with the `cluster` argument. The calculation of robust standard errors can be specified with the `robust=TRUE` argument; if `robust` is false, then the model assumes a frailty or random effects model, with either a default Gamma frailty (`RandDist="Gamma"`) or a normal random effect (`RandDist="LogN"`, using notation from the `frailtypack` package).

The syntax for the fitting the penalised models with `pstpm2` is very similar. A subset of the arguments are:

```
pstpm2(formula, data, smooth.formula = NULL,
       tvc = NULL,
       bhazard = NULL,
       sp=NULL,
       criterion=c("GCV","BIC"),
       link.type=c("PH","PO","probit","AH","AO"), theta.AO=0,
       robust=FALSE,
       frailty=!is.null(cluster) & !robust, cluster = NULL, RandDist=c("Gamma","LogN"),
       ...)
```

The penalised smoothers are specified using the `s()` function from the `mgcv` package within the `smooth.formula` argument; by default, not specifying `smooth.formula` will lead to `smooth.formula=~s(log(t` Interactions with time (both penalised and unpenalised) and penalised covariate effects should be specified using `smooth.formula`.

Note that the `df` argument is not included. By default, the smoothing parameter(s) are using the `criterion` argument; the smoothing parameters can also be fixed using the `sp` argument.

The specifications for relative survival, link type, and clustered data follow that for the `stpm2` function.

2.1. Post-estimation

One of the strengths of these models is varied post-estimation. Most of the estimators are described in Tables 2 and 3. These estimators are typically calculated from the `predict` function or from `plot` function calls. All of these calls require that the `newdata` argument is specified (in contrast to prediction in the `survival` package, which defaults to the average of each covariate).

For contrasts (e.g. survival differences, hazard ratios), the `newdata` argument is the "unexposed" group, while the exposed group is defined by either: (i) a unit change in a variable in `newdata` as defined by the `var` argument (e.g. `var="x"` for variable `x`); or (ii) an `exposed` function that takes a data-frame and returns the "exposed" group (e.g. `exposed=function(data) transform(data, x=1)`). The latter mechanism is quite general and allows for standardised survival, standardised hazards, and attributable fractions under possibly counterfactual exposures.

Standard errors for the post-estimators are calculated on a possibly transformed scale using the delta method. For the delta method, the partial derivatives of the post-estimators are calculated either directly or using finite differences.

3. Examples: Independent survival analysis

We begin with some simple proportional hazard models using the brcancer dataset. We can fit the models using very similar syntax to coxph, except that we need to specify the degrees of freedom for the baseline smoother. Typical values for df are 3-6. For this model the model parameters include an intercept term, time-invariant log-hazard ratios, and parameters for the baseline smoother. The default for the baseline smoother is to use the nsx function, which is a limited extension to the splines::ns function, with log of the time effect.

```
> fit <- stpm2(Surv(rectime,censrec==1)~hormon,
+              data=brcancer, df=4)
> summary(fit)
```

Description	Formulation[a]	`type`
Conditional link	$\eta(t,x;\hat\theta)$	`"link"`
Conditional survival	$S(t\|x;\hat\theta)=G(\eta(t,x;\hat\theta))$	`"surv"`
Conditional odds	$\text{Odds}(t\|x;\hat\theta)=S(t\|x;\hat\theta)/(1-S(t\|x;\hat\theta))$	`"odds"`
Conditional failure	$1-S(t\|x;\hat\theta)$	`"fail"`
Conditional cumulative hazard	$H(t\|x;\hat\theta)=-\log G(\eta(t,x;\hat\theta))$	`"cumhaz"`
Conditional density	$f(t\|x;\hat\theta)=G'(\eta(t,x;\hat\theta))\frac{\partial\eta(t,x;\hat\theta)}{\partial t}$	`"density"`
Conditional hazard	$h(t\|x;\hat\theta)=\frac{G'(\eta(t,x;\hat\theta))}{G(\eta(t,x;\hat\theta))}\frac{\partial\eta(t,x;\hat\theta)}{\partial t}$	`"hazard"`
Conditional log hazard	$\log h(t\|x;\hat\theta)$	`"loghazard"`
Conditional survival differences	$S(t\|x^*;\hat\theta)-S(t\|x;\hat\theta)$	`"survdiff"`
Conditional hazard differences	$h(t\|x^*;\hat\theta)-h(t\|x;\hat\theta)$	`"hazdiff"`
Conditional hazard ratios	$h(t\|x^*;\hat\theta)/h(t\|x;\hat\theta)$	`"hr"`
Conditional odds ratios	$\text{Odds}(t\|x^*;\hat\theta)/\text{Odds}(t\|x;\hat\theta)$	`"or"`
Restricted mean survival time	$\int_0^t S(u\|x;\hat\theta)du$	`"rmst"`
Standardised survival	$E_{X^*}S(t\|X^*;\hat\theta)$	`"meansurv"`
Standardised survival differences	$E_{X_1^*}S(t\|X_1^*;\hat\theta)-E_{X_0^*}S(t\|X_0^*;\hat\theta)$	`"meansurvdiff"`
Standardised hazard	$h_{X^*}(t\|X^*;\hat\theta)=\frac{E_{X^*}(S(t\|X^*;\hat\theta)h(t\|X^*;\hat\theta))}{E_{X^*}(S(t\|X^*;\hat\theta))}$	`"meanhaz"`
Standardised hazard ratio	$h_{X_1^*}(t\|X_1^*;\hat\theta)/h_{X_0^*}(t\|X_0^*\hat\theta)$	`"meanhazdiff"`
Attributable fraction	$\frac{E_{X^*}S(t\|X^*;\hat\theta)--E_XS(t\|X;\hat\theta)}{1-E_XS(t\|X;\hat\theta)}$	`"af"`

[a]Notation: x^* is a covariate pattern for the "exposed" group; X^* is a set of possibly counterfactual covariates; $E_X(g(X))$ is the expectation or average of $g(X)$ across the set X; X_0^* and X_1^* are sets of possibly counterfactual covariates for the "unexposed" and "exposed" sets, respectively.

Table 2: Types of conditional post-estimators

Description	Formulation[a]	`type`
Marginal survival	$S_M(t\|x;\hat\theta)=E_ZG(\eta(t,x,Z;\hat\theta))$	`"margsurv"`
Marginal hazard	$h_M(t\|x;\hat\theta)=E_Z(h(t,x,Z;\hat\theta))$	`"marghaz"`
Marginal survival differences	$S_M(t\|x^*;\hat\theta)-S_M(t\|x;\hat\theta)$	`"margsurvdiff"`
Marginal hazard ratios	$h_M(t\|x^*;\hat\theta)/h_M(t\|x;\hat\theta)$	`"marghr"`
Standardised marginal survival	$E_ZE_{X^*}S(t\|X^*,Z;\hat\theta)$	`"meanmargsurv"`
Standardised marginal survival differences	$E_ZE_{X^*}S(t\|X^*,Z;\hat\theta)-E_ZE_XS(t\|X,Z;\hat\theta)$	`"meansurvdiff"`
Attributable fraction	$\frac{E_ZE_{X^*}S(t\|X^*,Z;\hat\theta)-E_ZE_XS(t\|X,Z;\hat\theta)}{1-E_ZE_XS(t\|X,Z;\hat\theta)}$	`"af"`

[a]Notation: Z is a random effect or frailty; x^* is a covariate pattern for the "exposed" group; X^* is a set of possibly counterfactual covariates; $E_X(g(X))$ is the expectation or average of $g(X)$ across the set X; X_0^* and X_1^* are sets of possibly counterfactual covariates for the "unexposed" and "exposed" sets, respectively.

Table 3: Types of post-estimators for clustered data

```
Maximum likelihood estimation

Call:
mle2(minuslogl = negll, start = coef, eval.only = TRUE, vecpar = TRUE,
    gr = function (beta)
    {
        localargs <- args
        localargs$init <- beta
```

Functionality	Uncorrelated param.	Uncorrelated penal.	Param. gamma frailty	Penal. gamma frailty	Param. normal random effects	Penal. normal random effects
Multiple links	☒	☒	☒	☒	☒	☒
Right censoring	☒	☒	☒	☒	☒	☒
Left truncation	☒	☒	☒[a]	☒[a]	☒[a]	☒[a]
Interval censoring	☒	☒	☐	☐	☒	☒
Time-varying effects	☒	☒	☒	☒	☒	☒
Excess hazards	☒	☒	☒	☒	☒	☒
Conditional estimators[b]	☒	☒	☒	☒	☒	☒
Conditional standardisation[c]	☒	☒	☒	☒	☒	☒
Frailty/random effects variance			☒	☒	☒	☒
Marginal estimators[d]			☒	☒	☒	☒
Marginal standardisation[e]			☒	☒	☐	☐
Random intercept			☒	☒	☒	☒
Random slope					☒	☒
Multiple random effects					☐	☐

[a] Gradients not currently implemented.

[b] Estimators including survival, survival differences, hazards, hazard ratios, hazard differences, density, odds and odds ratios.

[c] Standardised estimators include mean survival, mean survival differences, mean hazards and attributable fractions.

[d] Marginal estimators include survival, survival differences, hazards, hazard ratios and hazard differences.

[e] Marginal standardised estimators include mean survival, mean survival differences and attributable fractions.

Table 4: Functionality for the different generalised survival models

```
        localargs$return_type <- "gradient"
        return(.Call("model_output", localargs, PACKAGE = "rstpm2"))
    }, control = list(parscale = c(`(Intercept)` = 1, hormon = 1,
    `nsx(log(rectime), df = 4)1` = 1, `nsx(log(rectime), df = 4)2` = 1,
    `nsx(log(rectime), df = 4)3` = 1, `nsx(log(rectime), df = 4)4` = 1
    ), maxit = 300), lower = -Inf, upper = Inf)

Coefficients:
                          Estimate Std. Error z value     Pr(z)
(Intercept)               -6.79773    0.72642 -9.3578 < 2.2e-16 ***
hormon                    -0.36406    0.12491 -2.9144  0.003563 **
nsx(log(rectime), df = 4)1  5.69995    0.71677  7.9523 1.830e-15 ***
nsx(log(rectime), df = 4)2  4.85614    0.48002 10.1166 < 2.2e-16 ***
nsx(log(rectime), df = 4)3 10.13328    1.41268  7.1731 7.331e-13 ***
nsx(log(rectime), df = 4)4  4.70626    0.33016 14.2545 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

-2 log L: 5212.943
```

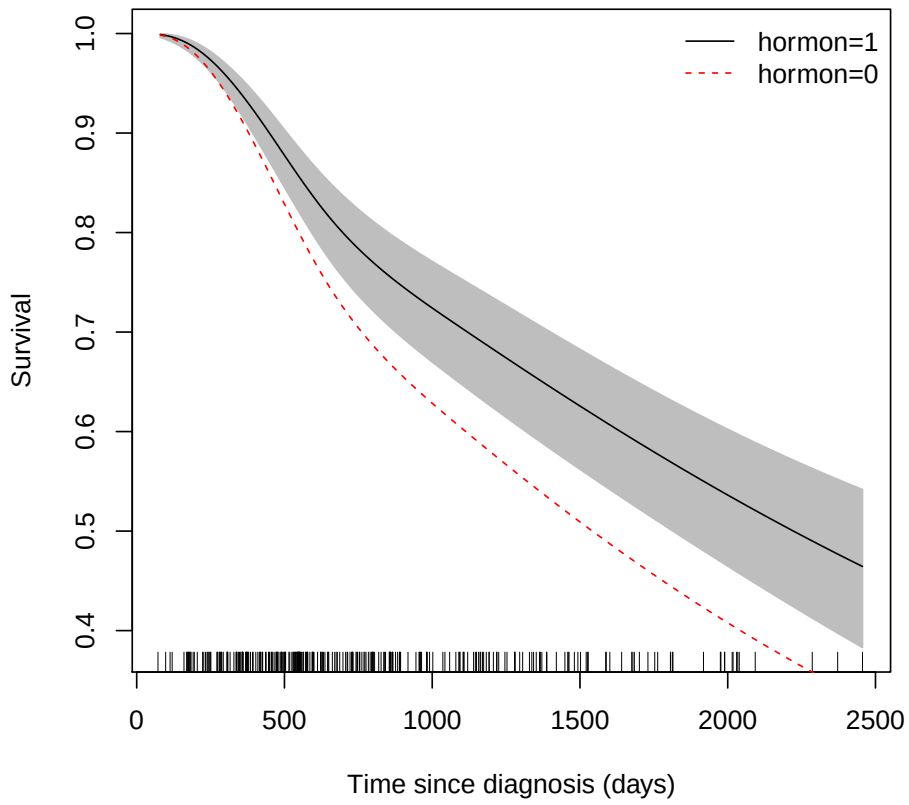
```
> ## utility to exponentiate the ith components
> expi <- function(x,i=1:length(x)) { x[i] <- exp(x[i]); x }
> fit.cox <- coxph(Surv(rectime,censrec==1)~hormon, data=brcancer)
> rbind(coxph=coef(summary(fit.cox)),
+       stpm2=expi(coef(summary(fit))["hormon",c(1,1,2:4)],2))
```

```
             coef exp(coef)  se(coef)         z    Pr(>|z|)
hormon -0.3640099 0.6948843 0.1250446 -2.911041 0.003602266
stpm2  -0.3640574 0.6948513 0.1249147 -2.914449 0.003563175
```

```
>
```

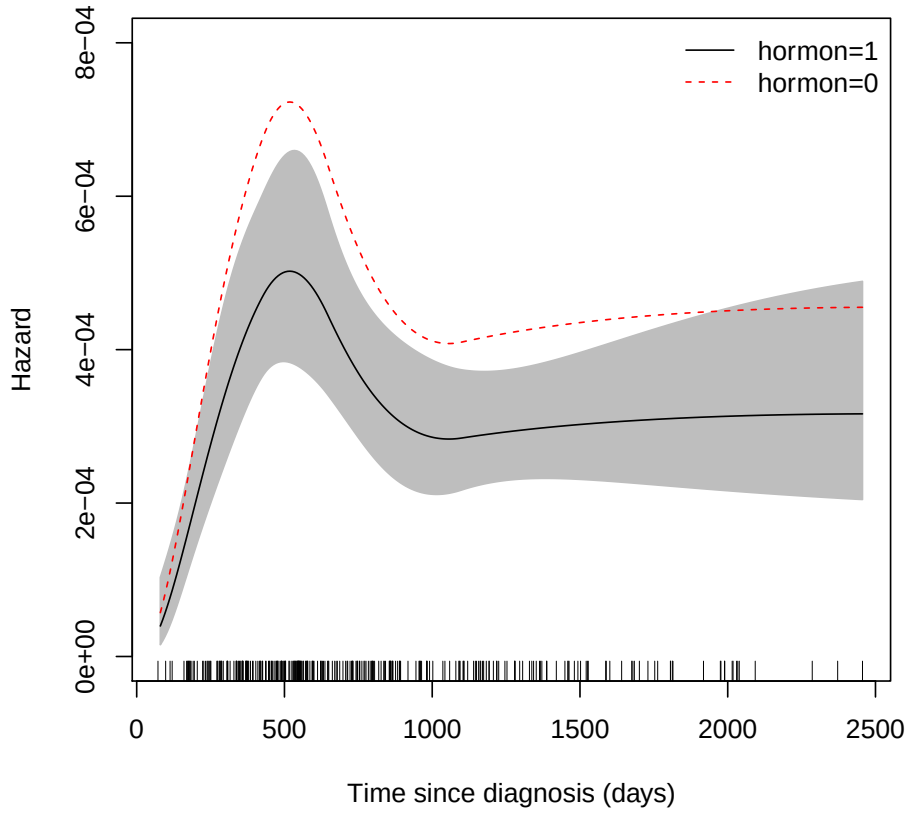
We see that the hazard ratios are very similar to the coxph model. The model fit can also be used to estimate a variety of parameters. For example, we can easily estimate survival for a given parameter.

```
> # tms=seq(0,10,length=301)[-1]
> plot(fit,newdata=data.frame(hormon=1),
+      xlab="Time since diagnosis (days)")
> lines(fit, newdata=data.frame(hormon=0), col=2, lty=2)
> legend("topright", c("hormon=1","hormon=0"),lty=1:2,col=1:2,bty="n")
>
```



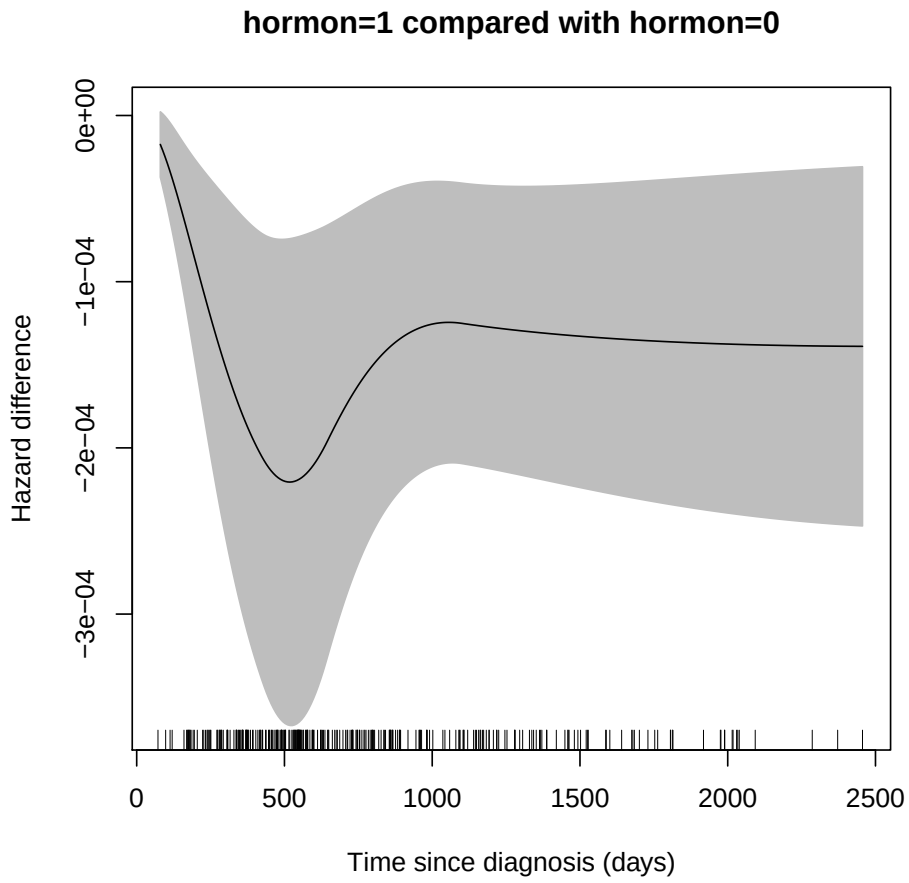
We can also calculate the hazards.

```
> # tms=seq(0,10,length=301)[-1]
> plot(fit,newdata=data.frame(hormon=1), type="hazard",
+      xlab="Time since diagnosis (days)", ylim=c(0,8e-4))
> lines(fit, newdata=data.frame(hormon=0), col=2, lty=2, type="hazard")
> legend("topright", c("hormon=1","hormon=0"),lty=1:2,col=1:2,bty="n")
>
```

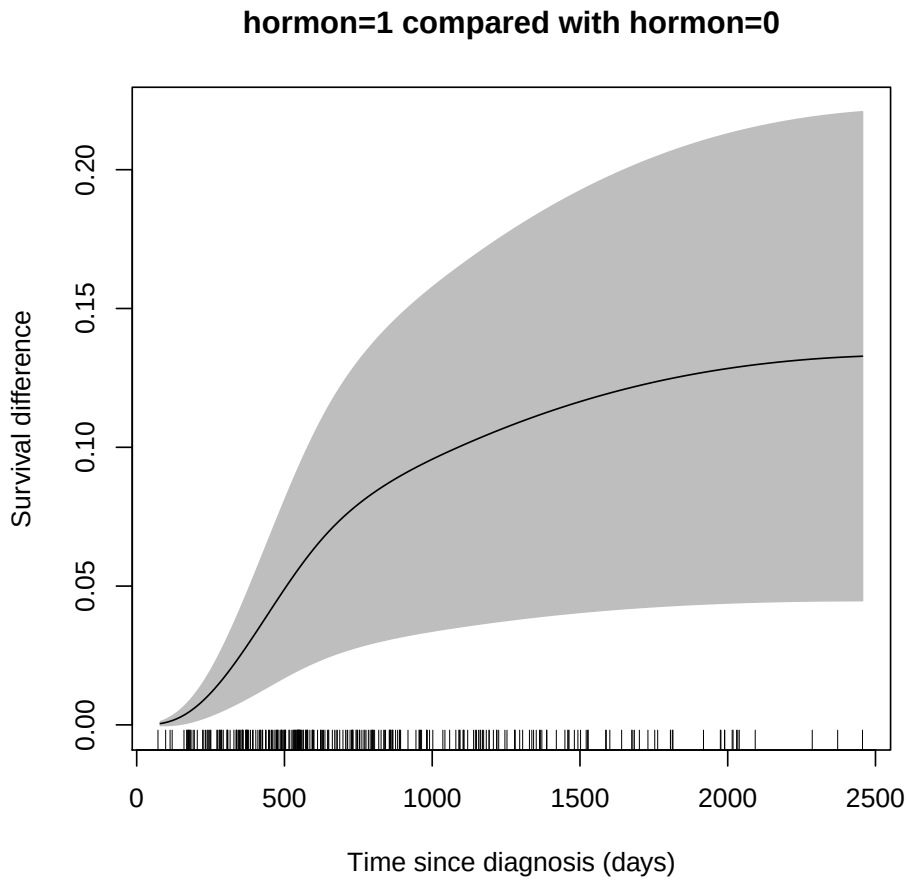


Usefully, we can also estimate survival differences and hazard differences. We define the survival differences using a reference covariate pattern using the newdata argument, and then define an exposed function which takes the newdata and transforms for the 'exposed' covariate pattern. As an example

```
> plot(fit,newdata=data.frame(hormon=0), type="hdiff",
+      exposed=function(data) transform(data, hormon=1),
+      main="hormon=1 compared with hormon=0",
+      xlab="Time since diagnosis (days)")
>
```




```
> plot(fit,newdata=data.frame(hormon=0), type="sdiff",
+      exposed=function(data) transform(data, hormon=1),
+      main="hormon=1 compared with hormon=0",
+      xlab="Time since diagnosis (days)")
>
```



4. Mean survival

This has a useful interpretation for causal inference.
$E_Z(S(t|Z, X = 1)) - E_Z(S(t|Z, X = 0))$

```
fit <- stpm2(...)
predict(fit,type="meansurv",newdata=data)
```

5. Cure models

For cure, we use the melanoma dataset used by Andersson and colleagues for cure models with Stata's stpm2 (see http://www.pauldickman.com/survival/).

Initially, we merge the patient data with the all cause mortality rates.

```
> popmort2 <- transform(rstpm2::popmort,exitage=age,exityear=year,age=NULL,year=NULL)
> colon2 <- within(rstpm2::colon, {
+   status <- ifelse(surv_mm>120.5,1,status)
+   tm <- pmin(surv_mm,120.5)/12
+   exit <- dx+tm*365.25
+   sex <- as.numeric(sex)
+   exitage <- pmin(floor(age+tm),99)
```

```
+   exityear <- floor(yydx+tm)
+   ##year8594 <- (year8594=="Diagnosed 85-94")
+ })
> colon2 <- merge(colon2,popmort2)
>
```

For comparisons, we fit the relative survival model without and with cure.

```
> fit0 <- stpm2(Surv(tm,status %in% 2:3)~I(year8594=="Diagnosed 85-94"),
+               data=colon2,
+               bhazard=colon2$rate, df=5)
>
```

```
> summary(fit <- stpm2(Surv(tm,status %in% 2:3)~I(year8594=="Diagnosed 85-94"),
+                      data=colon2,
+                      bhazard=colon2$rate,
+                      df=5,cure=TRUE))
```

```
Maximum likelihood estimation

Call:
mle2(minuslogl = negll, start = coef, eval.only = TRUE, vecpar = TRUE,
    gr = function (beta)
    {
        localargs <- args
        localargs$init <- beta
        localargs$return_type <- "gradient"
        return(.Call("model_output", localargs, PACKAGE = "rstpm2"))
    }, control = list(parscale = c(`(Intercept)` = 1, `I(year8594 == "Diagnosed 85-94")TRUE
    `nsx(log(tm), df = 5, cure = TRUE)1` = 1, `nsx(log(tm), df = 5, cure = TRUE)2` = 1,
    `nsx(log(tm), df = 5, cure = TRUE)3` = 1, `nsx(log(tm), df = 5, cure = TRUE)4` = 1,
    `nsx(log(tm), df = 5, cure = TRUE)5` = 1), maxit = 300),
    lower = -Inf, upper = Inf)

Coefficients:
                                    Estimate Std. Error  z value     Pr(z)
(Intercept)                        -3.977319   0.054777 -72.6096 < 2.2e-16
I(year8594 == "Diagnosed 85-94")TRUE -0.155610   0.025088  -6.2026 5.554e-10
nsx(log(tm), df = 5, cure = TRUE)1    3.323187   0.053164  62.5077 < 2.2e-16
nsx(log(tm), df = 5, cure = TRUE)2    3.628625   0.053158  68.2606 < 2.2e-16
nsx(log(tm), df = 5, cure = TRUE)3    1.634848   0.022464  72.7749 < 2.2e-16
nsx(log(tm), df = 5, cure = TRUE)4    6.592010   0.111501  59.1204 < 2.2e-16
nsx(log(tm), df = 5, cure = TRUE)5    3.371802   0.042787  78.8038 < 2.2e-16

(Intercept)                        ***
I(year8594 == "Diagnosed 85-94")TRUE ***
nsx(log(tm), df = 5, cure = TRUE)1   ***
nsx(log(tm), df = 5, cure = TRUE)2   ***
nsx(log(tm), df = 5, cure = TRUE)3   ***
nsx(log(tm), df = 5, cure = TRUE)4   ***
nsx(log(tm), df = 5, cure = TRUE)5   ***
```

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

-2 log L: 42190.77

> predict(fit,head(colon2),se.fit=TRUE)

   Estimate     lower     upper
1 0.8610828 0.8544630 0.8677538
2 0.7934654 0.7851997 0.8018182
3 0.6967405 0.6865297 0.7071032
4 0.8610828 0.8544630 0.8677538
5 0.8221244 0.8145060 0.8298140
6 0.8610828 0.8544630 0.8677538

>
```

The estimate for the year parameter from the model without cure is within three significant figures with that in Stata. For the predictions, the Stata model gives:

```
     +---------------------------------+
     |      surv   surv_lci   surv_uci |
     |---------------------------------|
  1. | .86108264   .8542898   .8675839 |
  2. | .79346526   .7850106   .8016309 |
  3. | .69674037   .6863196   .7068927 |
  4. | .86108264   .8542898   .8675839 |
  5. | .82212425   .8143227   .8296332 |
     |---------------------------------|
  6. | .86108264   .8542898   .8675839 |
     +---------------------------------+
```

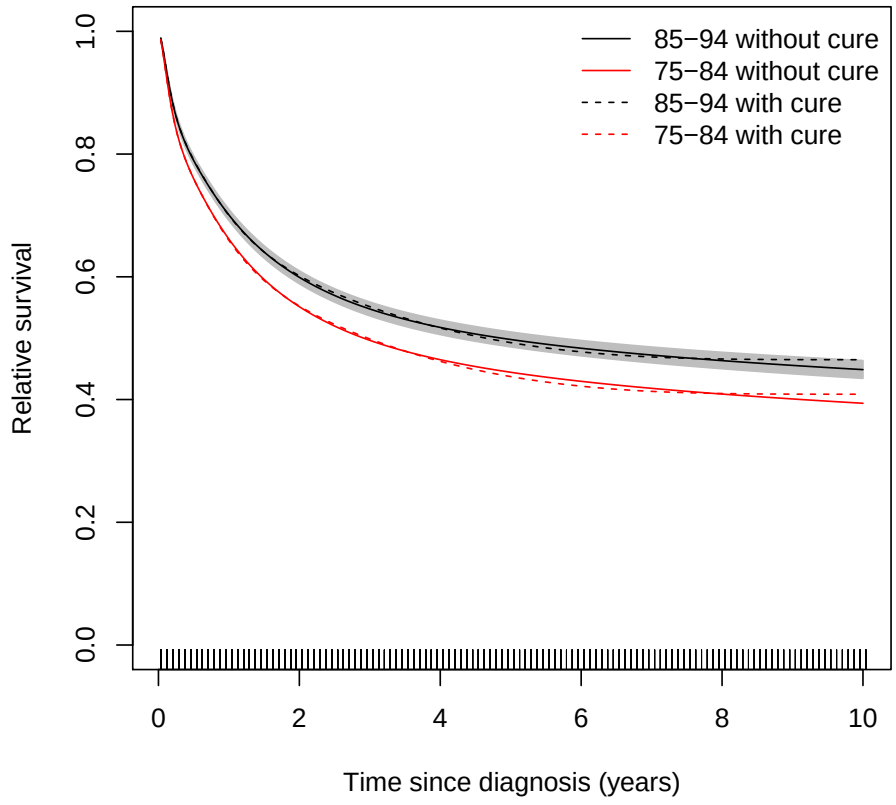
We can estimate the proportion of failures prior to the last event time:

```
> newdata.eof <- data.frame(year8594 = unique(colon2$year8594),
+                           tm=10)
> 1-predict(fit0, newdata.eof, type="surv", se.fit=TRUE)

   Estimate     lower     upper
1 0.6060914 0.6205876 0.5910413
2 0.5512434 0.5655493 0.5364663

> 1-predict(fit, newdata.eof, type="surv", se.fit=TRUE)

   Estimate     lower     upper
1 0.5913311 0.6052340 0.5769385
2 0.5350826 0.5482903 0.5214888

> predict(fit, newdata.eof, type="haz", se.fit=TRUE)

      Estimate        lower        upper
1 1.254142.... 1.081724e-06 1.426561e-06
2 1.073411.... 9.235373e-07 1.223286e-06
```

```
>
```

We can plot the predicted survival estimates:

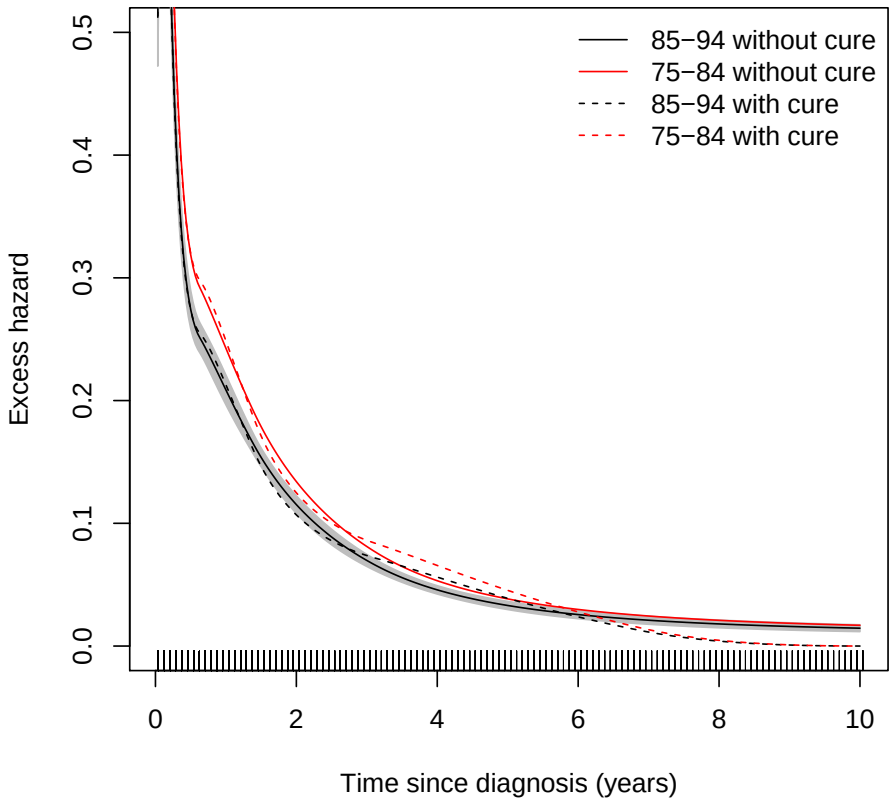
```
> tms=seq(0,10,length=301)[-1]
> plot(fit0,newdata=data.frame(year8594 = "Diagnosed 85-94", tm=tms), ylim=0:1,
+      xlab="Time since diagnosis (years)", ylab="Relative survival")
> plot(fit0,newdata=data.frame(year8594 = "Diagnosed 75-84",tm=tms),
+      add=TRUE,line.col="red",rug=FALSE)
> ## warnings: Predicted hazards less than zero for cure
> plot(fit,newdata=data.frame(year8594 = "Diagnosed 85-94",tm=tms),
+      add=TRUE,ci=FALSE,lty=2,rug=FALSE)
> plot(fit,newdata=data.frame(year8594="Diagnosed 75-84",tm=tms),
+      add=TRUE,rug=FALSE,line.col="red",ci=FALSE,lty=2)
> legend("topright",c("85-94 without cure","75-84 without cure",
+                     "85-94 with cure","75-84 with cure"),
+        col=c(1,2,1,2), lty=c(1,1,2,2), bty="n")
>
```



And the hazard curves:

```
> plot(fit0,newdata=data.frame(year8594 = "Diagnosed 85-94", tm=tms),
+      ylim=c(0,0.5), type="hazard",
+      xlab="Time since diagnosis (years)",ylab="Excess hazard")
```

```
> plot(fit0,newdata=data.frame(year8594 = "Diagnosed 75-84", tm=tms),
+      type="hazard",
+      add=TRUE,line.col="red",rug=FALSE)
> plot(fit,newdata=data.frame(year8594 = "Diagnosed 85-94", tm=tms),
+      type="hazard",
+      add=TRUE,ci=FALSE,lty=2,rug=FALSE)
> plot(fit,newdata=data.frame(year8594="Diagnosed 75-84", tm=tms),
+      type="hazard",
+      add=TRUE,rug=FALSE,line.col="red",ci=FALSE,lty=2)
> legend("topright",c("85-94 without cure","75-84 without cure",
+                     "85-94 with cure","75-84 with cure"),
+        col=c(1,2,1,2), lty=c(1,1,2,2), bty="n")
>
```



6. Potential limitations and next steps

- The calculation of the matrix $X_D(t,x)$ using finite differences can be inaccurate for small event times. The functions `stpm2` and `pstpm2` raise errors when the event times are less than 1e-4. One working solution is to scale the event times (e.g. multiply by 1000) for analysis. TODO: investigate whether we can calculate $X_D(t,x)$ more accurately using the `numDeriv` package.

- TODO: Extend the generalised survival models to use multiple random effects.
- TODO: Extend the generalised survival models to use automatic differentiation.

Affiliation:

Mark Clements
Department of Medical Epidemiology and Biostatistics
Karolinska Institutet
Email: mark.clements@ki.se