Spline terms in a Cox model

Terry Therneau

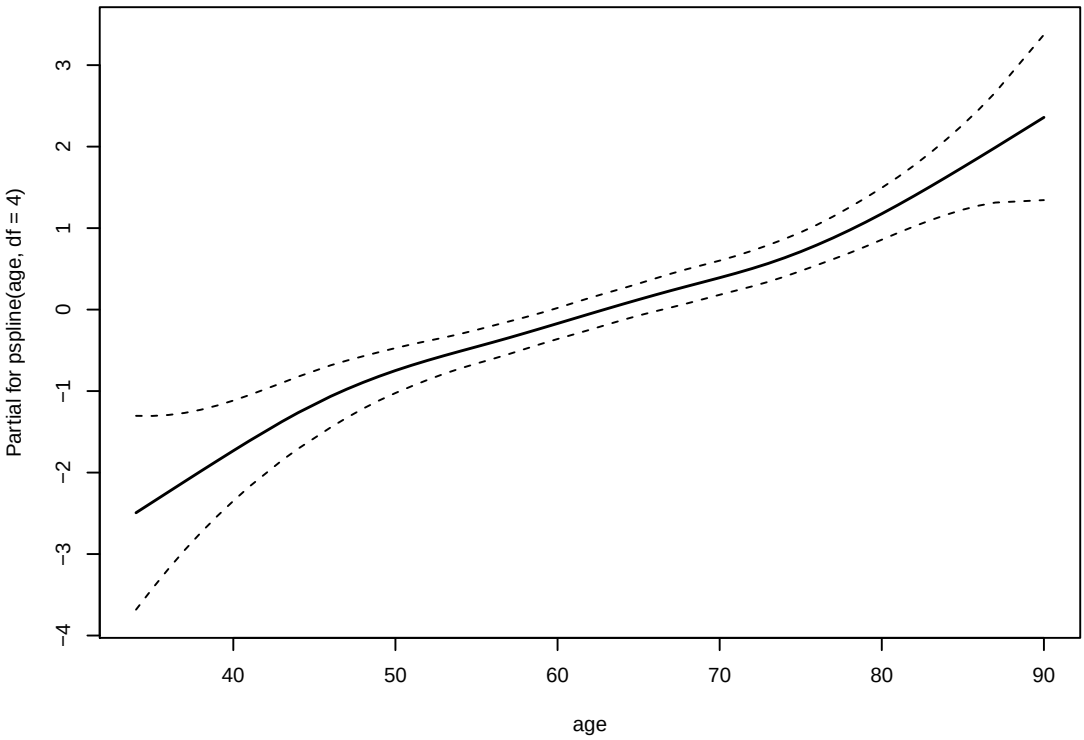
May 10, 2016

This is a trio of topics that comes up just often enough in my work that I end up re-discovering how to do it correctly about once a year. A note showing how may be useful to others, it is certainly a useful reference for me.

1 Plotting smooth terms

Here is a simple example using the MGUS data. I prefer a simpler color palette than the default found in termplot.

```
> require(survival)
> mfit <- coxph(Surv(futime, death) ~ sex + pspline(age, df=4), data=mgus)
> mfit
Call:
coxph(formula = Surv(futime, death) ~ sex + pspline(age, df = 4),
    data = mgus)

                             coef se(coef)      se2
sexmale                    0.22784  0.13883  0.13820
pspline(age, df = 4), lin  0.06682  0.00703  0.00703
pspline(age, df = 4), non
                             Chisq   DF      p
sexmale                    2.69335 1.00   0.10
pspline(age, df = 4), lin 90.22974 1.00 <2e-16
pspline(age, df = 4), non  3.44005 3.05   0.34

Iterations: 5 outer, 16 Newton-Raphson
     Theta= 0.851
Degrees of freedom for terms= 1.0 4.1
Likelihood ratio test=108  on 5.04 df, p=0  n= 241
> termplot(mfit, term=2, se=TRUE, col.term=1, col.se=1)
```



Note that the `term=2` option is passed directly from the `termplot` routine to a `predict(fit, type='terms')` call. For coxph models, the `predict` function allows terms to be specified either by position or name. Other routines, e.g. `gam`, respond only to a name. (This can be a bit of a pain since it must exactly match the *printed* call in both spelling and spacing; and the printed spacing in may not match what the user typed.)

Three questions of the plot are whether the curve is significantly non-linear, how the curve is centered and whether we can easily plot it on the hazard as opposed to the log hazard scale. The first question is answered by the printout, the solution to the others is to use the plot=FALSE option of termplot, which returns the data points that would be plotted back to the user.

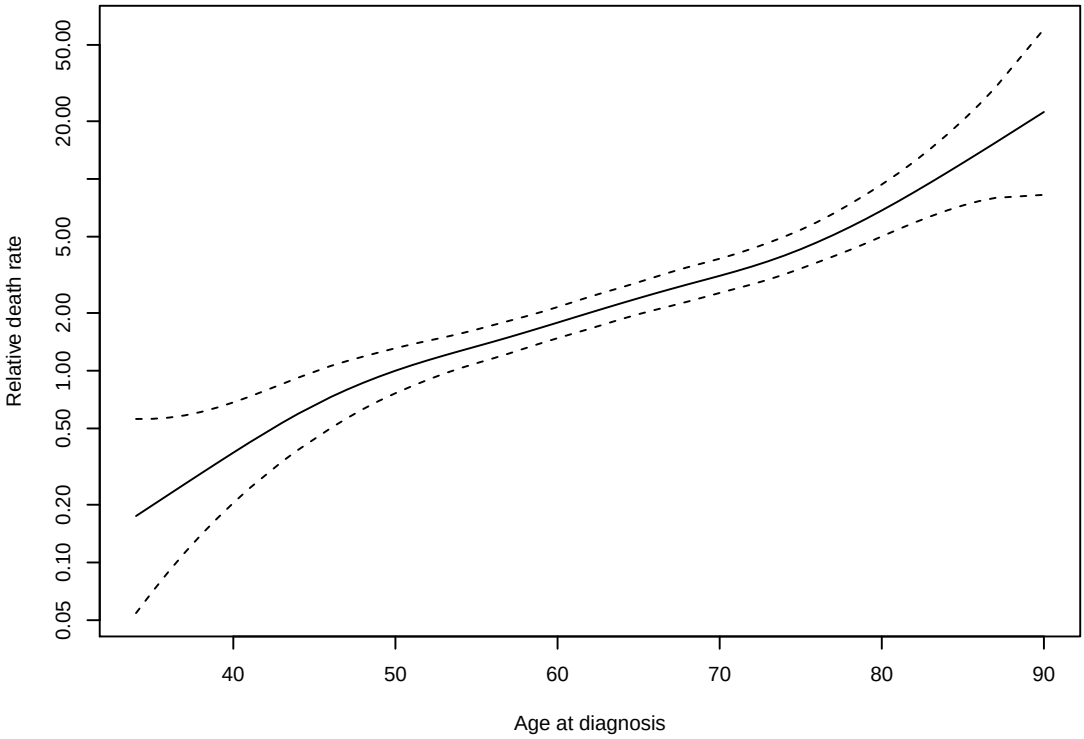
```
> ptemp <- termplot(mfit, se=TRUE, plot=FALSE)
> attributes(ptemp)
$constant
[1] 3.334793

$names
[1] "sex" "age"
> ptemp$age[1:4,]
   x         y        se
1 34 -2.493101 0.5946620
2 35 -2.365152 0.5301362
3 36 -2.237191 0.4720479
```

```
4 37 -2.109559 0.4208977
```

The termplot function depends on a call to predict with type='terms', which returns a centered set of predictions. Like a simple linear model fit, the intercept is a separate term, which is found in the "constant" attribute above, and each column of the result is centered so that the average predicted value is zero. Since any given x value may appear multiple times in the data and thus in the result of predict, and the termplot function removes duplicates, the plot will normally not be precisely centered at zero.

Now suppose we want to redraw this on log scale with age 50 as the reference, i.e., the risk is 1 for a 50 year old. Since the Cox model is a relative hazards model we can choose whatever center we like. (If there were no one of exactly age 50 in the data set the first line below would need to do an interpolation, e.g. by using the approx function.)

```
> ageterm <- ptemp$age  # this will be a data frame
> center <- with(ageterm, y[x==50])
> ytemp <- ageterm$y + outer(ageterm$se, c(0, -1.96, 1.96), '*')
> matplot(ageterm$x, exp(ytemp - center), log='y',
          type='l', lty=c(1,2,2), col=1,
          xlab="Age at diagnosis", ylab="Relative death rate")
```



Voila! We now have a plot that is more interpretable. The approach is appropriate for any term, not just psplines. The above plot uses log scale for the y axis which is appropriate for the question of whether a non-linear age effect was even necessary for this data set (it is not), one could remove the log argument to emphasize the Gomperzian effect of age on mortality.

2 Monotone splines

Consider the following model using the mgus2 data set.

```
> fit <- coxph(Surv(futime, death) ~ age + pspline(hgb, 4), mgus2)
> fit
Call:
coxph(formula = Surv(futime, death) ~ age + pspline(hgb, 4),
    data = mgus2)

                             coef  se(coef)       se2
age                       0.05394   0.00337   0.00336
pspline(hgb, 4), linear  -0.11579   0.01612   0.01612
pspline(hgb, 4), nonlin
                            Chisq   DF       p
age                     256.31805 1.00 < 2e-16
pspline(hgb, 4), linear  51.56708 1.00 6.9e-13
pspline(hgb, 4), nonlin  18.77963 3.06 0.00033

Iterations: 6 outer, 16 Newton-Raphson
     Theta= 0.942
Degrees of freedom for terms= 1.0 4.1
Likelihood ratio test=420  on 5.06 df, p=0
  n=1371 (13 observations deleted due to missingness)
> termplot(fit, se=TRUE, term=2, col.term=1, col.se=1,
           xlab="Hemoglobin level")
```

Low hemoglobin or anemia is a recognized marker of frailty in older age, so the rise in risk for low levels is not surprising. The rise on the right hand portion of the curve is less believeable — the normal range of HGB is 12-15.5 for women and 13.5 to 17.5 for men, why would we expect a rise there? A monotone fit that forces the curve to be horizontal from 14 onward fits well within the confidence bands, so we might want to force monotonicity.

There are two tools for this within the pspline function. The first is to decrease the overall degrees of freedom and the second is to use `combine` option to force equality of selected coefficients. Start by decreasing the degrees of freedom. The pspline function automatically picks the number of basis (nterms) to be "sufficiently large" for the given degrees of freedom. We fix it at a single value for the rest of this example to better isolate the effects of degrees of freedom and of constraints.

```
> termplot(fit, se=TRUE, col.term=1, col.se=1, term=2,
           xlab="Hemoglobin level", ylim=c(-.4, 1.3))
> df <- c(3, 2.5, 2)
> for (i in 1:3) {
      tfit <- coxph(Surv(futime, death) ~ age +
                    pspline(hgb, df[i], nterm=8), mgus2)
      temp <- termplot(tfit, se=FALSE, plot=FALSE, term=2)
      lines(temp$hgb$x, temp$hgb$y, col=i+1, lwd=2)
```

```
  }
> legend(14, 1, paste("df=", c(4, df)), lty=1, col=1:4, lwd=2)
```

This has reduced, but not eliminated, the right hand rise at the expense of a less sharp transition at the value of 14. The `combine` option makes use of a property of the P-spline basis, which is that the curve will be monotone if and only if the coefficients are monotone. We can then use a pool adjacent violators algorithm to sequentially force equality for those coefficients which go the wrong way. Look at the coefficients for the fit with 2.5 degrees of freedom.

```
> fit2a <- coxph(Surv(futime, death) ~ age + pspline(hgb, 2.5, nterm=8), mgus2)
> coef(fit2a)
        age    ps(hgb)3    ps(hgb)4    ps(hgb)5    ps(hgb)6
 0.05399819 -0.26149468 -0.52044496 -0.77438892 -1.04488355
   ps(hgb)7    ps(hgb)8    ps(hgb)9   ps(hgb)10   ps(hgb)11
-1.35106599 -1.59448244 -1.61297463 -1.50478222 -1.38368954
  ps(hgb)12
-1.26113182
> plot(1:10, coef(fit2a)[-1])
```

Now force the last 3 to be equal, then the last 4, and see how this changes the fit.

```
> temp <- c(1:7, 8,8,8)
> fit2b <- coxph(Surv(futime, death) ~ age +
                 pspline(hgb, 2.5, nterm=8, combine=temp),
                 data= mgus2)
> temp2 <- c(1:6, 7,7,7,7)
> fit2c <- coxph(Surv(futime, death) ~ age +
                 pspline(hgb, 2.5, nterm=8, combine=temp2),
                 data= mgus2)
> matplot(1:10, cbind(coef(fit2a)[-1], coef(fit2b)[temp+1],
                      coef(fit2c)[temp2+1]), type='b', pch='abc',
                      xlab="Term", ylab="Pspline coef")
```

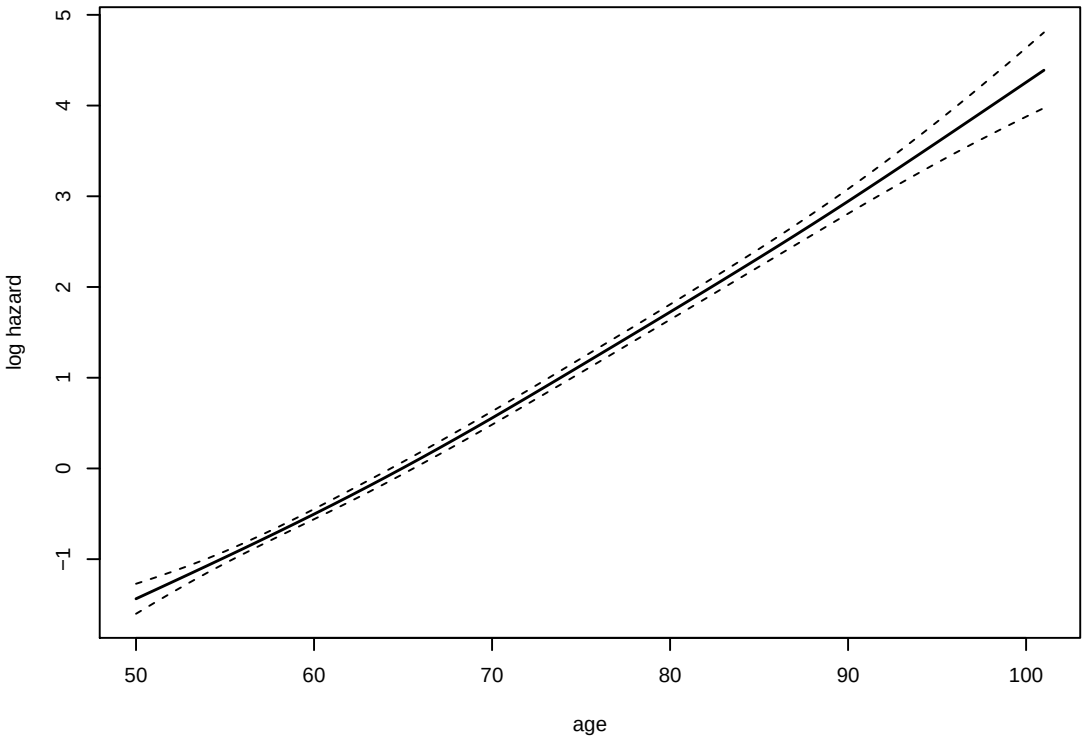
We see that constraining the last four terms along with a degrees of freedom of is almost enough to force monotonicity; it may be sufficient if our goal is a simple plot for display.

This dance between degrees of freedom, number of terms, and constraints has a component of artistry. When all three values become large the result will begin to approach a step function, reminiscent of non-parametric isotonic regression, whereas small values begin to approach a linear fit. The best compromise of smoothness and constraints will be problem specific.

3 Splines in an interaction

As an example we will use the effect of age on survival in the `flchain` data set, a population based sample of subjects from Olmsted County, Minnesota. If we look at a simple model using age and sex we see that both are very significant.

```
> options(show.signif.stars=FALSE) # display intelligence
> fit1 <- coxph(Surv(futime, death) ~ sex + pspline(age, 3), data=flchain)
> fit1
Call:
coxph(formula = Surv(futime, death) ~ sex + pspline(age, 3),
    data = flchain)

                           coef se(coef)      se2    Chisq
sexM                   4.09e-01 4.40e-02 4.40e-02 8.64e+01
pspline(age, 3), linear 1.12e-01 2.20e-03 2.20e-03 2.59e+03
pspline(age, 3), nonlin                           1.09e+01
                         DF      p
sexM                   1.00 <2e-16
pspline(age, 3), linear 1.00 <2e-16
pspline(age, 3), nonlin 2.09 0.0048

Iterations: 7 outer, 20 Newton-Raphson
     Theta= 0.989
Degrees of freedom for terms= 1.0 3.1
Likelihood ratio test=2629  on 4.09 df, p=0  n= 7874
> termplot(fit1, term=2, se=TRUE, col.term=1, col.se=1,
           ylab="log hazard")
```



We used a `pspline` term rather than `ns`, say, because the printout for a pspline nicely segregates the linear and non-linear age effects. The non-linearity is not very large, as compared to the linear portion, but still may be important.

We would like to go forward and fit separate age curves for the males and the females, since the above fit makes an unwarranted assumption that the male/female ratio of death rates will be the same at all ages. The primary problem is that a formula of `sex * pspline(age)` does not work; the coxph routine is not clever enough to do the right thing automatically. (Perhaps some future version will be sufficiently intelligent, but don't hold your breath). If we were using regression splines instead, e.g. `ns(age, df=4)`, a simple call coxph routine using the interaction term would succeed, but then termplot would fall short. The solution below suffices for all cases.

First, we need to create our own dummy variables to handle the interaction.

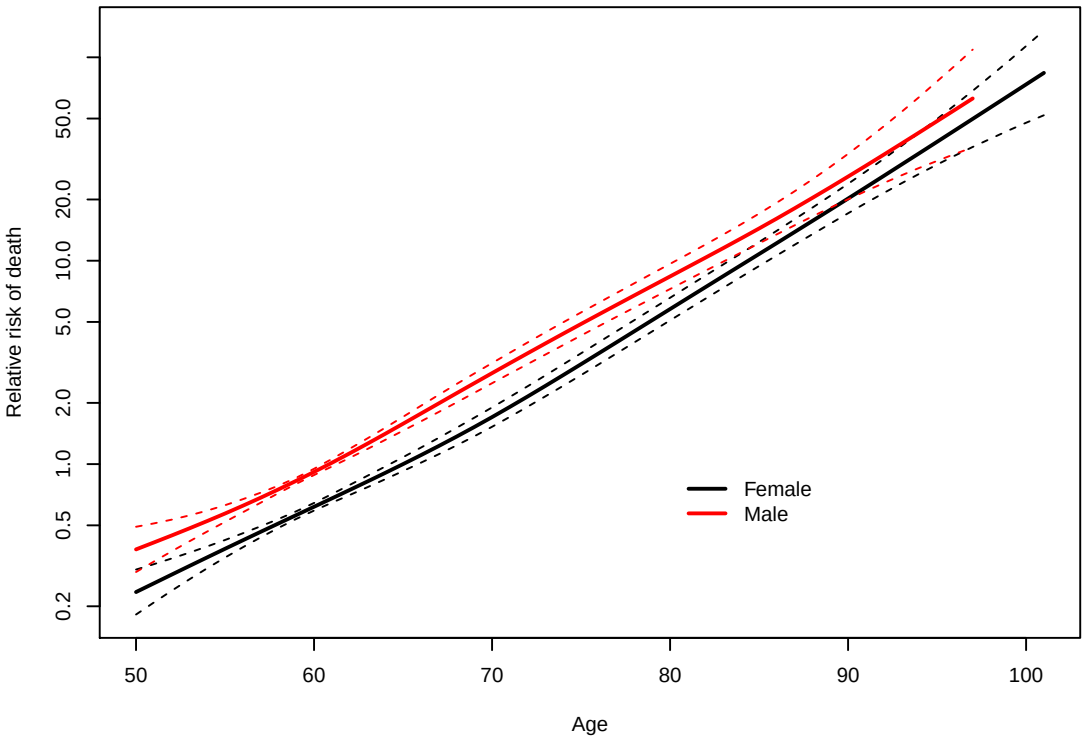
```
> agem <- with(flchain, ifelse(sex=="M", age, 60))
> agef <- with(flchain, ifelse(sex=="F", age, 60))
> fit2 <- coxph(Surv(futime, death) ~ sex + pspline(agef, df=3)
               + pspline(agem, df=3), data=flchain)
> anova(fit2, fit1)
Analysis of Deviance Table
 Cox model: response is  Surv(futime, death)
 Model 1: ~ sex + pspline(agef, df = 3) + pspline(agem, df = 3)
 Model 2: ~ sex + pspline(age, 3)
  loglik  Chisq     Df P(>|Chi|)
```

```
1 -17551
2 -17554 5.8211 2.8583    0.1096
```

The gain in this particular problem is not great, but we will forge ahead. You might well ask why we used 60 as a dummy value of `agem` for the females instead of 0? There is nothing special about the choice, and any value within the range of ages would do as well, though I try to pick one where the standard errors of the curves are not outrageous. If a value of 0 is used it forces the pspline function to create a basis set that includes all the empty space between 0 and 50, and do predictions at 0; these last can become numerically unstable leading to errors or incorrect values.

The Cox model deals with relative hazards, when doing a plot we will usually want to specify who our reference is. By default the termplot function uses an average reference, that is, any plot will be centered to have an average log hazard of 0. In this case, we decided to use 65 year old females as our reference, with all of the hazards relative to them.

```
> # predictions
> pterm <- termplot(fit2, term=2:3, se=TRUE, plot=FALSE)
> # reference
> refdata <- data.frame(sex=c('F', 'M'), agef=c(65, 60), agem=c(60,65))
> pred.ref <- predict(fit2, newdata=refdata, type="lp")
> # females
> tempf <- pterm$agef$y + outer(pterm$agef$se, c(0, -1.96, 1.96))
> frow <- which(pterm$agef$x == 65)
> tempf <- tempf  - tempf[frow,1]  # shift curves
> # males
> tempm <- pterm$agem$y + outer(pterm$agem$se, c(0, -1.96, 1.96))
> mrow  <- which(pterm$agem$x == 65)
> tempm <- tempm + diff(pred.ref) - tempm[mrow,1]
> # plot
> matplot(pterm$agef$x, exp(tempf), log='y', col=1,
          lty=c(1,2,2), type='l', lwd=c(2,1,1),
          xlab="Age", ylab="Relative risk of death")
> matlines(pterm$agem$x, exp(tempm), log='y',
           col=2, lwd=c(2,1,1), lty=c(1,2,2))
> legend(80, 1, c("Female", "Male"), lty=1, lwd=2, col=1:2, bty='n')
```



1. The termplot routine is used to get the data points for the plot, without executing a plot, by use of the `plot=FALSE` argument. The result is a list with one element per term; each element of the list contains x, y, and se components.

2. We had decided to center the female curve at age 65, risk =1. The relative offset for the male curve can be derived directly from `fit2` by adding up the right coefficients, and I used to do it that way but would get it wrong one time out of two. So instead use the `predict` routine to get predicted log hazards for males and females at a particular age. This tells me how far apart the curves should be at that point. We force the females to go through 0, which is exp(0) =1 on the hazard scale.

3. Get the predicted curve and confidence bands for the females as a matrix `tempf`, and then shift them by subtracting the value for a 65 year old female. Do the same for males, plus adding in the curve separation at age 65 from `pred.ref`.

4. The male and female portions don't have quite the same set of age values, there are no 95 year old males in the data set for example, so the plot needs to be done in two steps.

The final curves for males and female are not quite parallel. One thing the plot does not display is that the spacing between the male and female points also has a standard error. This moves the entire bundle of three red curves up and down. It is not clear how best to add this information into the plot. For questions of parallelism and shape, as here, it seemed best to ignore it, which is what the termplot function also does. If someone were reading individual male/female differences off the plot a different choice would be appropriate.