Using Time Dependent Covariates and Time Dependent Coefficients in the Cox Model

Terry Therneau Cindy Crowson
Mayo Clinic

February 26, 2013

1 Introduction

One of the strengths of the Cox model is its ability to encompass covariates that change over time, due to the theoretical foundation in martigales. A *martingale* (original definition) is a betting strategy in games of chance. One of the simplest and best known is doubling the bet each time you lose. For instance consider the following game of roulette:

Bet	Outcome	Win	Running total
R \$1	Red	2	1
R \$1	Black	0	0
R \$2	Black	0	-2
B \$4	Red	0	-6
R \$8	Black	0	-14
B \$16	Black	32	2
B \$1	Red	0	1
B \$2	Black	4	3
⋮	⋮	⋮	⋮

At the end of each cycle of bets the player is another \$1 ahead. The problem is that a modest sequence of losses will exhaust their stake.

The rule for time dependent covariates in a Cox model is simple and essentially the same as that for gambling: you cannot look into the future. A covariate may change in any way based on past data or outcomes, but it may not reach "forward" in time. One of the more well known examples of this error is analysis by response: at the end of a trial a survival curve is made comparing those who had an early response to treatment (shrinkage of tumor, lowering of cholesterol, or whatever), and it discovered that response predicts survival. The problem arises because subjects are classified as responders or non-responders from the beginning of the study, i.e., they are placed into group A or B before the response has occurred. As a consequence, any

early deaths that occur before response can be assessed will be assigned to the non-responder group, even deaths that have nothing to do with the condition under study.

There are many variations on the error: interpolation of the values of a laboratory test linearly between observation times, removing subjects who do not finish the treatment plan, imputing the date of an adverse event midway between observation times, etc. All of these are similar to running a red light in your car: disaster is not guarranteed — but it is likely.

The most common way to encode time-dependent covariates is to use the (start, stop] form of the model.

```
> fit <- coxph(Surv(time1, time2, status) ~ age + creatinine,
               data=mydata)
```

In this data set a patient might have the following observations

subject	time1	time2	status	age	creatinine	...
1	0	15	0	25	1.3	
1	15	46	0	25	1.5	
1	46	73	0	25	1.4	
1	73	100	1	25	1.6	

In this case the variable *age* = age at entry to the study stays the same from line to line, while the value of creatinine varies and is treated as 1.3 over the interval $(0, 15]$, 1.5 over $(15, 46]$, etc. The intervals are open on the left and closed on the right, which means that the creatinine is taken to be 1.3 on day 15. The status variable describes whether or not each interval ended in an event.

One commmon question with this data setup is whether we need to worry about correlated data, since a given subject has multiple observations. The answer is no, we do not. The reason is that this representation is simply a bookkeeping trick; the likelihood equations at any time point use only one copy of any subject, the program picks out the correct row of data at any given time. There are two exceptions to this rule, in which case the cluster variance is necessary:

- When subjects have multiple events.
- When a subject appears in overlapping intervals. This however is almost always a data error, since it corresponds to two copies of the subject being present at the same time, e.g., they could meet themselves on the sidewalk.

2 Examples

2.1 Multiple events

Chronic granulomatous disease (CGD) is a heterogenous group of uncommon inherited disorders characterized by recurrent pyogenic infections that usually begin early in life and may lead to death in childhood. Interferon gamma is a principal macrophage-activating factor shown to partially correct the metabolic defect in phagocytes. It was hypothesized that treatment with interferon might reduce the frequency of serious infections in patients with CGD. In 1986, Genentech, Inc. conducted a randomized, double-blind, placebo-controlled trial in 128 CGD patients who received Genentech's humanized interferon gamma (rIFN-g) or placebo three times

daily for a year. The primary endpoint of the study was the time to the first serious infection. However, data were collected on all serious infections until the end of followup, which occurred before day 400 for most patients. Thirty of the 65 patients in the placebo group and 14 of the 63 patients in the rIFN-g group had at least one serious infection. The total number of infections was 56 and 20 in the placebo and treatment groups, respectively. One patient was taken off on the day of his last infection; all others have some followup after their last episode. Below are the first 10 observations, but with the listing truncatated beyond the fourth infection. Subject 2 has 7 infections, no one in the study has more.

```
  1 204 082888 1 2 12 147.0  62.0 2 2 2 2 414 219 373
  2 204 082888 0 1 15 159.0  47.5 2 2 1 2 439   8  26 152 241 249 322 350
  3 204 082988 1 1 19 171.0  72.7 1 2 1 2 382
  4 204 091388 1 1 12 142.0  34.0 1 2 1 2 388
  5 238 092888 0 1 17 162.5  52.7 1 2 1 1 383 246 253
  6 245 093088 1 2 44 153.3  45.0 2 2 2 2 364
  7 245 093088 0 1 22 175.0  59.7 1 2 1 2 364 292
  8 245 093088 1 1  7 111.0  17.4 1 2 1 2 363
  9 238 100488 0 1 27 176.0  82.8 2 2 1 1 349 294
 10 238 100488 1 1  5 113.0  19.5 1 2 1 1 371
```

```
> cgdname <- c("id", "center", "random.dt", "trt", "sex", "age",
              "height", "weight", "inheritance", "steroids",
              "antibiotic", "inst", "futime", paste("e", 1:7, sep=''))
> cgd1 <- read.table('cgd.dat', header=F, fill=T, col.names=cgdname,
                     colClasses=c("integer", "integer", "character",
                                  rep("integer", 3), rep("numeric",2),
                                  rep("integer", 12)))
> cgd1$ninfect <- rowSums(!is.na(as.matrix(cgd1[,14:20])))
> cgd1$random.dt <- as.Date(cgd1$random.dt, format="%m%d%y")
> cgd1$sex <- factor(cgd1$sex, labels=c("M", "F"))
> cgd1$inst <- factor(cgd1$inst, labels=c("NIH", "US-Other",
                            "Amsterdam", "Europe-Other"))
> temp <- apply(as.matrix(cgd1[,13:20]), 1, function(x) {
     z <- as.vector(x[!is.na(x)])
     if (length(z)==1) cbind(0, z, 0)
     else {
         temp <- cbind(c(0, z[-1]),
                       c(z[-1], z[1]),
                       c(rep(1, length(z)-1), 0))
         if (z[1]== z[length(z)]) temp[-nrow(temp),]
         else temp
     }})
> index <- rep(1:nrow(cgd1), unlist(lapply(temp, nrow)))
> cgd2 <- data.frame( cgd1[index, 1:12],
                     time1= unlist(lapply(temp, function(x) x[,1])),
                     time2= unlist(lapply(temp, function(x) x[,2])),
                     event= unlist(lapply(temp, function(x) x[,3])),
                     enum = unlist(lapply(temp, function(x) 1:nrow(x))))
```

```
> cfit <- coxph(Surv(time1, time2, event) ~ trt + sex + age +
                inheritance + cluster(id), data=cgd2)
```

The logic for creating the time variables is

- If a subject has no events there is a single interval from 0 to last follow-up, with a status of 0.
- If there are events
  - There is an interval for each event (0, event1], (event1, event2], etc. each with status =1.
  - If the follow-up time exceeds the last event there will be a final interval with status =0.

The result of the `apply` function above will be a list with one item per patient, each item being a 3 column matrix with time1, time2, and status. The as.vector command removes names, which can speed the operation. The resulting data frame `cgd2` is in the proper format for the analysis.

2.2 Changing lab tests

To be filled in.

2.3 Predictable time-dependent covariates

Occasionaly one has a time-dependent covariate whose values in the future are predictable. The most obvious of these is patient age, occasionally this may also be true for the cumulative dose of a drug. If age is entered as a linear term in the model, then the effect of changing age can be ignored in a Cox model, due to the structure of the partial likelihood. Assume that subject i has an event at time t_i, with other subject $j \in R_i$ at risk at that time, with a denoting age. The partial likelihood term is

$$\frac{e^{\beta * a_i}}{\sum_{j \in R_i} e^{\beta * a_j}} = \frac{e^{\beta * (a_i + t_i)}}{\sum_{j \in R_i} e^{\beta * (a_j + t_i)}}$$

We see that using time-dependent age (the right hand version) or age at baseline (left hand), the partial likelihood term is identical since $\exp(\beta t_i)$ cancels out of the fraction. Howevever, if the effect of age on risk is *non-linear*, this cancellation does not occur.

Since age changes continuously, we would in theory need a very large (start, stop] data set to completely capture the effect — an interval per day to match the usual resolution for death times. In practice this level of resolution is not necessary; though we all grow older, risk does not increase so rapidly that we need to know out age to the day! For most medical applications year is sufficient, but this still leads to a large data set. One useful way to generate this data set is through use of the `pyears` function. The following example uses data on rehospitalization for cohort of rheumatoid arthritis patients who also have conjestive heart failure (CHF)[1]. The variables are

- patid: patient identifier
- agechf: age at onset of CHF
- yearchf calendar year of CHF, relative to the start of the study
- startday, stopday: an interval of risk
- hospevt: =1 if the interval ends with a hospitalization
- prevhosp: number of prior hosptializations
- duration: duration of RA prior to CHF
- male sex: 1 for male, 0 for female

The age and duration variables have been rounded to .1 year to maintain patient privacy.

```
> load('raheart.rda')
> age2 <- tcut(raheart$agechf*365.25, 0:110* 365.25, labels=0:109)
> rowid <- 1:nrow(raheart)
> pfit <- pyears(Surv(startday, stopday, hospevt) ~ age2 + rowid,
                 data=raheart, data.frame=TRUE, scale=1)
> print(pfit$offtable)

[1] 0

> pdata <- pfit$data
> print(pdata[1:6,])

  age2 rowid pyears n event
1   88     1   7.00 1     0
2   92     2 108.25 1     0
3   93     2 365.25 1     0
4   94     2 365.25 1     0
5   95     2 236.25 1     1
6   95     3 104.00 1     1
```

The `tcut` function attaches a set of cutpoints to the starting age for each subject, it's primary job is to mark the variable as time-increasingfor the `pyears` function. In the `pyears` call we set scale=1 to prevent the age intervals from being rescaled to years, this is not critical. Printing out the value of offtable is important, however. One of them most common mistakes in using `pyears` is mismatched scales, for instance if the age were in years and the follow-up time in days, and a common result of that error is to have follow-up time that fits into none of the categories. This will give a large amount of time that is outside the boundaries of the table. In the resulting data frame the first observation has 7 days of follow-up, exactly as in the starting data. The second observation has been broken into 4 rows, 109.6 days at age 92, then a year each at age 93 and 94, and a final 234.9 days at age 95 ending with a hospitalization. (The odd fractions of a day like .575 are a consequence of rounding the age values to 1 digit.)

Now we combine this with the original data set using the same indexing trick found in the first example. We also need a variable containing the end time for the prior row of each subject and zero for the first row of each subject, which is `lagtime` below. We then fit two models. The first looks at the effect of age at diagnosis of CHF, the second at the effect of current age.

```
> index <- as.integer(pdata$rowid)
> lagtime <- c(0, pdata$pyears[-nrow(pdata)])
> lagtime[1+ which(diff(index)==0)] <- 0 #starts at 0 for each subject
> temp <- raheart$startday[index] + lagtime  #start of each new interval
> data2 <- data.frame(raheart[index,],
                      time1= temp,
                      time2= temp + pdata$pyears,
                      event= pdata$event,
                      age2=  1+ as.numeric(pdata$age2) )
> afit1 <- coxph(Surv(startday, stopday, hospevt) ~ male + pspline(agechf),
                 data=raheart)
> afit2 <- coxph(Surv(time1, time2, event) ~ male + pspline(age2), data2)
> #termplot(afit1, terms=2, se=TRUE, xlab="Age at Diagnosis of CHF")
> #termplot(afit2, terms=2, se=TRUE, xlab="Current Age")
>
> table(with(raheart, tapply(hospevt, patid, sum)))

 0  1  2  3  4  5  6  7  8  9 10 11 13 14 15 18 19 20 22
28 22 18 14  9  7  7 10  3  4  1  2  3  1  1  2  2  1  1
```

In this particular case the two fits are quite similar. In retrospect this perhaps should have been expected: the mean age for onset of CHF in this group is 75 years, which does leave a lot of time for aging. (This analysis is very preliminary, however. As shown by the last line above there are a few patients with a very large number of admissions, sometimes referred to as "entering a revolving door" near the end of their disease. These have an untoward influence on the fit.)

2.4 Predictable covariates, method 2

Another method to create a time-changing covariate is to use the *time-transform* feature of coxph.

```
> afit2b <- coxph(Surv(startday, stopday, hospevt) ~ male + tt(agechf),
                  data=raheart,
                  tt=function(x, t, ...) pspline(x + t/365.25))
> afit2b

Call:
coxph(formula = Surv(startday, stopday, hospevt) ~ male + tt(agechf),
    data = raheart, tt = function(x, t, ...) pspline(x + t/365.25))

                   coef      se(coef) se2     Chisq DF
male                0.228763 0.08909  0.08899 6.59  1.00
```

```
tt(agechf), linear -0.000666 0.00426  0.00426 0.02  1.00
tt(agechf), nonlin                            6.02  3.06
                  p
male              0.01
tt(agechf), linear 0.88
tt(agechf), nonlin 0.12

Iterations: 6 outer, 16 Newton-Raphson
     Theta= 0.932
Degrees of freedom for terms= 1.0 4.1
Likelihood ratio test=13.7  on 5.05 df, p=0.0181  n= 658
```

If there are one or more terms on the right hand side of the equation marked with the tt() operator, the program will pre-compute the values of that variable for each unique event time and strata combination. A user-defined function is called with arguments of

- the covariate: whatever is inside the tt() call
- the event time
- the event number: if there are multiple strata and the same event time occurs in two of them, they are treated separately
- the weight for the observation, if the call used weights

There is a single call to the function with a very large x vector, it contains an element for each subject at risk at each event time. If there are multiple tt() terms in the formula, then the tt argument should be a list of functions with the requisite number of elements.

There are other interesting uses for the time-transform capability. One example is O'Brien's logit-rank test procedure [2]. He proposed replacing the covariate at each event time with a logit transform of its ranks. This removes the influence of any outliers in the predictor x. For this case we ingore the event time argument and concentrate on the groupings.

```
> function(x, t, riskset, weights){
     obrien <- function(x) {
         r <- rank(x)
         (r-.5)/(.5+length(r)-r)
     }
     unlist(tapply(x, riskset, obrien))
 }

function(x, t, riskset, weights){
    obrien <- function(x) {
        r <- rank(x)
        (r-.5)/(.5+length(r)-r)
    }
    unlist(tapply(x, riskset, obrien))
}
```

This relies on the fact that the input argments to tt() are ordered by the event number or riskset. This function is used as a default if no tt argument is present in the coxph call, but there are tt terms in the model formula. (Doing so allowed me to depreciate the survobrien function).

Another interesting useage is to replace the data by simple ranks, not rescaled to 0–1.

```
> function(x, t, riskset, weights)
     unlist(tapply(x, riskset, rank))

function(x, t, riskset, weights)
    unlist(tapply(x, riskset, rank))
```

The score statistic for this model is $(C - D)/2$, where C and D are the number of concordant and discordant pairs, see the survConcordance function. The score statistic from this fit is then a test for significance of the concordance statistics, and is in fact the basis for the standard error reported by survConcordance. The O'Brien test can be viewed as concordance statistic that gives equal weight to each event time, whereas the standard concordance weights each event proportionally to the size of the risk set. (The Cox score statistic depends on the mean x at each event time; since ranks go from 1 to number at risk the mean also scales.)

Although handy, the computational impact of the tt argument should be considered before using it. The Cox model requires computation of a weighted mean and variance of the covariates at each event time, a process that is inherently $O(ndp^2)$ where n = the sample size, d = the number of events and p= the number of covariates. Much of the algorithmic effort in coxph() is to use updating methods for the mean and variance matrices, reducing the compute time to $O((n+d)p^2)$. When a tt term appears updating is not possible; for even moderate size data sets the impact of nd versus $n + d$ can be surprising.

The time-transform is a new addition and still has some rough edges. At this moment the $x = T$ argment is needed to get proper residuals and predicted values, and termplot is unable to properly reconstruct the data to plot the spline fit. Please communicate any concerns or interesting examples to the author.

References

[1] Nicola PJ, Maradit-Kremers H, Roger VL, Jacobsen SJ, Crowson CS, Ballman KV, Gabriel SE. "The risk of Congestive Heart Failure in Rheumatoid Arthritis: a Population-Based Study Over 46 Years." *Arthritis Rheum* 52: 412–20, 2005.

[2] O'Brien, Peter, "A non-parametric test for association with censored data", *Biometrics* 34:243–250, 1978.