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

Terry Therneau Cindy Crowson
Mayo Clinic

July 1, 2015

1 Introduction

This vignette covers 3 different but interrelated concepts:

- An introduction to time dependent covariates, along with some of the most common mistakes.
- Tools for creating time-dependent covariates, or rather the data sets used to encode them.
- Time dependent coefficients.

2 Time dependent covariates

One of the strengths of the Cox model is its ability to encompass covariates that change over time. The practical reason that time-dependent covariates work is based on the underlying way in which the Cox model works: at each event time the program compares the current covariate values of the subject who had the event to the current values of all others who were at risk at that time. One can think of it as a lottery model, where at each death time there is a drawing to decide which subject "wins" the event. Each subject's risk score $\exp(X\beta)$ determines how likely they are to win, e.g., how many "tickets" they have purchased for the drawing. The model tries to assign a risk score to each subject that best predicts the outcome of each drawing based on

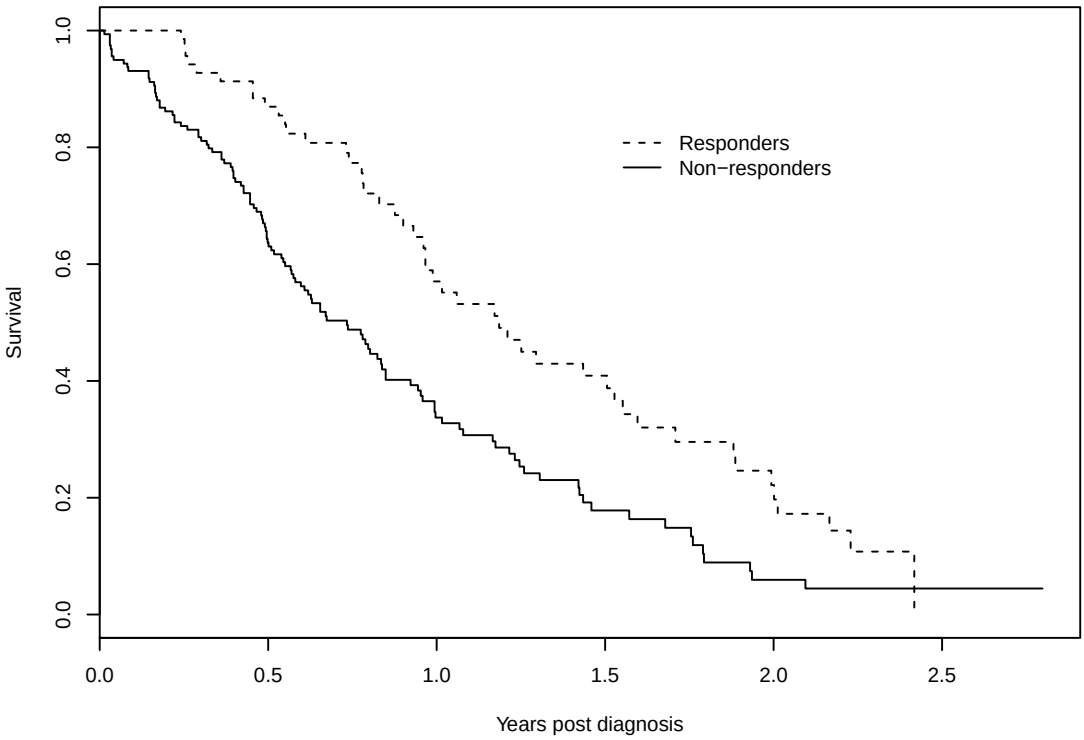
- The risk set: which subjects are present for each event; the set of those able to "win the prize".
- The covariate values of each subject just prior to the event time.

The model has a theoretical foundation in martingale theory, a mathematical construct which arose out of the study of games of chance. A key underlying condition for a martingale like game is that present actions depend only on the past. The decision of whether to play (is one in the risk set or not) and the size of a bet (covariates) can depend in any way on prior bets and patterns of won/lost, but cannot look into the future. If this holds then multiple properties can be proven about the resulting process.

The key rule for time dependent covariates in a Cox model is simple and essentially the same as that for gambling: *you cannot look the future*. A covariate may change in any way based on past data or outcomes, but it may not reach forward in time. A good example of this is found in a recent analysis from the Mayo Clinic study of aging (MCSA), a study which enrolled a stratified random sample from the population of Olmsted County and then has followed them forward in time. The occurence of mild cognitive impairment (MCI), dementia, and death are all of interest. The paper starts out with a table comparing baseline covariates for those who never progress to MCI versus those who ever did, there is also a table of baseline covariates versus survival. Both of these are fine: if you think in terms of an R formula they could be written as `future  past`. A model that predicts survival as a function of ever versus never MCI is not correct, however; that is a model with a future occurence on both sides of the equation.

One of the more well known examples of this error is analysis by treatment 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) to those who did not, and it discovered that responders have a better curve. A Cox model fit to the same data will demonstrate a strong "significant" effect. The problem arises because any early deaths, those that occur before response can be assessed, will all be assigned to the non-responder group, even deaths that have nothing to do with the condition under study. Below is a simple example based on the advanced lung cancer data set. Assume that subjects came in every monthly for 12 cycles of treatment, and randomly declare a "response" for 5% of the subjects at each visit.

```
> set.seed(1953)  # a good year
> nvisit <- floor(pmin(lung$time/30.5, 12))
> response <- rbinom(nrow(lung), nvisit, .05) > 0
> badfit <- survfit(Surv(time/365.25, status) ~ response, data=lung)
> plot(badfit, mark.time=FALSE, lty=1:2,
      xlab="Years post diagnosis", ylab="Survival")
> legend(1.5, .85, c("Responders", "Non-responders"),
        lty=2:1, bty='n')
```



What is most surprising about this error is the *size* of the false effect that is produced. A Cox model using the above data reports a hazard ratio of 1.9 fold with a p-value of less than 1 in 1000.

The alarm about this incorrect approach has been sounded often [1, 2, 7] but the analysis is routinely re-discovered. A slightly subtler form of the error is discussed in Redmond et al [6]. Breast cancer chemotherapy patients were divided into three groups based on whether the patient eventually received $> 85\%$, 65–85% or $< 65\%$ of the dose planned at the start of their treatment. The chemotherapy regiment spans 12 weeks of treatment and the early deaths, not surprisingly, do not get all their dose.

If response is instead coded as a time-dependent covariate whose values depend only on the past, then the problem disappears. For treatment response this will be a variable that starts at 0 for all subjects and is recoded to 1 only when the response occurs. For dose it would measure cumulative dose to date.

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 as midway between observation times, etc. Using future data will often generate large positive or negative bias in the coefficients, but sometimes it generates little bias at all. It is nearly impossible to predict a priori which of these will occur in any given data set. Using such a covariate is similar to jogging across the Los Angeles freeway: disaster is not guaranteed — 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 data set `mydata` 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 ends in an event.

One common 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 programming 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 each time. There two exceptions to this rule:

- When subjects have multiple events, then the rows for events are correlated and a cluster variance is needed.
- 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 in the same strata at the same time, e.g., they could meet themselves on the sidewalk.

A subject can be at risk in multiple strata at the same time, however. This corresponds to being simultaneously at risk for two distinct outcomes.

3 Building time-dependent sets with tmerge

3.1 The function

A useful function for building data sets is `tmerge`, which is part of the survival library. The motivating case for `tmerge` came from a particular problem: the Rochester Epidemiology Project has tracked all subjects living in Olmsted County, Minnesota, from 1965 to the present. For an investigation of cumulative comorbidity we had three data sets

- base: demographic data such as sex and birth date
- timeline: one or more rows for each subject containing age intervals during which they were a resident of the county. The important variables are id, age1 and age2; each (age1, age2) pair marks an interval of residence.
- outcome: one row for each age/outcome pair of interest. The outcomes were 20 comorbid conditions as defined by NIH.

The structure for building the data is shown below. (The data for this example unfortunately cannot be included in the survival library so the code is shown but not executed.)

```
> newd <- tmerge(data1=base, data2=timeline, id=repid, tstart=age1,
                 tstop=age2, options(id="repid"))
> newd <- tmerge(newd, outcome, id=repid, mtype = cumevent(age))
> newd <- with(subset(outcome, event='diabetes'),
               tmerge(newd, id=repid, diabetes= tdc(age)))
> newd <- with(subset(outcome, event='arthritis'),
               tmerge(newd, id=repid, event =tdc(age)))
```

The first call to tmerge adds the timeline for each observation to the baseline data. The `tstart` and `tstop` arguments refer to the starting and ending times for each subject and are taken from data set 2 (`data2`). The `options` argument tells the routine that the identifier variable in data set 1 is called 'repid', and will cause the identifier variable in the final output data set `newd` to have that name. By default, the names of the three key variables are "id", "tstart", and "tstop"; these uniquely identify each row of the final data set.

Each subsequent call adds a new variable to the data set. The second line creates an event variable which is a cumulative count of the number of comorbidities thus far, for each subject. The third line creates a time dependent covariate (tdc) which will be 0 until the age of diabetes and is 1 thereafter, the fourth line creates a time dependent variable for the presence of arthritis.

This is the basic working approach for `tmerge`: first establish baseline covariates and the range over which the subject is at risk, and then add events and/or time dependent covariates to the data set one by one. These additions will often increase the number of rows in the data set. Say at some stage subject Smith has time intervals of (0,15), (15,40) and (40,100) and we add a time dependent covariate `sbp` (systolic blood pressure) which is evaluated at months 6, 12, and 24 with values of 134, 126, and 140, respectively. In the resulting data set Smith will have intervals of

(0,6)	(6,12)	(12,15)	(15,24)	(24,40)	(40,100)
	134	126	126	140	140

The value over the interval (0,6) will be the value of the variable `sbp` in data set 1 — if the variable existed there it is assumed to contain the baseline value — or NA otherwise.

3.1.1 CGD data set

Chronic granulomatous disease (CGD) is a heterogeneous group of uncommon inherited disorders characterized by recurrent pyogenic infections that usually begin early in life and may lead to death in childhood. 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. Data were collected on all serious infections until the end of followup, which occurred before day 400 for most patients. 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, see the help page for `cgd0` for the full list of variable names. The last few columns contain the duration of follow-up for the subject followed by infection times. Subject 1 was followed for 414 days and had 2 infections on days 219 and 373, subject 2 had 7 infections, and subject 3 had none.

```
 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
```

The data set above is included as `cgd0` in the survival library. We want to turn this into a data set that has survival in a counting process form.

- Each row of the resulting data set represents a time interval (time1, time2] which is open on the left and closed on the right. Covariate values for that row are the covariate values that apply over that interval.
- The event variable for each row i is 1 if that time interval ends with an event and 0 otherwise.

We don't want the variables etime1–etime7 in the final data set, so they are left out of the data1 argument in the first call.

```
> newcgd <- tmerge(cgd0[, 1:13], cgd0, id=id, tstop=futime)
> newcgd <- tmerge(newcgd, cgd0, id=id, infect = event(etime1))
> newcgd <- with(cgd0, tmerge(newcgd, id=id, infect = event(etime2)))
> newcgd <- tmerge(newcgd, cgd0, id=id, infect = event(etime3))
> newcgd <- tmerge(newcgd, cgd0, id=id, infect = event(etime4),
                   infect= event(etime5), infect=event(etime6),
                   infect= event(etime7))
> attr(newcgd, "tcount")

       early late gap within boundary leading trailing tied
infect     0    0   0      3        0       0        0    0
infect     0    0   0      2        0       0        0    0
infect     0    0   0      1        0       0        0    0
infect     0    0   0      1        0       0        0    0

> newcgd <- tmerge(newcgd, newcgd, id, enum=cumtdc(tstart))
> all.equal(newcgd[, c("id", "tstart", "tstop", "infect")],
            cgd   [, c("id", "tstart", "tstop", "status")],
            check.attributes=FALSE)

[1] TRUE
```

- A standard way to build data sets is one addition at a time, as shown by the `event(etime1)` and `event(etime2)` lines above. When multiple additions are done using the same d the same data set, however, they can also be done using as single call as shown by the line that adds etime4–etime7. When there are multiple arguments they are processed sequentially.

- Additions with a missing time value are skipped.
- The result of `tmerge` is a data frame with a few extra attributes. One of these, tcount, is designed to help visualize the process, and was printed out after the last step above. (Printing after every step may often be useful.) Assume that a subject already had 3 intervals of (2,5), (5,10) and (14,40). A new event added at time 1 would be "early" while one at time 50 is after any interval and would be recorded as "late". An event at time 3 is within an interval, one at 5 is on the border of two intervals, one at 14 is at the leading edge of an interval and one at time 10 in on the trailing edge. In this data set all new additions fell strictly within prior intervals. We also see that etime6 and etime7 each added only a single event to the data.
- If two observations in data2 for a single person share exactly the same time, the created value will be the sum of the contributions. The "tied" column tells how often this happened; in some data sets this behavior might not be desired and one would need to break the ties before calling tmerge.
- Sometimes the "where" form of the call may be more convenient than using the data2 argument. An example is shown in the addition of etime2.
- The last addition above, after printing the tcount attribute, adds a simple time-dependent variable `enum` which is a running observation count for each subject. This can often be a useful variable in later models or processing, e.g. `enum==1` select off the first row for each subject.
- The extra attributes of the data frame are ephemeral: they will be lost as soon as any further manipulation is done. This is intentional.

The last line above shows that the created data set is identical to `cgd`, a (start, stop] version of the CGD data, also part of the survival library, which had been created by hand several years earlier.

3.2 Stanford heart transplant

The `jasa` data set contains information from the Stanford heart transplant study, in the form that it appeared in the paper of Crowley and Hu [3]. Each row also contain also contains the age, time to transplant and time to last follow-up calculated from these dates. Patients were on medical treatment from their entry to the study until a matching heart became available, at which time they transferred to surgical treatment. As is often the case with real data, this data set contains a few anomalies that need to be dealt with when setting up an analysis data set.

1. The coefficients in table 6.1 of the definitive analysis found in Kalbfliesch and Prentice [4] will only be obtained if covariates are defined in precisely the same way. For age this is (age in days)/ 365.25 - 40 years, and for year of enrollment it is the number of years since the start of the study: (entry date - 1967/10/1)/365.25.
2. One subject died on the day of entry. However (0,0) is an illegal time interval for the program. It suffices to have them die on day 0.5.

3. A subject transplanted on day 10 is considered to have been on medical treatment for days 1–10 and as transplanted starting on day 11. That is, except for patient 38 who died during the procedure on day 5. They should be treated as a transplant death; the problem is resolved by moving the transplant day to 4.5.

Since time is in days the fractional time of 0.5 could be any chosen value t with $0 < t < 1$, it will not affect the results.

```
> tdata <- jasa[, -(1:4)]  #leave off the dates, temporary data set
> tdata$futime <- pmax(.5, tdata$futime)  # the death on day 0
> indx <- with(tdata, which(wait.time == futime))
> tdata$wait.time[indx] <- tdata$wait.time[indx] - .5  #the tied transplant
> sdata <- tmerge(tdata, tdata, id=1:nrow(tdata),
                  death = event(futime, fustat),
                  trans = tdc(wait.time))
> attr(sdata, "tcount")

      early late gap within boundary leading trailing tied
death     0    0   0      0        0       0      103    0
trans     0    0   0     67        0       2        0    0

> coxph(Surv(tstart, tstop, death) ~ age + trans, sdata)

Call:
coxph(formula = Surv(tstart, tstop, death) ~ age + trans, data = sdata)


          coef exp(coef) se(coef)     z     p
age    0.03076   1.03124  0.01450  2.12 0.034
trans -0.00606   0.99396  0.31175 -0.02 0.984

Likelihood ratio test=5.17  on 2 df, p=0.0754
n= 170, number of events= 75
```

This example shows one special case for the `tmerge` function that is moderately common: when the data1 and data2 arguments are the same, and the first created variable is an event code, then the range for each subject is inferred to be from 0 to the event time: an explicit `tstop` argument is not required. It also makes use of a two argument form of `event`. Each of the `event`, `cumevent`, `tdc` and `cumtdc` functions may have a second argument, which will be used as the value or increment to the event code or time dependent covariate. If not present a value of 1 is used. Also note that if the variable being created is already a part of data1, then our updates make changes to that variable. Be careful of this. This feature is what allowed for the `infection` indicator to be build up incrementally in the cgd example given earlier, but quite surprising results can occur when you think a new variable is being created de novo but its name already is in use. (As an example change "trans" to "transplant" in the code just above). For a variable that is not in data1, the starting point is either a vector of NA values or a vector of zeros; the first is used for a `tdc` (or `cumtdc`) call that has two arguments, and the zero vector for all other cases.

The `tcount` table for the above fit shows all the deaths at the trailing edge of their interval, not surprising since last follow-up was used to define the interval of risk. Three of the transplants happened on day 0, one of which we moved to 0.5, the other 2 listed as occurring on the leading edge of the follow-up interval. The other 67 transplants were strictly within the (0, last follow up) interval of each subject.

As a further example of time dependent covariates consider the PBC data. The `pbc` data set contains baseline data and follow-up status for a set of subjects with primary biliary cirrhosis, while the `pbcseq` data set contains repeated laboratory values. The first data set contains data on 312 subjects in a clinical trial plus 106 that agreed to be followed off protocol, the second data set has data only on the protocol subjects.

```
> temp <- subset(pbc, id <= 312, select=c(id:sex, stage))
> pbc2 <- tmerge(temp, temp, id=id, status = event(time, status))
> pbc2 <- tmerge(pbc2, pbcseq, id=id, ascites = tdc(day, ascites),
                 bili = tdc(day, bili), albumin = tdc(day, albumin),
                 protime = tdc(day, protime), alkphos = tdc(day, alk.phos))
> coef(coxph(Surv(time, status==2) ~ log(bili) + log(protime), pbc))

   log(bili) log(protime)
    0.930592     2.890573

> coef(coxph(Surv(tstart, tstop, status==2) ~ log(bili) + log(protime), pbc2))

   log(bili) log(protime)
    1.241214     3.983400
```

The coefficients of bilirubin and prothrombin time are somewhat larger in the time-dependent analysis than using only baseline values. In this autoimmune disease there is steady progression of liver damage, along with a steady rise in these two markers of dysfunction. The baseline analysis captures patients' disease status at the start, the time-dependent also is able to account for those who progress more quickly.

```
> attr(pbc2, "tcount")

        early late gap within boundary leading trailing
ascites     0  138   0   1495        0     312        0
bili        0  138   0      0     1495     312        0
albumin     0  138   0      0     1495     312        0
protime     0  138   0      0     1495     312        0
alkphos     0  138   0      0     1495     312        0
        tied
ascites    0
bili       0
albumin    0
protime    0
alkphos    0
```

The tcount results are interesting. For the first addition of ascites we have 312 observations on a leading edge of follow up (all of the lab values at time 0) and 1495 within the subjects' follow-up interval. This causes a new break point to be added at each laboratory date, and for subsequent additions these 1495 are all on a boundary of two intervals. Another 138 lab values occurred after the last follow-up date of the pbc data set and are ignored. The data for the pbcseq data set was gathered at a later calendar time. Since having lab tests done is a certain marker of survival, would a better analysis have first used this information to extend the last follow-up date for these 138 subjects? Not necessarily.

Odd things happen in survival analysis when risk sets are extended piecemeal. A basic tenet of the Cox model (or a survival curve) is that if someone is marked as being "at risk" over some interval (s, t), this means that "if they had had an event over that interval, we would have recorded it." Say someone ended their initial follow-up time at 3000 days and then had a lab test at 3350 days (subjects returned about once a year). If we only extend the time of those who had a test, then saying that this subject was at risk during the interval (3000, 3350) is false: if they had died in that interval, they would not have had the lab test and not obtained the extension. We need to extend the followup time of all subjects. In the study all subjects were actively followed up and the results of that endeavor are in the futime and status variables of the pbcseq data set. Extension needs to use those variables and not just the presence of another laboratory result.

4 Time dependent coefficients

Time dependent covariates and time dependent coefficients are two different extensions of a Cox model, as shown in the two equations below.

$$\lambda(t) = \lambda_0(t) e^{\beta X(t)} \tag{1}$$

$$\lambda(t) = \lambda_0(t) e^{\beta(t) X} \tag{2}$$

Equation (1) is a time dependent covariate, a commonly used and well understood usage. Equation (2) has a time dependent coefficient. These models are much less common, but represent one way to deal with non-proportional hazards – the proportional hazard assumption is precisely that the coefficient does not change over time: $\beta(t) = c$. The `cox.zph` function will plot an estimate of $\beta(t)$ for a study and is used to diagnose and understand non-proportional hazards. Here for example is a test case using the veterans cancer data.

```
> options(show.signif.stars = FALSE)  # display intelligence
> vfit <- coxph(Surv(time, status) ~ trt + prior + karno, veteran)
> vfit

Call:
coxph(formula = Surv(time, status) ~ trt + prior + karno, data = veteran)


          coef exp(coef) se(coef)     z     p
trt    0.18020   1.19745  0.18347  0.98  0.33
prior -0.00555   0.99446  0.02023 -0.27  0.78
```

```
karno -0.03377   0.96679  0.00511 -6.60 4e-11

Likelihood ratio test=43  on 3 df, p=2.41e-09
n= 137, number of events= 128
```

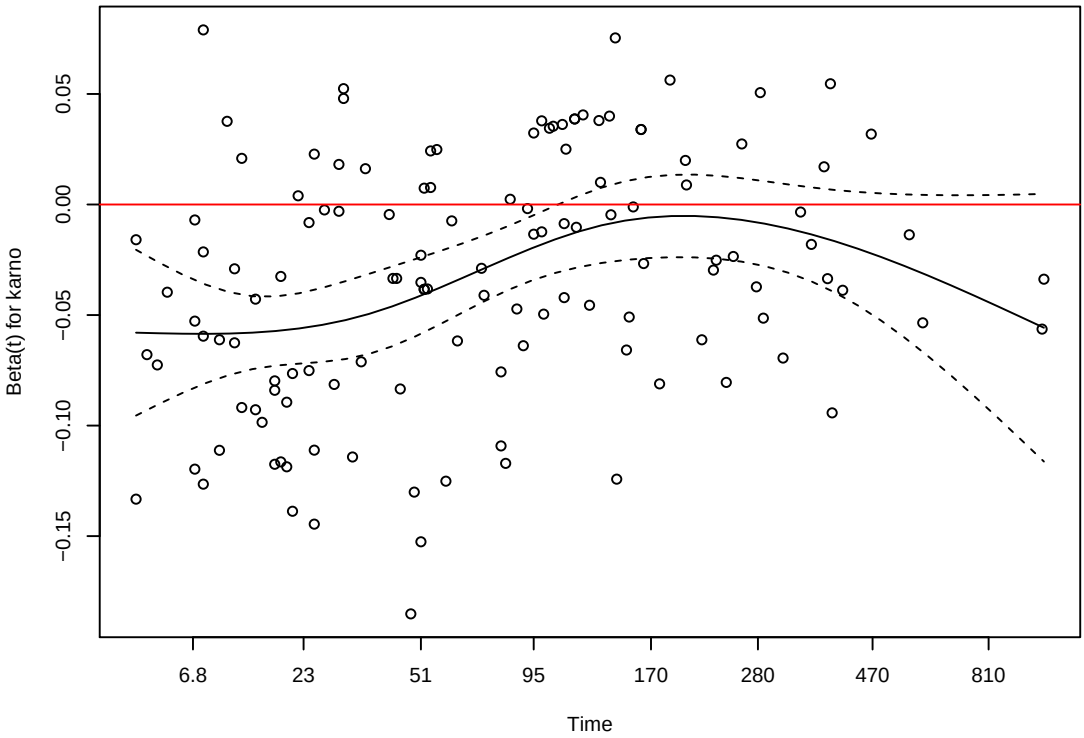
```
> quantile(veteran$karno)

  0%  25%  50%  75% 100%
  10   40   60   75   99
```

```
> zp <- cox.zph(vfit, transform= function(time) log(time +20))
> zp

           rho  chisq       p
trt    -0.0598  0.477 0.48976
prior  -0.1535  3.213 0.07307
karno   0.2890  9.951 0.00161
GLOBAL      NA 12.133 0.00694
```

```
> plot(zp[3])
> abline(0,0, col=2)
```



Karnofsky score is a very important predictor, but its effect is not constant over time as shown by both the test and the plot. Early on it has a large negative effect: the risk of someone

at the first quartile is approximately exp(35*.03377) = 3.2 fold times that of someone at the third quartile, but by 200 days this has waned. The cox.zph function does not produce a formal fit, however, only a graph and a linear test of whether the graph is "flat". What if we want to actually fit the model?

If $\beta(t)$ is assumed to have a simple functional form we can fool an ordinary Cox model program in to doing the fit. The particular form $\beta(t) = a + b\log(t)$ has for instance often been assumed. Then $\beta(t)x = ax + b\log(t)x = ax + bz$ for the special time dependent covariate $z = \log(t)x$. The time scale for the plot above of $\log(t + 20)$ was chosen to make the first part of the plot roughly linear. The simple linear model does not fit over the entire range, but we will forge ahead. (After all, most who fit the log(t) form have not bothered to even look at a plot.) An obvious but incorrect approach is

```
> vfit2 <- coxph(Surv(time, status) ~ trt + prior + karno +
                 I(karno * log(time + 20)), data=veteran)
```

This mistake has been made often enough the the `coxph` routine has been updated to print an error message for such attempts. The issue is that the above code does not actually create a time dependent covariate, rather it creates a time-static value for each subject based on their value for the covariate `time`; no differently than if we had constructed the variable outside of a `coxph` call. This variable most definitely breaks the rule about not looking into the future, and one would quickly find the circularity: large values of `time` predict long survival, because long survival leads to large values for `time`; the resulting model coefficient is large.

A true time-dependent covariate can be constructed using the *time-transform* functionality of coxph.

```
> vfit3 <-  coxph(Surv(time, status) ~ trt + prior + karno + tt(karno),
                 data=veteran,
                 tt = function(x, t, ...) x * log(t+20))
> vfit3

Call:
coxph(formula = Surv(time, status) ~ trt + prior + karno + tt(karno),
    data = veteran, tt = function(x, t, ...) x * log(t + 20))


              coef exp(coef) se(coef)     z       p
trt        0.01648   1.01661  0.19071  0.09  0.9311
prior     -0.00932   0.99073  0.02030 -0.46  0.6462
karno     -0.12466   0.88279  0.02879 -4.33 1.5e-05
tt(karno)  0.02131   1.02154  0.00661  3.23  0.0013

Likelihood ratio test=53.8  on 4 df, p=5.7e-11
n= 137, number of events= 128
```

The time dependent coefficient is estimated to be $\beta(t)$ = -0.125 + 0.021 * log(t + 20). We can add said line to the above plot via `abline(coef(vfit3)[3:4])`. Not surprisingly, the result is rather too high for time > 200.

(The same dichotomy exists in SAS phreg, by the way. A new covariate based on `time` will be fixed, while a programming statement within phreg that uses `time` as a variable will generate time-dependent objects. The error is less likely there because phreg's model statement has no equivalent to the `I()` function, i.e., you cannot create new variables on-the-fly.)

5 Predictable time-dependent covariates

Occasionally 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. However, 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 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 our age to the day!

One method to create a time-changing covariate is to use the *time-transform* feature of coxph. Below is an example using the pbc data set. The longest follow-up time in that data set is over 13 years, follow-up time is in days, and we might worry that the intermediate data set would be huge. The program only needs the value of the time dependent covariate(s) for each subject at the times of events, however, so the maximum number of rows in the intermediate data set is the number of subjects times the number of unique event times.

```
> pfit1 <- coxph(Surv(time, status==2) ~ log(bili) + ascites + age, pbc)
> pfit2 <- coxph(Surv(time, status==2) ~ log(bili) + ascites + tt(age),
                 data=pbc,
                 tt=function(x, t, ...) {
                     age <- x + t/365.25
                     cbind(age=age, age2= (age-50)^2, age3= (age-50)^3)
                 })
> pfit2

Call:
coxph(formula = Surv(time, status == 2) ~ log(bili) + ascites +
    tt(age), data = pbc, tt = function(x, t, ...) {
    age <- x + t/365.25
    cbind(age = age, age2 = (age - 50)^2, age3 = (age - 50)^3)
})
```

```
                 coef exp(coef)  se(coef)     z       p
log(bili)    1.05e+00  2.85e+00  9.45e-02 11.09 < 2e-16
ascites      1.29e+00  3.64e+00  2.63e-01  4.91 9.2e-07
tt(age)age   6.66e-02  1.07e+00  1.47e-02  4.53 5.8e-06
tt(age)age2  3.04e-04  1.00e+00  1.24e-03  0.24   0.807
tt(age)age3 -9.17e-05  1.00e+00  5.00e-05 -1.83   0.067

Likelihood ratio test=179  on 5 df, p=0
n= 312, number of events= 125
   (106 observations deleted due to missingness)
```

```
> anova(pfit2)
```

```
Analysis of Deviance Table
 Cox model: response is Surv(time, status == 2)
Terms added sequentially (first to last)

           loglik    Chisq Df Pr(>|Chi|)
NULL      -639.97
log(bili) -956.62 -633.304  1          1
ascites   -939.07   35.098  1  3.135e-09
tt(age)   -550.50  777.133  3  < 2.2e-16
```

```
> # anova(pfit1, pfit2)  #this fails
> 2*(pfit2$loglik - pfit1$loglik)[2]
```

```
[1] 10.80621
```

Since initial age is in years and time is in days, it was important to scale within the pspline function. The likelihood ratio of 10.8 on 2 degrees of freedom shows that the additional terms are mildly significant.

When 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. 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 can be treated separately
- the weight for the observation, if the call used weights

There is a single call to the function with a 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 [5]. 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 ignore 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 arguments 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 usage 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 $x = TRUE$ argument is needed to get proper residuals and predicted values, and termplot

is unable to properly reconstruct the data to plot a fit. Please communicate any concerns or interesting examples to the author.

References

[1] Anderson JR, Cain KC, and Gelber RD. Analysis of survival by tumor response. J Clinical Oncology 1:710–719, 1983.

[2] M Buyse and P Piedbois. The relationship between response to treatment and survival time. Stat in Med 15:2797–2812, 1996.

[3] J Crowley and M Hu. Covariance analysis of heart transplant survival data. J American Statistical Assoc, 72:27–36, 1977.

[4] J Kalbfleisch and R Prentice. The statistical analysis of failure time data, second edition. Wiley, 2002.

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

[6] Redmond C, Fisher B, Wieand HS. The methodologic dilemma in retrospectively correlating the amount of chemotherapy received in adjuvant therapy protocols with disease free survival: a commentary. Cancer Treatment Reports 67:519–526, 1983.

[7] S Suissa. Immortal time bias in pharmacoepidemiology. Am J Epi, 167:492-499, 2008.