Roundoff error and Tied Times

Terry M Therneau

March 1, 2017

1 Round off error

The heart of the issue can be shown with a simple example. Calculate the following set of intervals for subjects with the same birth date who were enrolled in a study from September 14 through October 23, and then followed for 2–3 months.

```
> birth <- as.Date("1973/03/10")
> start <- as.Date("1998/09/13") + 1:40
> end   <- as.Date("1998/12/03") + rep(1:10, 4)
> interval <- (end-start)
> table(interval)
interval
51 61 71 81
10 10 10 10
```

Each interval has a different start and end date, but there are only 4 unique interval lengths, each of which appears 10 times. Now convert this to an age scale.

```
> start.age <- as.numeric(start-birth)/365.25
> end.age   <- as.numeric(end  -birth)/365.25
> age.interval <- end.age - start.age
> length(unique(age.interval))
[1] 8
> table(match(age.interval, sort(unique(age.interval))))
1 2 3 4 5 6 7 8
7 3 1 9 5 5 9 1
```

There are now eight different age intervals instead of 4, and the 8 unique values appear between 1 and 9 times each. We have become a victim of floating point precision. The exact results above, i.e. how many 'unique' time intervals are found, may depend on your computer system.

Some users prefer to use time in days and some prefer time in years, and those users reasonably expect survival analysis results to be identical on the two scales. Both the coxph and survfit routines treat tied event times in a special way, however, and this roundoff can make actual ties appear as non-tied values. In that case results will differ. Parametric survival routines such

as `survreg` are not affected by the problem since they do not treat ties differently than other values.

In survival version 2.40 this issue has been addressed for the coxph and survfit routines; input times are subjected to the same logic found in the all.equal routine in order to determine actual ties. This may change the results for some data sets.

For the following test case cox1 and cox2 have identical results in in version 2.40 but different results in prior versions of the survival package.

```
> ndata <- data.frame(id=1:30,
                      birth.dt = rep(as.Date("1953/03/10"), 30),
                      enroll.dt= as.Date("1993/03/10") + 1:30,
                      end.dt   = as.Date("1996/10/21") + 1:30 +
                          rep(1:10, 3),
                      status= rep(0:1, length=30),
                      x = 1:30)
> ndata$enroll.age <- with(ndata, as.numeric(enroll.dt - birth.dt))/365.25
> ndata$end.age    <- with(ndata, as.numeric(end.dt - birth.dt))/365.25
> fudays <- with(ndata, as.numeric(end.dt - enroll.dt))
> fuyrs  <- with(ndata, as.numeric(end.age- enroll.age))
> cox1 <- coxph(Surv(fudays, status) ~ x, data=ndata)
> cox2 <- coxph(Surv(fuyrs,  status) ~ x, data=ndata)
```

A downside to the new procedure is that the code will now give an error message for some constructed data sets. An example sent by one user had several time intervals of length 1e-9, which is less than the roundoff precision used by the `all.equal` routine and consequently turned them into illegal intervals of zero length. The `timefix` argument of `coxph.control` can be used to address this.

This general issue of floating point precision arises often enough in R that it is part of the frequently asked questions, see FAQ 7.31 on CRAN. The author of the survival routines (me) has always used days as the scale for analysis – just by habit, not for any particluarly scientific reason – so the issue had never appeared in my work nor in the survival package's test suite. Due to user input, near ties had been addressed earlier in the survfit routine, but only when the status variable was 0/1, not when it is a factor. The new code uses a single routine `aeqSurv` to deal with ties in a uniform way for all the affected functions.

As a final footnote, the simple data set above also gives different results when using the SAS phreg procedure and I suspect the problem exists in other software as well — the R routines are not alone.[1] As a consequence, the maintainer expects to get new emails that "we have found a bug in your code: it gives a different answer than SAS". (This is an actual quote.)

[1] I have reported this to SAS as of November 2016 and was told that they plan to address the problem.