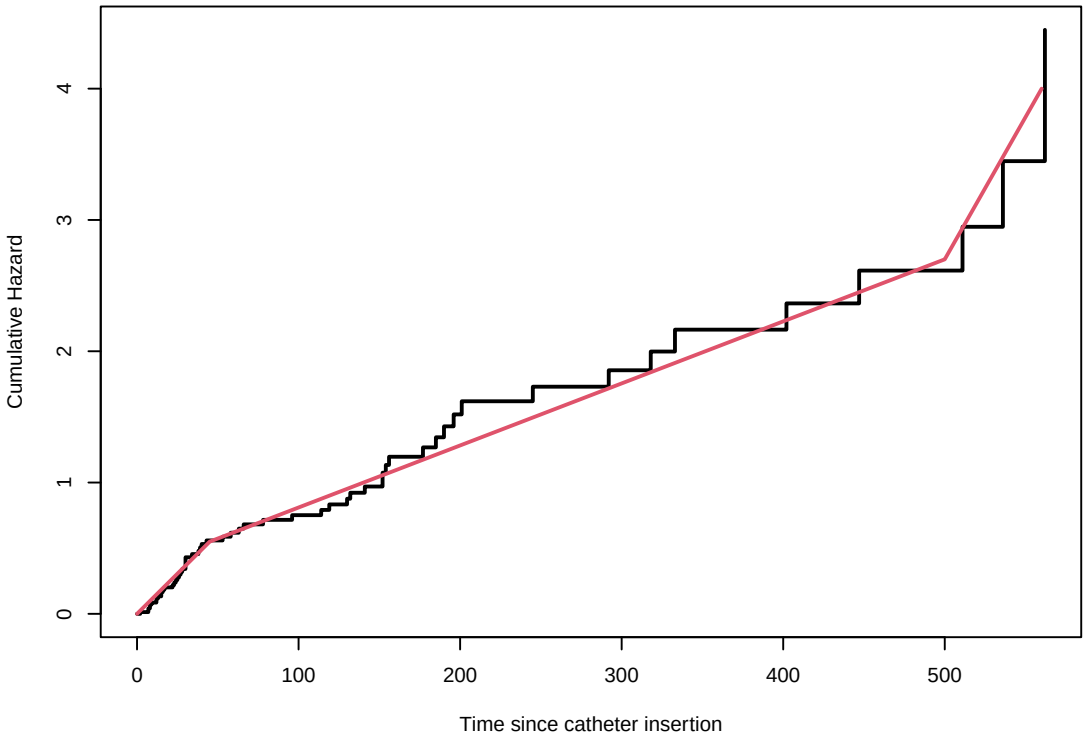
Approximating the Cox Model

Chenyang Zhong, Robert Tibshirani and Terry Therneau

May 2019

The Cox model can be approximated using Poisson regression, a trick that was well known when Cox models were not yet common in the major software packages [**?**, **?**]. Start by plotting the cumulative hazard for the data set, and approximate it with a set of connected line segments. The kidney cancer data set, for instance, is moderately well approximated using cutpoints at 45 and 500 days.

```
> ksurv <- survfit(Surv(time, status) ~1, data=kidney)
> plot(ksurv, fun="cumhaz", conf.int=FALSE, lwd=2,
       xlab="Time since catheter insertion", ylab="Cumulative Hazard")
> lines(c(0, 45, 500, 560), c(0, .55, 2.7, 4), col=2, lwd=2)
```



Break the time scale into intervals based on these cutpoints, and fit a Poisson regression with one intercept per interval.

```
> kdata2 <- survSplit(Surv(time, status) ~., data=kidney, cut=c(45, 500),
      episode="interval")
> kfit1 <- coxph(Surv(time, status) ~ age + sex, kidney, ties='breslow')
> kfit2 <- glm(status ~ age + sex + factor(interval) -1 +
                   offset(log(time-tstart)), family=poisson, data=kdata2)
> cbind(Cox= summary(kfit1)$coefficients[,c(1,3)],
        poisson = summary(kfit2)$coefficients[1:2, 1:2])
            coef    se(coef)     Estimate  Std. Error
age  0.002181516 0.009224643  0.003156994 0.009259435
sex -0.820995315 0.298719655 -0.751242318 0.294277223
```

We see that the coefficients and standard errors are very similar.

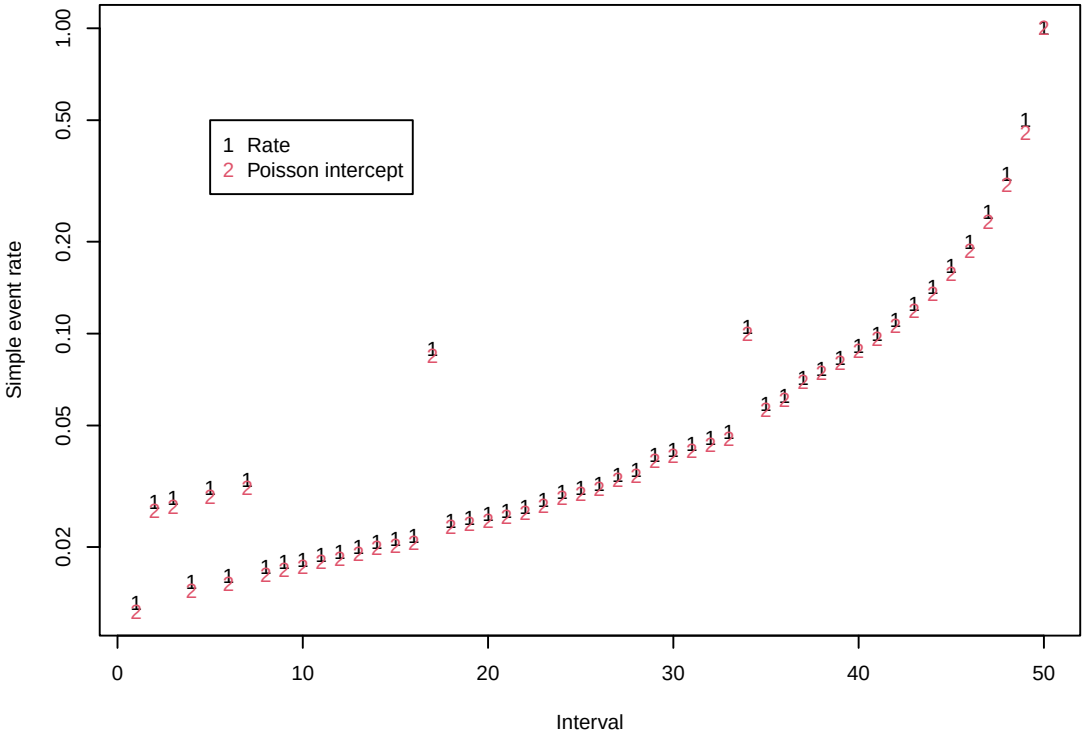
If each unique death time is made into its own interval the Cox result can be duplicated exactly. One more correction is needed for perfect agreement, which is to toss away any partial contributions. If there were unique death times at 100 and 110 days, for instance, and a subject were censored at time 103, those 3 days are not counted in the in the 100–110 interval. If there is someone with follow-up after the last event, that is removed as well. After removal, everyone in the same interval will have the same number of days at risk, which means that an offset correction is no longer needed.

```
> utime <- sort(unique(kidney$time[kidney$status==1])) # unique deaths
> kdata3 <- survSplit(Surv(time, status) ~., data=kidney, cut=utime,
      episode="interval")
> kdata3 <- subset(kdata3, time == c(utime,0)[interval]) # remove partials
> kfit3 <- glm(status ~ age + sex + factor(interval) -1,
                   family=poisson, data=kdata3)
> kfit4 <- glm(status ~ age + sex + factor(interval) -1,
                   family=binomial, data=kdata3)
> rbind(poisson= coef(kfit3)[1:2], binomial = coef(kfit4)[1:2])
                 age        sex
poisson  0.002181516 -0.8209953
binomial 0.002753787 -0.8992513
```

The Poisson coefficients now exactly match those of the Cox model. Almost all of the intervals in `kdata3` have only a single event, i.e., both the event count and the event rate are low, which is the case in which the binomial and Poisson distributions closely approximate each other. Consequently, the last model shows that binomial fits are also effective. This computational trick can be particularly useful in contexts where there is readily available code for binomial outcomes but time-to-event models are lacking, e.g., machine learning.

We can make the connection easier to expliot by pre-centering the data. The plot shows that when this is done, the intercepts are very close to the simple event rate for each interval of (number of events) / (number of observations). We can add this variable to the model as an offset.

```
> counts <- c(table(kdata3$interval))  # subjects in each interval
> xmat <- as.matrix(kdata3[,c('age', 'sex')])
> centers <- rowsum(xmat, kdata3$interval) / counts
> xmat2 <- xmat - centers[kdata3$interval,]
> kfit4a <- glm(status ~ xmat2 + factor(interval) -1, poisson, kdata3)
> temp <- coef(kfit4a)[-(1:2)]   # intercepts
> phat <- with(kdata3, tapply(status, interval, sum)) /counts
> matplot(1:length(counts), cbind(phat, exp(temp)), log='y',
      xlab="Interval", ylab="Simple event rate")
> legend(5, .5, c("Rate", "Poisson intercept"), pch="12", col=1:2)
```




```
> kdata3$phat <- phat[kdata3$interval]  # add phat to the data set
> logit <- function(x) log(x/(1-x))
> kfit4b <- glm(status ~ xmat2 + offset(log(phat)), poisson, kdata3)
> kfit4c <- glm(status ~ xmat2, poisson, kdata3)
> kfit4d <- glm(status ~ xmat2 + offset(logit(phat)), binomial, kdata3,
                subset=(phat<1))
> kfit4e <- glm(status ~ xmat2, binomial, kdata3,
                subset=(phat<1))
> rbind(Cox= coef(kfit1), poisson=coef(kfit4a)[1:2],
        poisson2 = coef(kfit4b)[2:3], poisson3 = coef(kfit4c)[2:3],
        binomial2 = coef(kfit4d)[2:3], binomial3 = coef(kfit4e)[2:3])
```

```
                  age        sex
Cox       0.002181516 -0.8209953
poisson   0.002181516 -0.8209953
poisson2  0.002171066 -0.8184549
poisson3  0.003810010 -0.7885167
binomial2 0.002781817 -0.8961874
binomial3 0.004027962 -0.8370518
>
```

The fits show that adding an approximate per-interval intercept via the offset term gives a very close approximation to the `coxph` fit with the Poisson and a reasonable one with the binomial. Using no intercept at all produces some bias, the size of which will be related to the correlation between `phat` and the covariate in question. The advantage of the offset models is a smaller number of coefficients, particularly for large data sets where the number of intercepts would be excessive.