Describing Continuous-Time Event Occurrence Data

James H. Steiger

Department of Psychology and Human Development
Vanderbilt University

GCM, 2010

Describing Continuous-Time Event Occurrence Data

Introduction

In this module, we introduce methods for describing continuous-time survival data.

Recall that, when we discussed probability theory in Psychology 310, we saw that there was a key distinction between *discrete* and *continuous* probability, necessitated by the fact that any range of values on the number line can be broken up into infinitely many intervals, and that any of those intervals can, in turn, be broken into infinitely many sub-intervals. Consequently, while we can define the probability of the event $X = 1$ in discrete probability theory, in general we cannot in continuous probability. The probability of any numerical value is infinitesimally small.

It turns out that a similar distinction holds in survival analysis, so our notion of *hazard* is different in continuous-time survival analysis from the simple and very convenient notion of a conditional probability that worked so nicely for us in the discrete-time framework.

Introduction

We begin our discussion by emphasizing the similarities between the new, continuous time framework, and the discrete-time framework we've worked in so far.

Then we shall examine the key mathematical differences, the difficulties created by these differences, and the mathematical approaches used to circumvent these difficulties.

Introduction

In a sense, all time recording is discrete. The question is a matter of degree. If the recording is sufficiently fine-grained, we tend to think of it as continuous, even though, in a formal sense, it really isn't.

No matter how fine-grained our recording of time is, we can always "discretize" the data by breaking the time continuum into intervals.

Moreover, as we've already seen, discrete-time survival analysis methods offer definitions of hazard and survival functions that are intuitively straightforward and easily related to our previous studies in regression and probability theory.

Therefore, the question naturally arises: "Why not simply use discrete-time methods with fine-grained data."

Introduction

The answer is that continuous (or nearly continuous) data contain a richer store of information than discrete data, and “discretizing” the data throws some of that information away.

Salient Features

Since time is infinitely divisible, the distribution of event times displays two key properties:

1. The probability of observing any exact event time is infinitesimally small
2. The probability that two individuals will have a truly identical event time is also infinitesimally small

An Illustrative Data Set

To illustrate some of the key differences between discrete and continuous survival data, S&W present data from a study by Diekmann, Jungbauer-Gans, Krassnig, & Lorenz (1996). In this study, motorists were blocked at a busy intersection in Munich, Germany, and the time until they honked their horn was measured to the nearest .01 second.

The Hazard Function

Let T stand for the time the event of interest occurs.

In the general continuous case the *hazard function*, $h(t_j)$, is defined as

$$h(t_j) = \lim_{\Delta t \to 0} \frac{\Pr\left(t \leq T < t + \Delta t \mid T \geq t\right)}{\Delta t} \tag{1}$$

To decipher this expression, look inside the limit. The numerator is the conditional probability that the event occurs in the interval between $T = t$ and $T = t + \Delta t$, given that it did not occur earlier. By dividing this probability by Δt, we *transform it to a rate.* By taking the limit as the width of the interval (i.e., Δt) becomes infinitesimally small, we are calculating the *instantaneous rate* at which the event occurs at a given time. The hazard function in the continuous case is never negative, and, unlike the discrete case, is not bounded above by 1. It can vary from 0 to infinity.

The Survival Function

The *survival function* (or "survivor" function, as some authors refer to it) $S(t_j)$ is defined as that an individual will survive *beyond* time t_j. Formally, if T is the survival time,

$$S(t_j) = \Pr(T > t_j) \tag{2}$$

Note the essential equivalence of this definition with its discrete-time counterpart.

Introduction

Estimation of the hazard and survival functions in continuous-time survival analysis begins with a life table similar to the one we employed in discrete-time survival analysis.

This table is based on the simple notion of constructing intervals, and collecting events (and censored observations) that occur in those intervals.

For example, Table 13.2 on page 477 of S&W breaks the 18 second interval during which the events occurred into 9 intervals (the first of which had no events). The first 8 intervals are 1 second long, and the last interval is 10 seconds long.

Constructing a Grouped Life Table

Once the intervals have been determined, events are tabulated, and an *event conditional probability* (corresponding to what we called the hazard probability in the discrete-time case) is tabulated. This is defined as

$$\hat{p}(t_j) = \frac{n\ \ events_j}{n\ \ at\ \ risk_j} \tag{3}$$

The code on the next slide computes these probabilities for the selected intervals and displays the first 5 columns of Table 13.2

Constructing a Grouped Life Table

```
> honk<-read.table("honking.csv", sep=",", header=T)
> interval<- rep(dim(honk)[1], 0)
> for (i in 1:dim(honk)[1]){
+   interval[i] = min(floor(honk$SECONDS[i]), 8)}
> int.event <- rep(0, 8)
> int.censor <- rep(0, 8)
> for (i in 1:dim(honk)[1]){
+   int <- interval[i]
+   if (honk$CENSOR[i] == 0) int.event[int] <- int.event[int]+1
+   if (honk$CENSOR[i] == 1) int.censor[int] <- int.censor[int] + 1}
> interval <- c(1:8)
> interval.end <- c(2:8, 18)
> interval.w <- interval.end - interval
> int.risk <- c(57, rep(0, 7))
> for (i in 2:8){
+   int.risk[i] <- int.risk[i-1] - (int.event[i-1] + int.censor[i-1])}
> phat = (int.event/int.risk)
> int.table <- cbind(interval, interval.end, int.risk, int.event,
+   int.censor, phat)
```

A Grouped Life Table

Here is the table.

```
> int.table
```

```
     interval interval.end int.risk int.event int.censor    phat
[1,]        1            2       57         5          1 0.08772
[2,]        2            3       51        14          3 0.27451
[3,]        3            4       34         9          2 0.26471
[4,]        4            5       23         6          4 0.26087
[5,]        5            6       13         2          2 0.15385
[6,]        6            7        9         2          2 0.22222
[7,]        7            8        5         1          0 0.20000
[8,]        8           18        4         3          1 0.75000
```

The Discrete-Time Survival Function Estimate

The discrete-time method is very straightforward, and essentially parallels the approach taken with discrete-time data. The survival function is estimates using a product-limit approach, i.e., $\hat{S}(t_0) = 1$, and, for $j = 1, \ldots, J$, we use the recursive formula

$$\hat{S}(t_j) = \hat{S}(t_{j-1}) \times (1 - \hat{p}(t_j)) \tag{4}$$

The Discrete-Time Hazard Function Estimate

The estimated hazard function values using the simple discrete-time approach are obtained by converting the conditional probability estimates to rates by dividing by the interval width, i.e.,

$$\hat{h}(t_j) = \frac{\hat{p}(t_j)}{width_j} \tag{5}$$

Calculating the Estimates

The following code computes the discrete-estimates and adds them to the table.

```
> discrete.h = (phat/interval.w)
> discrete.s <- c(1-discrete.h[1], rep(0, 7))
> for (i in 2:8){
+   discrete.s[i] <- (1 - phat[i])*discrete.s[i-1]
+   }
> int.table <- cbind(int.table,discrete.h,discrete.s)
```

The Table

Here is the augmented table.

```
> int.table
```

```
     interval interval.end int.risk int.event int.censor    phat discrete.h
[1,]        1            2       57         5          1 0.08772    0.08772
[2,]        2            3       51        14          3 0.27451    0.27451
[3,]        3            4       34         9          2 0.26471    0.26471
[4,]        4            5       23         6          4 0.26087    0.26087
[5,]        5            6       13         2          2 0.15385    0.15385
[6,]        6            7        9         2          2 0.22222    0.22222
[7,]        7            8        5         1          0 0.20000    0.20000
[8,]        8           18        4         3          1 0.75000    0.07500
     discrete.s
[1,]    0.91228
[2,]    0.66185
[3,]    0.48665
[4,]    0.35970
[5,]    0.30436
[6,]    0.23673
[7,]    0.18938
[8,]    0.04735
```

The Actuarial Method for the Survivor Function

The Actuarial Method works very similarly to the discrete-time method, except that the actuarial method "corrects for continuity" by assuming that events and censoring are equally likely to occur anywhere in the interval in which they occurred. The number at risk is redefined in terms of what it means to be "at risk for surviving." Once an individual is censored, that individual is no longer at risk to survive. Consequently, the mean number at risk for an interval can be estimated for the survivor function as

$$n' \ at \ risk_j = n \ at \ risk_j - \frac{n \ censored_j}{2} \tag{6}$$

The actuarial estimates of the survivor function are obtained using the same formula as the discrete-time method, except substituting $n' \ at \ risk_j$ for $n \ at \ risk_j$. That is, $\hat{S}(t_0) = 1$, and, for $j = 1, \ldots, J$,

$$\begin{aligned} \hat{S}(t_j) &= \hat{S}(t_{j-1}) \times (1 - \hat{p}(t_j)) \\ &= \hat{S}(t_{j-1}) \times (1 - \frac{n \ events_j}{n' \ at \ risk_j} \\ &= \hat{S}(t_{j-1}) \times (1 - \frac{n \ events_j}{n \ at \ risk_j - n \ censored_j/2} \end{aligned} \tag{7}$$

The Actuarial Method for the Hazard Function

When redefining the number "at risk" in an interval with respect to the hazard function, we need to correct, not just for censoring, but for event occurrence itself, since once an event occurs, the risk set for future occurrences is reduced. So, in computing the estimated hazard function via the actuarial method, we use the redefined risk set formula

$$n'' \ at \ risk_j = n \ at \ risk_j - \frac{n \ censored_j}{2} - \frac{n \ events_j}{2} \tag{8}$$

The estimated hazard function is now

$$\begin{aligned} \hat{h}''(t_j) &= \frac{\hat{p}''(t_j)}{width_j} \\ &= \frac{n \ events_j / n'' \ at \ risk_j}{width_j} \end{aligned} \tag{9}$$

The Actuarial Method

Here is some R code to compute the actuarial estimates and add them to the previous table:

```
> actuarial.s <- c((1-5/(57-1/2)), rep(0,7))
> for (i in 2:8){
+ actuarial.h = (int.event/(int.risk - (int.censor/2) -
+   (int.event/2)))/interval.w
+ actuarial.s[i] <- actuarial.s[i-1]*(1-int.event[i]/(int.risk[i] -
+   int.censor[i]/2))
+ }
> int.table <- cbind(int.table,actuarial.h,actuarial.s)
```

The Actuarial Method

Here is the updated table:

```
> int.table
```

```
     interval interval.end int.risk int.event int.censor    phat discrete.h
[1,]        1            2       57         5          1 0.08772    0.08772
[2,]        2            3       51        14          3 0.27451    0.27451
[3,]        3            4       34         9          2 0.26471    0.26471
[4,]        4            5       23         6          4 0.26087    0.26087
[5,]        5            6       13         2          2 0.15385    0.15385
[6,]        6            7        9         2          2 0.22222    0.22222
[7,]        7            8        5         1          0 0.20000    0.20000
[8,]        8           18        4         3          1 0.75000    0.07500
     discrete.s actuarial.h actuarial.s
[1,]    0.91228      0.0926     0.91150
[2,]    0.66185      0.3294     0.65371
[3,]    0.48665      0.3158     0.47542
[4,]    0.35970      0.3333     0.33959
[5,]    0.30436      0.1818     0.28299
[6,]    0.23673      0.2857     0.21224
[7,]    0.18938      0.2222     0.16979
[8,]    0.04735      0.1500     0.02426
```

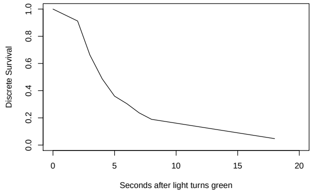
Graphical Comparison of the Two Methods

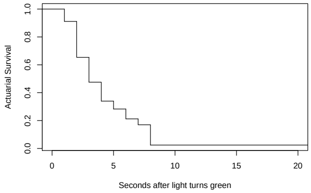
Here is come code for creating graphs of the survivor and hazard functions created by the two methods. Note that the hazard functions are step functions, and are plotted that way.

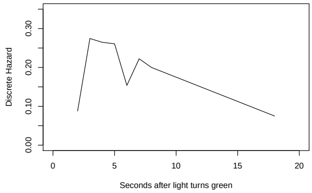
```
> attach(data.frame(int.table))
```

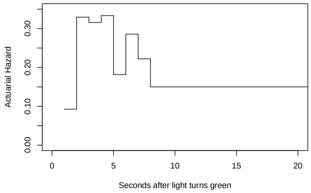
```
> par(mfrow=c(2,2))
> plot(c(0, interval.end), c(1,discrete.s), type = "l",
+ xlim = c(0, 20), ylim = c(0,1),ylab = "Discrete Survival",
+ xlab = "Seconds after light turns green")
> s.hat.steps <- stepfun(interval, c(1, actuarial.s))
> plot(s.hat.steps, do.points = FALSE, xlim = c(0, 20),
+   ylab = "Actuarial Survival",
+   xlab = "Seconds after light turns green", main = "")
> plot(interval.end, discrete.h, type = "l",
+   xlim = c(0, 20), ylim = c(0, .35),ylab = "Discrete Hazard",
+   xlab = "Seconds after light turns green")
> h.hat.steps <- stepfun(interval, c(NA, actuarial.h))
> plot(h.hat.steps, do.points = FALSE, xlim = c(0, 20),
+  ylim = c(0, .35), ylab = "Actuarial Hazard",
+  xlab = "Seconds after light turns green", main = "")
```

Graphical Comparison of the Two Methods









Introduction

A fundamental problem with methods based on dividing the time line into discrete intervals is that information is discarded.

The Kaplan-Meier method circumvents the “discretization” of the data by a clever employment of the product-limit formula.

Single-Unit Intervals

The Kaplan-Meier method, in practice, works very much like the discrete-time methods we've been using so far. However, there is a fundamental difference—instead of rounding times, each event constitutes its own interval. Consequently, each "interval" contains just one event time (unless there are ties, which should virtually never occur). Each interval on the time line begins at one observed event time, and ends just before the next. The first interval is arbitrarily defined to begin at time 0 and end just before the first event (and, of course, contains no events).

Once the Kaplan-Meier intervals are constructed, the hazard and survivor functions are constructed via the discrete-time method

Kaplan-Meier Computations

Table 13.3 in S&W gives Kaplan-Meier computations for the honking data. This table has some minor errors (numbering of intervals is off), and, unfortunately, the code on the UCLA website (at least for now) does not work with more recent versions of R, with the updated `survival` package. Even when it did work, it did not include or properly label the start time and end time columns, nor did it number the intervals starting from zero. Here is some code for performing the calculations. This code works with R 2.10.1.

Kaplan-Meier Computations

```
> library(survival)
> ## Note, UCLA code will not work with latest versions of R, which need "~1"
> t13.3 <-summary( survfit(Surv(honk$SECONDS, abs(honk$CENSOR - 1))~1))
> ## survfit has done much of the work for us
> ## We extract the appropriate columns
> ## Note, we had to reverse the CENSOR variable!
> s.hat <- t13.3[1][[1]]
> time <- t13.3[2][[1]]
> n.atrisk <- t13.3[3][[1]]
> n.events <- t13.3[4][[1]]
> ## Next line has changed from UCLA code
> se.s.hat <- t13.3$std.err
> n.censored <- width <- p.hat <- rep(0, length(t13.3[[1]]))
> for (i in 1:(length(t13.3[[1]]))){
+   p.hat[i] <- n.events[i] / n.atrisk[i]
+   if (i < (length(t13.3[[1]]))){
+     n.censored[i] <- (n.atrisk[i]-n.atrisk[i+1]) - n.events[i]
+     width[i] <- time[i+1] - time[i]}
+   if (i == (length(t13.3[[1]]))){
+     n.censored[i] <- (n.atrisk[i]- n.events[i])
+     width[i] <- NA}}
> h.hat <- p.hat/width
> ## Note, here I patch the UCLA code to add the missing rows and columns
> t.start <- time
> J <- length(t.start)
> t.end <- 0
> for(i in 1:J-1) t.end[i] <- t.start[i+1]
> t.end[J]=Inf
> tab13.3 <- cbind(t.start,t.end,n.atrisk,n.events,n.censored,p.hat,s.hat,se.s.hat,width,h.hat)
> row.0 <- matrix(c(0,t.start[1],n.atrisk[1],0,0,NA,1,NA,NA,NA),1,10)
> tab13.3 <- rbind(row.0,tab13.3)
> row.names(tab13.3) <- 0:J
```

Kaplan-Meier Computations

Here is the table:

```
> options(digits=3,width=100)
> tab13.3
```

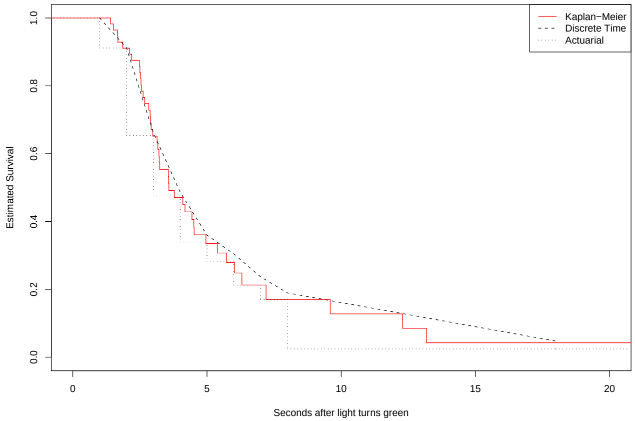
```
   t.start t.end n.atrisk n.events n.censored  p.hat  s.hat se.s.hat width  h.hat
0     0.00  1.41       57        0          0     NA 1.0000       NA    NA     NA
1     1.41  1.51       57        1          1 0.0175 0.9825   0.0174  0.10 0.1754
2     1.51  1.67       55        1          0 0.0182 0.9646   0.0246  0.16 0.1136
3     1.67  1.68       54        1          0 0.0185 0.9467   0.0299  0.01 1.8519
4     1.68  1.86       53        1          0 0.0189 0.9289   0.0343  0.18 0.1048
5     1.86  2.12       52        1          0 0.0192 0.9110   0.0380  0.26 0.0740
6     2.12  2.19       51        1          0 0.0196 0.8931   0.0412  0.07 0.2801
7     2.19  2.48       50        1          1 0.0200 0.8753   0.0441  0.29 0.0690
8     2.48  2.50       48        1          0 0.0208 0.8570   0.0468  0.02 1.0417
9     2.50  2.53       47        1          0 0.0213 0.8388   0.0492  0.03 0.7092
10    2.53  2.54       46        1          0 0.0217 0.8206   0.0514  0.01 2.1739
11    2.54  2.56       45        1          0 0.0222 0.8023   0.0534  0.02 1.1111
12    2.56  2.62       44        1          0 0.0227 0.7841   0.0552  0.06 0.3788
13    2.62  2.68       43        1          0 0.0233 0.7659   0.0569  0.06 0.3876
14    2.68  2.83       42        1          2 0.0238 0.7476   0.0584  0.15 0.1587
15    2.83  2.88       39        1          0 0.0256 0.7285   0.0599  0.05 0.5128
16    2.88  2.89       38        1          0 0.0263 0.7093   0.0614  0.01 2.6316
17    2.89  2.92       37        1          0 0.0270 0.6901   0.0626  0.03 0.9009
18    2.92  2.98       36        1          0 0.0278 0.6710   0.0637  0.06 0.4630
19    2.98  3.14       35        1          1 0.0286 0.6518   0.0647  0.16 0.1786
20    3.14  3.17       33        1          0 0.0303 0.6320   0.0657  0.03 1.0101
21    3.17  3.21       32        1          0 0.0312 0.6123   0.0666  0.04 0.7812
22    3.21  3.22       31        1          0 0.0323 0.5925   0.0673  0.01 3.2258
23    3.22  3.24       30        1          0 0.0333 0.5728   0.0679  0.02 1.6667
24    3.24  3.56       29        1          1 0.0345 0.5530   0.0684  0.32 0.1078
25    3.56  3.57       27        1          0 0.0370 0.5325   0.0688  0.01 3.7037
26    3.57  3.58       26        1          0 0.0385 0.5121   0.0692  0.01 3.8462
27    3.58  3.78       25        1          0 0.0400 0.4916   0.0694  0.20 0.2000
28    3.78  4.10       24        1          1 0.0417 0.4711   0.0694  0.32 0.1302
29    4.10  4.18       22        1          0 0.0455 0.4497   0.0695  0.08 0.5682
30    4.18  4.44       21        1          1 0.0476 0.4283   0.0694  0.26 0.1832
31    4.44  4.51       19        1          0 0.0526 0.4057   0.0693  0.07 0.7519
32    4.51  4.52       18        1          0 0.0556 0.3832   0.0690  0.01 5.5556
33    4.52  4.96       17        1          2 0.0588 0.3606   0.0686  0.44 0.1337
34    4.96  5.39       14        1          1 0.0714 0.3349   0.0683  0.43 0.1661
35    5.39  5.73       12        1          0 0.0833 0.3070   0.0681  0.34 0.2451
36    5.73  6.03       11        1          1 0.0909 0.2791   0.0674  0.30 0.3030
37    6.03  6.30        9        1          1 0.1111 0.2481   0.0666  0.27 0.4115
38    6.30  7.20        7        1          1 0.1429 0.2126   0.0659  0.90 0.1587
39    7.20  9.59        5        1          0 0.2000 0.1701   0.0650  2.39 0.0837
40    9.59 12.29        4        1          0 0.2500 0.1276   0.0611  2.70 0.0926
41   12.29 13.18        3        1          0 0.3333 0.0851   0.0535  0.89 0.3745
42   13.18   Inf        2        1          1 0.5000 0.0425   0.0403    NA     NA
```

Graphing Kaplan-Meier Estimates

Here is some code for graphing the Kaplan-Meier survivor function estimates, and comparing them with their actuarial and discrete-time counterparts. I left out the first panel in Figure 13.2 in the book, since its information is repeated in the second panel.

```
> par(mfrow=c(1,1))
> s.hat.steps <- stepfun(time, c(1, s.hat))
> plot(s.hat.steps, do.points = FALSE, xlim = c(0, 20),
+   ylim = c(0,1), ylab = "Estimated Survival",
+   xlab = "Seconds after light turns green",
+   main = "",col="red")
> ##Then we overlay the actuarial and discrete-time survival
> ##estimates using the imposed intervals from Table 13.2.
> points(c(1, interval.end), c(1, discrete.s),
+  type = "l", lty = 2)
> s.hat.actuarial <- stepfun(c(interval, 18),
+  c(1, actuarial.s, 0.02425622))
> lines(s.hat.actuarial , do.points = FALSE, lty = 3)
> legend("topright", c("Kaplan-Meier", "Discrete Time",
+ "Actuarial"),lty = c(1, 2, 3),col=c("red","black","black"))
```

Graphing Kaplan-Meier Estimates



The Cumulative Hazard Function

The fact that hazard is a probability in discrete-time survival analysis and an instantaneous rate in continuous-time survival analysis is a major point of differentiation between the two methods, and creates challenges in estimation in the continuous case. For one thing, as the text points out, since each interval has only one observation, the interval widths used in computing estimated hazard values vary widely, and so the hazard values are quite erratic. Consequently, these estimates are seldom provided by standard packages. To circumvent this difficulty, we deal with the *cumulative hazard function* $H(t_j)$, which is the cumulation of the instantaneous hazards from t_0 to t_j.

The Meaning of Cumulative Hazard

Technically, of course $H(t_j) = \int_{t_0}^{t_1} h(x)dx$. So the hazard at t_j is the derivative of $H(t_j)$, that is, the slope of the cumlative hazard function at t_j. This simple fact leads to some obvious conclusions, which S&W explore graphically in great detail in section 13.4.1 of their text. You should study that section, with the associated graphs, and be sure it makes sense to you.

The Meaning of Cumulative Hazard

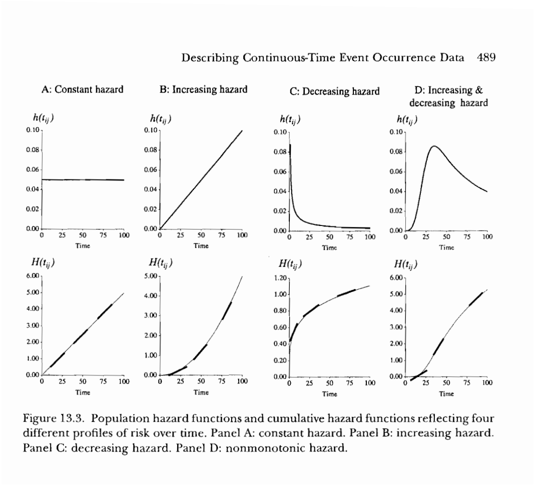


Figure 13.3. Population hazard functions and cumulative hazard functions reflecting four different profiles of risk over time. Panel A: constant hazard. Panel B: increasing hazard. Panel C: decreasing hazard. Panel D: nonmonotonic hazard.

The Nelson-Aalen Method

The Nelson-Aalen estimator of cumulative hazard at t_j sums up interval-specific estimates of total hazard for all intervals up to and including j. The total hazard for interval i is $\hat{h}_{KM}(t_i) \times width_i$.

The Negative Log Survivor-Function Method

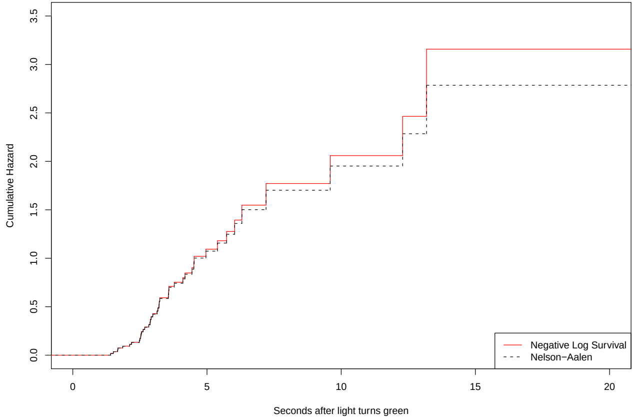
The negative log method exploits a well-known relationship between the cumulative hazard and survivor functions. Specifically, $H(t_j) = -\ln S(t_j)$. The negative log survivor-function estimates are obtained by applying the above equation directly to the Kaplan-Meier estimates of the survivor function.

Plotting Estimates of the Cumulative Hazard Function

Here is some code for replicating the upper panel of Figure 13.4 in the S&W text. This panel shows the negative log survivor and Nelson-Aalen estimates of the cumulative hazard function on the same graph.

```
> nl.s <- -log(s.hat)
> nls.steps <- stepfun(time, c(0, nl.s))
> NA.est <- c(0, rep(0, length(s.hat)))
> for (i in 2:length(NA.est)){
+   NA.est[i] <- NA.est[i-1]+p.hat[i-1]}
> NA.steps <- stepfun(time, NA.est)
> par(mfrow=c(1,1))
> plot(nls.steps, do.points = FALSE, xlim = c(0, 20), ylim = c(0,3.5),
+   ylab = "Cumulative Hazard", xlab = "Seconds after light turns green",
+    main = "",col="red")
> lines(NA.steps, do.points = FALSE, lty =2, xlim = c(0,20))
> legend("bottomright", c("Negative Log Survival", "Nelson-Aalen"),
+ lty = c(1, 2),col=c("red","black"))
```

Plotting Estimates of the Cumulative Hazard Function



Estimating the Hazard Function

So far, we've talked about estimation of the *cumulative* hazard function. Estimation of the hazard function itself is somewhat more challenging.

Since hazard is the slope of the cumulative hazard function, one way to obtain crude estimates of the hazard at any point is to take the difference between cumulative hazard values and use them to estimate the slope of the cumulative hazard function at that time.

These estimates will be crude and quite variable, so we might smooth them by taking an average of their values over a window ranging around the time of interest. The larger the window, the smoother the resulting estimated graph will tend to be.

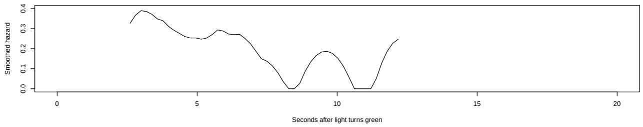
Choice of an interval width, much like the choice of a bin width in histogram construction, is an art as well as a science. You should experiment with several widths.

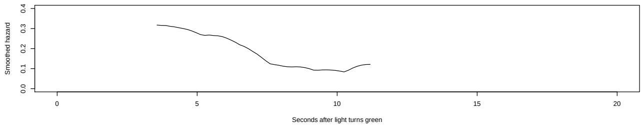
Kernel Smoothed Estimate

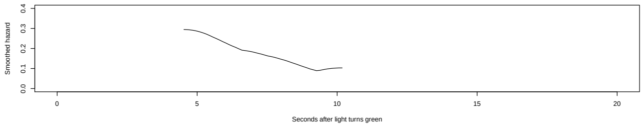
Here is some R code for generating and graphing kernel-smoothed estimates.

```
> smooth<- function(width, time, survive){
+   n=length(time)
+   lo=time[1] + width
+   hi=time[n] - width
+   npt=50
+   inc=(hi-lo)/npt
+   s=lo+t(c(1:npt))*inc
+   slag = c(1, survive[1:n-1])
+   h=1-survive/slag
+   x1 = as.vector(rep(1, npt))%*%(t(time))
+   x2 = t(s)%*%as.vector(rep(1,n))
+   x = (x1 - x2) / width
+   k=.75*(1-x*x)*(abs(x)<=1)
+   lambda=(k%*%h)/width
+   smoothed= list(x = s, y = lambda)
+   return(smoothed)
+ }
> par(mfrow=c(3,1))
> bw1 <- smooth(1, time, s.hat)
> plot(bw1$x, bw1$y, type = "l", xlim = c(0, 20), ylim = c(0, .4),
+   xlab = "Seconds after light turns green", ylab = "Smoothed hazard")
> bw2 <- smooth(2, time, s.hat)
> plot(bw2$x, bw2$y, type = "l", xlim = c(0, 20), ylim = c(0, .4),
+   xlab = "Seconds after light turns green", ylab = "Smoothed hazard")
> bw3 <- smooth(3, time, s.hat)
> plot(bw3$x, bw3$y, type = "l", xlim = c(0, 20), ylim = c(0, .4),
+   xlab = "Seconds after light turns green", ylab = "Smoothed hazard")
```

Kernel Smoothed Estimates



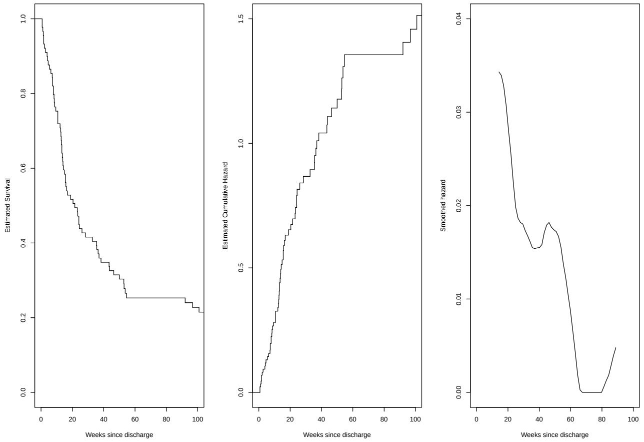




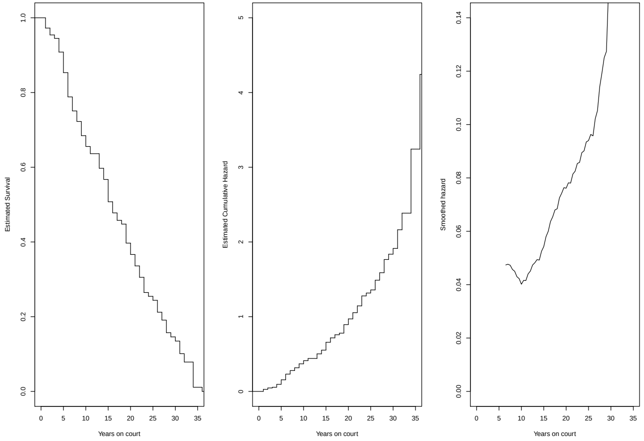
Putting it All Together

In the final section of Chapter 13, Singer & Willett present several examples and urge you to consider them carefully to develop your intuitions about survivor, cumulative hazard, and kernel-smoothed hazard functions.

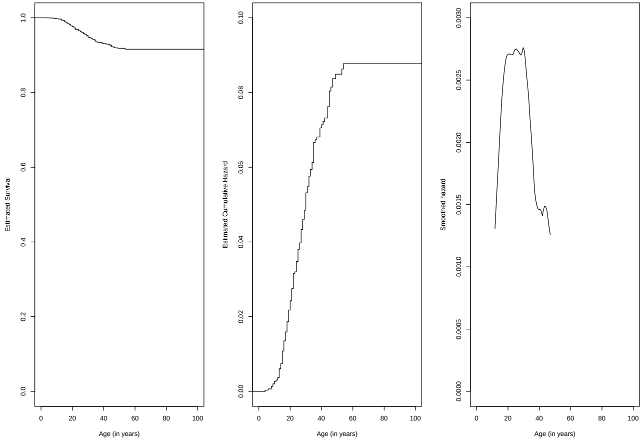
Time to Relapse among Alcoholics



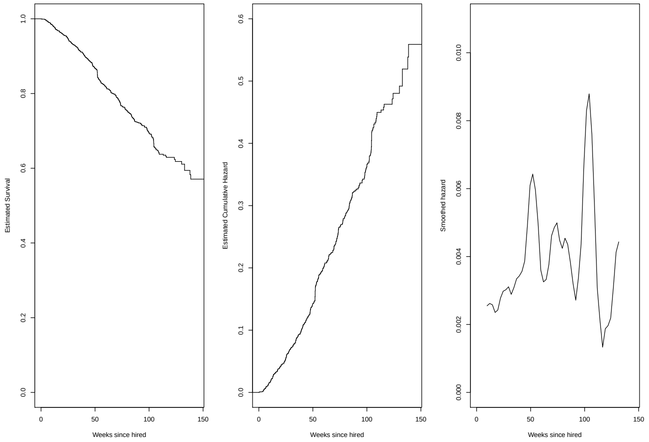
Time to Retirement among Supreme Court Judges



Age at First Depressive Episode



Employment Duration among Health Care Workers


Estimated Survival
Weeks since hired
Estimated Cumulative Hazard
Weeks since hired
Smoothed hazard
Weeks since hired