Multi-state models as a data exploration tool

Terry Therneau

October 29, 2016

1 Multi-state curves

Consider the simple `survfit` call

```
> curves <- survfit(Surv(time, status) ~ group, data=mydata)
```

In the classic case `status` is either a logical or 0/1 numeric variable that represents censoring (0 or false) or an event (1 or true), and the result is a survival curve for each group. If `status` is a factor, however, the result is a multi-state estimate. In this case the first level of `status` is used to code censoring while the remaining ones are possible states. Here is a simple competing risks example where the three endpoints are labeled as a, b and c.

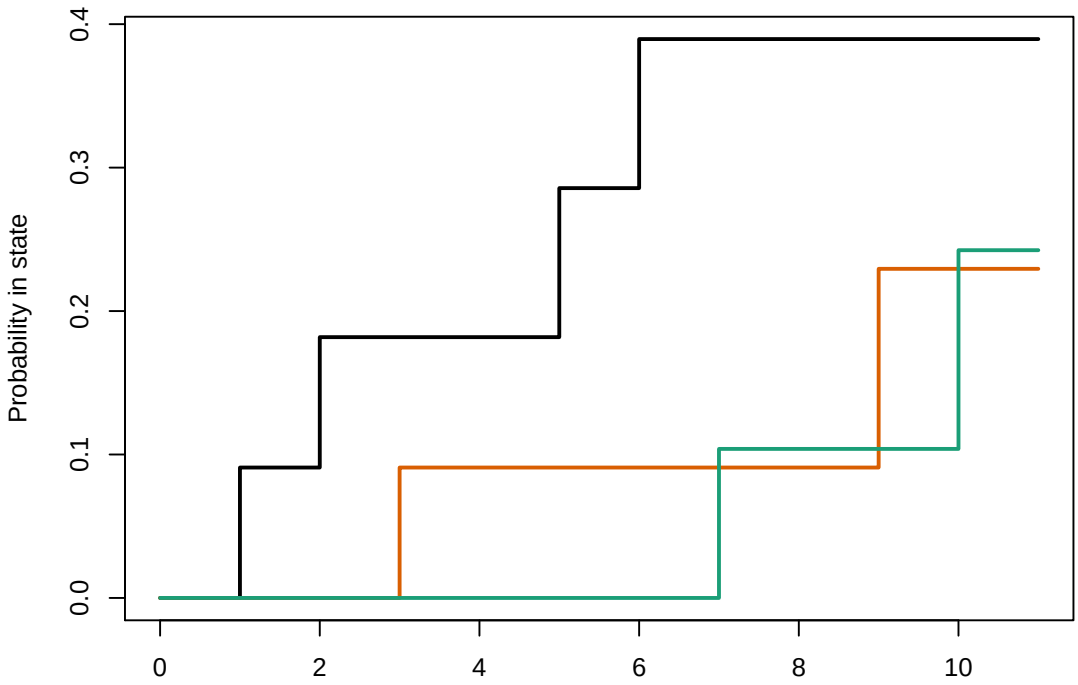
```
> set.seed(1952)
> crdata <- data.frame(time=1:11,
                       endpoint=factor(c(1,1,2,0,1,1,3,0,2,3,0),
                                       labels=c("censor", "a", "b", "c")))
> tfit  <- survfit(Surv(time, endpoint) ~ 1, data=crdata)
> dim(tfit)
[1] 1 4
> summary(tfit)
Call: survfit(formula = Surv(time, endpoint) ~ 1, data = crdata)

 time n.risk n.event   P(a)   P(b)  P(c)   P()
    1     11       1 0.0909 0.0000 0.000 0.909
    2     10       1 0.1818 0.0000 0.000 0.818
    3      9       1 0.1818 0.0909 0.000 0.727
    5      7       1 0.2857 0.0909 0.000 0.623
    6      6       1 0.3896 0.0909 0.000 0.519
    7      5       1 0.3896 0.0909 0.104 0.416
    9      3       1 0.3896 0.2294 0.104 0.277
   10      2       1 0.3896 0.2294 0.242 0.139
```

The resulting object `tfit` contains an estimate of P(state), the probability of being in each state. P is a matrix with one row for each time and one column for each of the four states a–c and the "no event as of yet" state; we will often refer to the latter as the entry state. By definition each

row of P sums to 1. The plot of the fit will have 3 curves, by default the curve for an unnamed state is not displayed. (Since they sum to 1 one of the 4 curves is redundant, and the entry state is normally the least interesting of the set.)

```
> plot(tfit, col=1:3, lwd=2, ylab="Probability in state")
```



The resulting `survfms` object appears as a matrix and can be subscripted as such, with a column for each state and rows for each group that was created by any variables on the right hand side of the formula. This makes it simple to display a subset of the curves using plot or lines commands. The unnamed state in the above fit, for instance, can be displayed with `plot(tfit[,4])`.

The curves are computed using the Aalen-Johansen estimator. The Kaplan-Meier estimate and the cumulative incidence esitmate (for competing risks) are each a special case of the AJ estimate. It is more general that that, however; a given subject can have multiple transitions from state to state, including transitions to a state that was visited earlier. In this case the dataset structure is similar to that for time varying covariates in a Cox model: the time variable will be intervals $(t_1, t_2]$ which are open on the left and closed on the right, and the status variable contains the state that was entered at time t_2, and a subject will have multiple lines of data. There are a few restrictions.

- An identifier variable is required which indicates which rows of the data frame belong to each subject. If the `id` argument is missing the code assumes that each row of data is a separate subject, which leads to a nonsense estimate when there are actually multiple rows for each.

- Subjects do not have to enter at time 0 or all at the same time, but each must traverse a connected segment of time. Disjoint intervals such as the pair $(0, 5]$, $(8, 10]$ are illegal.
- A subject cannot change groups. Any covariates on the right hand side of the formula must remain constant within subject. (This function is not a way to creat supposed 'time-dependent' survival curves.)
- Subjects may have case weights, and these weights may change over time for a subject.

By default every subject is assumed to start in an unnamed common entry state. The `istate` argument can instead be used to designate an entry state for each subject; like variables in the formula it is searched for in the `data` argument. The distribution of states at the first event time is treated as the initial distribution of states; like ordinary survival an observation which is censored before the first event time has no impact on the results.

The extended example below is intended to give more information about the routines.

2 Data set

The `myeloid` data set contains simulated data which mimics that from a trial in subjects with acute myeloid leukemia. In this comparison of two conditioning regimens the canonical path for a subject is initial therapy $\rightarrow$ complete response (CR) $\rightarrow$ hematologic stem cell transplant (HSCT) $\rightarrow$ sustained remission, followed by relapse or death.

```
> myeloid[1:5,]
  id trt futime death txtime crtime rltime
1  1   B    235     1     NA     44    113
2  2   A    286     1    200     NA     NA
3  3   A   1983     0     NA     38     NA
4  4   B   2137     0    245     25     NA
5  5   B    326     1    112     56    200
```

The first few rows of data are shown above. The data set contains the follow-up time and status at last follow-up for each subject, along with the time to transplant (txtime), complete response (crtime) or relapse after CR (rltime). Subject 1 did not receive a transplant, as shown by the NA value, and subject 2 did not achieve CR.

Overall survival curves for the data are shown in figure 1. The difference between the treatment arms A and B is substantial. A goal of this analysis is to better understand this difference. Here is the code to generate the simple survival curves:

```
> sfit0 <- survfit(Surv(futime, death) ~ trt, myeloid)
> plot(sfit0, xscale=365.25, xaxs='r', col=1:2, lwd=2,
      xlab="Years post enrollment", ylab="Survival")
> legend(20, .4, c("Arm A", "Arm B"),
        col=1:2, lwd=2, bty='n')
```

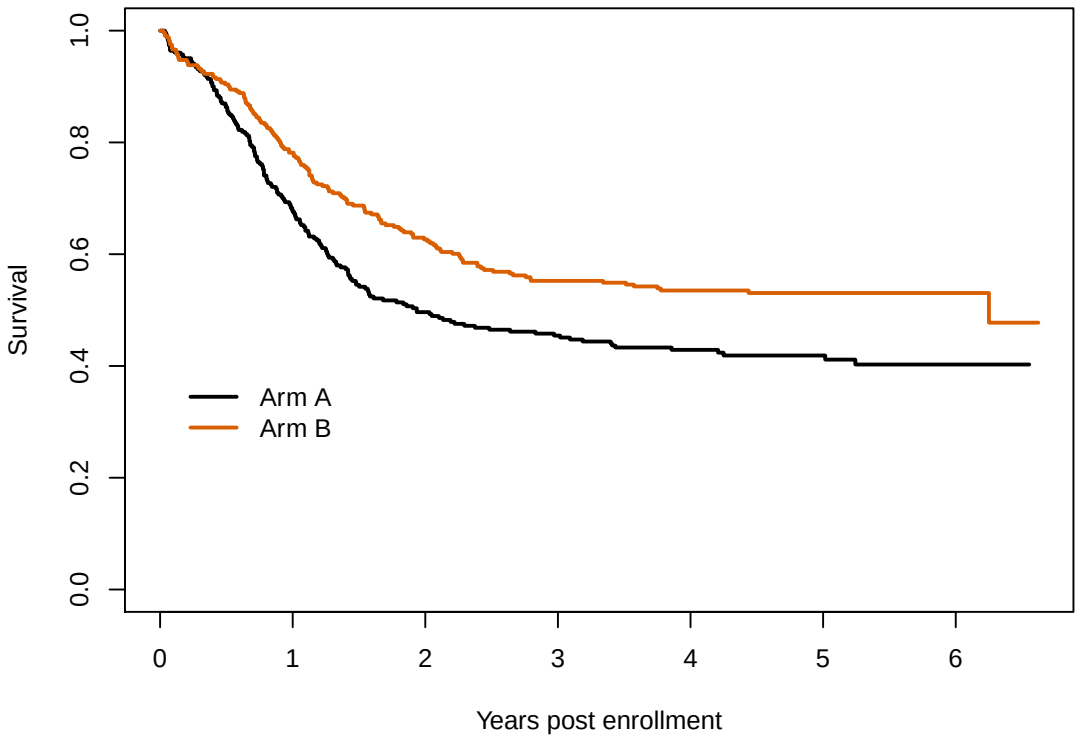


Figure 1: Overall survival curves for the two treatments.

3 Competing risks

A first step towards deeper analysis is to look at intermediate states one at a time, e.g., how many subjects ever achieve a CR or ever receive a transplant. Create a working data set that contains variables for simple 2-state competing risks for the pairs CR/death and transplant/death. For competing risks each subject has a single row of data, so this data set simply adds two new variables and redefines two others. At the same time we will convert from days to months. This is the natural time scale for our plots, and forestalls adding the `xscale` argument to every plot call.

```
> data1 <- myeloid
> data1$crstat <- factor(with(data1, ifelse(is.na(crtime), death, 2)),
                         labels=c("censor", "death", "CR"))
> data1$crtime <- with(data1, ifelse(crstat=="CR", crtime, futime))
> data1$txstat <- factor(with(data1, ifelse(is.na(txtime), death, 2)),
                         labels=c("censor", "death", "transplant"))
> data1$txtime <- with(data1, ifelse(txstat=="transplant", txtime, futime))
> for (i in c("futime", "crtime", "txtime", "rltime"))
     data1[[i]] <- data1[[i]] * 12/365.25  #rescale to months
```

This data set is the basis for our first set of curves, which are shown in figure 2. The plot overlays three separate `survfit` calls: standard surivival until death, complete response with death as a competing risk, and transplant with death as a competing risk. For each fit we have shown one selected state: the fraction who have died, fraction ever in CR, and fraction ever to recieve transplant, respectively. Most of the CR events happen before 2 months (the green vertical line) and nearly all the additional CRs conferred by treatment B occur between months 2 and 8. Most transplants happen after 2 months, which is consistent with the clinical guide of transplant after CR. The survival advantage for treatment B begins between 4 and 5 months, which argues that it could be at least partially a consequence of the additional CR events.

The code to draw figure 2 is below. It can be separated into 5 parts:

1. Fits for the 3 endpoints are simple and found in the first 3 lines. The `crstat` and `txstat` variables are factors, which causes a multi-state curve to be generated.

2. The `layout` and `par` commands are used to create a multi-part plot with curves on the left and state space diagrams on the right, and to reduce the amount of white space between them.

3. Draw a subset of the curves via subscripting. A multi-state survfit object appears as a matrix of curves, with one row for each group (treatment) and one column for each state. The CR state is the second column in `sfit2`, for instance. The CR fit was drawn first simply because it has the greatest y-axis range, then the other curves added using the lines command.

4. Decoration of the plots. This includes the line types, colors, legend, choice of x-axis labels, etc.

5. Add the state space diagrams. The functions for this are described in the last section of the vignette.

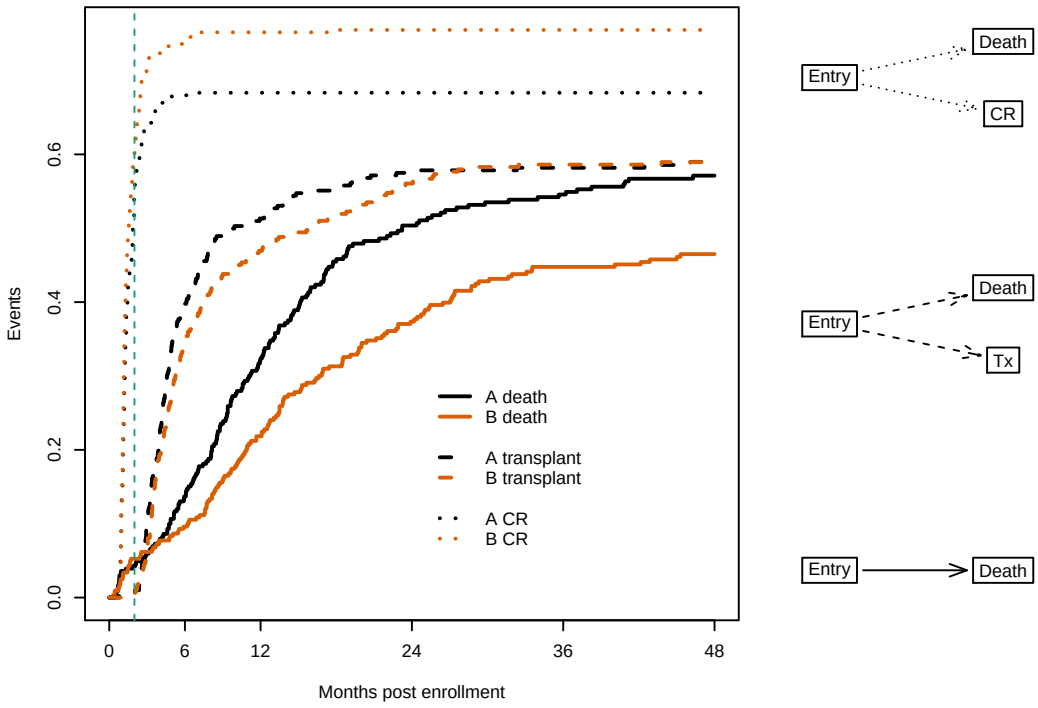


Figure 2: Overall survival curves: time to death, to transplant (Tx), and to complete response (CR). Each shows the estimated fraction of subjects who have ever reached the given state. The vertical line at 2 months is for reference. The curves were limited to the first 48 months to more clearly show early events. The right hand panel shows the state-space model for each pair of curves.

```
> sfit1 <- survfit(Surv(futime, death) ~ trt, data1) #survival
> sfit2 <- survfit(Surv(crtime, crstat) ~ trt, data1) # CR
> sfit3 <- survfit(Surv(txtime, txstat) ~ trt, data1)
> layout(matrix(c(1,1,1,2,3,4), 3,2), widths=2:1)
> oldpar <- par(mar=c(5.1, 4.1, 1.1, .1))
> plot(sfit2[,2], mark.time=FALSE, fun='event', xmax=48,
         lty=3, lwd=2, col=1:2, xaxt='n',
      xlab="Months post enrollment", ylab="Events")
> lines(sfit1, mark.time=FALSE, xmax=48, fun='event', col=1:2, lwd=2)
> lines(sfit3[,2], mark.time=FALSE, xmax=48, fun='event', col=1:2,
          lty=2, lwd=2)
> xtime <- c(0, 6, 12, 24, 36, 48)
> axis(1, xtime, xtime) #marks every year rather than 10 months
> temp <- outer(c("A", "B"), c("death", "transplant", "CR"),  paste)
> temp[7] <- ""
> legend(25, .3, temp[c(1,2,7,3,4,7,5,6,7)], lty=c(1,1,1, 2,2,2 ,3,3,3),
       col=c(1,2,0), bty='n', lwd=2)
> abline(v=2, lty=2, col=3)
> # add the state space diagrams
> par(mar=c(4,.1,1,1))
> crisk(c("Entry","Death", "CR"), alty=3)
> crisk(c("Entry","Death", "Tx"), alty=2)
> crisk(c("Entry","Death"))
> par(oldpar)
```

The association between a particular curve and its corresponding state space diagram is critical. As we will see below, many different models are possible and it is easy to get confused. Attachment of a diagram directly to each curve, as was done above, will not necessarily be day-to-day practice, but the state space should always be foremost. If nothing else, draw it on a scrap of paper and tape it to the side of the terminal when creating a data set and plots.

Figure 3 shows the transplant curves overlaid with the naive KM that censors subjects at death. There is no difference in the initial portion as no deaths have yet intervened, but the final portion overstates the transplant outcome by more than 10%.

1. The key problem with the naive estimate is that subjects who die can never have a transplant. The result of censoring them is an estimate of the "fraction who would be transplanted, if death before transplant were abolished". This is not a real world quantity.

2. In order to estimate this fictional quantity one needs to assume that death is uninformative with respect to future disease progression. The early deaths in months 0–2, before transplant begins, are however a very different class of patient. Non-informative censoring is untenable.

We are left with an unreliable estimate of an uninteresting quantity. Mislabeling any true state as censoring is always a mistake, one that will not be repeated here. Here is the code for figure 3. The use of a logical (true/false) as the status variable in the `Surv` call leads to ordinary survival calculations.

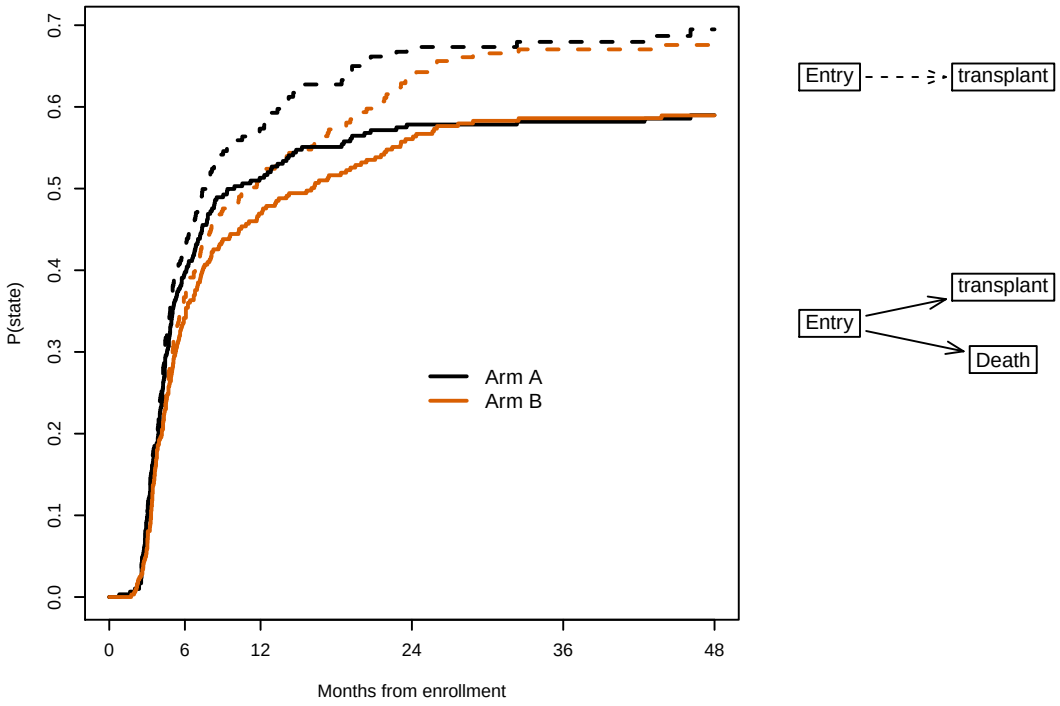


Figure 3: Correct (solid) and invalid (dashed) estimates of the number of subjects transplanted.

```
> badfit <- survfit(Surv(txtime, txstat=="transplant") ~ trt, data1)
> layout(matrix(c(1,1,1,2,3,4), 3,2), widths=2:1)
> oldpar <- par(mar=c(5.1, 4.1, 1.1, .1))
> plot(badfit, fun="event", xmax=48, xaxt='n', col=1:2, lty=2, lwd=2,
      xlab="Months from enrollment", ylab="P(state)")
> axis(1, xtime, xtime)
> lines(sfit3[,2], fun='event',  xmax=48, col=1:2, lwd=2)
> legend(24, .3, c("Arm A", "Arm B"), lty=1, lwd=2,
        col=1:2, bty='n', cex=1.2)
> par(mar=c(4,.1,1,1))
> crisk(c("Entry", "transplant"), alty=2, cex=1.2)
> crisk(c("Entry","transplant", "Death"), cex=1.2)
> par(oldpar)
```

4 Multi-state models

The multi-state models are based on a second data set which looks very much like the (start, stop] data sets that are used for time dependent covariates. Consider subject 5 who experienced CR on day 56, relapse on day 112 and death on day 200. In the expanded data set this subject will have 3 lines, one for each of the intervals (0,56], (56, 112] and (112, 200]. The first interval ends with CR and the second with relapse.

What if someone has two endpoints on the same day? Creation of a zero length interval will lead to a justifiable complaint from the programs; subjects are not allowed to do instantaneous transitions. For each such observation a decision needs to be made, preferably based on rational scientific argument rather than statistical or programming convenience. It turns out that we do have one such case in the myeloid data: one subject who was declared to be a CR on the day of their transplant. Since complete response will occur before its clinical detection I decided to make the tied CR one day earlier. This issue didn't come up in creating `data1` only because it dealt with the pairs CR:death and transplant:death, and neither of these has a tie.

We create the data set using the `tmerge` function in R, code is shown below. (Because such start-stop data sets are commonly used for Cox models with time-dependent covariates, this is a familiar task to many users and they will have developed their own favorite work flow; tmerge is a useful but not essential tool.) The tmerge function uses a baseline data set, in this case the variables from the starting data that are constant over time, and then adds rows to it. Each `event` and `tdc` statement sequentially adds either an endpoint or time-dependent covariate as new rows to the data, in much the same way that one would insert new folders into the proper position in a file drawer. Each addition will split a subject's time interval as necessary.

```
> temp <- myeloid
> id <- which(temp$crtime == temp$txtime) # the one special person
> temp$crtime[id] <- temp$crtime[id] -1   # move their CR back by 1 day
> data2 <- tmerge(myeloid[, c('id', 'trt')], temp,
                  id=id, death=event(futime, death),
                         transplant = event(txtime),
                         response   = event(crtime),
```

```
                        relapse    = event(rltime),
                        priortx    = tdc(txtime),
                        priorcr    = tdc(crtime))
> attr(data2, "tcount")
           early late gap within boundary leading trailing
death          0    0   0      0        0       0      646
transplant     0    0   0    363        0       0        1
response       0    0   0    454        0       0        0
relapse        0    0   0    226        0       0        0
priortx        0    0   0      0      363       0        1
priorcr        0    0   0      0      454       0        0
           tied
death         0
transplant    0
response      0
relapse       0
priortx       0
priorcr       0
> data2$event <- with(data2, factor(death + 2*response + 3*transplant +
                                4*relapse, 0:4,
                                labels=c("censor", "death", "CR",
                                         "transplant", "relapse")))
> data2[1:10,c(1:4, 11, 9, 10)]
   id trt tstart tstop      event priortx priorcr
1   1   B      0    44         CR       0       0
2   1   B     44   113    relapse       0       1
3   1   B    113   235      death       0       1
4   2   A      0   200 transplant       0       0
5   2   A    200   286      death       1       0
6   3   A      0    38         CR       0       0
7   3   A     38  1983     censor       0       1
8   4   B      0    25         CR       0       0
9   4   B     25   245 transplant       0       1
10  4   B    245  2137     censor       1       1
```

The tmerge call starts by adding death/censoring time, which appears in the 'trailing' column of the tcount table since it defines the right endpoint for each subject, and thus by definition occurs at the trailing end of their interval. Then transplant is added which has 363 within and 1 trailing: there is one subject whose transplant date is also their last follow-up date. Response and relapse times all fall within a prior interval. Looking above at the first 4 subjects in `data2`, the fourth follows the canonical path of CR followed by transplant. Subject 1 relapses after CR, without transplant, and subject 2 has transplant without a CR. A critical step in any multi-state model is to print out some portion of the created data set and *read* it. This data set is key, and any errors will invalidate all the analysis which follows. This step has been abbreviated for the vignette; inspection of only the first 4 subjects is a very small sample.

Rescale the data set from days to months and look at three more summaries.

```
> for (i in c("tstart", "tstop"))
     data2[[i]] <- data2[[i]] *12/365.25  #scale to months
> ctab <- table(table(data2$id))
> ctab
  1   2   3   4
 84 216 211 135
> with(data2, table( table(id, event)))
   0    1
1541 1689
> etab <- table(data2$event, useNA="ifany")
> etab
    censor      death         CR transplant    relapse
       325        320        454        364        226
```

In the final result there are 84 subjects with only a single row of data, 216 with 2 rows, etc. The table of `id` by `event` contains only 0 and 1 as values, i.e., no one has two events of the same type, which is correct for this data set. Overall 454 of the 646 subjects experience a CR at some point in the study.

Complete response is a goal of the initial therapy; figure 4 looks more closely at this. As was noted before arm B has an increased number of late responses. The duration of response is also increased: the solid curves show the number of subjects still in response, and we see that they spread farther apart than the dotted "ever in response" curves. The figure shows only the first eight months in order to better visualize the details, but continuing the curves out to 48 months reveals a similar pattern. Here is the code to create the figure.

```
> crstat <- data2$event
> crstat[crstat=="transplant"] <- "censor" # ignore transplants
> crsurv <- survfit(Surv(tstart, tstop, crstat) ~ trt,
                    data= data2, id=id, influence=TRUE)
> layout(matrix(c(1,1,2,3), 2,2), widths=2:1)
> oldpar <- par(mar=c(5.1, 4.1, 1.1, .1))
> plot(sfit2[,2], lty=3, lwd=2, col=1:2, xmax=12,
      xlab="Months", ylab="CR")
> lines(crsurv[,2], lty=1, lwd=2, col=1:2, xmax=12)
> par(mar=c(4, .1, 1, 1))
> crisk( c("Entry","CR", "Death"), alty=3)
> state3(c("Entry", "CR", "Death/Relapse"))
> par(oldpar)
```

Rather than create a new data set, the above code modifies the event variable so as to ignore transitions to the transplant state. They become a non-event, in the same way that extra lines with a status of zero are used to create time-dependent covariates for a Cox model fit.

The `survfit` call above included the `influence=TRUE` argument, which causes the influence array to be calculated and returned. It contains, for each subject, that subject's influence on the time by state matrix of results and allows for calcuation of the standard error of the restricted mean. We will return to this in a later section.

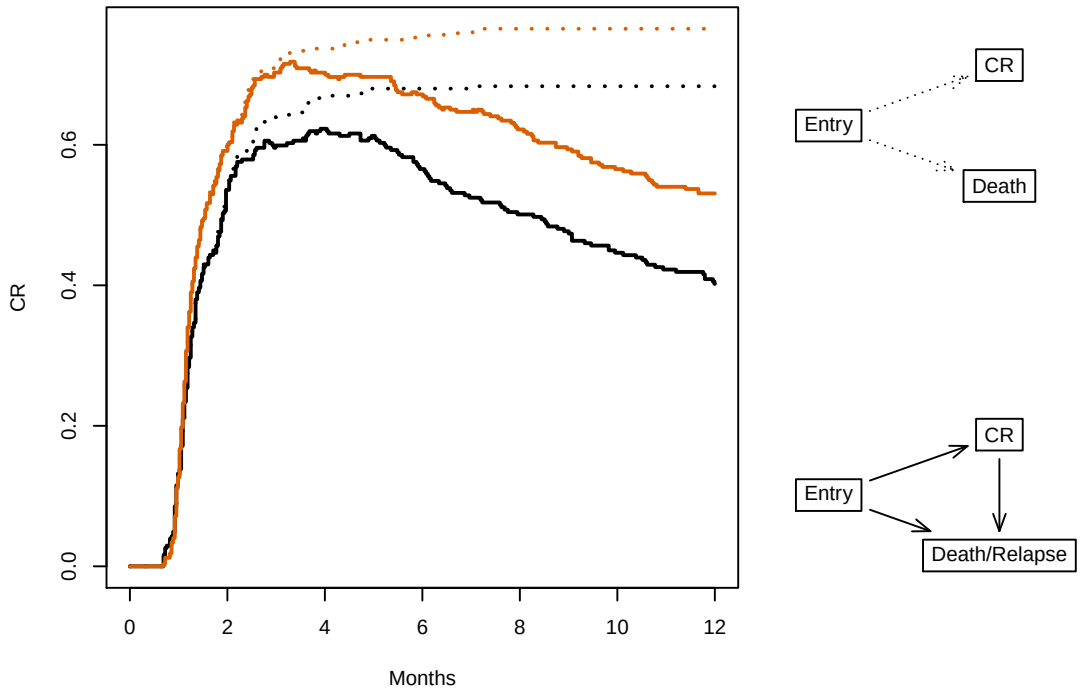


Figure 4: Models for 'ever in CR' and 'currently in CR'; the only difference is an addional transition. Both models ignore transplant.

```
> print(crsurv, rmean=48, digits=2)
Call: survfit(formula = Surv(tstart, tstop, crstat) ~ trt, data = data2,
    id = id, influence = TRUE)

                     n nevent rmean std(rmean)*
trt=A, death       807    171  20.2        1.10
trt=B, death       882    149  15.6        1.01
trt=A, CR          807    206  16.3        1.13
trt=B, CR          882    248  21.2        1.12
trt=A, transplant 807      0   0.0        0.00
trt=B, transplant 882      0   0.0        0.00
trt=A, relapse     807    109   4.3        0.56
trt=B, relapse     882    117   5.5        0.61
trt=A,             807      0   7.1        0.78
trt=B,             882      0   5.6        0.65
   *mean time in state, restricted (max time = 48 )
```

The restricted mean time in the CR state is extended by 21.2 - 16.3 = 4.89 months. A question which immediately gets asked is whether this difference is "significant", to which there are two answers. The first and more important is to ask whether 5 months is an important gain from either a clinical or patient perspective. The overall restricted mean survival for the study is approximately 30 of the first 48 months post entry (use print(sfit1, rmean=48)); on this backdrop an extra 5 months in CR might or might not be an meaningful advantage from a patient's point of view. The less important answer is to test whether the apparent gain is sufficiently rare from a mathematical point of view, i.e., "statistical" significance. The standard errors of the two values are 1.1 and 1.1, and since they are based on disjoint subjects the values are independent, leading to a standard error for the difference of $\sqrt{1.1^2 + 1.2^2} = 1.6$. The difference is over 3 standard errors.

In summary

- Arm B adds late complete responses (about 4%); there are 212/310 in arm B vs. 244/338 in arm B.
- The difference in 4 year survival is about 6%.
- There is approximately 2 months longer average duration of CR (of 48).

CR $\rightarrow$ transplant is the target treatment path for a patient; given the improvements listed above why does figure 2 show no change in the number transplanted? Figure 5 shows the transplants broken down by whether this happened before or after complete response. Most of the non-CR transplants happen by 10 months. One possible explanation is that once it is apparent to the patient/physician pair that CR is not going to occur, they proceed forward with other treatment options. The extra CR events on arm B, which occur between 2 and 8 months, lead to a consequent increase in transplant as well, but at a later time of 12–24 months: for a subject in CR we can perhaps afford to defer the tranplant date.

Computation is again based on a manipulation of the event variable: in this case dividing the transplant state into two sub-states based on the presence of a prior CR. The code makes use

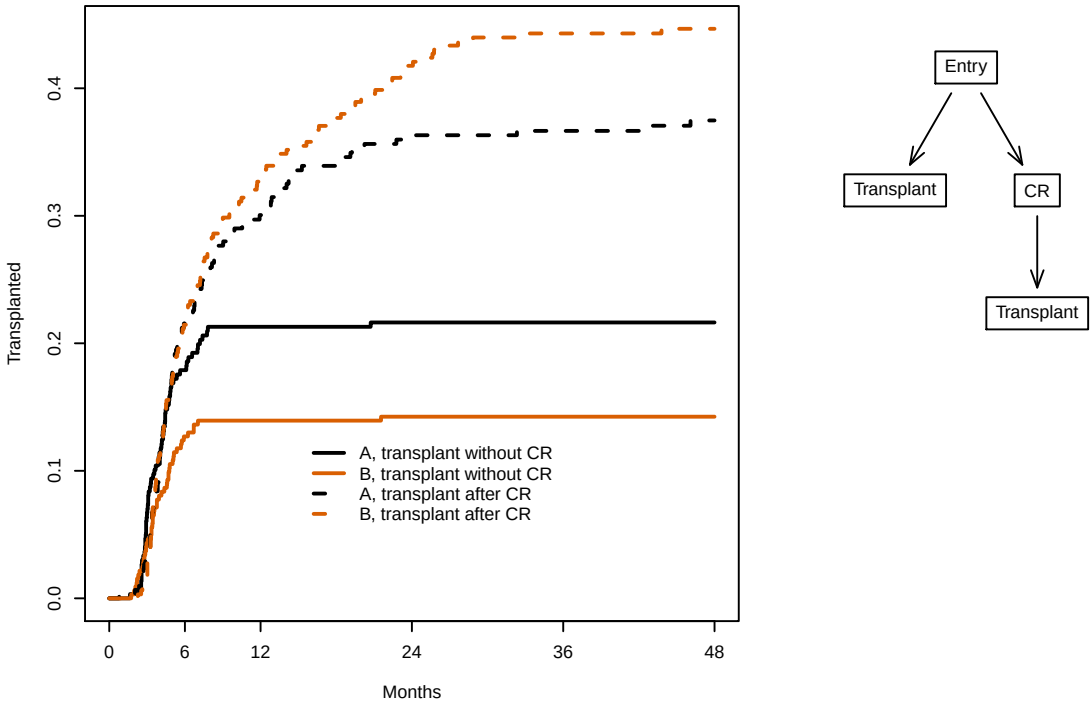


Figure 5: Transplant status of the subjects, broken down by whether it occured before or after CR.

of the time-dependent covariate priorcr. (Because of scheduling constraints within a hospital it is unlikely that a CR that is within a few days prior to transplant could have effected the decision to schedule a transplant, however. An alternate breakdown that might be useful would be "transplant without CR or within 7 days after CR" versus those that are more than a week later. There are many sensible questions that can be asked.)

```
> event2 <- with(data2, ifelse(event=="transplant" & priorcr==1, 6,
                 as.numeric(event)))
> event2 <- factor(event2, 1:6, c(levels(data2$event), "tx after CR"))
> txsurv <- survfit(Surv(tstart, tstop, event2) ~ trt, data2, id=id,
                    subset=(priortx ==0))
> layout(matrix(c(1,1,1,2,2,0),3,2), widths=2:1)
> oldpar <- par(mar=c(5.1, 4.1, 1,.1))
> plot(txsurv[,c(3,5)], col=1:2, lty=c(1,1,2,2), lwd=2, xmax=48,
      xaxt='n', xlab="Months", ylab="Transplanted")
> axis(1, xtime, xtime)
> legend(15, .13, c("A, transplant without CR", "B, transplant without CR",
                  "A, transplant after CR", "B, transplant after CR"),
        col=1:2, lty=c(1,1,2,2), lwd=2, bty='n')
> state4()  # add the state figure
> par(oldpar)
```

Figure 6 shows the full set of state occupancy probabilities for the cohort over the first 4 years. At each point in time the curves estimate the fraction of subjects currently in that state. The total who are in the transplant state peaks at about 9 months and then decreases as subjects relapse or die; the curve rises whenever someone receives a transplant and goes down whenever someone leaves the state. At 36 months treatment arm B (dashed) has a lower fraction who have died, the survivors are about evenly split between those who have received a transplant and those whose last state is a complete response (only a few of the latter are post transplant). The fraction currently in relapse – a transient state – is about 5% for each arm. The figure omits the curve for "still in the entry state". The reason is that at any point in time the sum of the 5 possible states is 1 — everyone has to be somewhere. Thus one of the curves is redundant, and the fraction still in the entry state is the least interesting of them. (A multi-state survfit call that does not include the istate argument will assume that everyone starts in an unnamed entry state. The default plot behavior is to omit the curves for any unnamed states.)

```
> sfit4 <- survfit(Surv(tstart, tstop, event) ~ trt, data2, id=id)
> sfit4$transitions
           to
from         death  CR transplant relapse
  death          0   0          0       0
  CR            17   0        159     168
  transplant   149  11          0      45
  relapse       99   0         99       0
                55 443        106      13
> layout(matrix(1:2,1,2), widths=2:1)
> oldpar <- par(mar=c(5.1, 4.1, 1,.1))
```

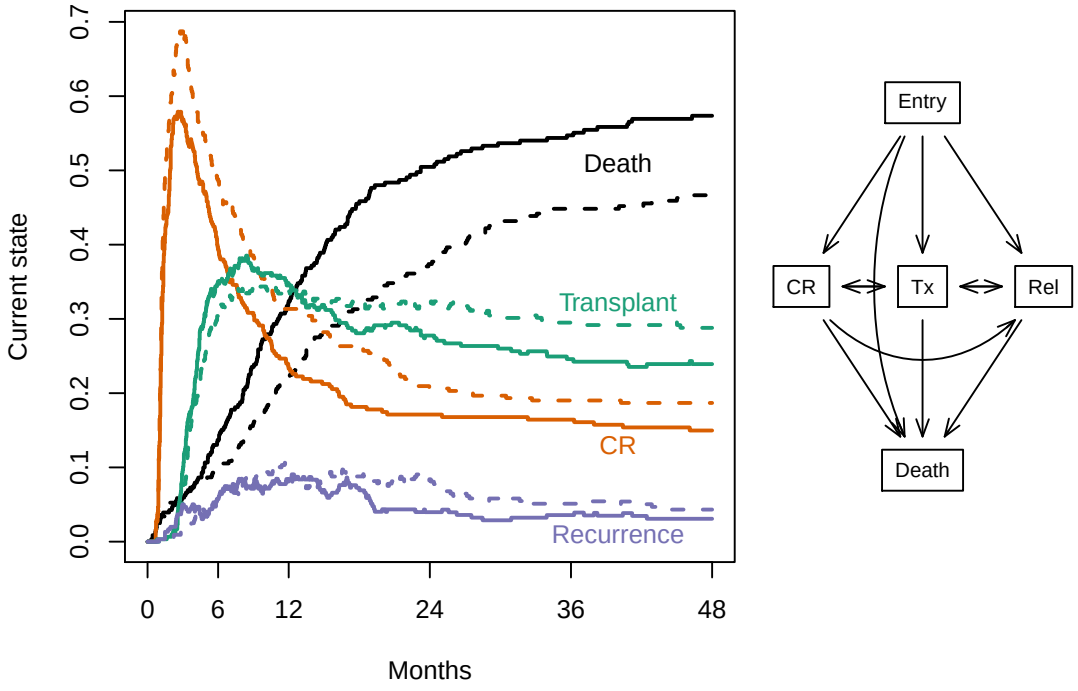


Figure 6: The full multi-state curves for the two treatment arms.

```
> plot(sfit4, col=rep(1:4,each=2), lwd=2, lty=1:2, xmax=48, xaxt='n',
      xlab="Months", ylab="Current state")
> axis(1, xtime, xtime)
> text(c(40, 40, 40, 40), c(.51, .13, .32, .01),
      c("Death", "CR", "Transplant", "Recurrence"), col=1:4)
> par(mar=c(5.1, .1, 1, .1))
> state5()
> par(oldpar)
```

The transitions table above shows 55 direct transitions from entry to death, i.e., subjects who die without experiencing any of the other intermediate points, 159 who go from CR to transplant (as expected), 11 who go from transplant to CR, etc. No one was observed to go from relapse to CR in the data set, this serves as a data check since it should not be possible per the data entry plan.

5 Influence matrix

For one of the curves above we returned the influence array. For each value in the matrix $P =$ probability in state and each subject i in the data set, this contains the effect of that subject on each value in P. Formally,

$$I_{ij}(t) = \left.\frac{\partial p_j(t)}{\partial w_i}\right|_w$$

where $I_{ij}(t)$ is the influence of subject i on $p_j(t)$, and $p_j(t)$ is the estimated probability for state j at time t. This is known as the infinitesimal jackknife (among other labels).

```
> crsurv <- survfit(Surv(tstart, tstop, crstat) ~ trt,
                    data= data2, id=id, influence=TRUE)
> curveA <- crsurv[1,]  # select treatment A
> dim(curveA$pstate)    # P matrix for treatement A
[1] 473   5
> dim(curveA$influence) # influence matrix for treatment A
[1] 317 474   5
> table(data1$trt)
  A   B
317 329
> curveA$p0             # state distribution at time 0
     death         CR transplant    relapse
         0          0          0          0          1
```

For treatment arm A there are 317 subjects and 473 time points in the P matrix. The influence array has subject as the first dimension, and for each subject it has an image of the P matrix containing that subject's influence on each value in P, i.e., `influence[1■]` is the influence of subject 1 on P. The influence has one extra row, however; the first row for each subjects is the influence of that subject on p_0, the initial state probabilities. For this data set

everyone starts in the entry state, p_0 will always be (0, 0, 0, 1), and so this first influence row will be zero; this does not hold if not all subjects start in the same state.

As an exercise we will calculate the mean time in state out to 48 weeks. This is the area under the individual curves from time 0 to 48. Since the curves are step functions this is simple sum of rectangles, treating any intervals after 48 months as having 0 width.

```
> t48 <- pmin(48, curveA$time)
> delta <- diff(c(0, t48, 48))  # width of intervals
> rfun <- function(pmat, delta) colSums(pmat * delta)  #area under the curve
> rmean <- rfun(rbind(curveA$p0, curveA$pstate), delta)
> round(rmean, 2)
     death         CR transplant    relapse
     20.24      16.34       0.00       4.31       7.10
> # Apply the same calculation to each subject's influence slice
> inf <- apply(curveA$influence, 1, rfun, delta=delta)
> # inf is now a 5 state by 310 subject matrix, containing the IJ estimates
> #  on the AUC or mean time.  The sum of squares is a variance.
> se.rmean <- sqrt(rowSums(inf^2))
> round(se.rmean, 2)
[1] 1.10 1.13 0.00 0.56 0.78
```

In general, let U_i be the influence of subject i. For some function $f(P)$ of the prevalence matrix, the influence of subject i will be $\delta_i = f(P + U_i) - f(P)$ and the infinitesimal jackknife estimate of variance will be $\sum_i \delta^2$. For the simple case of adding up rectangles $f(P+U_i)-f(P) = f(U_i)$ leading to particularly simple code, but this will not always be the case.

6 State space figures

The state space figures in this document were drawn with a simple utility function `statefig`. It has two primary arguments along with standard graphical options of color, line type, etc.

1. A layout vector or matrix. A vector with values of (1, 3, 1) for instance will allocate one state, then a column with 3 states, then one more state, proceeding from left to right. A matrix with a single row will do the same, whereas a matrix with one column will proceed from top to bottom.

2. A k by k connection matrix C where k is the number of states. If $C_{ij} \neq 0$ then an arrow is drawn from state i to state j. The row or column names of the matrix are used to label the states. The lines connecting the states can be straight or curved, see the help file for the function for an example.

The first few state space diagrams were competing risk models, which use the following derived function. It accepts a vector of state names, where the first name is the starting state and the remainder are the possible outcomes.

```
> crisk <- function(what, horizontal = TRUE, ...) {
     nstate <- length(what)
     connect <- matrix(0, nstate, nstate,
                       dimnames=list(what, what))
     connect[1,-1] <- 1  # an arrow from state 1 to each of the others
     if (horizontal) statefig(c(1, nstate-1),  connect, ...)
     else statefig(matrix(c(1, nstate-1), ncol=1), connect, ...)
 }
```

This next function draws a variation of the illness-death model. It has an initial state, an absorbing state (normally death), and an optional intermediate state.

```
> state3 <- function(what, horizontal=TRUE, ...) {
     if (length(what) != 3) stop("Should be 3 states")
     connect <- matrix(c(0,0,0, 1,0,0, 1,1,0), 3,3,
                       dimnames=list(what, what))
     if (horizontal) statefig(1:2, connect, ...)
     else statefig(matrix(1:2, ncol=1), connect, ...)
 }
```

The most complex of the state space figures has all 5 states.

```
> state5 <- function(what, ...) {
     sname <- c("Entry", "CR", "Tx", "Rel", "Death")
     connect <- matrix(0, 5, 5, dimnames=list(sname, sname))
     connect[1, -1] <- c(1,1,1, 1.4)
     connect[2, 3:5] <- c(1, 1.4, 1)
     connect[3, c(2,4,5)] <- 1
     connect[4, c(3,5)]  <- 1
     statefig(matrix(c(1,3,1)), connect, cex=.8, ...)
 }
```

For figure 5 I want a third row with a single state, but don't want that state centered. For this I need to create my own (x,y) coordinate list as the layout parameter. Coordinates must be between 0 and 1.

```
> state4 <- function() {
     sname <- c("Entry", "CR", "Transplant", "Transplant")
     layout <- cbind(x =c(1/2, 3/4, 1/4, 3/4),
                     y =c(5/6, 1/2, 1/2, 1/6))
     connect <- matrix(0,4,4, dimnames=list(sname, sname))
     connect[1, 2:3] <- 1
     connect[2,4] <- 1
     statefig(layout, connect)
 }
```

The statefig function was written to do "good enough" state space figures quickly and easily, in the hope that users will find it simple enough that diagrams are drawn early and often. Other packages such as diagram, DiagrammeR, or dagR are far more flexible and can create more nuanced and well decorated results.

7 Conclusion

With a data set such as this we can fit many different multi-state models. These fits are easy to do, and can give substantial further insight into a data set.