Multistate models in Epi: The DLI example from JSS: the etm package by Allignol *et al.*

SDCC
http://BendixCarstensen.com/AdvCoh/MultiState/Papers/
February 2018
Version 3

Compiled Monday 5th February, 2018, 11:11
from: /home/bendix/stat/R/lib.src/Epi/pkg/vignettes/etm.tex

Bendix Carstensen Steno Diabetes Center, Gentofte, Denmark
& Department of Biostatistics, University of Copenhagen
bxc@steno.dk
http://BendixCarstensen.com

Contents

1 Introduction

This is an illustration of how to set up the multistate data from `dli.data` used as illustration in [1]. In this note I use the `Lexis` machinery from the the `Epi` package[3, 2], which makes things more transparent, because it requires an initial specification of the multistate *data* in terms of transitions and timescales. The latter is largely bypassed as an issue in [1] as only one time scale is used. The `Lexis` machinery also includes a couple of summary and graphical functions that facilitate the overview of the data.

We have downloaded the data, and we start by loading the `Epi` package and the data:

```
> options( width=90 )
> library( Epi )
> library( etm )
> library( survival )
> library( splines )
> print( sessionInfo(), l=F )
R version 3.4.3 (2017-11-30)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 14.04.5 LTS

Matrix products: default
BLAS: /usr/lib/openblas-base/libopenblas.so.0
LAPACK: /usr/lib/lapack/liblapack.so.3.0

attached base packages:
[1] splines   utils     datasets  graphics  grDevices stats     methods   base

other attached packages:
[1] etm_0.6-2      survival_2.41-3 Epi_2.24

loaded via a namespace (and not attached):
 [1] cmprsk_2.2-7      zoo_1.8-0         MASS_7.3-48       compiler_3.4.3
 [5] Matrix_1.2-11     plyr_1.8.4        parallel_3.4.3    Rcpp_0.12.12
 [9] grid_3.4.3        numDeriv_2016.8-1 lattice_0.20-35

> load( url("https://www.jstatsoft.org/index.php/jss/article/downloadSuppFile/v038i04/dli.data.rda")
> # save( dli.data, file="./dli.Rda" )
> # load(           file="./dli.Rda" )
> str( dli.data )

'data.frame':	536 obs. of  5 variables:
 $ id      : int  2 3 4 5 6 11 12 13 15 16 ...
 $ from    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ to      : chr  "cens" "cens" "cens" "cens" ...
 $ time    : num  19.9 15.2 21 16 21.1 ...
 $ distatpr: int  1 1 1 2 1 1 1 1 1 1 ...
```

In order to understand how data is coded we list data for a few select persons:

```
> subset( dli.data, id %in% c(2,5,388,511,531,600) )
     id from   to       time distatpr
1     2    0 cens 19.8841843        1
4     5    0 cens 16.0316320        2
202 388    0    2  2.8467381        1
203 388    2 cens  3.8754478        1
328 511    0    2  0.3127550        1
329 511    2    4  0.6354438        1
330 511    4 cens  1.3810754        1
355 531    0    2  0.4244171        3
356 531    2    4  0.4117015        3
```

```
357 531    4    5  0.5622658        3
505 600    0    2  0.4632695        1
506 600    2    4  1.0560224        1
507 600    4    6  1.3696801        1
508 600    6 cens  3.0239852        1
```

The time records from persons 531 is clearly flawed because exit from state 2 occurs before entry to it. There is only this one instance in the dataset:

```
> subset( dli.data, c(TRUE,diff(time)<0 & diff(id)==0 ) )
     id from   to       time distatpr
1     2    0 cens 19.8841843        1
356 531    2    4  0.4117015        3
```

So we doctor it (more or less arbitrarily):

```
> dli.data[dli.data$id==531 & dli.data$from==2,"time"] <- 0.45
```

We can now inspect how the transitions occur between states:

```
> with( dli.data, table( from, to ) )
    to
from   1   2   3   4   5   6   7   8 cens
   0 100 123   0   0   0   0   0   0   81
   2   0   0  40  65   0   0   0   0   18
   4   0   0   0   0  10  44   0   0   11
   6   0   0   0   0   0   0   2   4   38
```

2 Setting up a Lexis object

In order to be able to put the data into a `Lexis` object, we must know the entry times for each state, not only the exit times. This is done by using the `ave` function, that allows operation within each level of a grouping (in this case `id`). Moreover, we also change the censoring value `"cens"` to the value of the state in which the censoring occurs (`as.numeric("cens")` returns `NA`).

```
> dli <- transform( dli.data,
+                     to = as.numeric(to),
+                     ti = ave( time, id, FUN=function( x ) c(0,x[-length(x)]) ) )
> dli$to <- ifelse( is.na(dli$to), dli$from, dli$to )
> subset( dli, id %in% c(5,388,511,600) )
     id from to       time distatpr        ti
4     5    0  0 16.0316320        2 0.0000000
202 388    0  2  2.8467381        1 0.0000000
203 388    2  2  3.8754478        1 2.8467381
328 511    0  2  0.3127550        1 0.0000000
329 511    2  4  0.6354438        1 0.3127550
330 511    4  4  1.3810754        1 0.6354438
505 600    0  2  0.4632695        1 0.0000000
506 600    2  4  1.0560224        1 0.4632695
507 600    4  6  1.3696801        1 1.0560224
508 600    6  6  3.0239852        1 1.3696801
> with( dli, table( from, to ) )
    to
from   0   1   2   3   4   5   6   7   8
   0  81 100 123   0   0   0   0   0   0
   2   0   0  18  40  65   0   0   0   0
   4   0   0   0   0  11  10  44   0   0
   6   0   0   0   0   0   0  38   2   4
```

2.1 Other timescales

Apart from the underlying timescale, time from inclusion in the study (*i.e.* treatment to remission), it may be of interest to be able to assess the effect of other timescales, such as current age and calendar time. These are not available in the data set and might even not be relevant depending on the range of these in data.

However it would be most interesting to be able to use the time since entry into each of the three intermediate states. So for the state 2 (1st relapse) we can construct the time of entry into the state and subsequently the time *since* entry for all other pieces of follow-up:

```
> tmp <- subset(dli,to==2 & from!=to)[,c("id","time")]
> names(tmp)[2] <- "tr"
> dli <- merge( dli, tmp, all.x=TRUE )
> dli$tr <- with( dli, ifelse( ti-tr>=0, ti-tr, NA ) )
> subset(dli,id %in% c(600,603,608) )
     id from to      time distatpr        ti        tr
505 600    0  2 0.4632695        1 0.0000000        NA
506 600    2  4 1.0560224        1 0.4632695 0.0000000
507 600    4  6 1.3696801        1 1.0560224 0.5927529
508 600    6  6 3.0239852        1 1.3696801 0.9064107
513 603    0  2 1.9036507        1 0.0000000        NA
514 603    2  4 2.7898033        1 1.9036507 0.0000000
515 603    4  6 3.9490625        1 2.7898033 0.8861527
516 603    6  7 8.8798961        1 3.9490625 2.0454118
525 608    0  2 3.8078026        1 0.0000000        NA
526 608    2  4 4.7707204        1 3.8078026 0.0000000
527 608    4  6 5.4684705        1 4.7707204 0.9629178
528 608    6  8 5.7731360        1 5.4684705 1.6606679
> str( dli )
'data.frame':        536 obs. of  7 variables:
 $ id      : int  2 3 4 5 6 11 12 13 15 16 ...
 $ from    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ to      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ time    : num  19.9 15.2 21 16 21.1 ...
 $ distatpr: int  1 1 1 2 1 1 1 1 1 1 ...
 $ ti      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ tr      : num  NA NA NA NA NA NA NA NA NA NA ...
```

The same is now repeated for the other two intermediate states:

```
> # DLI
> tmp <- subset(dli,to==4 & from!=to)[,c("id","time")]
> names(tmp)[2] <- "tD"
> dli <- merge( dli, tmp, all.x=TRUE )
> dli$tD <- with( dli, ifelse( ti-tD>=0, ti-tD, NA ) )
> # Rm2
> tmp <- subset(dli,to==6 & from!=to)[,c("id","time")]
> names(tmp)[2] <- "tR"
> dli <- merge( dli, tmp, all.x=TRUE )
> dli$tR <- with( dli, ifelse( ti-tR>=0, ti-tR, NA ) )
> subset(dli,id %in% c(600,603,608) )
     id from to      time distatpr        ti        tr        tD tR
505 600    0  2 0.4632695        1 0.0000000        NA        NA NA
506 600    2  4 1.0560224        1 0.4632695 0.0000000        NA NA
507 600    4  6 1.3696801        1 1.0560224 0.5927529 0.0000000 NA
508 600    6  6 3.0239852        1 1.3696801 0.9064107 0.3136578  0
513 603    0  2 1.9036507        1 0.0000000        NA        NA NA
514 603    2  4 2.7898033        1 1.9036507 0.0000000        NA NA
515 603    4  6 3.9490625        1 2.7898033 0.8861527 0.0000000 NA
```

```
516 603    6  7 8.8798961        1 3.9490625 2.0454118 1.1592591  0
525 608    0  2 3.8078026        1 0.0000000        NA        NA NA
526 608    2  4 4.7707204        1 3.8078026 0.0000000        NA NA
527 608    4  6 5.4684705        1 4.7707204 0.9629178 0.0000000 NA
528 608    6  8 5.7731360        1 5.4684705 1.6606679 0.6977501  0
```

2.2 Defining the Lexis object

Now we can define a Lexis object, using the factor facility in R to label the states:

```
> state.names <- c("Rem" , "D/Rem",
+                  "Rel" , "D/Rel",
+                  "DLI" , "D/DLI",
+                  "Rem2", "D/Rem2",
+                  "Rel2")
> dli <- Lexis( entry        = list( tfi=ti, tfr=tr, tfD=tD, tfR=tR ),
+               entry.status = factor( from, levels=0:8, labels=state.names ),
+                exit        = list( tfi=time ),
+                exit.status = factor(   to, levels=0:8, labels=state.names ),
+                         id = id,
+                       data = dli )
> print.data.frame(
+ subset( dli, id %in% c(600,603,608) )[,1:13], digits=3)
      tfi   tfr   tfD tfR lex.dur lex.Cst lex.Xst lex.id  id from to  time distatpr
505 0.000    NA    NA  NA   0.463     Rem     Rel    600 600    0  2 0.463        1
506 0.463 0.000    NA  NA   0.593     Rel     DLI    600 600    2  4 1.056        1
507 1.056 0.593 0.000  NA   0.314     DLI    Rem2    600 600    4  6 1.370        1
508 1.370 0.906 0.314   0   1.654    Rem2    Rem2    600 600    6  6 3.024        1
513 0.000    NA    NA  NA   1.904     Rem     Rel    603 603    0  2 1.904        1
514 1.904 0.000    NA  NA   0.886     Rel     DLI    603 603    2  4 2.790        1
515 2.790 0.886 0.000  NA   1.159     DLI    Rem2    603 603    4  6 3.949        1
516 3.949 2.045 1.159   0   4.931    Rem2  D/Rem2    603 603    6  7 8.880        1
525 0.000    NA    NA  NA   3.808     Rem     Rel    608 608    0  2 3.808        1
526 3.808 0.000    NA  NA   0.963     Rel     DLI    608 608    2  4 4.771        1
527 4.771 0.963 0.000  NA   0.698     DLI    Rem2    608 608    4  6 5.468        1
528 5.468 1.661 0.698   0   0.305    Rem2    Rel2    608 608    6  8 5.773        1
```

Setting up a Lexis object for this dataset basically means replacing the time of event for each transition by the time of entry into a state (tfi) and the sojourn time (lex.dur) in the state, and making clear that with this convention from (or lex.Cst) is the state in which the person spends lex.dur, and to (or lex.Xst) is the state to which the persons moves (using the convention that if from=to we have a censored observation.

2.3 Summarizing the Lexis object

Using the summary facility for Lexis objects shows the transitions and the risk time in each state:

```
> summary( dli )
Transitions:
     To
From    Rem D/Rem Rel D/Rel DLI D/DLI Rem2 D/Rem2 Rel2  Records:  Events: Risk time:
  Rem    81   100 123     0   0     0    0      0    0       304      223    1042.59
  Rel     0     0  18    40  65     0    0      0    0       123      105     246.98
  DLI     0     0   0     0  11    10   44      0    0        65       54      92.18
  Rem2    0     0   0     0   0     0   38      2    4        44        6     189.38
```

```
  Sum    81    100  141     40   76     10    82       2     4        536       388      1571.13

Transitions:
     To
From     Persons:
  Rem          304
  Rel          123
  DLI           65
  Rem2          44
  Sum          304
```

There is also a facility to show the states in a graph with person-years (risk time in the units given) shown in the boxes and no. of transitions shown between them:

```
> boxes( dli )
```

This is an interactive facility where you are asked to click on the plot where the boxes should go; but you can also explicitly supply the positions of the centers of the boxes. For later use we first define the number of states, and a set of colors to be used for the states in this and subsequent plots. We also scale the printed transition rates between states by 100, so they will appear as percent per year (formally cases per 100 PY). Finally we put a black frame around the dead states.

```
> n.st <- nlevels( dli$lex.Cst )
> st.col <- rainbow( n.st )
> # A slightly more transparent color
> # substr(st.col,8,9) <- "BB"
> boxes( dli, wmult=1.1, hmult=1.2,
+             boxpos=list(x=c(10,30,30,50,50,70,70,90,90),
+                         y=c(25, 8,42,25,59,42,76,59,93)),
+             scale.R=100, show.BE=TRUE,
+             col.bg=st.col, col.txt="white",
+             col.border=c("white","black")[c(1,2,1,2,1,2,1,2,1)] )
```

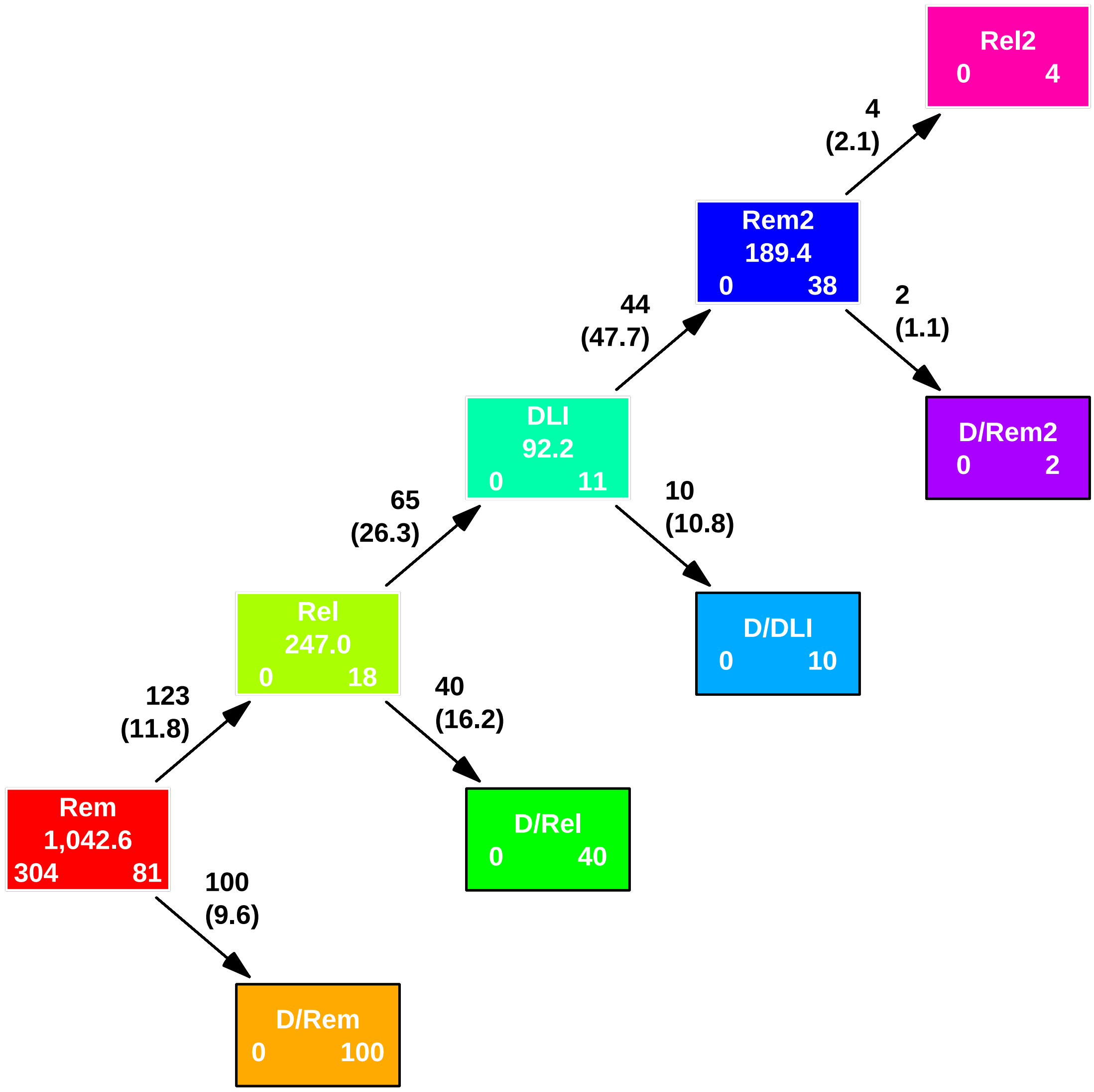

Figure 1: *Display of the actually occurring transitions and -rates (in %/year) and the corresponding risk time in each state. In each state is also shown the number of persons at start and end of the study in each state. Note that the two rightmost boxes are interchanged relative to the figure 4 in [1] in order to have all mortality rates point South-East.*

3 The mortality rates

Once the data is set up in a `Lexis` object is quite easy to model the mortality rates. If we want to fit a joint model for all transition rates, we need a stacked dataset; that is a dataset with person years and events for each of the transitions. This means (for this model) that in the stacked dataset all person-years are represented twice, one time for each transition (since all transient states have two transitions out of them). This is accomplished by the `stack.Lexis` function, and since `dli` has class `Lexis`, we can just do:

```
> st.dli <- stack( dli )
> str( st.dli )
Classes ‘stacked.Lexis’ and 'data.frame':       1072 obs. of  19 variables:
 $ tfi     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ tfr     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ tfD     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ tfR     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ lex.dur : num  19.9 15.2 21 16 21.1 ...
 $ lex.Cst : Factor w/ 9 levels "Rem","D/Rem",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Xst : Factor w/ 9 levels "Rem","D/Rem",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Tr  : Factor w/ 8 levels "Rem->D/Rem","Rem->Rel",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Fail: logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ lex.id  : int  2 3 4 5 6 11 12 13 15 16 ...
 $ id      : int  2 3 4 5 6 11 12 13 15 16 ...
 $ from    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ to      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ time    : num  19.9 15.2 21 16 21.1 ...
 $ distatpr: int  1 1 1 2 1 1 1 1 1 1 ...
 $ ti      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ tr      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ tD      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ tR      : num  NA NA NA NA NA NA NA NA NA NA ...
 - attr(*, "breaks")=List of 4
  ..$ tfi: NULL
  ..$ tfr: NULL
  ..$ tfD: NULL
  ..$ tfR: NULL
 - attr(*, "time.scales")= chr  "tfi" "tfr" "tfD" "tfR"
> round(
+ cbind( "Original"=with(    dli, tapply( lex.dur, lex.Cst, sum ) ),
+        "Stacked" =with( st.dli, tapply( lex.dur, lex.Cst, sum ) ) ), 1 )

       Original Stacked
Rem      1042.6  2085.2
D/Rem        NA      NA
Rel       247.0   494.0
D/Rel        NA      NA
DLI        92.2   184.4
D/DLI        NA      NA
Rem2      189.4   378.8
D/Rem2       NA      NA
Rel2         NA      NA

> round( xtabs( cbind( lex.Fail, lex.dur ) ~ lex.Tr, data = st.dli ), 1 )

lex.Tr         lex.Fail lex.dur
  Rem->D/Rem      100.0  1042.6
  Rem->Rel        123.0  1042.6
  Rel->D/Rel       40.0   247.0
  Rel->DLI         65.0   247.0
  DLI->D/DLI       10.0    92.2
  DLI->Rem2        44.0    92.2
  Rem2->D/Rem2      2.0   189.4
  Rem2->Rel2        4.0   189.4
```

For the analysis of the deaths we need only the transitions 1,3,5 and 7:

```
> dd.dli <- subset( st.dli, lex.Tr %in% levels(lex.Tr)[c(1,3,5,7)] )
> table( dd.dli$lex.Tr )
  Rem->D/Rem      Rem->Rel    Rel->D/Rel      Rel->DLI    DLI->D/DLI      DLI->Rem2
         304             0           123             0            65              0
Rem2->D/Rem2    Rem2->Rel2
          44             0
```

In order not to mess up the reporting etc. we remove the non-existent levels of the factor `lex.Tr`:

```
> dd.dli$lex.Tr <- factor( dd.dli$lex.Tr )
> round( xtabs( cbind( lex.Fail, lex.dur ) ~ lex.Tr, data=dd.dli ), 1 )
lex.Tr         lex.Fail lex.dur
  Rem->D/Rem      100.0  1042.6
  Rel->D/Rel       40.0   247.0
  DLI->D/DLI       10.0    92.2
  Rem2->D/Rem2      2.0   189.4
```

Now we can do a Cox-analysis using time as the underlying timescale, and assuming that the mortality rates are proportional on this scale:

```
> str( dd.dli )
Classes 'stacked.Lexis' and 'data.frame':       536 obs. of  19 variables:
 $ tfi     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ tfr     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ tfD     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ tfR     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ lex.dur : num  19.9 15.2 21 16 21.1 ...
 $ lex.Cst : Factor w/ 9 levels "Rem","D/Rem",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Xst : Factor w/ 9 levels "Rem","D/Rem",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Tr  : Factor w/ 4 levels "Rem->D/Rem","Rel->D/Rel",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Fail: logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ lex.id  : int  2 3 4 5 6 11 12 13 15 16 ...
 $ id      : int  2 3 4 5 6 11 12 13 15 16 ...
 $ from    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ to      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ time    : num  19.9 15.2 21 16 21.1 ...
 $ distatpr: int  1 1 1 2 1 1 1 1 1 1 ...
 $ ti      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ tr      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ tD      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ tR      : num  NA NA NA NA NA NA NA NA NA NA ...
 - attr(*, "breaks")=List of 4
  ..$ tfi: NULL
  ..$ tfr: NULL
  ..$ tfD: NULL
  ..$ tfR: NULL
 - attr(*, "time.scales")= chr  "tfi" "tfr" "tfD" "tfR"
> c0 <- coxph( Surv(tfi,tfi+lex.dur,lex.Fail) ~ lex.Tr, data=dd.dli )
> summary( c0 )
Call:
coxph(formula = Surv(tfi, tfi + lex.dur, lex.Fail) ~ lex.Tr,
    data = dd.dli)

  n= 536, number of events= 152

                      coef exp(coef) se(coef)     z Pr(>|z|)
lex.TrRel->D/Rel    1.2396    3.4544   0.2103 5.895 3.75e-09
```

```
lex.TrDLI->D/DLI    1.1652    3.2064   0.3604  3.233  0.00123
lex.TrRem2->D/Rem2 -0.9073    0.4036   0.7320 -1.239  0.21518

                   exp(coef) exp(-coef) lower .95 upper .95
lex.TrRel->D/Rel      3.4544     0.2895   2.28756     5.216
lex.TrDLI->D/DLI      3.2064     0.3119   1.58203     6.499
lex.TrRem2->D/Rem2    0.4036     2.4775   0.09614     1.695

Concordance= 0.573  (se = 0.013 )
Rsquare= 0.073   (max possible= 0.95 )
Likelihood ratio test= 40.41  on 3 df,   p=8.721e-09
Wald test            = 42.01  on 3 df,   p=3.998e-09
Score (logrank) test = 47.96  on 3 df,   p=2.171e-10
```

We see that this is pretty consistent with patients being in remission having a lower mortality than those in relapse, and with no real difference between types of relapse ("Rel" or "DLI"). A formal pooling of levels can be achieved by `Relevel`, and we can the fit the reduced model and compare to the more elaborate:

```
> dd.dli$Rst <- Relevel( dd.dli$lex.Tr, list(Remis=c(1,4),Relapse=2:3) )
> with( dd.dli, table( lex.Tr, Rst ) )
              Rst
lex.Tr          Remis Relapse
  Rem->D/Rem      304       0
  Rel->D/Rel        0     123
  DLI->D/DLI        0      65
  Rem2->D/Rem2     44       0

> c1 <- coxph( Surv(tfi,tfi+lex.dur,lex.Fail) ~ Rst, data=dd.dli )
> summary( c1 )

Call:
coxph(formula = Surv(tfi, tfi + lex.dur, lex.Fail) ~ Rst, data = dd.dli)

  n= 536, number of events= 152

             coef exp(coef) se(coef)     z Pr(>|z|)
RstRelapse 1.2829    3.6072   0.1982 6.473  9.6e-11

           exp(coef) exp(-coef) lower .95 upper .95
RstRelapse     3.607     0.2772     2.446     5.319

Concordance= 0.57  (se = 0.012 )
Rsquare= 0.069   (max possible= 0.95 )
Likelihood ratio test= 38.38  on 1 df,   p=5.837e-10
Wald test            = 41.9  on 1 df,   p=9.604e-11
Score (logrank) test = 46.75  on 1 df,   p=8.051e-12

> anova( c0, c1, test="Chisq" )

Analysis of Deviance Table
 Cox model: response is  Surv(tfi, tfi + lex.dur, lex.Fail)
 Model 1: ~ lex.Tr
 Model 2: ~ Rst
   loglik  Chisq Df P(>|Chi|)
1 -781.39
2 -782.41 2.0352  2    0.3615
```

We now leave the Cox-model and turn to a more natural model for the underlying rates; a model that assumes a hazard smoothly varying by time since initiation.

3.1 Parametric description of rates

If we want to get an overview of how mortality rates actually look as a function of time since entry into the study, we can invoke a Poisson model for time-split data. From the summary on page 5 we see that the total amount of follow-up is about 1500 PY among some 300 patients, so roughly 5 years per person. Hence we split data in pieces of 1/10 year, and should get some 15,000 records:

```
> sp.dli <- splitLexis( dli, breaks=seq(0,50,1/10) )
> print.data.frame( subset( sp.dli, id==603 )[,1:13], digits=3 )
      lex.id  tfi    tfr    tfD    tfR lex.dur lex.Cst lex.Xst  id from to time distatpr
15739    603 0.00     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15740    603 0.10     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15741    603 0.20     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15742    603 0.30     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15743    603 0.40     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15744    603 0.50     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15745    603 0.60     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15746    603 0.70     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15747    603 0.80     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15748    603 0.90     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15749    603 1.00     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15750    603 1.10     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15751    603 1.20     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15752    603 1.30     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15753    603 1.40     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15754    603 1.50     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15755    603 1.60     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15756    603 1.70     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15757    603 1.80     NA     NA     NA 0.10000     Rem     Rem 603    0  2 1.90        1
15758    603 1.90     NA     NA     NA 0.00365     Rem     Rel 603    0  2 1.90        1
15759    603 1.90 0.0000     NA     NA 0.09635     Rel     Rel 603    2  4 2.79        1
15760    603 2.00 0.0963     NA     NA 0.10000     Rel     Rel 603    2  4 2.79        1
15761    603 2.10 0.1963     NA     NA 0.10000     Rel     Rel 603    2  4 2.79        1
15762    603 2.20 0.2963     NA     NA 0.10000     Rel     Rel 603    2  4 2.79        1
15763    603 2.30 0.3963     NA     NA 0.10000     Rel     Rel 603    2  4 2.79        1
15764    603 2.40 0.4963     NA     NA 0.10000     Rel     Rel 603    2  4 2.79        1
15765    603 2.50 0.5963     NA     NA 0.10000     Rel     Rel 603    2  4 2.79        1
15766    603 2.60 0.6963     NA     NA 0.10000     Rel     Rel 603    2  4 2.79        1
15767    603 2.70 0.7963     NA     NA 0.08980     Rel     DLI 603    2  4 2.79        1
15768    603 2.79 0.8862 0.0000     NA 0.01020     DLI     DLI 603    4  6 3.95        1
15769    603 2.80 0.8963 0.0102     NA 0.10000     DLI     DLI 603    4  6 3.95        1
15770    603 2.90 0.9963 0.1102     NA 0.10000     DLI     DLI 603    4  6 3.95        1
15771    603 3.00 1.0963 0.2102     NA 0.10000     DLI     DLI 603    4  6 3.95        1
15772    603 3.10 1.1963 0.3102     NA 0.10000     DLI     DLI 603    4  6 3.95        1
15773    603 3.20 1.2963 0.4102     NA 0.10000     DLI     DLI 603    4  6 3.95        1
15774    603 3.30 1.3963 0.5102     NA 0.10000     DLI     DLI 603    4  6 3.95        1
15775    603 3.40 1.4963 0.6102     NA 0.10000     DLI     DLI 603    4  6 3.95        1
15776    603 3.50 1.5963 0.7102     NA 0.10000     DLI     DLI 603    4  6 3.95        1
15777    603 3.60 1.6963 0.8102     NA 0.10000     DLI     DLI 603    4  6 3.95        1
15778    603 3.70 1.7963 0.9102     NA 0.10000     DLI     DLI 603    4  6 3.95        1
15779    603 3.80 1.8963 1.0102     NA 0.10000     DLI     DLI 603    4  6 3.95        1
15780    603 3.90 1.9963 1.1102     NA 0.04906     DLI    Rem2 603    4  6 3.95        1
15781    603 3.95 2.0454 1.1593 0.0000 0.05094    Rem2    Rem2 603    6  7 8.88        1
15782    603 4.00 2.0963 1.2102 0.0509 0.10000    Rem2    Rem2 603    6  7 8.88        1
15783    603 4.10 2.1963 1.3102 0.1509 0.10000    Rem2    Rem2 603    6  7 8.88        1
15784    603 4.20 2.2963 1.4102 0.2509 0.10000    Rem2    Rem2 603    6  7 8.88        1
15785    603 4.30 2.3963 1.5102 0.3509 0.10000    Rem2    Rem2 603    6  7 8.88        1
15786    603 4.40 2.4963 1.6102 0.4509 0.10000    Rem2    Rem2 603    6  7 8.88        1
15787    603 4.50 2.5963 1.7102 0.5509 0.10000    Rem2    Rem2 603    6  7 8.88        1
15788    603 4.60 2.6963 1.8102 0.6509 0.10000    Rem2    Rem2 603    6  7 8.88        1
15789    603 4.70 2.7963 1.9102 0.7509 0.10000    Rem2    Rem2 603    6  7 8.88        1
```

```
15790    603 4.80 2.8963 2.0102 0.8509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15791    603 4.90 2.9963 2.1102 0.9509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15792    603 5.00 3.0963 2.2102 1.0509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15793    603 5.10 3.1963 2.3102 1.1509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15794    603 5.20 3.2963 2.4102 1.2509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15795    603 5.30 3.3963 2.5102 1.3509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15796    603 5.40 3.4963 2.6102 1.4509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15797    603 5.50 3.5963 2.7102 1.5509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15798    603 5.60 3.6963 2.8102 1.6509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15799    603 5.70 3.7963 2.9102 1.7509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15800    603 5.80 3.8963 3.0102 1.8509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15801    603 5.90 3.9963 3.1102 1.9509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15802    603 6.00 4.0963 3.2102 2.0509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15803    603 6.10 4.1963 3.3102 2.1509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15804    603 6.20 4.2963 3.4102 2.2509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15805    603 6.30 4.3963 3.5102 2.3509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15806    603 6.40 4.4963 3.6102 2.4509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15807    603 6.50 4.5963 3.7102 2.5509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15808    603 6.60 4.6963 3.8102 2.6509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15809    603 6.70 4.7963 3.9102 2.7509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15810    603 6.80 4.8963 4.0102 2.8509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15811    603 6.90 4.9963 4.1102 2.9509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15812    603 7.00 5.0963 4.2102 3.0509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15813    603 7.10 5.1963 4.3102 3.1509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15814    603 7.20 5.2963 4.4102 3.2509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15815    603 7.30 5.3963 4.5102 3.3509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15816    603 7.40 5.4963 4.6102 3.4509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15817    603 7.50 5.5963 4.7102 3.5509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15818    603 7.60 5.6963 4.8102 3.6509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15819    603 7.70 5.7963 4.9102 3.7509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15820    603 7.80 5.8963 5.0102 3.8509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15821    603 7.90 5.9963 5.1102 3.9509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15822    603 8.00 6.0963 5.2102 4.0509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15823    603 8.10 6.1963 5.3102 4.1509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15824    603 8.20 6.2963 5.4102 4.2509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15825    603 8.30 6.3963 5.5102 4.3509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15826    603 8.40 6.4963 5.6102 4.4509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15827    603 8.50 6.5963 5.7102 4.5509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15828    603 8.60 6.6963 5.8102 4.6509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15829    603 8.70 6.7963 5.9102 4.7509 0.10000     Rem2     Rem2 603     6   7 8.88          1
15830    603 8.80 6.8963 6.0102 4.8509 0.07990     Rem2   D/Rem2 603     6   7 8.88          1
> summary( sp.dli )

Transitions:
     To
From      Rem D/Rem  Rel D/Rel DLI D/DLI Rem2 D/Rem2 Rel2  Records:  Events: Risk time:
  Rem   10356   100  123     0   0     0    0      0    0     10579      223    1042.59
  Rel       0     0 2485    40  65     0    0      0    0      2590      105     246.98
  DLI       0     0    0     0 931    10   44      0    0       985       54      92.18
  Rem2      0     0    0     0   0     0 1933      2    4      1939        6     189.38
  Sum   10356   100 2608    40 996    10 1977      2    4     16093      388    1571.13

Transitions:
     To
From     Persons:
  Rem         304
  Rel         123
  DLI          65
  Rem2         44
  Sum         304
```

We observe that the split dataset contains the same amount of person-years and event in the appropriate states and transitions, only the number of records is larger. The point is that each interval now has a different value of the timescales (`tfi`, `tfr`, `tfD`, and `tfR`), so that these can be used as a covariates. In the first instance we shall only use `tfi`

This dataset can now be stacked, and subsetted as before to allow for analysis of the mortality rates; later we also extract the other part of the stacked dataset, referring to the transitions between disease states:

```
> ss.dli <- stack( sp.dli )
> m.dli <- subset( ss.dli, lex.Tr %in% levels(lex.Tr)[c(1,3,5,7)] )
> m.dli$lex.Tr <- factor( m.dli$lex.Tr )
> m.dli$Rst <- Relevel( m.dli$lex.Tr, list(Remis=c(1,4),Relapse=2:3) )
> with( m.dli, ftable( Rst, lex.Tr, lex.Fail, col.vars=3 ) )
                     lex.Fail FALSE  TRUE
Rst     lex.Tr
Remis   Rem->D/Rem             10479   100
        Rel->D/Rel                 0     0
        DLI->D/DLI                 0     0
        Rem2->D/Rem2            1937     2
Relapse Rem->D/Rem                 0     0
        Rel->D/Rel              2550    40
        DLI->D/DLI               975    10
        Rem2->D/Rem2               0     0
```

We also inspect how the deaths are distributed over the intervals of time (since entry, *i.e.* first remission), albeit only for the first two years:

```
> YDtab <- xtabs( cbind(lex.dur,lex.Fail) ~ I(floor(tfi*10)/10) + Rst,
+                  data=subset(m.dli,tfi<2.1) )
> dnam <- dimnames(YDtab)
> dnam[[3]] <- c("Y","D","rate")
> YDrate <- array( NA, dimnames=dnam, dim=sapply(dnam,length) )
> YDrate[,,1:2] <- YDtab
> YDrate[,,3] <- YDrate[,,2]/YDrate[,,1]*1000
> round( ftable( YDrate, row.vars=1 ), 1 )
                      Rst  Remis                Relapse
                               Y      D   rate        Y      D   rate
I(floor(tfi * 10)/10)
0                           30.1   11.0  365.4      0.0    0.0    0.0
0.1                         27.9   24.0  861.5      0.4    3.0 8377.5
0.2                         24.4   20.0  818.3      1.0    1.0  988.2
0.3                         22.1    5.0  226.2      2.1    3.0 1442.2
0.4                         20.4    6.0  294.0      2.8    1.0  356.8
0.5                         18.1    5.0  275.5      4.4    1.0  228.5
0.6                         16.9    5.0  295.5      5.0    2.0  397.7
0.7                         16.0    2.0  124.9      5.4    2.0  372.6
0.8                         15.2    2.0  131.2      5.8    0.0    0.0
0.9                         14.9    0.0    0.0      6.0    1.0  165.6
1                           14.4    2.0  139.2      6.4    0.0    0.0
1.1                         13.9    2.0  144.1      6.6    1.0  151.8
1.2                         13.6    1.0   73.6      6.6    2.0  303.0
1.3                         13.7    1.0   73.0      6.2    2.0  322.4
1.4                         13.4    0.0    0.0      6.1    1.0  163.4
1.5                         13.1    0.0    0.0      6.3    1.0  159.7
1.6                         13.1    0.0    0.0      6.0    1.0  166.5
1.7                         13.0    0.0    0.0      5.7    1.0  174.8
1.8                         12.8    0.0    0.0      5.7    1.0  175.3
1.9                         12.5    1.0   80.2      5.8    0.0    0.0
2                           12.2    0.0    0.0      5.7    1.0  174.0
```

We see that there is a very large mortality in the “Relapse” group in beginning, essentially an immediate death after very early relapse (albeit based only on very few cases).

We use natural splines with knots that are located at 0 and at the suitable quantiles of the death times:

```
> ( m.kn <- with( subset( m.dli, lex.Fail ),
+                 c(0,quantile( tfi+lex.dur, probs=1:4/5 )) ) )
                20%       40%       60%       80%
0.0000000 0.1781533 0.3484463 0.8011891 2.6473747
```

We will use natural splines that are generated by the command `ns` from the `splines` package, but the interface is a bit clumsy, so we use the wrapper `Ns` from the `Epi` package when we model the mortality rates as in the Cox-model, but now with an explicit parametric form of the underlying mortality (natural splines, using `Ns`):

```
> m0 <- glm( lex.Fail ~ Ns( tfi, knots=m.kn ) + lex.Tr,
+            family = poisson, offset=log(lex.dur), data=m.dli )
> summary( m0 )
Call:
glm(formula = lex.Fail ~ Ns(tfi, knots = m.kn) + lex.Tr, family = poisson,
    data = m.dli, offset = log(lex.dur))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.7315  -0.1321  -0.0786  -0.0637   4.6827

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)             -0.7812     0.2490  -3.138  0.00170
Ns(tfi, knots = m.kn)1  -1.1510     0.3699  -3.111  0.00186
Ns(tfi, knots = m.kn)2  -2.9237     0.2850 -10.258  < 2e-16
Ns(tfi, knots = m.kn)3  -1.8039     0.6375  -2.830  0.00466
Ns(tfi, knots = m.kn)4  -3.1014     0.2505 -12.383  < 2e-16
lex.TrRel->D/Rel         1.2586     0.2100   5.993 2.06e-09
lex.TrDLI->D/DLI         1.1406     0.3601   3.168  0.00154
lex.TrRem2->D/Rem2      -0.9060     0.7302  -1.241  0.21465

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1689.7  on 16092  degrees of freedom
Residual deviance: 1425.8  on 16085  degrees of freedom
AIC: 1745.8

Number of Fisher Scoring iterations: 8
> round(ci.lin( m0, E=T ),3)
                       Estimate StdErr       z     P exp(Est.)  2.5% 97.5%
(Intercept)              -0.781  0.249  -3.138 0.002     0.458 0.281 0.746
Ns(tfi, knots = m.kn)1   -1.151  0.370  -3.111 0.002     0.316 0.153 0.653
Ns(tfi, knots = m.kn)2   -2.924  0.285 -10.258 0.000     0.054 0.031 0.094
Ns(tfi, knots = m.kn)3   -1.804  0.637  -2.830 0.005     0.165 0.047 0.574
Ns(tfi, knots = m.kn)4   -3.101  0.250 -12.383 0.000     0.045 0.028 0.073
lex.TrRel->D/Rel          1.259  0.210   5.993 0.000     3.520 2.333 5.313
lex.TrDLI->D/DLI          1.141  0.360   3.168 0.002     3.129 1.545 6.337
lex.TrRem2->D/Rem2       -0.906  0.730  -1.241 0.215     0.404 0.097 1.691
```

We see that the value of the regression parameters and their s.e. are virtually the same as from the Cox-model:

```
> round(cbind( ci.lin( m0, subset="->" )[,1:2],
+              ci.lin( c0             )[,1:2] ), 4 )
```

```
                    Estimate StdErr Estimate StdErr
lex.TrRel->D/Rel      1.2586 0.2100   1.2396 0.2103
lex.TrDLI->D/DLI      1.1406 0.3601   1.1652 0.3604
lex.TrRem2->D/Rem2   -0.9060 0.7302  -0.9073 0.7320
> round(cbind( ci.lin( m0, subset="->" )[,1:2]/
+              ci.lin( c0                )[,1:2] ), 4 )
                    Estimate StdErr
lex.TrRel->D/Rel      1.0153 0.9987
lex.TrDLI->D/DLI      0.9789 0.9990
lex.TrRem2->D/Rem2    0.9987 0.9975
```

We make the same model reduction as before and make a similar test:

```
> m1 <- update( m0, . ~ . - lex.Tr + Rst )
> anova( m0, m1, test="Chisq" )
Analysis of Deviance Table

Model 1: lex.Fail ~ Ns(tfi, knots = m.kn) + lex.Tr
Model 2: lex.Fail ~ Ns(tfi, knots = m.kn) + Rst
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1     16085     1425.8
2     16087     1427.9 -2  -2.0942    0.351
```

with pretty much the same result. But with the Poisson model we can easily make a likelihood-ratio tests for non-proportionality, both in the case with 4 different transitions, and in the case where we have assumed them to be pairwise equivalent:

```
> m0i <- update( m0, . ~ . + Ns( tfi, knots=m.kn ):lex.Tr )
> m1i <- update( m1, . ~ . + Ns( tfi, knots=m.kn ):Rst )
> anova( m0, m0i, test="Chisq" )
Analysis of Deviance Table

Model 1: lex.Fail ~ Ns(tfi, knots = m.kn) + lex.Tr
Model 2: lex.Fail ~ Ns(tfi, knots = m.kn) + lex.Tr + Ns(tfi, knots = m.kn):lex.Tr
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1     16085     1425.8
2     16074     1407.4 11   18.407   0.0726
> anova( m1, m1i, test="Chisq" )
Analysis of Deviance Table

Model 1: lex.Fail ~ Ns(tfi, knots = m.kn) + Rst
Model 2: lex.Fail ~ Ns(tfi, knots = m.kn) + Rst + Ns(tfi, knots = m.kn):Rst
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1     16087     1427.9
2     16083     1415.5  4   12.414  0.01452
```

It is clear that the test for proportional hazards between the 4 transitions to death is swamped by too many d.f. and that there actually is pretty clear evidence that the mortality rates from the remission states and from the relapsed states are not proportional (on the `tfi` scale). This type of analysis would not be straight-forward had we used Cox-models, despite the simplicity of the question.

Also we can test whether the reduction to two sets of mortality rates is OK in the stratified model. This is a formal test of whether the four different mortality rates can be reduced to two:

```
> anova( m0i, m1i, test="Chisq" )
```

```
Analysis of Deviance Table

Model 1: lex.Fail ~ Ns(tfi, knots = m.kn) + lex.Tr + Ns(tfi, knots = m.kn):lex.Tr
Model 2: lex.Fail ~ Ns(tfi, knots = m.kn) + Rst + Ns(tfi, knots = m.kn):Rst
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1     16074     1407.4
2     16083     1415.5 -9  -8.0876   0.5253
```

Thus, there is some evidence that there is non-proportionality between rates from the remission states and the relapse states, but not much evidence that the two sets of rates from remission states and the two sets of rates from relapse states are different. Therefore it is of interest to see how the underlying rates look in the two groups.

To this end we take a look at the parameters of the model:

```
> ci.lin( m1i )[,1:2]
                                      Estimate    StdErr
(Intercept)                         -0.8117037 0.2544885
Ns(tfi, knots = m.kn)1              -0.6527816 0.3984492
Ns(tfi, knots = m.kn)2              -3.1250602 0.3717075
Ns(tfi, knots = m.kn)3              -2.1104346 0.6978965
Ns(tfi, knots = m.kn)4              -3.3206608 0.3112183
RstRelapse                           6.5399827 1.9364236
Ns(tfi, knots = m.kn)1:RstRelapse  -6.1985860 1.8849812
Ns(tfi, knots = m.kn)2:RstRelapse  -2.5218787 1.0573918
Ns(tfi, knots = m.kn)3:RstRelapse -10.4111758 4.3371388
Ns(tfi, knots = m.kn)4:RstRelapse  -1.8725306 0.8284777
```

In order to show the rates we need to multiply the parameters by a matrix where each row are numbers to be multiplied to the parameter estimates to produce the estimated log-rates at a given point in time. This matrix is set up using the same machinery of natural splines that was used in the model, but now not with the observed values of `tfi` as in the data, but with the values where we want the predictions made, *in casu* `pr.pt`:

```
> pr.pt <- seq(0,10,0.02)
>  n.pt <- length( pr.pt )
> CM <- cbind( 1, Ns( pr.pt, knots=m.kn ) )
```

This is the matrix that we must multiply to the parameter vector (or subset of it) in order to get the log-mortality rates evaluated at the points in `pr.pt`. The function `ci.exp` exponentiates the results so that they are given as rates with 95% c.i.s:

```
> Rem.Dead <- ci.exp( m1i, ctr.mat=cbind(CM,CM*0) )
> Rel.Dead <- ci.exp( m1i, ctr.mat=cbind(CM,CM  ) )
```

Once we have the rates, we can plot them:

```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( pr.pt, cbind( Rem.Dead, Rel.Dead )*100, log="y", ylim=c(1,400),
+          type="l", lty=1, lwd=c(3,1,1), las=1,
+          col=rep(clr<-c("limegreen","red"),each=3),
+          ylab="Mortality rates per 100 PY", xlab="Time since entry" )
> text( c(10,10), 400*c(0.8,1), c("Remission","Relapse"), col=clr,
+       adj=c(1,1) )
```

We do not show this plot, because we also want to include RR between the two rates as a function of time. Note that the RR is easily obtained by using a different contrast matrix:

```
> RR.RmRl <- ci.exp( m1i, ctr.mat=cbind(CM*0,CM) )
> par( mar=c(3,4,1,4), mgp=c(3,1,0)/1.6 )
> matplot( pr.pt, cbind( Rem.Dead, Rel.Dead )*100, log="y", ylim=c(1,400),
+          type="l", lty=1, lwd=c(3,1,1),
+          col=rep(c("limegreen","red"),each=3),
+          ylab="Rate per 100 PY", xlab="Time since entry", yaxt="n" )
> abline( h=10 )
> text( c(10,10), 400*c(0.8,1),
+       c("Remission","Relapse"), col=c("limegreen","red"),
+       adj=c(1,1), font=2 )
> matlines( pr.pt, RR.RmRl*10, type="l", lty=1, lwd=c(3,1,1), col="blue" )
> yt <- outer(c(1,2,5),c(1,10,100,1000),"*")
> axis( side=2, at=yt, labels=yt    , las=1 )
> axis( side=4, at=yt, labels=yt/10, las=1 )
> mtext( "Rate ratio", side=4, line=2, col="blue" )
```

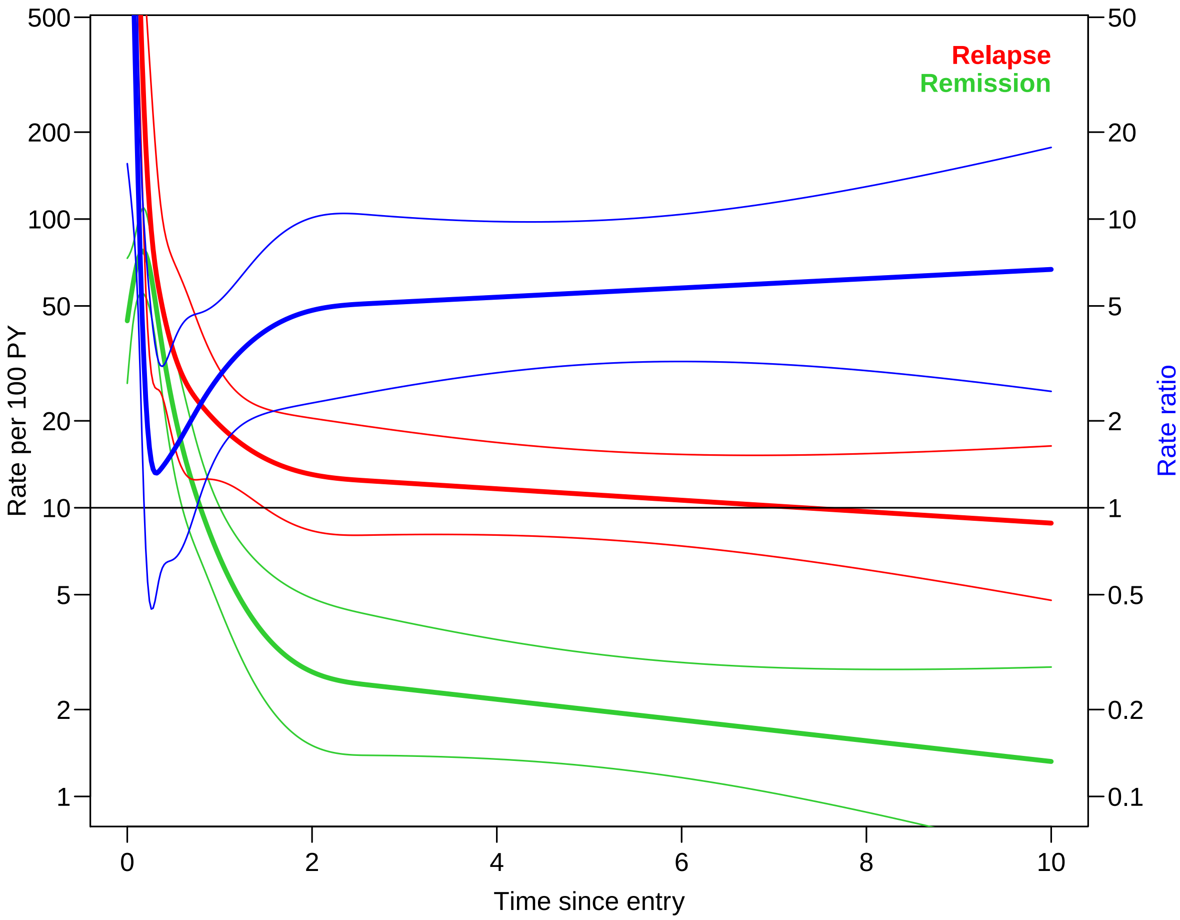

Figure 2: *Estimated baseline mortality rates from remission (green) and relapse (red), and the ratio of them (blue).*

From the figure 2 we clearly see the assumption about linear log-rates beyond the rightmost knot (at 2.3) where only 25% of deaths occur. Also it is clear that it is during the first 2 years the mortality rates from relapse and remission are non-proportional.

4 Transitions between disease states

If we want a description of how patients fare through the states we must provide statistical model for `all` transitions.

For the non-mortality transition rates there is not much sense in having common rates, so we fit a separate rate for each.

However, we first explore how events are distributed along the timescale for the different transitions:

```
> str( ss.dli )
Classes ‘stacked.Lexis’ and 'data.frame':       32186 obs. of  19 variables:
 $ lex.id  : int  2 2 2 2 2 2 2 2 2 2 ...
 $ tfi     : num  0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 ...
 $ tfr     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ tfD     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ tfR     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ lex.dur : num  0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 ...
 $ lex.Cst : Factor w/ 9 levels "Rem","D/Rem",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Xst : Factor w/ 9 levels "Rem","D/Rem",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Tr  : Factor w/ 8 levels "Rem->D/Rem","Rem->Rel",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Fail: logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ id      : int  2 2 2 2 2 2 2 2 2 2 ...
 $ from    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ to      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ time    : num  19.9 19.9 19.9 19.9 19.9 ...
 $ distatpr: int  1 1 1 1 1 1 1 1 1 1 ...
 $ ti      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ tr      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ tD      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ tR      : num  NA NA NA NA NA NA NA NA NA NA ...
 - attr(*, "breaks")=List of 4
  ..$ tfi: num  0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 ...
  ..$ tfr: NULL
  ..$ tfD: NULL
  ..$ tfR: NULL
 - attr(*, "time.scales")= chr  "tfi" "tfr" "tfD" "tfR"
```

```
> with( subset( ss.dli, lex.Fail & lex.Tr %in% levels(lex.Tr)[-c(1,3,5,7)] ),
+       dotchart( tfi+lex.dur, groups=factor(lex.Tr), pch=16, cex=0.8) )
```

From figure 3 it seems that there is a reasonable distribution of event times for all 4 types of transitions, so we use the same set of knots for all transitions:

```
> p.dli <- subset( ss.dli, lex.Tr %in% levels(lex.Tr)[-c(1,3,5,7)] )
> p.dli$lex.Tr <- factor( p.dli$lex.Tr )
> ( p.kn <-with( subset( p.dli, lex.Fail ),
+               c(0,quantile( tfi+lex.dur, probs=1:3/4 )) ) )
               25%      50%      75%
0.000000 0.553869 1.200929 2.828907
```

A quick glance at figure 1, however shows that there is virtually no information to estimate the transition from Rem2→Rel2, so we pool this transition with the transition Rem→Rel:

```
> p.dli$lex.Tr <- Relevel( p.dli$lex.Tr, list("Rem->Rel"=c(1,4), 2, 3 ) )
```

With the new set of knots defined we now for separate transition rates for the four remaining transitions (modeled as 3). Note that we use `intercept=TRUE` to include the intercept with the natural spline:

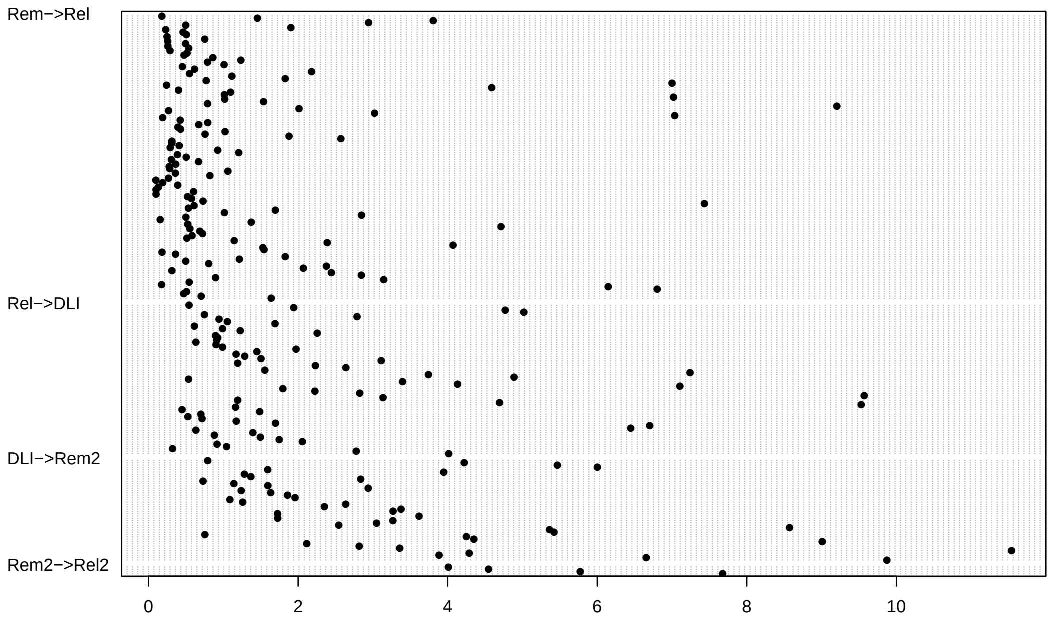

Figure 3: *Distribution of transition times for the non-death events.*

```
> p2 <- glm( lex.Fail ~ Ns( tfi, knots=p.kn, intercept=TRUE ):lex.Tr -1 ,
+            family = poisson, offset=log(lex.dur), data=p.dli )
> summary( p2 )

Call:
glm(formula = lex.Fail ~ Ns(tfi, knots = p.kn, intercept = TRUE):lex.Tr -
    1, family = poisson, data = p.dli, offset = log(lex.dur))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.4388  -0.2086  -0.0994  -0.0611   4.2521

Coefficients:
                                                Estimate Std. Error z value
Ns(tfi, knots = p.kn, intercept = TRUE)1:lex.TrRem->Rel   0.15491    0.25007   0.619
Ns(tfi, knots = p.kn, intercept = TRUE)2:lex.TrRem->Rel  -1.89374    0.25749  -7.355
Ns(tfi, knots = p.kn, intercept = TRUE)3:lex.TrRem->Rel  -4.32120    0.32392 -13.340
Ns(tfi, knots = p.kn, intercept = TRUE)4:lex.TrRem->Rel  -1.94709    0.22720  -8.570
Ns(tfi, knots = p.kn, intercept = TRUE)1:lex.TrRel->DLI   0.69701    0.59036   1.181
Ns(tfi, knots = p.kn, intercept = TRUE)2:lex.TrRel->DLI  -0.08021    0.59508  -0.135
Ns(tfi, knots = p.kn, intercept = TRUE)3:lex.TrRel->DLI  -3.74838    1.49175  -2.513
Ns(tfi, knots = p.kn, intercept = TRUE)4:lex.TrRel->DLI   0.04084    0.81607   0.050
Ns(tfi, knots = p.kn, intercept = TRUE)1:lex.TrDLI->Rem2  0.85071    1.51811   0.560
Ns(tfi, knots = p.kn, intercept = TRUE)2:lex.TrDLI->Rem2  0.47027    2.25737   0.208
Ns(tfi, knots = p.kn, intercept = TRUE)3:lex.TrDLI->Rem2 -3.44492    5.56219  -0.619
Ns(tfi, knots = p.kn, intercept = TRUE)4:lex.TrDLI->Rem2  0.82517    3.17607   0.260
                                                Pr(>|z|)
Ns(tfi, knots = p.kn, intercept = TRUE)1:lex.TrRem->Rel     0.536
Ns(tfi, knots = p.kn, intercept = TRUE)2:lex.TrRem->Rel  1.91e-13
Ns(tfi, knots = p.kn, intercept = TRUE)3:lex.TrRem->Rel   < 2e-16
Ns(tfi, knots = p.kn, intercept = TRUE)4:lex.TrRem->Rel   < 2e-16
Ns(tfi, knots = p.kn, intercept = TRUE)1:lex.TrRel->DLI     0.238
```

```
Ns(tfi, knots = p.kn, intercept = TRUE)2:lex.TrRel->DLI     0.893
Ns(tfi, knots = p.kn, intercept = TRUE)3:lex.TrRel->DLI     0.012
Ns(tfi, knots = p.kn, intercept = TRUE)4:lex.TrRel->DLI     0.960
Ns(tfi, knots = p.kn, intercept = TRUE)1:lex.TrDLI->Rem2    0.575
Ns(tfi, knots = p.kn, intercept = TRUE)2:lex.TrDLI->Rem2    0.835
Ns(tfi, knots = p.kn, intercept = TRUE)3:lex.TrDLI->Rem2    0.536
Ns(tfi, knots = p.kn, intercept = TRUE)4:lex.TrDLI->Rem2    0.795

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 4250.4  on 16093  degrees of freedom
Residual deviance: 2155.9  on 16081  degrees of freedom
AIC: 2651.9

Number of Fisher Scoring iterations: 9
```

```
> round(ci.exp( p2, Exp=FALSE ), 2 )
```

```
                                                         Estimate   2.5% 97.5%
Ns(tfi, knots = p.kn, intercept = TRUE)1:lex.TrRem->Rel      0.15  -0.34  0.65
Ns(tfi, knots = p.kn, intercept = TRUE)2:lex.TrRem->Rel     -1.89  -2.40 -1.39
Ns(tfi, knots = p.kn, intercept = TRUE)3:lex.TrRem->Rel     -4.32  -4.96 -3.69
Ns(tfi, knots = p.kn, intercept = TRUE)4:lex.TrRem->Rel     -1.95  -2.39 -1.50
Ns(tfi, knots = p.kn, intercept = TRUE)1:lex.TrRel->DLI      0.70  -0.46  1.85
Ns(tfi, knots = p.kn, intercept = TRUE)2:lex.TrRel->DLI     -0.08  -1.25  1.09
Ns(tfi, knots = p.kn, intercept = TRUE)3:lex.TrRel->DLI     -3.75  -6.67 -0.82
Ns(tfi, knots = p.kn, intercept = TRUE)4:lex.TrRel->DLI      0.04  -1.56  1.64
Ns(tfi, knots = p.kn, intercept = TRUE)1:lex.TrDLI->Rem2     0.85  -2.12  3.83
Ns(tfi, knots = p.kn, intercept = TRUE)2:lex.TrDLI->Rem2     0.47  -3.95  4.89
Ns(tfi, knots = p.kn, intercept = TRUE)3:lex.TrDLI->Rem2    -3.44 -14.35  7.46
Ns(tfi, knots = p.kn, intercept = TRUE)4:lex.TrDLI->Rem2     0.83  -5.40  7.05
```

We now apply the same machinery as before, and construct a contrast matrix to extract the transition rates:

```
> CP <- Ns( pr.pt, knots=p.kn, intercept=TRUE )
> Rem.Rel <- ci.exp( p2, subset="TrRem", ctr.mat=CP )
> Rel.DLI <- ci.exp( p2, subset="TrRel", ctr.mat=CP )
> DLI.Rem <- ci.exp( p2, subset="TrDLI", ctr.mat=CP )
```

and we can plot these three *progression* rates together:

```
> par( mar=c(3,4,1,4), mgp=c(3,1,0)/1.6 )
> matplot( pr.pt, cbind(Rem.Rel,Rel.DLI,DLI.Rem)*100, log="y", ylim=c(1,500),
+          type="l", lty=1, lwd=c(3,1,1),
+          col=rep(c("limegreen","red","blue"),each=3),
+          ylab="Progression rate per 100 PY", xlab="Time since entry" )
> par(font=2)
> legend( "topright", legend=c("Remission > Relapse",
+                              "Relapse > DLI",
+                              "DLI > Remission"), bty="n",
+         text.col=c("limegreen","red","blue"), xjust=1 )
```

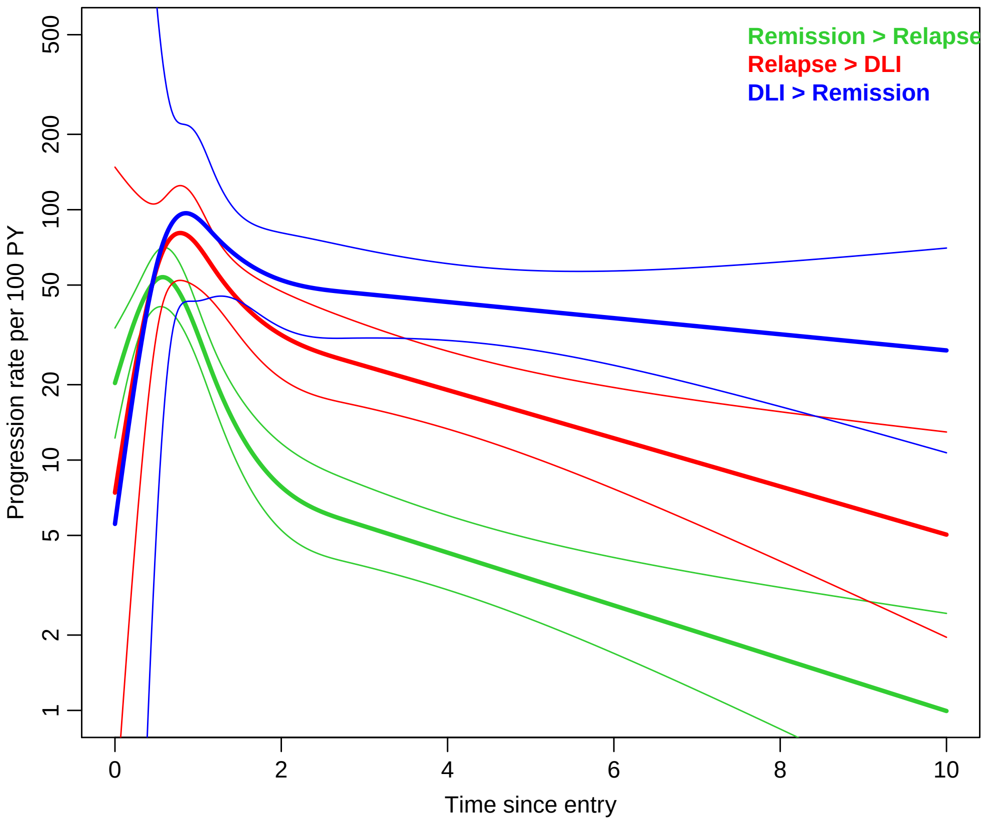

Figure 4: *Estimated disease progression rates.*

5 State occupancy probability

In the paper by Allignol *et al.*[1], the probabilities are referred to as transition probabilities. But since the authors only ever consider transition probabilities from time 0, they are just probabilities of being in a particular state at some time (given of course that the person starts in the study at time 0).

The progression rates as shown are difficult to interpret, so we will instead have a look at how state occupancy looks. Also to be able to show how the overall remission probability evolves, that is how the probability of being in one of the states "Rem" and "Rem2" changes with time since entry into the study.

We can compute the probability of being in a particular state at various times by multiplying the initial state distribution vector (in this case $(1, 0, 0, 0, 0, 0, 0, 0, 0)$), by multiplying it by the transition matrix, that is the matrix of transition probabilities between states. Now this varies by time, because the transition rates vary. But we have computed the transition rates at narrowly spaced intervals, so we can make a very good approximation of the transition probabilities by computing them under the assumption of constant rates in these small intervals. Using `unique` demonstrates that differences are never reliable in in computing:

```
> pr.pt[1:5]
[1] 0.00 0.02 0.04 0.06 0.08
> unique( diff(pr.pt) )
 [1] 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
> ( il <- mean( diff(pr.pt) ) )
[1] 0.02
```

What needs to be done now is to set up a structure (array) with the transition probability matrices as slices and the third dimension being the times in `pr.pt`, so that each slice represents the transition probability at that point.

First we fetch the state names from the `Lexis` object, and then set up the array with transition probabilities at different times:

```
> states <- levels( dli$lex.Cst )
> dnam <- list(From=states, To=states, time=pr.pt )
> AR <- array( 0, dimnames=dnam, dim=sapply(dnam,length) )
> str( AR )
 num [1:9, 1:9, 1:501] 0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, "dimnames")=List of 3
  ..$ From: chr [1:9] "Rem" "D/Rem" "Rel" "D/Rel" ...
  ..$ To  : chr [1:9] "Rem" "D/Rem" "Rel" "D/Rel" ...
  ..$ time: chr [1:501] "0" "0.02" "0.04" "0.06" ...
```

Then we fill the array `AR` with the rates for the actually existing transitions, and from these later compute the one-step transition probabilities in `AP`:

```
> AR["Rem" ,"D/Rem" ,] <- Rem.Dead[,1]
> AR["Rem2","D/Rem2",] <- Rem.Dead[,1]
> AR["Rel" ,"D/Rel" ,] <- Rel.Dead[,1]
> AR["DLI" ,"D/DLI" ,] <- Rel.Dead[,1]
> AR["Rem" ,"Rel"   ,] <-  Rem.Rel[,1]
> AR["Rem2","Rel2"  ,] <-  Rem.Rel[,1]
> AR["Rel" ,"DLI"   ,] <-  Rel.DLI[,1]
> AR["DLI" ,"Rem2"  ,] <-  DLI.Rem[,1]
```

Now we need to fill in the transition probabilities corresponding to a single step of length `il`. Here we must use the formulae for transition probabilities under competing risks:

$$\begin{aligned}\text{P}\{\text{event of type 1 before } b \mid \text{alive at } a\} &= \int_a^b \lambda_1(s)\exp\left(\int_a^s \lambda_1(u)+\lambda_2(u)\,\text{d}u\right)\,\text{d}s\\ &= \frac{\lambda_1}{\lambda_1+\lambda_2}\Big(1-\exp\big(-(b-a)(\lambda_1+\lambda_2)\big)\Big)\end{aligned}$$

So first we compute the sum of the intensities *out* of each state (dimension 1 of `AR`) in each interval (dimension 3 of `AR`):

```
> SI <- apply(AR,c(1,3),sum)
```

This must now be swept through the array to form the transition probabilities as well as the probabilities of remaining in the state:

```
> AP <- AR
> for( i in 1:(dim(AP)[3]) )
+     {
+     AP[,,i] <- AR[,,i]/SI[,i] * (1-exp(-SI[,i]*il))
+     diag(AP[,,i]) <- exp(-SI[,i]*il)
+     }
> AP[is.na(AP)] <- 0
> round( SI[,1], 4 )
     Rem    D/Rem      Rel    D/Rel      DLI    D/DLI     Rem2   D/Rem2     Rel2
  0.6471   0.0000 307.5138   0.0000 307.4953   0.0000   0.6471   0.0000   0.0000
```

```
> round( ftable( AR[,,50+1:2], row.vars=c(3,1)), 4 )
           To     Rem  D/Rem    Rel  D/Rel    DLI  D/DLI   Rem2 D/Rem2   Rel2
time From
1    Rem       0.0000 0.0673 0.3155 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     D/Rem     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     Rel       0.0000 0.0000 0.0000 0.1930 0.7166 0.0000 0.0000 0.0000 0.0000
     D/Rel     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     DLI       0.0000 0.0000 0.0000 0.0000 0.0000 0.1930 0.9204 0.0000 0.0000
     D/DLI     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     Rem2      0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0673 0.3155
     D/Rem2    0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     Rel2      0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
1.02 Rem       0.0000 0.0651 0.3030 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     D/Rem     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     Rel       0.0000 0.0000 0.0000 0.1903 0.7026 0.0000 0.0000 0.0000 0.0000
     D/Rel     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     DLI       0.0000 0.0000 0.0000 0.0000 0.0000 0.1903 0.9086 0.0000 0.0000
     D/DLI     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     Rem2      0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0651 0.3030
     D/Rem2    0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     Rel2      0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
```

```
> round( ftable( AP[,,50+1:2], row.vars=c(3,1)), 4 )
           To     Rem  D/Rem    Rel  D/Rel    DLI  D/DLI   Rem2 D/Rem2   Rel2
time From
1    Rem       0.9924 0.0013 0.0063 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     D/Rem     0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     Rel       0.0000 0.0000 0.9820 0.0038 0.0142 0.0000 0.0000 0.0000 0.0000
     D/Rel     0.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     DLI       0.0000 0.0000 0.0000 0.0000 0.9780 0.0038 0.0182 0.0000 0.0000
     D/DLI     0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000
     Rem2      0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.9924 0.0013 0.0063
```

```
     D/Rem2    0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000
     Rel2      0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000
1.02 Rem       0.9927 0.0013 0.0060 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     D/Rem     0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     Rel       0.0000 0.0000 0.9823 0.0038 0.0139 0.0000 0.0000 0.0000 0.0000
     D/Rel     0.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     DLI       0.0000 0.0000 0.0000 0.0000 0.9783 0.0038 0.0180 0.0000 0.0000
     D/DLI     0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000
     Rem2      0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.9927 0.0013 0.0060
     D/Rem2    0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000
     Rel2      0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000
```

Each of the slices of the array `AP` is a matrix of transition probabilities.

Suppose the initial state distribution is π_0, in this case a vector of length 9: $\pi_o = (1,0,0,0,0,0,0,0,0)$. The probability distribution after one step is then $\pi_1 = \pi_0 A_1$, where A_1 is the transition matrix for the first interval. After two steps it is $\pi_1 A_2$ and so forth.

So now we set up an array to store the state-distribution at different times; it has the same structure as the `SI` array:

```
> pi0 <- c(1,rep(0,8))
> ST <- SI*0
> ST[,1] <- pi0 %*% AP[,,1]
> for( i in 2:dim(ST)[2] ) ST[,i] <- ST[,i-1] %*% AP[,,i]
> str( ST )
 num [1:9, 1:501] 0.98714 0.00882 0.00403 0 0 ...
 - attr(*, "dimnames")=List of 2
  ..$ From: chr [1:9] "Rem" "D/Rem" "Rel" "D/Rel" ...
  ..$ time: chr [1:501] "0" "0.02" "0.04" "0.06" ...
```

```
> round(t(ST[,1:10]),3)
      From
time     Rem D/Rem   Rel D/Rel DLI D/DLI Rem2 D/Rem2 Rel2
  0    0.987 0.009 0.004 0.000   0     0    0      0    0
  0.02 0.973 0.018 0.004 0.004   0     0    0      0    0
  0.04 0.959 0.029 0.005 0.007   0     0    0      0    0
  0.06 0.943 0.040 0.006 0.011   0     0    0      0    0
  0.08 0.926 0.052 0.008 0.013   0     0    0      0    0
  0.1  0.909 0.065 0.011 0.016   0     0    0      0    0
  0.12 0.891 0.078 0.014 0.017   0     0    0      0    0
  0.14 0.872 0.091 0.018 0.019   0     0    0      0    0
  0.16 0.854 0.104 0.021 0.020   0     0    0      0    0
  0.18 0.835 0.118 0.026 0.021   0     0    0      0    0
```

The estimated occupancy probabilities for each of the states can now easily be plotted:

```
> matplot( pr.pt, t(ST)[,-1], type="l", ylim=c(0,0.5), las=1,
+          lty=1, lwd=2, col=rainbow(8),
+          xlab="Time since 1st remission", ylab="Fraction of patients" )
> legend( "topleft", legend=rownames(ST)[-1],
+         bty="n", lty=1, lwd=2, col=rainbow(8),
+         ncol=1 )
```

Once we have that, we can plot the more easily understandable *cumulative* state-occupancies, by cumulating over the 1st dimension of `ST` (that is a loop over dimension 2), which in the case of using `cumsum` returns a vector, and so adds a dimension equal to the length of this as the *first* dimension of the result:

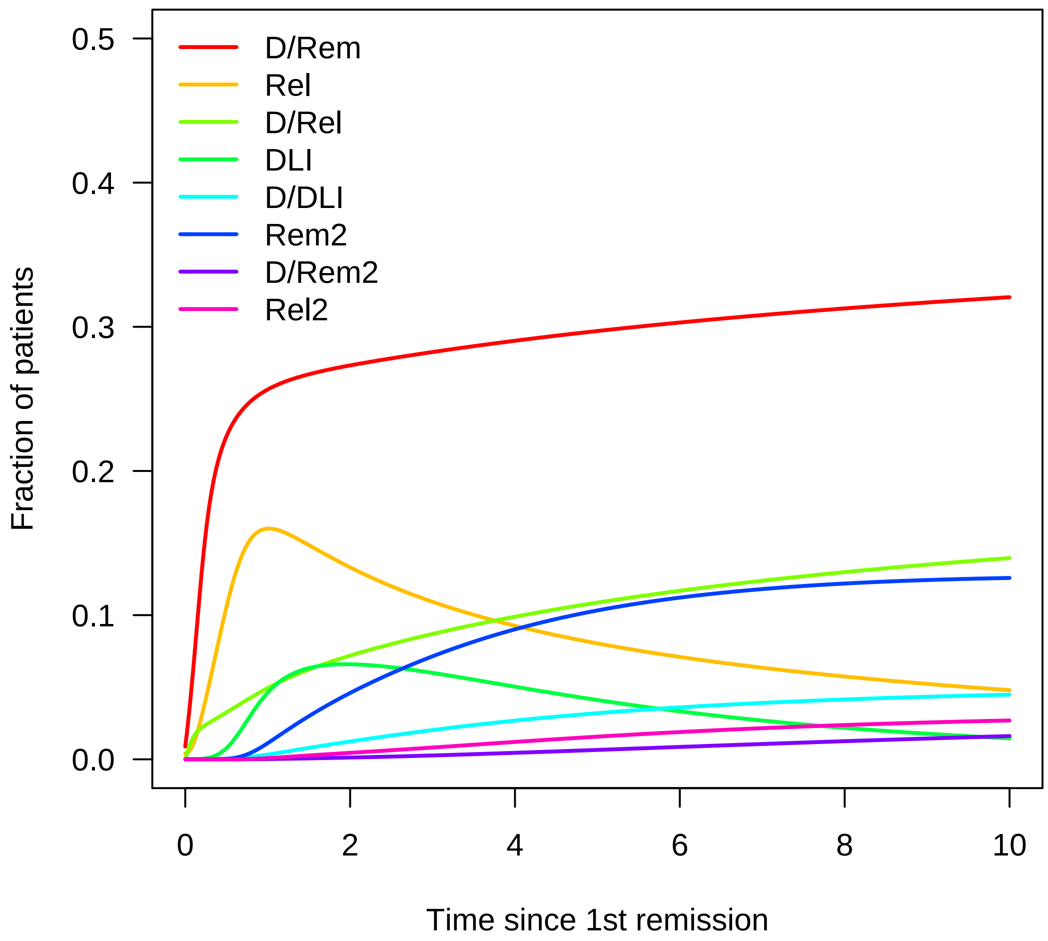

Figure 5: *Estimated state occupancy probabilities for each state.*

```
> cST <- apply(ST,2,cumsum)
```

This could easily be plotted using *e.g.*:

```
> matplot( pr.pt, t(cST), type="l", lty=1, lwd=2 )
```

It would however be nicer to be able to show the states in an arbitrary order; but this is easily packed into one command, by defining a vector `perm` which is a permutation of the levels:

```
> perm <- c(1,3,5,7,9,2,4,6,8)
> dimnames(ST)[[1]][perm]
[1] "Rem"    "Rel"    "DLI"    "Rem2"   "Rel2"   "D/Rem"  "D/Rel"  "D/DLI"  "D/Rem2"
> matplot( pr.pt, t(apply(ST[perm,],2,cumsum)), type="l", lty=1, lwd=2 )
```

This is not extremely informative, because it is the space between the lines that is of interest. Moreover it would be relevant to be able to write the state-names at the ends, for example:

```
> endpos <- cumsum(ST[perm,n.pt]) - ST[perm,n.pt]/2
> matplot( pr.pt, t(apply(ST[perm,],2,cumsum)),
+          type="l", lty=1, lwd=2, col=gray(c(0.6,0)[c(1,1,1,1,2,1,1,1,1)]),
+          xlim=c(0,11.5), ylim=c(0,1), yaxs="i" )
> text( 10.05, endpos, dimnames(ST)[[1]][perm], font=2, adj=0 )
```

Furthermore, if we want to color the areas between the curves, we just start from the top and color the area underneath each curve. We also make sure that there is place for the labels at the edges (although the algorithm for this is not complete yet). We put it all in a function with `perm` as an argument, to allow us to change the ordering of the states:

```
> state.occ <-
+ function( perm=1:n.st, line=NULL )
+ {
+ clr <- st.col[perm]
+
+ mindist <- 1/40
+ endpos <- cumsum(ST[perm,n.pt]) - ST[perm,n.pt]/2
+ minpos <- (1:n.st-0.5)*mindist
+ maxpos <- 1-rev(minpos)
+ endpos <- pmin( pmax( endpos, minpos ), maxpos )
+
+ stkcrv <- t(apply(ST[perm,],2,cumsum))
+ matplot( pr.pt, stkcrv,
+          type="l", lty=1, lwd=1, col="transparent",
+          xlim=c(0,11.5), ylim=c(0,1), yaxs="i", xaxs="i", bty="n",
+          xlab="Time since 1st remission", ylab="Fraction of patients" )
+ text( 10.05, endpos, dimnames(ST)[[1]][perm], font=2, adj=0, col=clr )
+ for( i in 9:1 )
+ polygon( c( pr.pt, rev(pr.pt) ),
+          c( stkcrv[,i], if(i>1) rev(stkcrv[,i-1]) else rep(0,n.pt) ),
+          col=clr[i], border=clr[i] )
+ if( !is.null(line) )
+ matlines( pr.pt, stkcrv[,line], type="l", lty=1, lwd=3, col="black" )
+ }
```

With this function in place we can make plots for different orderings of the states, but maintaining the colors for each state. Also note that we have grouped the “alive” states together, so we can sensibly show the survival curve, separating the alive from the dead:

```
> state.occ( perm=1:n.st )
```

```
> state.occ( perm=c(1,3,5,7,9,2,4,6,8), line=5 )
```

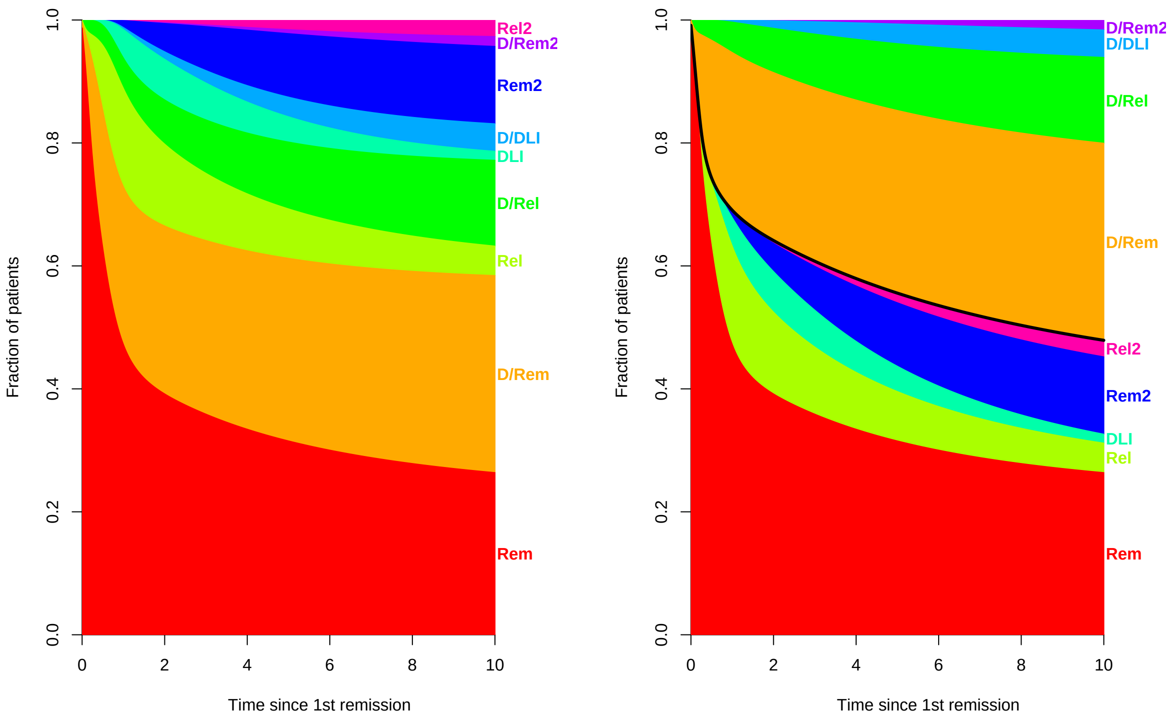

Figure 6: *Estimated state occupancy probabilities. The only difference is the ordering of the states. The black line in the rightmost display is the estimated survival curve.*

5.1 Confidence interval for state occupancies

The uncertainty of these cumulative probabilities are not easily assessed, because they are non-linear functions of the parameters of the glms used to fit the data.

The natural thing to do would therefore be to simulate in some way. Either by bootstrapping the original data and redo the entire analysis for each bootstrap sample, thereby giving a sample from the "posterior" distribution of the relevant quantities.

Or (slightly simpler) by simulation from an assumed normal distribution of the parameter estimates in the model. The latter is achieved by using the `sample=` argument to `ci.lin`, which when set to a number takes a random sample of that size from a multivariate normal distribution with mean equal to the estimated parameters and variance-covariance matrix equal to the estimated variance-covariance matrix of the parameters.

In order for this to work we should wrap all the previous stuff up in a couple of functions that does the work in suitable bits:

- Extract the transition rates and stuff them in a structure — `get.rates`
- Transform the transition matrices to probabilities
- Draw the curves (areas) in color and superpose them with confidence intervals.

When drawing a sample from the posterior we use the argument `sample` to `ci.lin`, that draws a sample from a multivariate normal with mean equal to the parameter estimates and variance equal to the estimated variance-covariance matrix, and transforms each sample by the supplied contrast matrix:

```
> get.rates <- function( N=10 )
+ {
+ Rem.Dead <- ci.lin( m1i,                  ctr.mat=cbind(CM,CM*0), sample=N )
+ Rel.Dead <- ci.lin( m1i,                  ctr.mat=cbind(CM,CM  ), sample=N )
+ Rem.Rel  <- ci.lin( p2 , subset="TrRem", ctr.mat=CP             , sample=N )
+ Rel.DLI  <- ci.lin( p2 , subset="TrRel", ctr.mat=CP             , sample=N )
+ DLI.Rem  <- ci.lin( p2 , subset="TrDLI", ctr.mat=CP             , sample=N )
+ states <- levels( dli$lex.Cst )
+ dnam <- list( From = states,
+                 To = states,
+               time = pr.pt,
+             sample = 1:N )
+ AR <- AP <- array( 0, dimnames=dnam, dim=sapply(dnam,length) )
+ AR["Rem" ,"D/Rem" ,,] <- exp(Rem.Dead)
+ AR["Rem2","D/Rem2",,] <- exp(Rem.Dead)
+ AR["Rel" ,"D/Rel" ,,] <- exp(Rel.Dead)
+ AR["DLI" ,"D/DLI" ,,] <- exp(Rel.Dead)
+ AR["Rem" ,"Rel"   ,,] <- exp(Rem.Rel )
+ AR["Rem2","Rel2"  ,,] <- exp(Rem.Rel )
+ AR["Rel" ,"DLI"   ,,] <- exp(Rel.DLI )
+ AR["DLI" ,"Rem2"  ,,] <- exp(DLI.Rem )
+ AR
+ }
> system.time( AR <- get.rates(1000) )

   user  system elapsed
  0.703   0.619   0.597

> str( AR )
```

```
 num [1:9, 1:9, 1:501, 1:1000] 0 0 0 0 0 ...
 - attr(*, "dimnames")=List of 4
  ..$ From  : chr [1:9] "Rem" "D/Rem" "Rel" "D/Rel" ...
  ..$ To    : chr [1:9] "Rem" "D/Rem" "Rel" "D/Rel" ...
  ..$ time  : chr [1:501] "0" "0.02" "0.04" "0.06" ...
  ..$ sample: chr [1:1000] "1" "2" "3" "4" ...
```

Thus in this general setup, we now have a function that produces an array classified by states×states×times×samples, which contains (samples of) the transition rates in the appropriate places in the resulting array.

Then we need a function that transforms this to a similar array of transition probabilities:

```
> trans.prob <-
+ function( AR )
+ {
+ # A matrix for transition probabilities:
+ AP <- AR * 0
+ # Compute the interval length for the give rates
+ il <- mean( diff( as.numeric( dimnames(AR)[[3]] ) ) )
+ # Sum of the Intensities out of each state
+ SI <- apply(AR,c(1,3,4),sum)
+ for( i in 1:dim(AR)[3] ) # Loop over times
+ for( j in 1:dim(AR)[4] ) # Loop over samples
+    {
+    AP[,,i,j] <- AR[,,i,j]/SI[,i,j] * (1-exp(-SI[,i,j]*il))
+    diag(AP[,,i,j]) <- exp(-SI[,i,j]*il)
+    }
+ AP[is.na(AP)] <- 0
+ invisible( AP )
+ }
> system.time( AP <- trans.prob( AR ) )
   user  system elapsed
 25.437   0.263  25.782
```

We have now computed the transition probabilities for each time for each of the samples from the parameter estimates. So with this we can now compute the state occupancy probabilities:

```
> pi0 <- rep(1:0,c(1,n.st-1))
> ST  <- AP[1,,,]*0
> names( dimnames(ST) )[1] <- "State"
> str( ST )
 num [1:9, 1:501, 1:1000] 0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, "dimnames")=List of 3
  ..$ State : chr [1:9] "Rem" "D/Rem" "Rel" "D/Rel" ...
  ..$ time  : chr [1:501] "0" "0.02" "0.04" "0.06" ...
  ..$ sample: chr [1:1000] "1" "2" "3" "4" ...
> system.time(
+ for( j in 1:dim(ST)[3] )
+    {
+    ST[,1,j] <- pi0 %*% AP[,,1,j]
+    for( i in 2:dim(ST)[2] ) ST[,i,j] <- ST[,i-1,j] %*% AP[,,i,j]
+    }
+            )
   user  system elapsed
  8.195   8.814   5.689
> str( ST )
```

```
num [1:9, 1:501, 1:1000] 0.98749 0.00843 0.00408 0 0 ...
- attr(*, "dimnames")=List of 3
 ..$ State : chr [1:9] "Rem" "D/Rem" "Rel" "D/Rel" ...
 ..$ time  : chr [1:501] "0" "0.02" "0.04" "0.06" ...
 ..$ sample: chr [1:1000] "1" "2" "3" "4" ...
```

By this time we have $N = 1000$ samples of occupancy probabilities as functions of time. If we want to plot these in a given order, we should *first* make the cumulative sums for each of these over states as indicated by `perm`. Then we take for example the 5, 50 and 95% quantiles, and use the median to demarcate the colored chunks, and the 5th and 95th percentiles to make shaded areas showing the (point-wise) 90% confidence for the probabilities.

This amounts to a slight modification and expansion of the `state.occ` function. We show the confidence bands by overlaying a transparent gray shade over the demarcations between the colors showing the occupancy probabilities. This is done using the color `rgb(1/9,1/9,1/9,1/9)`, which is a transparent light gray.

```
> state.occ.sim <-
+ function( perm = 1:n.st,
+             pct = c(5,95),
+           cicol = rgb(1/9,1/9,1/9,1/9),
+            line = NULL )
+ {
+ clr <- st.col[perm]
+ cST <- apply(ST[perm,,],2:3,cumsum)
+ cST <- apply(cST,1:2,quantile,probs=c(pct/100,0.5) )
+ mindist <- 1/40
+ endpos <- cST["50%",,n.pt] - diff(c(0,cST["50%",,n.pt]))/2
+ minpos <- (1:n.st-0.5)*mindist
+ maxpos <- 1-rev(minpos)
+ endpos <- pmin( pmax( endpos, minpos ), maxpos )
+
+ matplot( pr.pt, t(cST["50%",,]),
+          type="n", # lty=1, lwd=2, col=gray(c(0.6,0)[c(1,2,1,2,1,2,1,2,1)]),
+          xlim=c(0,11.5), ylim=c(0,1), yaxs="i", xaxs="i", bty="n",
+          xlab="Time since 1st remission", ylab="Fraction of patients" )
+ text( 10.05, endpos, dimnames(ST)[[1]][perm], font=2, adj=0, col=clr )
+ for( i in n.st:1 )
+ polygon( c( pr.pt, rev(pr.pt) ),
+          c( cST["50%",i,], if(i>1) rev(cST["50%",i-1,]) else rep(0,dim(cST)[3]) ),
+          col=clr[i], border=clr[i] )
+ for( i in n.st:1 )
+ polygon( c( pr.pt, rev(pr.pt) ),
+          c( cST[1,i,], rev(cST[2,i,]) ),
+          col=cicol, border=cicol )
+ if( !is.null(line) ) matlines( pr.pt, t(cST["50%",c(line,NA),]),
+                                type="l", lty=1, lwd=3, col="black" )
+ }
```

With this function we can now plot various versions of the display, to show how the shaded confidence bands work:

```
> par(mar=c(3,3,1,1),mgp=c(3,1,0)/1.6,las=1)
> state.occ.sim( pct=c(5,95) )
```

```
> par(mar=c(3,3,1,1),mgp=c(3,1,0)/1.6,las=1)
> state.occ.sim( perm=c(1,3,5,7,9,2,4,6,8), pct=c(5,95), line=5 )
```

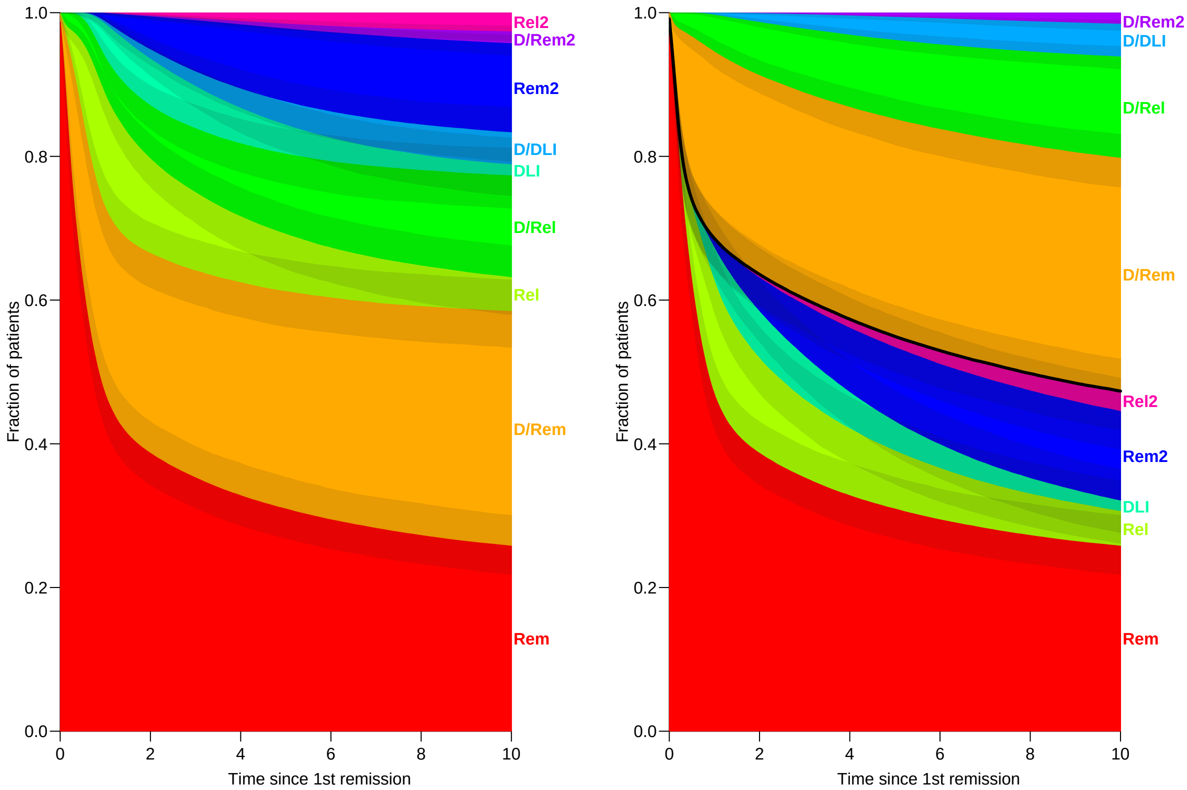

Figure 7: *Estimated state occupancy probabilities with 90% confidence bands based on a simulated sample of 1000 from the "posterior" distribution of the parameters from the Poisson model(s) for the transitions. The only difference is the ordering of the states; in the right hand panel the alive states are grouped together and the black line represents the survival curve.*

6 Probability of being in remission

Allignol *et al.*[1] produce a plot of the probability of being in remission, currently leukaemia free survivor (CLFS) that is in either state "Rem" or "Rem2". Incidentally this can easily be shown by changing the ordering in the above plot, as shown in figure **??**:

```
> par(mar=c(3,3,1,1),mgp=c(3,1,0)/1.6,las=1)
> state.occ.sim( perm=c(1,7,3,5,9,4,6,8,2), pct=c(5,95), line=c(2,5) )
```

With our posterior sample we can also quite easily give a parametrically estimated counterpart of the 95% confidence interval to compare with the estimate from the `etm` package:

```
> CLFS <- apply( ST["Rem",,]+ST["Rem2",,], 1, quantile, probs=c(500,25,975)/1000 )
> str( CLFS )
 num [1:3, 1:501] 0.987 0.981 0.991 0.973 0.962 ...
 - attr(*, "dimnames")=List of 2
  ..$     : chr [1:3] "50%" "2.5%" "97.5%"
  ..$ time: chr [1:501] "0" "0.02" "0.04" "0.06" ...
```

This can now be plotted and compared with the results using the `etm` package:

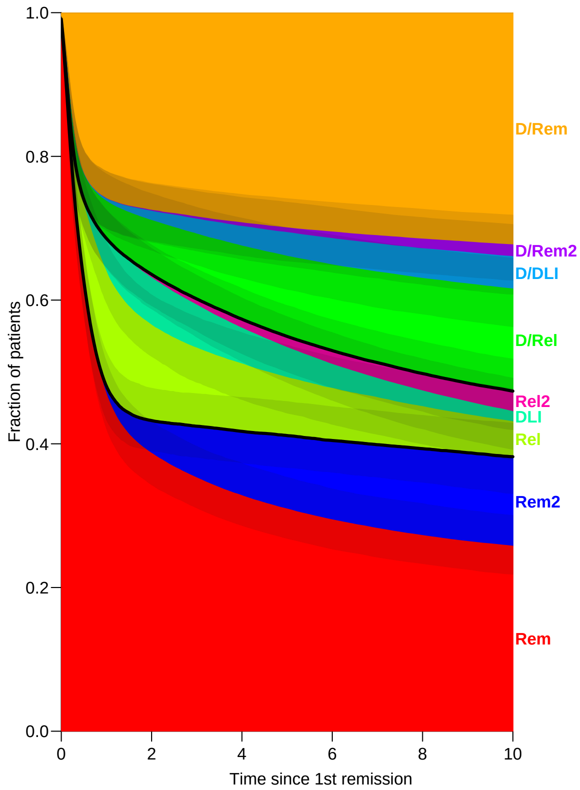

Figure 8: *Estimated state occupancy probabilities with 90% confidence bands based on a simulated sample of 1000 from the "posterior" distribution of the parameters from the Poisson model(s) for the transitions. The black lines represent the probability of being alive in remission and the probability of being alive at all. The only difference to figure 7 is the ordering of the states.*

```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( pr.pt, t(CLFS),
+          type="l", lty=1,lwd=c(3,1,1), col="black",
+          ylim=c(0,1),
+          ylab="P(CLFS)", xlab="Time since first remission" )
```

We do not show this figure, because we will overlay the estimated curve on the non-parametrically estimated curve that that can be computed from the etm package. This is essentially code from the paper [1], tidied a bit for readability:

```
> tra <- matrix(FALSE, 9, 9,
+               dimnames = list(as.character(0:8), as.character(0:8)))
> tra[1, 2:3] <- TRUE
> tra[3, 4:5] <- TRUE
> tra[5, 6:7] <- TRUE
> tra[7, 8:9] <- TRUE
> ### computation of the transition probabilities
> dli.etm <- etm(dli.data, as.character(0:8), tra, "cens", s = 0)
> str(dli.etm)
List of 12
 $ est        : num [1:9, 1:9, 1:536] 0.997 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ : chr [1:9] "0" "1" "2" "3" ...
```

```
  .. ..$ : chr [1:9] "0" "1" "2" "3" ...
  .. ..$ : chr [1:536] "0.0329688509639716" "0.0555592940852114" "0.0593206943292609" "0.06653457953
 $ cov        : num [1:81, 1:81, 1:536] 1.08e-05 0.00 0.00 0.00 0.00 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ : chr [1:81, 1] "0 0" "1 0" "2 0" "3 0" ...
  .. ..$ : chr [1:81, 1] "0 0" "1 0" "2 0" "3 0" ...
  .. ..$ : chr [1:536] "0.0329688509639716" "0.0555592940852114" "0.0593206943292609" "0.06653457953
 $ time       : num [1:536] 0.033 0.0556 0.0593 0.0665 0.0667 ...
 $ s          : num 0
 $ t          : num 21.1
 $ trans      :'data.frame':       8 obs. of  2 variables:
  ..$ from: Factor w/ 4 levels "0","2","4","6": 1 1 2 2 3 3 4 4
  ..$ to  : Factor w/ 8 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8
 $ state.names: chr [1:9] "0" "1" "2" "3" ...
 $ cens.name  : chr "cens"
 $ n.risk     : int [1:536, 1:4] 304 303 302 301 300 299 298 297 296 295 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:4] "0" "2" "4" "6"
 $ n.event    : int [1:9, 1:9, 1:536] 0 0 0 0 0 0 0 0 0 1 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ : chr [1:9] "0" "1" "2" "3" ...
  .. ..$ : chr [1:9] "0" "1" "2" "3" ...
  .. ..$ : chr [1:536] "0.0329688509639716" "0.0555592940852114" "0.0593206943292609" "0.06653457953
 $ delta.na   : num [1:9, 1:9, 1:536] -0.00329 0 0 0 0 ...
 $ ind.n.risk : num 5
 - attr(*, "class")= chr "etm"
```

```
> ### Computation of the clfs + var clfs
> clfs <- dli.etm$est["0", "0", ] +
+         dli.etm$est["0", "6", ]
> var.clfs <- dli.etm$cov["0 0", "0 0", ] +
+             dli.etm$cov["0 6", "0 6", ] +
+         2 * dli.etm$cov["0 0", "0 6", ]
> ## computation of the 95% CIs + plot
> ciplus  <- clfs + qnorm(0.975) * sqrt(var.clfs)
> cimoins <- clfs - qnorm(0.975) * sqrt(var.clfs)
> plot(dli.etm$time, clfs, type = "s", lwd=3,
+      bty = "n", ylim = c(0, 1), yaxs="i", las=1,
+      xlab = "Time since 1st remission (years)",
+      ylab = "P(CLFS)" )
> lines(dli.etm$time, cimoins, lty = 3, type = "s")
> lines(dli.etm$time, ciplus , lty = 3, type = "s")
> matlines( pr.pt, t(CLFS), lty=c(1,3,3), lwd=c(3,1,1), col="red" )
```

From figure 9 it is pretty clear that the non-parametric approach gives qualitatively the same result as the parametric approach, but also that the non-parametric modeling represents an over-modeling of the shape of the probability of CLFS which is not biologically credible and hence in general should be avoided.

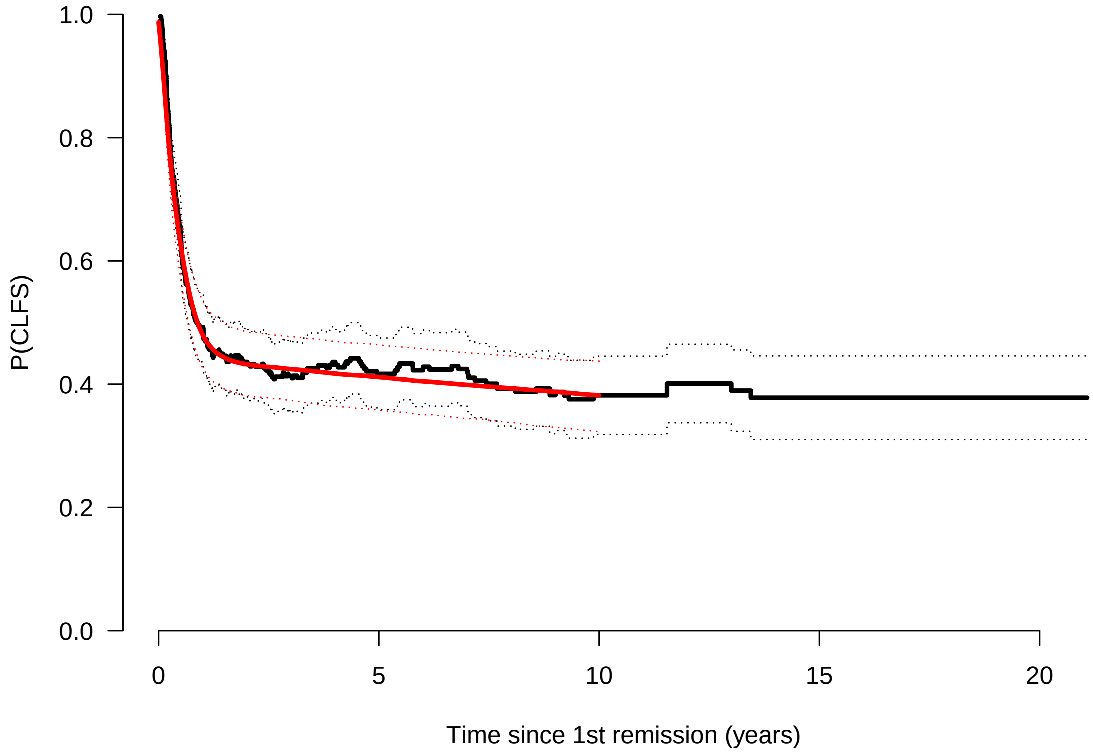

Figure 9: *Estimated probability of CLFS, with confidence bands based on simulation from an assumed normal distribution of the estimates (red curve), compared with the non-parametric curve for the* `etm` *package.*

7 Linking the Epi and etm packages.

For the sake of completeness we have included a function in Epi that automatically converts a Lexis object to a data frame of the relevant structure for input into the etm package. Note that even though there is an etm method for Lexis objects, we must call etm.Lexis explicitly, because we loaded the etm package *after* the Epi package, and thus Epi:::etm will be masked by etm:::etm. Instead we could call Epi:::etm explicitly, but that would be really goofy looking code.

```
> xdli.etm <- etm.Lexis( dli )
> str( xdli.etm )
List of 12
 $ est        : num [1:9, 1:9, 1:540] 0.997 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ : chr [1:9] "Rem" "D/Rem" "Rel" "D/Rel" ...
  .. ..$ : chr [1:9] "Rem" "D/Rem" "Rel" "D/Rel" ...
  .. ..$ : chr [1:540] "0.0329688509639716" "0.0555592940852114" "0.0593206943292609" "0.06653457953
 $ cov        : num [1:81, 1:81, 1:540] 1.08e-05 0.00 0.00 0.00 0.00 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ : chr [1:81, 1] "Rem Rem" "D/Rem Rem" "Rel Rem" "D/Rel Rem" ...
  .. ..$ : chr [1:81, 1] "Rem Rem" "D/Rem Rem" "Rel Rem" "D/Rel Rem" ...
  .. ..$ : chr [1:540] "0.0329688509639716" "0.0555592940852114" "0.0593206943292609" "0.06653457953
 $ time       : num [1:540] 0.033 0.0556 0.0593 0.0665 0.0667 ...
 $ s          : num 0
 $ t          : num 21.1
 $ trans      :'data.frame':        8 obs. of  2 variables:
  ..$ from: Factor w/ 4 levels "DLI","Rel","Rem",..: 3 3 2 2 1 1 4 4
  ..$ to  : Factor w/ 8 levels "D/DLI","DLI",..: 4 6 3 2 1 8 5 7
 $ state.names: chr [1:9] "Rem" "D/Rem" "Rel" "D/Rel" ...
 $ cens.name  : chr "cens"
 $ n.risk     : int [1:540, 1:4] 304 303 302 301 300 299 298 297 296 295 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:4] "Rem" "Rel" "DLI" "Rem2"
 $ n.event    : int [1:9, 1:9, 1:540] 0 0 0 0 0 0 0 0 0 1 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ : chr [1:9] "Rem" "D/Rem" "Rel" "D/Rel" ...
  .. ..$ : chr [1:9] "Rem" "D/Rem" "Rel" "D/Rel" ...
  .. ..$ : chr [1:540] "0.0329688509639716" "0.0555592940852114" "0.0593206943292609" "0.06653457953
 $ delta.na   : num [1:9, 1:9, 1:540] -0.00329 0 0 0 0 ...
 $ ind.n.risk : num 5
 - attr(*, "class")= chr "etm"
```

```
> plot( xdli.etm, col=rainbow(15), lty=1, lwd=3,
+       legend.pos="topright", bty="n", yaxs="i" )
```

The resulting graph is shown in figure 10

References

[1] Arthur Allignol, Martin Schumacher, and Jan Beyersmann. Empirical transition matrix of multi-state models: The etm package. *Journal of Statistical Software*, 38(4):1–15, 1 2011.

[2] Bendix Carstensen and Martyn Plummer. Using Lexis objects for multi-state models in R. *Journal of Statistical Software*, 38(6):1–18, 1 2011.

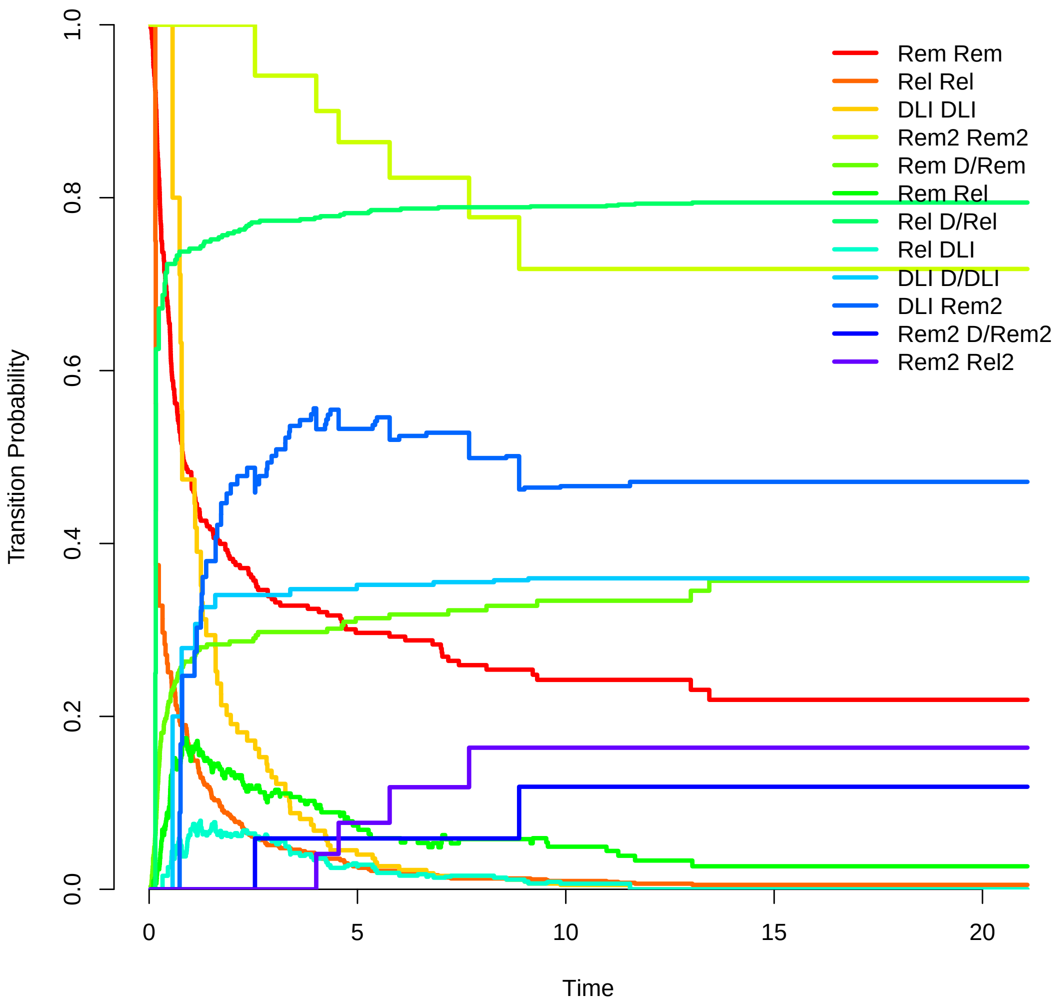

Figure 10: *Estimated transition probabilities between states using the* **etm.Lexis** *function.*

[3] Martyn Plummer and Bendix Carstensen. Lexis: An R class for epidemiological studies with long-term follow-up. *Journal of Statistical Software*, 38(5):1–12, 1 2011.