Multistate models in Epi

Version 1
(compiled Thursday 24th September, 2009, 00:28)

September 2009

www.biostat.ku.dk/~bxc/MS/

Bendix Carstensen Steno Diabetes Center, Gentofte, Denmark
& Department of Biostatistics, University of Copenhagen
bxc@steno.dk
http://www.biostat.ku.dk/~bxc/

Contents

1 Introduction

The following exposition of multistate manipulations follows (parts of) the exposition by Hein Putter which in turn is a computational companion for R to the tutorial by Putter *et al.* [?] from Stat. Med. 2007.

2 The example data

The dataset used is about the clinical course of bone-marrow transplant patients. The major event of interest is relapse or death (RD), and an intermediate event is platelet recovery (PR).

We first show the current version and then load the Epi and the mstate packages:

```
> options( "width" )

$width
[1] 80

> options( width=120 )
> R.version[c("platform","version.string")]


               _
platform       i386-pc-mingw32
version.string R version 2.9.2 (2009-08-24)

> installed.packages()[c("Epi","mstate"),c("Version","Built")]

       Version Built
Epi    "1.1.7" "2.9.2"
mstate "0.2.1" "2.9.1"

> library(Epi)
> library(mstate)
```

From the mstate package we get the ebmt3 dataset:

```
> data( ebmt3 )
> str( ebmt3 )

'data.frame':        2204 obs. of  9 variables:
 $ id     : int  1 2 3 4 5 6 7 8 9 10 ...
 $ prtime : num  23 35 26 22 29 ...
 $ prstat : int  1 1 1 1 1 1 1 1 0 0 ...
 $ rfstime: num  744 360 135 995 422 ...
 $ rfsstat: int  0 1 1 0 1 1 0 0 0 1 ...
 $ dissub : Factor w/ 3 levels "AML","ALL","CML": 3 3 3 1 1 2 3 2 1 3 ...
 $ age    : Factor w/ 3 levels "<=20","20-40",..: 3 3 3 2 2 3 2 1 2 3 ...
 $ drmatch: Factor w/ 2 levels "No gender mismatch",..: 2 1 1 1 1 1 2 1 1 1 ...
 $ tcd    : Factor w/ 2 levels "No TCD","TCD": 1 1 1 1 1 1 1 1 1 1 ...

> head( ebmt3 )
```

```
  id prtime prstat rfstime rfsstat dissub   age            drmatch    tcd
1  1     23      1     744       0    CML   >40    Gender mismatch No TCD
2  2     35      1     360       1    CML   >40 No gender mismatch No TCD
3  3     26      1     135       1    CML   >40 No gender mismatch No TCD
4  4     22      1     995       0    AML 20-40 No gender mismatch No TCD
5  5     29      1     422       1    AML 20-40 No gender mismatch No TCD
6  6     38      1     119       1    ALL   >40 No gender mismatch No TCD
```

The times are in days since transplant (`Tx`), they are time to relapse or death (`RD`) in the variable `rfstime`, and time to platelet recovery (`PR`) in the variable `prtime`. `PR` is considered an intermediate state.

Some of the platelet recovery times are larger than the recurrence times; these are (supposedly) instances where platelet recovery occurs *after* a relapse. However death and relapse are not distinguished in the available dataset.

```
> with(ebmt3,plot(prtime,rfstime,pch=16,col=c("gray","red")[prstat+1]))
```

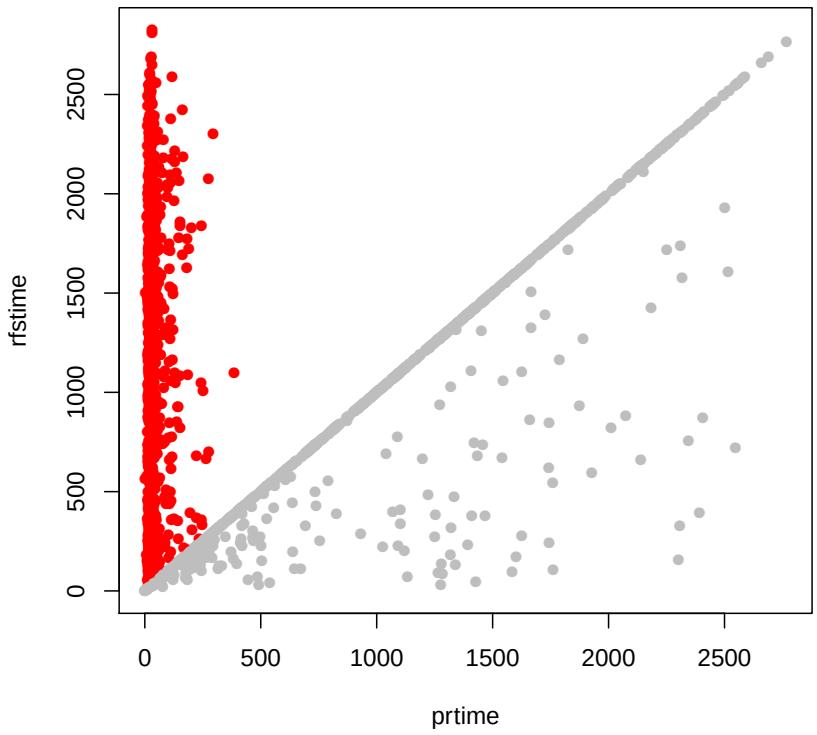

Figure 1: *The two time variables, color coded by the value of* `prstat`*, 0 is gray, 1 is red.*

3 A `Lexis` object

A convenient way of storing the information on the multistate structure is a `Lexis` object as supported by the `Epi` package. A `Lexis` object is basically a data frame with some extra

columns added. Each row represents a piece of follow-up, with indication of starting point on the timescale(s) and length of follow-up. The former are variables for each timescale, the latter (which is common on all time scales) in the variable (`lex.dur` (`dur`ation). The state in which the follow-up takes place is called `lex.Cst` (`C`urrent `st`ate) and the state occupied immediately after the follow-up `lex.Xst` (`X`it `st`ate). This type of object is constructed using the `Lexis` function.

The entry and exit times are given as named lists with times on the timescales of interest. For one of these, only one timescale need be specified, since the duration (length of follow-up) is the same on all timescales. One of these can optionally be replaced by duration.

In this study everyone enters at time 0, so we need not specify the argument `entry` to the `Lexis` function (it is by default set to 0 of only either `exit` or `duration` is given). Moreover, since everyone starts in state `Tx`, we just need to specify this state as the first level of the state factor, in order to have everyone assigned this as entry state by default:

```
> bmt <- Lexis(  exit = list(tft=rfstime/365.25),
+         exit.status = factor(rfsstat,labels=c("Tx","RD")),
+                data = ebmt3 )

NOTE: entry.status has been set to "Tx" for all.
NOTE: entry is assumed to be 0 on the tft timescale.

> str( bmt )

Classes 'Lexis' and 'data.frame':        2204 obs. of  14 variables:
 $ tft    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ lex.dur: num  2.037 0.986 0.37 2.724 1.155 ...
 $ lex.Cst: Factor w/ 2 levels "Tx","RD": 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Xst: Factor w/ 2 levels "Tx","RD": 1 2 2 1 2 2 1 1 1 2 ...
 $ lex.id : int  1 2 3 4 5 6 7 8 9 10 ...
 $ id     : int  1 2 3 4 5 6 7 8 9 10 ...
 $ prtime : num  23 35 26 22 29 ...
 $ prstat : int  1 1 1 1 1 1 1 1 0 0 ...
 $ rfstime: num  744 360 135 995 422 ...
 $ rfsstat: int  0 1 1 0 1 1 0 0 0 1 ...
 $ dissub : Factor w/ 3 levels "AML","ALL","CML": 3 3 3 1 1 2 3 2 1 3 ...
 $ age    : Factor w/ 3 levels "<=20","20-40",..: 3 3 3 2 2 3 2 1 2 3 ...
 $ drmatch: Factor w/ 2 levels "No gender mismatch",..: 2 1 1 1 1 1 2 1 1 1 ...
 $ tcd    : Factor w/ 2 levels "No TCD","TCD": 1 1 1 1 1 1 1 1 1 1 ...
 - attr(*, "time.scales")= chr "tft"
 - attr(*, "breaks")=List of 1
  ..$ tft: NULL
```

The time scaling was redefined to years on the fly, but we only defined the two-state model with transition from `Tx` to `RD`. This simple structure is shown by the `summary` command:

```
> summary( bmt )

Transitions:
     To
From   Tx  RD  Records:  Events: Risk time:  Persons:
  Tx 1363 841      2204      841    5612.46      2204

Rates:
     To
From Tx   RD Total
  Tx  0 0.15  0.15
```

Thus we have 841 events during 5612 person-years, corresponding to a rate of 0.15 events per year.

We have only one timescale in operation, so the default plot is a so-called 1-dimensional Lexis diagram:

```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( bmt )
```

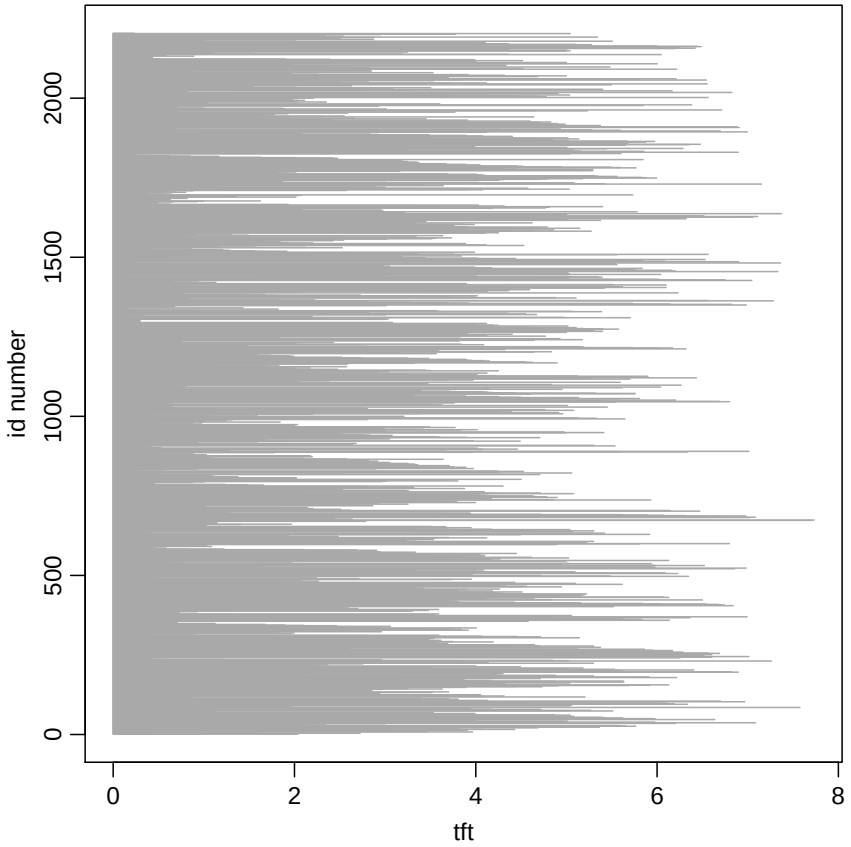

Figure 2: *One-dimensional Lexis diagram of the bone-marrow transplant data from* `ebmt3`.

Now we want to introduce the intermediate event platelet recovery, `PR`. We have the date of recovery in the variable `prtime`. Since all the censored values for `prtime` are after the relapse/death time, we could just use this variable. However, for the sake of the argument we first set the censored values of `prtime` to `NA`:

```
> bmt$prtime[bmt$prstat==0] <- NA
```

Now we can introduce the new state, and subdivide the follow-up time of the patients according to platelet recovery. We must of course specify the name of the new state in which patients enter at `prtime`, but also the so called precursor states, that is states that are overridden by the new state. This is only `Tx` because the `RD` state is absorbing. In practice it means that if a person exits the study in state `Tx` (any precursor state), then an entry to the new state (`PR`) will mean that the person exits in the new state, whereas if the person exits the study to state `RD` (any non-precursor state) the exit from the study will still be to this (original) state, regardless of whether the intermediate event has occurred or not:

```
> bmtr <- cutLexis( bmt, cut=bmt$prtime/365.25,
+                        pre="Tx",
+                        new.state="PR" )
> str( bmtr )

Classes 'Lexis' and 'data.frame':        3373 obs. of  14 variables:
 $ tft    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ lex.dur: num  0.063 0.0958 0.0712 0.0602 0.0794 ...
 $ lex.Cst: Factor w/ 3 levels "Tx","PR","RD": 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Xst: Factor w/ 3 levels "Tx","PR","RD": 2 2 2 2 2 2 2 2 1 3 ...
 $ lex.id : int  1 2 3 4 5 6 7 8 9 10 ...
 $ id     : int  1 2 3 4 5 6 7 8 9 10 ...
 $ prtime : num  23 35 26 22 29 38 30 35 NA NA ...
 $ prstat : int  1 1 1 1 1 1 1 1 0 0 ...
 $ rfstime: num  744 360 135 995 422 ...
 $ rfsstat: int  0 1 1 0 1 1 0 0 0 1 ...
 $ dissub : Factor w/ 3 levels "AML","ALL","CML": 3 3 3 1 1 2 3 2 1 3 ...
 $ age    : Factor w/ 3 levels "<=20","20-40",..: 3 3 3 2 2 3 2 1 2 3 ...
 $ drmatch: Factor w/ 2 levels "No gender mismatch",..: 2 1 1 1 1 1 2 1 1 1 ...
 $ tcd    : Factor w/ 2 levels "No TCD","TCD": 1 1 1 1 1 1 1 1 1 1 ...
 - attr(*, "time.scales")= chr "tft"
 - attr(*, "breaks")=List of 1
  ..$ tft: NULL

> summary( bmtr )

Transitions:
     To
From   Tx   PR  RD  Records:  Events: Risk time:  Persons:
  Tx  577 1169 458      2204     1627    2439.43      2204
  PR    0  786 383      1169      383    3173.03      1169
  Sum 577 1955 841      3373     2010    5612.46      2204

Rates:
     To
From Tx   PR   RD Total
  Tx  0 0.48 0.19  0.67
  PR  0 0.00 0.12  0.12
```

We see that we now have 3373 records instead of 2204, because of the 1169 transitions to PR. We still have the same amount of risk time (5612 person-years) and deaths (841) in total.

By default the newly created state is placed after the precursor states nominated, and before all non-procursor states.

3.1 Clock-back approach

We may be interested in the "clock back" approach, that is using the *duration* of platelet recovery as timescale. This can be accomplished by the `new.scale` argument to `cutLexis`. If set to `TRUE` a new timescale will generated, named `PR.dur`; alternatively we can set the argument to the name of the new timescale:

```
> bmtr <- cutLexis( bmt, cut=bmt$prtime/365.25,
+                        pre="Tx",
+                        new.state="PR", new.scale="tfPR" )
> str( bmtr )
```

```
Classes 'Lexis' and 'data.frame':        3373 obs. of  15 variables:
 $ tft    : num  0 0.063 0 0.0958 0 ...
 $ tfPR   : num  NA 0 NA 0 NA 0 NA 0 NA 0 ...
 $ lex.dur: num  0.063 1.974 0.0958 0.8898 0.0712 ...
 $ lex.Cst: Factor w/ 3 levels "Tx","PR","RD": 1 2 1 2 1 2 1 2 1 2 ...
 $ lex.Xst: Factor w/ 3 levels "Tx","PR","RD": 2 2 2 3 2 3 2 2 2 3 ...
 $ lex.id : int  1 1 2 2 3 3 4 4 5 5 ...
 $ id     : int  1 1 2 2 3 3 4 4 5 5 ...
 $ prtime : num  23 23 35 35 26 26 22 22 29 29 ...
 $ prstat : int  1 1 1 1 1 1 1 1 1 1 ...
 $ rfstime: num  744 744 360 360 135 135 995 995 422 422 ...
 $ rfsstat: int  0 0 1 1 1 1 0 0 1 1 ...
 $ dissub : Factor w/ 3 levels "AML","ALL","CML": 3 3 3 3 3 3 1 1 1 1 ...
 $ age    : Factor w/ 3 levels "<=20","20-40",..: 3 3 3 3 3 3 2 2 2 2 ...
 $ drmatch: Factor w/ 2 levels "No gender mismatch",..: 2 2 1 1 1 1 1 1 1 1 ...
 $ tcd    : Factor w/ 2 levels "No TCD","TCD": 1 1 1 1 1 1 1 1 1 1 ...
 - attr(*, "time.scales")= chr  "tft" "tfPR"
 - attr(*, "breaks")=List of 2
  ..$ tft : NULL
  ..$ tfPR: NULL
```

```
> summary( bmtr )

Transitions:
     To
From   Tx   PR  RD  Records:  Events: Risk time:  Persons:
  Tx  577 1169 458      2204     1627    2439.43      2204
  PR    0  786 383      1169      383    3173.03      1169
  Sum 577 1955 841      3373     2010    5612.46      2204

Rates:
     To
From Tx   PR   RD Total
  Tx  0 0.48 0.19  0.67
  PR  0 0.00 0.12  0.12
```

It is seen that this representation retains the total amount of risk time in the study; it merely subdivides it by state. It is seen that the output of the `summary.Lexis` function is pretty similar to the output from the `events` function from the `mstate` package.

Moreover, since we now have two timescales we can make a two-dimensional Lexis diagram, as shown in figure 1. The diagram only shows follow-up *after* `PR`, since one of the timescales is time since `PR`, which obviously is not defined before the event.

```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( bmtr )
```

3.2 Subdividing states

We may also be interested in subdividing the states subsequent to `PR` according to whether this event has occurred or not. In this example only `RD` is subsequent to the event `PR`:

```
> bmtr <- cutLexis( bmt, cut=bmt$prtime/365.25,
+                        pre="Tx",
+                        new.state="PR", new.scale="tfPR",
+                        split.states=TRUE )
> summary( bmtr )
```

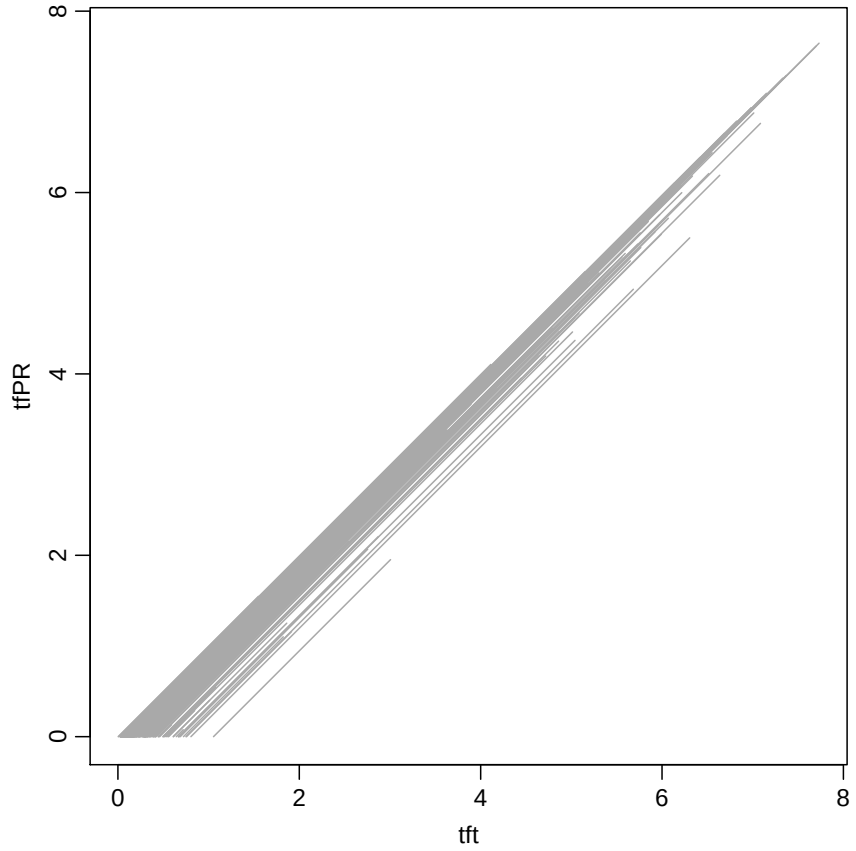

Figure 3: *Two-dimensional Lexis diagram of the bone-marrow transplant data from* `ebmt3`. *Note that this diagram only shows follow-up* after `PR`, *as the timescale* `tfPR` *is undefined (=missing) for follow-up prior to* `PR`.

```
Transitions:
     To
From   Tx   PR  RD RD(PR)  Records:  Events: Risk time:  Persons:
  Tx  577 1169 458      0      2204     1627    2439.43      2204
  PR    0  786   0    383      1169      383    3173.03      1169
  Sum 577 1955 458    383      3373     2010    5612.46      2204

Rates:
     To
From Tx   PR   RD RD(PR) Total
  Tx  0 0.48 0.19   0.00  0.67
  PR  0 0.00 0.00   0.12  0.12
```

In practical analyses this will have no effect, but it is useful in the drawing of multistate models as box-diagrams as shown in the next section. It can also be of interest if simulation-based inference on occupation probabilities for various states are of interest.

4 Illustrating the model

In order to illustrate the model in a box diagram, we can use the interactive function `boxes.Lexis`, which will prompt you for a click for the position of each of the states, and subsequently draw arrows between those states where transitions occur. If we put `boxpos=TRUE` (the default is `FALSE`), the states are put in a circular arrangement which usually is not very nice:

```
> boxes( bmtr, boxpos=TRUE )
```

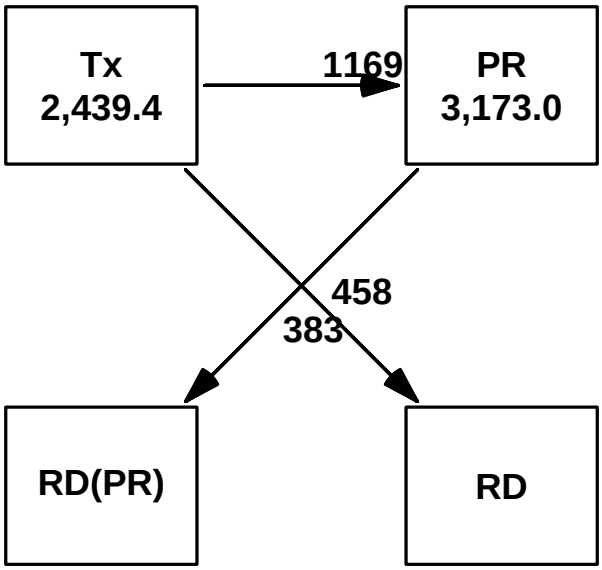

Figure 4: *Illustration of the transitions between defined states for the* `ebmt3` *data, using the defualt layout.*

That can be improved by using the interactive facility, using the default `boxpos=FALSE`, which will prompt you to click on the screen to place the various boxes.

Frequently one would need the exact same plot with slight modifications, for example with one or more of the arrows in a different color. To this end it is possible to use the argument `file=` to specify a file for the code generating the display. This can then be modified:

```
> boxes( bmtr, file="bmt-boxes.r" )
```

The following is the content of the file `bmt-boxes.r`, generated interactively (the annotaion of the code comes from the `boxes.Lexis` function), and below the resulting plot:

```
> obj <-  bmtr
> if( inherits(obj,"Lexis") ) tm <- tmat.Lexis( obj ) else tm <- obj
> ### Position of the boxes:
>       xx <- c( 16, 75, 24, 82 )
>       yy <- c( 71, 78, 21, 26 )
>      cex <- 1.5       # How should text and numbers be scaled
>    wmult <- 1.5     # Extra box-width relative to string width
>    hmult <- 2.25     # Extra box-height relative to string height
>    eq.wd <- TRUE     # All boxes the same width
>    eq.ht <- TRUE     # All boxes the same height
>   show.Y <- TRUE    # Show number of person-years in boxes
>  scale.Y <- 1   # How should person-years be scaled
> digits.Y <- 1  # How should person-years be printed
>   show.D <- TRUE    # Show number of events on arrows
>  scale.D <- FALSE   # How should rates be scaled
> digits.D <- 0  # How should rates be printed
>                       st.nam <- colnames( tm )
> if( is.null(st.nam) ) st.nam <- paste(1:ncol(tm))
>             pl.nam <- st.nam
>       n.st <- length( st.nam )
> # If we want to show person-years and events / rates
>   SM <- summary(obj,simplify=FALSE,scale=scale.Y)
>   Y <- SM[[1]][-n.st-1,"Risk time:"]
>   D <- SM[[1+as.logical(scale.D)]][1:n.st,1:n.st] * ifelse(scale.D,scale.D,1)
> # No extra line with person-years when they are NA
> # and adjust the printing of the person-Years
> if( show.Y )
+   {
+   pl.nam <- gsub( "\\\nNA", "",
+                   paste( st.nam,
+                          formatC( Y, format="f", digits= 1 ,
+                                   big.mark=","),
+                          sep="\n" ) )
+   }
> ######################################################################
> # Here comes the plot
> par( mar=c(0,0,0,0), cex=cex )
> plot( NA, bty="n",
+       xlim=0:1*100, ylim=0:1*100, xaxt="n", yaxt="n", xlab="", ylab="" )
> # String height and width only meaningful after a plot has been called
> ht <- strheight( pl.nam ) * hmult
> wd <- strwidth(  pl.nam ) * wmult
> # Should all boxes be of same height / width:
> if( eq.ht ) ht <- rep( max(ht), length(ht) )
> if( eq.wd ) wd <- rep( max(wd), length(wd) )
> # Plot the boxes
> b <- list()
> for( i in 1:n.st )
+    b[[i]] <- tbox( pl.nam[i], xx[i], yy[i], wd[i], ht[i] )
> # Plot the arrows and the text along them
> for( i in 1:n.st ) for( j in 1:n.st )
+    {
+    if( !is.na(tm[i,j]) )
+      {
+      arr <- boxarr( b[[i]], b[[j]], offset=!is.na(tm[j,i]) )
+      if( show.D ) text( arr$x-arr$d[2], arr$y+arr$d[1],
+                         formatC( D[i,j], format="f", digits=digits.D ),
```

```
+                          adj=as.numeric(c(arr$d[2]>0,arr$d[1]<0)),
+                          font=2, col="black" )
+      }
+    }
> # Redraw the boxes with white background to move the arrows behind them
> for( i in 1:n.st ) tbox( pl.nam[i], xx[i], yy[i], wd[i], ht[i], col="white" )
```

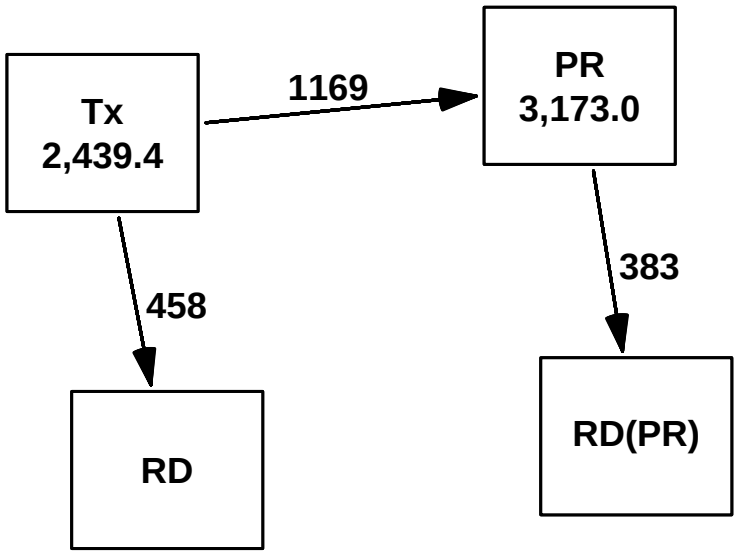

Figure 5: *Illustration of the transitions between defined states for the* `ebmt3` *data.*

The first argument to the `boxes` function is a `Lexis` object, and so the defult behaviour is to show the risk time (person-years) in those boxes where they are accumulated, and the number of transitions along the arrows. The first argument may also be a transition matrix, in which case only the boxes and arrows are drawn.

5 Analysis of rates

In the first instance we use a Cox-model for the analysis of the three rates. We can either analyse the transitions separately by selecting the appropriate rows and choosing the right event-type. We see that there is minimal difference between the codes; it is only the failure type (from `lex.Xst`) and the subset (from `lex.Cst`) that is different:

```
> m.Tx.PR <- coxph( Surv( tft, tft+lex.dur, lex.Xst=="PR" ) ~
+                   dissub + age + drmatch + tcd,
+                   data = subset(bmtr, lex.Cst=="Tx"), method="breslow" )
> m.Tx.RD <- coxph( Surv( tft, tft+lex.dur, lex.Xst=="RD" ) ~
+                   dissub + age + drmatch + tcd,
+                   data = subset(bmtr, lex.Cst=="Tx"), method="breslow" )
> m.PR.RD <- coxph( Surv( tft, tft+lex.dur, lex.Xst=="RD(PR)" ) ~
```

```
+                   dissub + age + drmatch + tcd,
+                   data = subset(bmtr, lex.Cst=="PR"), method="breslow" )
```

We can stack these sets of parameters nicely if we want, using the facilities in `ci.lin`:

```
> round( rbind( ci.lin( m.Tx.PR, E=T ),
+               ci.lin( m.Tx.RD, E=T ),
+               ci.lin( m.PR.RD, E=T ) ), 3 )

                      Estimate StdErr      z     P exp(Est.)  2.5% 97.5%
dissubALL               -0.044  0.078 -0.560 0.576     0.957 0.822 1.115
dissubCML               -0.297  0.068 -4.371 0.000     0.743 0.650 0.849
age20-40                -0.165  0.079 -2.082 0.037     0.848 0.726 0.990
age>40                  -0.090  0.086 -1.038 0.299     0.914 0.772 1.083
drmatchGender mismatch   0.046  0.067  0.687 0.492     1.047 0.919 1.193
tcdTCD                   0.429  0.080  5.335 0.000     1.536 1.312 1.798
dissubALL                0.256  0.135  1.893 0.058     1.292 0.991 1.684
dissubCML                0.017  0.108  0.155 0.877     1.017 0.822 1.258
age20-40                 0.255  0.151  1.689 0.091     1.291 0.960 1.735
age>40                   0.526  0.158  3.334 0.001     1.693 1.242 2.307
drmatchGender mismatch  -0.075  0.110 -0.682 0.495     0.928 0.747 1.151
tcdTCD                   0.297  0.150  1.977 0.048     1.345 1.003 1.806
dissubALL                0.137  0.148  0.923 0.356     1.146 0.858 1.532
dissubCML                0.247  0.117  2.115 0.034     1.280 1.018 1.610
age20-40                 0.061  0.153  0.400 0.689     1.063 0.787 1.436
age>40                   0.581  0.160  3.627 0.000     1.788 1.306 2.447
drmatchGender mismatch   0.173  0.115  1.510 0.131     1.189 0.950 1.488
tcdTCD                   0.201  0.126  1.589 0.112     1.222 0.954 1.566
```

However, in order to analyse the transition rates using joint parameters between transitions, we must do the analyses based on a stacked dataset; that is a dataset which is made by stacking the three subsets we used in the three separate analyses above. Note that these three sets are not disjoint subsets, and moreover the response variable (event indicator) is not the same in all three.

This can conveniently be constructed by using the `stack.Lexis` function:

```
> bmts <- stack( bmtr )
> str( bmts )

Classes 'stacked.Lexis' and 'data.frame':        5577 obs. of  17 variables:
 $ tft     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ tfPR    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ lex.dur : num  0.063 0.0958 0.0712 0.0602 0.0794 ...
 $ lex.Cst : Factor w/ 4 levels "Tx","PR","RD",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Xst : Factor w/ 4 levels "Tx","PR","RD",..: 2 2 2 2 2 2 2 2 1 3 ...
 $ lex.Tr  : Factor w/ 3 levels "Tx->PR","Tx->RD",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Fail: logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
 $ lex.id  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ prtime  : num  23 35 26 22 29 38 30 35 NA NA ...
 $ prstat  : int  1 1 1 1 1 1 1 1 0 0 ...
 $ rfstime : num  744 360 135 995 422 ...
 $ rfsstat : int  0 1 1 0 1 1 0 0 0 1 ...
 $ dissub  : Factor w/ 3 levels "AML","ALL","CML": 3 3 3 1 1 2 3 2 1 3 ...
 $ age     : Factor w/ 3 levels "<=20","20-40",..: 3 3 3 2 2 3 2 1 2 3 ...
 $ drmatch : Factor w/ 2 levels "No gender mismatch",..: 2 1 1 1 1 1 2 1 1 1 ...
 $ tcd     : Factor w/ 2 levels "No TCD","TCD": 1 1 1 1 1 1 1 1 1 1 ...
```

We see that two extra variables `lex.Tr` and `lex.Fail` have been added. The `lex.Tr` is a factor indicating the transition, and `lex.Fail` is the (logical) failure indicator for the transition.

Since the variables `lex.Cst` and `lex.Xst` have lost their meaning, the resulting object is not a `Lexis` object.

We can now reproduce the results from above by using the stacked dataset and the interactions with `lex.Tr` — note that the interaction with time is given in the `strata()` argument in the model formula:

```
> c1 <- coxph( Surv(tft,tft+lex.dur,lex.Fail) ~
+              lex.Tr:( dissub + age + drmatch + tcd ) +
+              strata( lex.Tr ),
+              data=bmts, method="breslow" )
> c1

Call:
coxph(formula = Surv(tft, tft + lex.dur, lex.Fail) ~ lex.Tr:(dissub +
    age + drmatch + tcd) + strata(lex.Tr), data = bmts, method = "breslow")


                                      coef exp(coef) se(coef)      z       p
lex.TrTx->PR:dissubAML              0.2972     1.346   0.0680  4.371 1.2e-05
lex.TrTx->RD:dissubAML             -0.0167     0.983   0.1084 -0.155 8.8e-01
lex.TrPR->RD(PR):dissubAML         -0.2471     0.781   0.1169 -2.115 3.4e-02
lex.TrTx->PR:dissubALL              0.2536     1.289   0.0830  3.057 2.2e-03
lex.TrTx->RD:dissubALL              0.2391     1.270   0.1313  1.821 6.9e-02
lex.TrPR->RD(PR):dissubALL         -0.1105     0.895   0.1491 -0.741 4.6e-01
lex.TrTx->PR:dissubCML                 NA        NA   0.0000     NA      NA
lex.TrTx->RD:dissubCML                 NA        NA   0.0000     NA      NA
lex.TrPR->RD(PR):dissubCML             NA        NA   0.0000     NA      NA
lex.TrTx->PR:age20-40              -0.1646     0.848   0.0791 -2.082 3.7e-02
lex.TrTx->RD:age20-40               0.2552     1.291   0.1510  1.689 9.1e-02
lex.TrPR->RD(PR):age20-40           0.0614     1.063   0.1534  0.400 6.9e-01
lex.TrTx->PR:age>40                -0.0898     0.914   0.0865 -1.038 3.0e-01
lex.TrTx->RD:age>40                 0.5265     1.693   0.1579  3.334 8.6e-04
lex.TrPR->RD(PR):age>40             0.5809     1.788   0.1601  3.627 2.9e-04
lex.TrTx->PR:drmatchGender mismatch 0.0458     1.047   0.0666  0.687 4.9e-01
lex.TrTx->RD:drmatchGender mismatch -0.0753     0.928   0.1103 -0.682 5.0e-01
lex.TrPR->RD(PR):drmatchGender mismatch 0.1729 1.189   0.1145  1.510 1.3e-01
lex.TrTx->PR:tcdTCD                 0.4291     1.536   0.0804  5.335 9.6e-08
lex.TrTx->RD:tcdTCD                 0.2967     1.345   0.1501  1.977 4.8e-02
lex.TrPR->RD(PR):tcdTCD             0.2007     1.222   0.1264  1.589 1.1e-01

Likelihood ratio test=118  on 18 df, p=1.11e-16  n= 5577
```

We see that the default model matrix set-up for interactions generates an interaction assuming that there is an intercept. Which there indeed is not in a Cox-model. Hence we get aliased parameters when all three levels of `dissub` have interaction columns with `lex.Tr` generated.

It gives the correct model, but the reference level for the factor `dissub` is the last rather than the first. This can be remedied by explicitly tampering with the model matrix before fitting the model; we just generate the model matrix and then remove the columns relating to the reference level. This has the advantage that we maintain the annotation of the parameters:

```
> mm <- model.matrix( ~ lex.Tr:( dissub + age + drmatch + tcd ), bmts )
> rm <- grep( "AML", colnames(mm) )
> mm <- mm[,-c(1,rm)]
> c1 <- coxph( Surv(tft,tft+lex.dur,lex.Fail) ~
+              mm +
+              strata( lex.Tr ),
+              data=cbind(bmts,mm), method="breslow" )
> c1

Call:
coxph(formula = Surv(tft, tft + lex.dur, lex.Fail) ~ mm + strata(lex.Tr),
    data = cbind(bmts, mm), method = "breslow")


                                        coef exp(coef) se(coef)      z       p
mmlex.TrTx->PR:dissubALL               -0.0436     0.957   0.0779 -0.560 5.8e-01
mmlex.TrTx->RD:dissubALL                0.2559     1.292   0.1352  1.893 5.8e-02
mmlex.TrPR->RD(PR):dissubALL            0.1366     1.146   0.1480  0.923 3.6e-01
mmlex.TrTx->PR:dissubCML               -0.2972     0.743   0.0680 -4.371 1.2e-05
mmlex.TrTx->RD:dissubCML                0.0167     1.017   0.1084  0.155 8.8e-01
mmlex.TrPR->RD(PR):dissubCML            0.2471     1.280   0.1169  2.115 3.4e-02
mmlex.TrTx->PR:age20-40                -0.1646     0.848   0.0791 -2.082 3.7e-02
mmlex.TrTx->RD:age20-40                 0.2552     1.291   0.1510  1.689 9.1e-02
mmlex.TrPR->RD(PR):age20-40             0.0614     1.063   0.1534  0.400 6.9e-01
mmlex.TrTx->PR:age>40                  -0.0898     0.914   0.0865 -1.038 3.0e-01
mmlex.TrTx->RD:age>40                   0.5265     1.693   0.1579  3.334 8.6e-04
mmlex.TrPR->RD(PR):age>40               0.5809     1.788   0.1601  3.627 2.9e-04
mmlex.TrTx->PR:drmatchGender mismatch   0.0458     1.047   0.0666  0.687 4.9e-01
mmlex.TrTx->RD:drmatchGender mismatch  -0.0753     0.928   0.1103 -0.682 5.0e-01
mmlex.TrPR->RD(PR):drmatchGender mismatch  0.1729     1.189   0.1145  1.510 1.3e-01
mmlex.TrTx->PR:tcdTCD                   0.4291     1.536   0.0804  5.335 9.6e-08
mmlex.TrTx->RD:tcdTCD                   0.2967     1.345   0.1501  1.977 4.8e-02
mmlex.TrPR->RD(PR):tcdTCD               0.2007     1.222   0.1264  1.589 1.1e-01

Likelihood ratio test=118  on 18 df, p=1.11e-16  n= 5577
```

Comparison with the estimates from the 3 separate analyses and from the `mstate` paper we can just use `ci.lin` to extract estimates, and the reshuffle the rows:

```
> round( ci.lin( c1, E=T )[c(0:5*3+1,0:5*3+2,0:5*3+3),-(3:4)], 3 )

                                       Estimate StdErr exp(Est.)  2.5% 97.5%
mmlex.TrTx->PR:dissubALL                 -0.044  0.078     0.957 0.822 1.115
mmlex.TrTx->PR:dissubCML                 -0.297  0.068     0.743 0.650 0.849
mmlex.TrTx->PR:age20-40                  -0.165  0.079     0.848 0.726 0.990
mmlex.TrTx->PR:age>40                    -0.090  0.086     0.914 0.772 1.083
mmlex.TrTx->PR:drmatchGender mismatch     0.046  0.067     1.047 0.919 1.193
mmlex.TrTx->PR:tcdTCD                     0.429  0.080     1.536 1.312 1.798
mmlex.TrTx->RD:dissubALL                  0.256  0.135     1.292 0.991 1.684
mmlex.TrTx->RD:dissubCML                  0.017  0.108     1.017 0.822 1.258
mmlex.TrTx->RD:age20-40                   0.255  0.151     1.291 0.960 1.735
mmlex.TrTx->RD:age>40                     0.526  0.158     1.693 1.242 2.307
mmlex.TrTx->RD:drmatchGender mismatch    -0.075  0.110     0.928 0.747 1.151
mmlex.TrTx->RD:tcdTCD                     0.297  0.150     1.345 1.003 1.806
mmlex.TrPR->RD(PR):dissubALL              0.137  0.148     1.146 0.858 1.532
mmlex.TrPR->RD(PR):dissubCML              0.247  0.117     1.280 1.018 1.610
mmlex.TrPR->RD(PR):age20-40               0.061  0.153     1.063 0.787 1.436
```

```
mmlex.TrPR->RD(PR):age>40                    0.581  0.160     1.788 1.306 2.447
mmlex.TrPR->RD(PR):drmatchGender mismatch    0.173  0.115     1.189 0.950 1.488
mmlex.TrPR->RD(PR):tcdTCD                    0.201  0.126     1.222 0.954 1.566
```

5.1 Joint parameters between transitions

Of course it is not of much interest to make a joint model which gives the same results as the three separate ones. The point comes from *reducing* this model to one which is more parsimonious yet sensible. One such reduction would be to assume that the two relapse/mortality rates, `Tx->RD` and `PR->RD` are proportional. This means that the Cox model should be changed so that the stratum variable only has two levels, but at the same time we should introduce an indicator of being in state `PR`. We use the `Relevel` command from `Epi` that allows grouping of factor levels, and repeat the tampering with the model matrix:

```
> bmts$inc.mort <- Relevel( bmts$lex.Tr, list(1,2:3), coll=" & " )
> with( bmts, table(lex.Tr,inc.mort) )

            inc.mort
lex.Tr       Tx->PR Tx->RD & PR->RD(PR)
  Tx->PR       2204                    0
  Tx->RD          0                 2204
  PR->RD(PR)      0                 1169

> mm <- model.matrix( ~ lex.Tr:( dissub + age + drmatch + tcd ), bmts )
> rm <- grep( "AML", colnames(mm) )
> mm <- mm[,-c(1,rm)]
> c2 <- coxph( Surv(tft,tft+lex.dur,lex.Fail) ~
+              mm + I(lex.Cst=="PR") +
+              strata( inc.mort ),
+              data=cbind(bmts,mm), method="breslow" )
```

Note that the logical expression identifying platelet recovery is wrapped in an "`I()`" in order to make `coxph` create an extra column adjacent to the supplied model matrix (if not `coxph` will crash):

```
> c2

Call:
coxph(formula = Surv(tft, tft + lex.dur, lex.Fail) ~ mm + I(lex.Cst ==
    "PR") + strata(inc.mort), data = cbind(bmts, mm), method = "breslow")


                                          coef exp(coef) se(coef)       z       p
mmlex.TrTx->PR:dissubALL                -0.04359     0.957   0.0779 -0.5597 5.8e-01
mmlex.TrTx->RD:dissubALL                 0.26100     1.298   0.1352  1.9308 5.4e-02
mmlex.TrPR->RD(PR):dissubALL             0.13993     1.150   0.1480  0.9456 3.4e-01
mmlex.TrTx->PR:dissubCML                -0.29724     0.743   0.0680 -4.3714 1.2e-05
mmlex.TrTx->RD:dissubCML                 0.00351     1.004   0.1084  0.0324 9.7e-01
mmlex.TrPR->RD(PR):dissubCML             0.25051     1.285   0.1168  2.1450 3.2e-02
mmlex.TrTx->PR:age20-40                 -0.16461     0.848   0.0791 -2.0823 3.7e-02
mmlex.TrTx->RD:age20-40                  0.25083     1.285   0.1511  1.6605 9.7e-02
mmlex.TrPR->RD(PR):age20-40              0.05554     1.057   0.1534  0.3621 7.2e-01
mmlex.TrTx->PR:age>40                   -0.08979     0.914   0.0865 -1.0384 3.0e-01
```

```
mmlex.TrTx->RD:age>40                        0.52577     1.692   0.1579  3.3299 8.7e-04
mmlex.TrPR->RD(PR):age>40                    0.56267     1.755   0.1600  3.5173 4.4e-04
mmlex.TrTx->PR:drmatchGender mismatch        0.04575     1.047   0.0666  0.6869 4.9e-01
mmlex.TrTx->RD:drmatchGender mismatch       -0.07205     0.930   0.1103 -0.6535 5.1e-01
mmlex.TrPR->RD(PR):drmatchGender mismatch    0.16939     1.185   0.1144  1.4801 1.4e-01
mmlex.TrTx->PR:tcdTCD                        0.42907     1.536   0.0804  5.3346 9.6e-08
mmlex.TrTx->RD:tcdTCD                        0.31886     1.376   0.1500  2.1262 3.3e-02
mmlex.TrPR->RD(PR):tcdTCD                    0.21103     1.235   0.1262  1.6722 9.4e-02
I(lex.Cst == "PR")TRUE                      -0.38036     0.684   0.2115 -1.7982 7.2e-02

Likelihood ratio test=136  on 19 df, p=0  n= 5577
```

6 Connecting to the `mstate` package

Alternatively we may throw away some of the state annotation and convenience in parameter labeling and use the `mstate` package:

```
> ms <- mstate( bmt )
> head( bmt )
```

```
  tft   lex.dur lex.Cst lex.Xst lex.id id prtime prstat rfstime rfsstat dissub   age
1   0 2.0369610      Tx      Tx      1  1     23      1     744       0    CML   >40    Gen
2   0 0.9856263      Tx      RD      2  2     35      1     360       1    CML   >40 No gen
3   0 0.3696099      Tx      RD      3  3     26      1     135       1    CML   >40 No gen
4   0 2.7241615      Tx      Tx      4  4     22      1     995       0    AML 20-40 No gen
5   0 1.1553730      Tx      RD      5  5     29      1     422       1    AML 20-40 No gen
6   0 0.3258042      Tx      RD      6  6     38      1     119       1    ALL   >40 No gen
```

```
> head( ms )
```

```
  id Tstart     Tstop from to trans status id.1 prtime prstat rfstime rfsstat dissub   age
1  1      0 2.0369610    1  1     1  FALSE    1     23      1     744       0    CML   >40
2  2      0 0.9856263    1  1     1   TRUE    2     35      1     360       1    CML   >40
3  3      0 0.3696099    1  1     1   TRUE    3     26      1     135       1    CML   >40
4  4      0 2.7241615    1  1     1  FALSE    4     22      1     995       0    AML 20-40
5  5      0 1.1553730    1  1     1   TRUE    5     29      1     422       1    AML 20-40
6  6      0 0.3258042    1  1     1   TRUE    6     38      1     119       1    ALL   >40
```

7 Poisson modeling

An alternative to the Cox-modeling of the rates is to use a Poisson model; i.e. to assume that rates are constant in (small) intervals, and then invoke a model for the rates in these intervals that assume that the rates are constant in each interval, but that the magnitude of the rates follow some smooth function.

In order for this to make sense we must split the data along one or more of the timescales of relevance. We split along the timescale `tft` (time from transplant). This is accomplished by the function `splitLexis` (note that it is *not* called `split.Lexis`):

```
> bmtx <- splitLexis( bmtr, time.scale="tft", breaks=c(0:19/10,seq(2,10,0.5)) )
> summary( bmtx )
```

```
Transitions:
     To
From     Tx    PR  RD RD(PR)  Records:  Events: Risk time:  Persons:
  Tx  15645  1169 458      0     17272     1627    2439.43      2204
  PR      0 20405   0    383     20788      383    3173.03      1169
  Sum 15645 21574 458    383     38060     2010    5612.46      2204

Rates:
     To
From Tx   PR   RD RD(PR) Total
  Tx  0 0.48 0.19   0.00  0.67
  PR  0 0.00 0.00   0.12  0.12
```

We see that the number of events and amount of risk time is unchanged in `bmtx`; the only difference is that it is distributed across many more records than in `bmtr`:

```
> summary( bmtr )

Transitions:
     To
From   Tx   PR  RD RD(PR)  Records:  Events: Risk time:  Persons:
  Tx  577 1169 458      0      2204     1627    2439.43      2204
  PR    0  786   0    383      1169      383    3173.03      1169
  Sum 577 1955 458    383      3373     2010    5612.46      2204

Rates:
     To
From Tx   PR   RD RD(PR) Total
  Tx  0 0.48 0.19   0.00  0.67
  PR  0 0.00 0.00   0.12  0.12
```

We can now redo the rate-modeling above by using a Poisson model with a suitable smooth version of the time-effect.

```
> library( splines )
> p.Tx.PR <- glm( as.numeric( lex.Xst=="PR" ) ~
+                 ns( tft, knots=c(seq(0.2,2,0.2),3:6), Bo=c(0,7) ) +
+                 dissub + age + drmatch + tcd,
+                 family=poisson,
+                 data = subset(bmtx, lex.Cst=="Tx") )
> p.Tx.RD <- glm( as.numeric( lex.Xst=="RD" ) ~
+                 ns( tft, knots=c(seq(0.2,2,0.2),3:6), Bo=c(0,7) ) +
+                 dissub + age + drmatch + tcd,
+                 family=poisson,
+                 data = subset(bmtx, lex.Cst=="Tx") )
> p.PR.RD <- glm( as.numeric( lex.Xst=="RD(PR)" ) ~
+                 ns( tft, knots=c(seq(0.2,2,0.2),3:6), Bo=c(0,7) ) +
+                 dissub + age + drmatch + tcd,
+                 family=poisson,
+                 data = subset(bmtx, lex.Cst=="PR") )
```

We then compare the results for the regression parameters from the Poisson-model with those from the Cox-model:

```
> sb <- c("dissub","age","drmatch","tcd")
> round( rbind( ci.lin( p.Tx.PR, subset=sb, E=T ),
+               ci.lin( p.Tx.RD, subset=sb, E=T ),
+               ci.lin( p.PR.RD, subset=sb, E=T ) ), 3 )
```

```
                       Estimate StdErr      z     P exp(Est.)  2.5% 97.5%
dissubALL                -0.013  0.078 -0.165 0.869     0.987 0.847 1.150
dissubCML                -0.242  0.068 -3.557 0.000     0.785 0.687 0.897
age20-40                 -0.158  0.079 -2.001 0.045     0.854 0.731 0.997
age>40                   -0.110  0.087 -1.269 0.204     0.896 0.756 1.062
drmatchGender mismatch    0.059  0.067  0.884 0.376     1.061 0.931 1.209
tcdTCD                    0.394  0.080  4.913 0.000     1.483 1.267 1.735
dissubALL                 0.255  0.135  1.889 0.059     1.291 0.990 1.683
dissubCML                 0.036  0.108  0.330 0.742     1.036 0.838 1.282
age20-40                  0.267  0.151  1.767 0.077     1.306 0.971 1.755
age>40                    0.528  0.158  3.346 0.001     1.696 1.245 2.311
drmatchGender mismatch   -0.072  0.110 -0.655 0.512     0.930 0.749 1.155
tcdTCD                    0.256  0.150  1.707 0.088     1.292 0.963 1.734
dissubALL                 0.132  0.148  0.893 0.372     1.141 0.854 1.525
dissubCML                 0.256  0.117  2.193 0.028     1.292 1.028 1.624
age20-40                  0.066  0.153  0.430 0.667     1.068 0.791 1.442
age>40                    0.578  0.160  3.612 0.000     1.782 1.303 2.439
drmatchGender mismatch    0.164  0.115  1.436 0.151     1.179 0.942 1.475
tcdTCD                    0.206  0.126  1.631 0.103     1.229 0.959 1.574
```

It gives a better impression if we divide the two sets of regression estimates to see the relative differences in RR estimates (the last 3 columns):

```
> round( rbind( ci.lin( m.Tx.PR, E=T ),
+               ci.lin( m.Tx.RD, E=T ),
+               ci.lin( m.PR.RD, E=T ) ) /
+        rbind( ci.lin( p.Tx.PR, subset=sb, E=T ),
+               ci.lin( p.Tx.RD, subset=sb, E=T ),
+               ci.lin( p.PR.RD, subset=sb, E=T ) ), 3 )
```

```
                       Estimate StdErr     z     P exp(Est.)  2.5% 97.5%
dissubALL                 3.386  0.998 3.392 0.663     0.970 0.970 0.970
dissubCML                 1.229  1.000 1.229 0.033     0.946 0.946 0.946
age20-40                  1.042  1.001 1.041 0.822     0.993 0.993 0.994
age>40                    0.815  0.996 0.818 1.463     1.021 1.021 1.020
drmatchGender mismatch    0.777  1.000 0.777 1.307     0.987 0.987 0.987
tcdTCD                    1.089  1.003 1.086 0.107     1.036 1.035 1.036
dissubALL                 1.002  1.000 1.002 0.991     1.000 1.001 1.000
dissubCML                 0.469  1.000 0.469 1.183     0.981 0.981 0.981
age20-40                  0.956  1.000 0.956 1.179     0.988 0.988 0.989
age>40                    0.997  1.000 0.997 1.042     0.998 0.998 0.998
drmatchGender mismatch    1.041  1.000 1.041 0.966     0.997 0.997 0.997
tcdTCD                    1.158  1.000 1.158 0.547     1.041 1.041 1.041
dissubALL                 1.034  1.001 1.033 0.958     1.005 1.004 1.005
dissubCML                 0.965  1.000 0.965 1.216     0.991 0.991 0.991
age20-40                  0.933  1.001 0.932 1.032     0.996 0.995 0.996
age>40                    1.005  1.001 1.004 0.944     1.003 1.003 1.003
drmatchGender mismatch    1.052  1.000 1.052 0.868     1.009 1.009 1.009
tcdTCD                    0.974  1.000 0.974 1.089     0.995 0.995 0.995
```

As before we can model this in one go by stacking the dataset using `stack.Lexis`. Note that it is impossible to try and use `splitLexis` on a stacked dataset because it is no longer a `Lexis` object. Hence stacking must be done *after* splitting data.

```
> bmtxs <- stack( bmtx )
> str( bmts )
```

```
Classes 'stacked.Lexis' and 'data.frame':       5577 obs. of  18 variables:
 $ tft     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ tfPR    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ lex.dur : num  0.063 0.0958 0.0712 0.0602 0.0794 ...
 $ lex.Cst : Factor w/ 4 levels "Tx","PR","RD",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Xst : Factor w/ 4 levels "Tx","PR","RD",..: 2 2 2 2 2 2 2 2 1 3 ...
 $ lex.Tr  : Factor w/ 3 levels "Tx->PR","Tx->RD",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Fail: logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
 $ lex.id  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ prtime  : num  23 35 26 22 29 38 30 35 NA NA ...
 $ prstat  : int  1 1 1 1 1 1 1 1 0 0 ...
 $ rfstime : num  744 360 135 995 422 ...
 $ rfsstat : int  0 1 1 0 1 1 0 0 0 1 ...
 $ dissub  : Factor w/ 3 levels "AML","ALL","CML": 3 3 3 1 1 2 3 2 1 3 ...
 $ age     : Factor w/ 3 levels "<=20","20-40",..: 3 3 3 2 2 3 2 1 2 3 ...
 $ drmatch : Factor w/ 2 levels "No gender mismatch",..: 2 1 1 1 1 1 2 1 1 1 ...
 $ tcd     : Factor w/ 2 levels "No TCD","TCD": 1 1 1 1 1 1 1 1 1 1 ...
 $ inc.mort: Factor w/ 2 levels "Tx->PR","Tx->RD & PR->RD(PR)": 1 1 1 1 1 1 1 1 1 1 ...
```

We can now redo the analysis with in one step. Moreover we can now get a global test for the proportionality of the two mortality rates as a simple likelihood-ratio test. Note that in order to test for proportionality of rates between the two mortality rates, we must have an interaction between the timescale (`ns(tft,...)` and a factor where the two mortalities are grouped together. But we must still have a main effect of `lex.Tr`, the type of transition:

```
> p1 <- glm( as.numeric(lex.Fail) ~
+              lex.Tr +
+              lex.Tr:( ns( tft, knots=c(seq(0.2,2,0.2),3:6), Bo=c(0,7) ) +
+                       dissub + age + drmatch + tcd ),
+              family=poisson,
+              data=bmtxs )
```

To fit a model with proportional mortality rates (from `Tx` and from `PR`) we must group the two transitions into one level, but in order to avaoid a model that assumes *identical* mortalities, we still include the three level factor `lex.Tr` as a main effect:

```
> bmtxs$inc.mort <- Relevel( bmtxs$lex.Tr, list(1,2:3), coll=" & " )
> p2 <- glm( as.numeric(lex.Fail) ~
+              lex.Tr +
+              inc.mort:ns( tft, knots=c(seq(0.2,2,0.2),3:6), Bo=c(0,7) ) +
+              lex.Tr:( dissub + age + drmatch + tcd ),
+              family=poisson,
+              data=bmtxs )
> round( ci.lin( p2, subset=sb, E=T ), 3 )[,5:7]

                                      exp(Est.)  2.5% 97.5%
lex.TrTx->PR:dissubALL                    0.987 0.847 1.150
lex.TrTx->RD:dissubALL                    1.297 0.995 1.691
lex.TrPR->RD(PR):dissubALL                1.142 0.855 1.526
lex.TrTx->PR:dissubCML                    0.785 0.687 0.897
lex.TrTx->RD:dissubCML                    1.021 0.826 1.263
lex.TrPR->RD(PR):dissubCML                1.295 1.030 1.628
lex.TrTx->PR:age20-40                     0.854 0.731 0.997
lex.TrTx->RD:age20-40                     1.297 0.965 1.744
lex.TrPR->RD(PR):age20-40                 1.059 0.785 1.430
```

```
lex.TrTx->PR:age>40                          0.896 0.756 1.062
lex.TrTx->RD:age>40                          1.695 1.243 2.309
lex.TrPR->RD(PR):age>40                      1.741 1.273 2.382
lex.TrTx->PR:drmatchGender mismatch          1.061 0.931 1.209
lex.TrTx->RD:drmatchGender mismatch          0.932 0.751 1.157
lex.TrPR->RD(PR):drmatchGender mismatch      1.174 0.938 1.469
lex.TrTx->PR:tcdTCD                          1.483 1.267 1.735
lex.TrTx->RD:tcdTCD                          1.332 0.993 1.786
lex.TrPR->RD(PR):tcdTCD                      1.243 0.971 1.592
```

```
> anova( p1, p2, test="Chisq" )

Analysis of Deviance Table

Model 1: as.numeric(lex.Fail) ~ lex.Tr + lex.Tr:(ns(tft, knots = c(seq(0.2,
    2, 0.2), 3:6), Bo = c(0, 7)) + dissub + age + drmatch + tcd)
Model 2: as.numeric(lex.Fail) ~ lex.Tr + inc.mort:ns(tft, knots = c(seq(0.2,
    2, 0.2), 3:6), Bo = c(0, 7)) + lex.Tr:(dissub + age + drmatch +
    tcd)
  Resid. Df Resid. Dev    Df Deviance P(>|Chi|)
1     55266     9125.4
2     55281     9164.1   -15    -38.7  0.000703
```

The test is highly significant, but it would be relevant to see what the actual differences in shape between the two mortality rates were.

Hence we extract the effect of `tft` from the model `p1` for the transitions `Tx`→`RD` and `PR`→`RD`.

To this end we construct a contrast matrix to multiply with the relevant spline parameters, by setting up the splines for a predefined set of times. Since we are mainly interested in the relative behaviour of the rates, we compute the RR relative to the mortality 1 year after entry:

```
> p.pt <- seq(0,8,,200)
> CM <- ns(              p.pt  , knots=c(seq(0.2,2,0.2),3:6), Bo=c(0,7) )
> C1 <- ns( rep(1,length(p.pt)), knots=c(seq(0.2,2,0.2),3:6), Bo=c(0,7) )
```

This matrix can now be applied to the estimates for the splines for the two transitions. The relevant parameters are selected using the `subset` argument to `ci.lin`. The `Exp=TRUE` gives the exponential of the computed parameter functions with c.i. i columns 5 to 7 of the result:

```
> rr.Tx <- ci.lin( p1, subset="Tx->RD:ns", ctr.mat=CM-C1, Exp=TRUE )[,5:7]
> rr.PR <- ci.lin( p1, subset="PR->RD\\(PR\\):ns", ctr.mat=CM-C1, Exp=TRUE )[,5:7]
```

Note the use of "\\" in the `subset` argument — it is because "(" and ")" have special meanings in regular expressions, so the characters need to be escaped in order to work properly. However, we would like the see the estimated rate-ratio between the two transitions as well. This is done by extracting the two sets of parameters and them multiplying them by a contrast matrix which is made by joining the contrast matrix by the negative of the same:

```
> RR <- ci.lin( p1, subset=c("Tx->RD:ns","PR->RD\\(PR\\):ns"),
+                   ctr.mat=cbind(CM,-CM), Exp=TRUE )[,5:7]
```

```
> matplot( p.pt, cbind( rr.Tx, rr.PR ),
+          log="y", ylim=c(0.1,10)/2,
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("red","blue"),each=3) )
> points( 1, 1, pch=16 )
> matlines( p.pt, RR, type="l", lty=1, lwd=c(3,1,1), col="black" )
```

Clearly the underlying rates are over-modeled (although not to the extent as in the Cox-model), but also it is pretty clear that despite the overmodeling there is a tendency that the mortality rates for those in platelet recovery (`PR`, blue line) do not decrease so rapidly the first two years. Apparently the entire difference takes place during the first year.

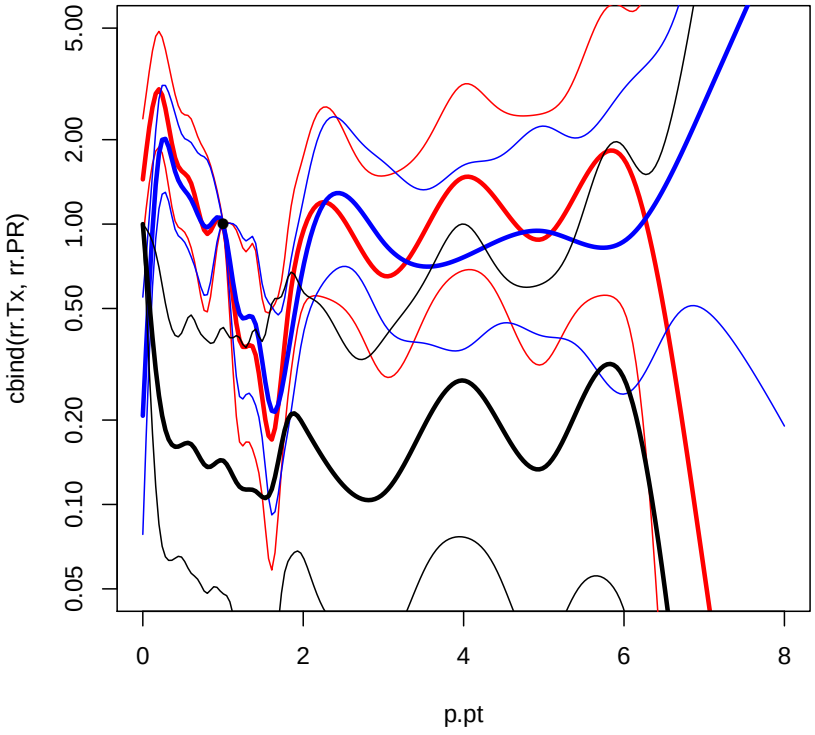

Figure 6: *RR relative to year 1 after transplant for persons in states* Tx *(red) and* PR *(blue), as well as the RR between the mortality rates in* Tx *and* PR *(black). Thin lines are estimated 95% confidence intervals.*

8 Competing risks

The special case of a multistate model with only one initial state ("alive") and a number of absorbing states ("causes of death"), we can get away with setting up a `Lexis` object particularly simple. There are two defaults that are used in the setup of the object for the `aidssi` data:

1. If only the `entry` argument is omitted, all entries are assumed to be at 0 at the only timescale — it is non-sensical to have more timescales in this instance.

2. If no `entry.state` is specified is by default assumed that all entires are in the state which is the *first* level of the `exit.state`.

Thus we must make sure that the "event-free" is the first level of the status factor:

```
> data(aidssi)
> LA <- Lexis( exit = list(zeit=time),
+              exit.status = factor(cause,levels=c("event-free","AIDS","SI")),
+              data = aidssi )

NOTE: entry.status has been set to "event-free" for all.
NOTE: entry is assumed to be 0 on the zeit timescale.

> summary( LA )

Transitions:
     To
From         event-free AIDS  SI  Records:  Events: Risk time:  Persons:
  event-free        107  114 108       329      222    2274.55       329

Rates:
     To
From         event-free AIDS   SI Total
  event-free          0 0.05 0.05   0.1
```

Figure 7 is the result of using the default layout of `boxes.Lexis`:

```
> boxes( LA, boxpos=TRUE )
```

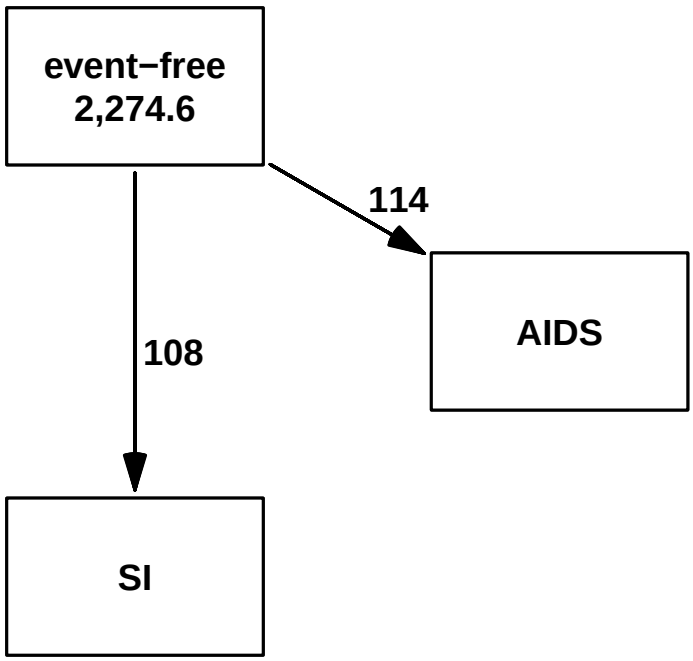

Figure 7: *Illustration of the transitions in the competing risks model for the* `aidssi` *data.*