Journal of Statistical Software
January 2011, Volume 38, Issue 6. http://www.jstatsoft.org/

Using `Lexis` Objects for Multi-State Models in R

Bendix Carstensen
Steno Diabetes Center

Martyn Plummer
International Agency for Research on Cancer

Abstract

The `Lexis` class in the R package **Epi** provides tools for creation, manipulation and display of data from multi-state models. Transitions between states are described by rates (intensities); `Lexis` objects represent this kind of data and provide tools to show states and transitions annotated by relevant summary numbers. Data can be transformed to a form that allows modelling of several transition rates with common parameters.

Keywords: epidemiology, survival analysis, multi-state models, R.

1. Introduction

Multi-state data arise from epidemiological follow-up studies in which a participant may go through different disease states during follow-up. The most common examples are the *illness-death* model and the *competing risks* model. Illness-death models are used in studies of mortality when an intermediate state ("ill") may occur between the entry state ("healthy") and the mortality endpoint ("dead"). The competing risk model is a model with one "healthy" state and a number of mutually exclusive absorbing disease endpoints (causes of death, for example).

The `Lexis` class in the **Epi** package (Carstensen, Plummer, Laara, and Hills 2010), for R (R Development Core Team 2010), which is introduced in a companion paper (Plummer and Carstensen 2011), is designed to facilitate long-term follow-up studies. It includes the following extensions to handle multi-state data:

- Allowing the use of factors for states, as well as numeric codes. This allows the states to have descriptive labels.
- Taking account of the ordering of disease states when splitting follow-up time.
- Creating new time scale variables to describe the time spent in a state.

These capabilities are described in the current paper. In this paper we are only looking at multi-state data for which the transition between states is known exactly. Panel studies, for which the disease state is only known at certain dates and transition times are not known exactly, are *not* covered in this paper.

2. The example data

The dataset we shall use for illustration is the `ebmt3` dataset from the R package **mstate** (de Wreede, Fiocco, and Putter 2010, 2011). It shows the clinical course of bone marrow transplant for 2204 patients. A more detailed introduction to the dataset is given in a tutorial paper by Putter, Fiocco, and Geskus (2007).

```
  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 major event of interest is "relapse or death", and an intermediate event is "platelet recovery". The time variables in the `embt3` dataset are given in days since transplant, they are time to relapse or death in the variable `rfstime`, and time to platelet recovery in the variable `prtime`. It is of interest to compare the rates of relapse or death between the states "transplant" and "platelet recovery", but also to model the transition rate from "transplant" to "platelet recovery". The structure of the multi-state model is illustrated in Figure 1.

3. Statistical model

For each transition between two states (an "event"), we can describe the process by the transition *rate*:

$$\lambda(t) = \lim_{h \to 0} \mathrm{P}\{\text{event in } (t, t+h) | \text{at risk at } t\}/h$$

If transition rates are known for all possible transitions in a multi-state set-up, we have a complete probability model, that is a model that would allow simulation of data of the same structure as the observed.

If rates are assumed constant, the likelihood for the model will be proportional to a Poisson likelihood using the number of transitions as Poisson variate and the log person-years as offset in a log-link.

If we assume that rates are merely constant in smaller time-intervals, the likelihood will still be proportional to a Poisson likelihood, but now with one 0/1 contribution from each person and interval as response, and the log of the interval length as offset (Carstensen 2006).

If we make no assumptions at all for the transition rates we can fit a model for the cumulative rates:

$$\Lambda(t) = \int_0^t \lambda(s)\,\mathrm{d}s$$

using e.g. the Nelson-Aalen estimator or using a Cox model to incorporate covariate effects. An extensive overview of these issues is given in Andersen and Keiding (2002).

The purpose of setting up a `Lexis` object is to facilitate required manipulations of data for these types of models.

4. A `Lexis` object

The required data for this type of analysis is for each person the starting time and state and the transitions and -times and the state and time of end of observation. Depending on availability of covariates, further timescales can be defined (current age, current disease duration, etc.). Essentially what is needed is the allocation of follow-up time to states, and timescales.

`ebmt3` is merely a data frame with two time-variables in it, one for each of the events "platelet recovery" and "relapse or death". So if we want to put the follow-up recorded in the `embt3` data frame into a `Lexis` object, we must decide on the timescale(s) we want to use (in this case only one). The first thing we do is to only consider the transition from "transplant" to "relapse or death", and then later incorporate the "platelet recovery".

The data frame `ebmt3` is converted to a `Lexis` object (which is also a dataframe) as follows:

```
R> 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.
```

This example illustrates the use of a factor to represent the states in the model. The numeric codes 0,1 of the `rfsstat` variable are given informative labels "Tx" and "RD". If no entry state is given, then the `Lexis` function assumes that everyone starts in the first factor level, which is `Tx` in this case. This is noted by the `Lexis` function.

The `Lexis` object has a single time scale — `tft` — measuring **t**ime **f**rom **t**ransplant in years. If no entry time is given by the argument `entry`, the `Lexis` function assumes a default entry time of 0. Since follow-up time begins on the day of transplant, this is the correct entry time on the `tft` scale. This is also noted by the `Lexis` function.

So far we have only defined a simple two-state model with transition from `Tx` to `RD`:

```
R> summary(bmt)

Transitions:
     To
From   Tx  RD  Records:  Events: Risk time:  Persons:
  Tx 1363 841      2204      841       5612      2204

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

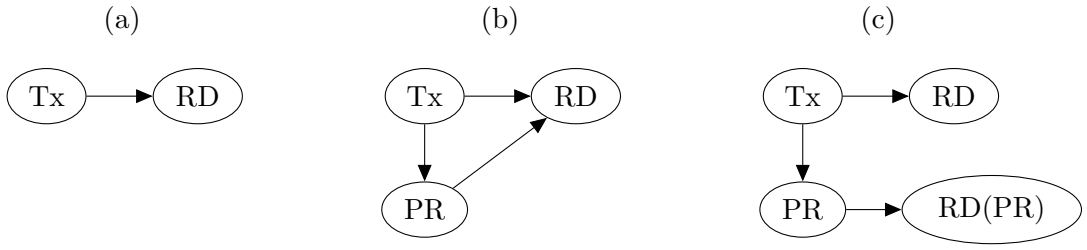

Figure 1: Three different representations of the ebmt3 data: (a) the simple survival type model ignoring the platelet recovery state, (b) the illness-death-type model where platelet recovery is an intermediate state, (c) the extension of the illness-death model where the absorbing state is subdivided by previous occurrence of platelet recovery.

Thus, among the 2204 persons we have 841 events during 5612 person-years, corresponding to a rate of 0.15 events per year. Now we introduce the state state "platelet recovery" (PR) by dividing the follow-up time at the time of platelet recovery using the cutLexis function. This transforms the model from panel (a) to panel (b) of Figure 1.

```
R> bmtr <- cutLexis(bmt, cut = bmt$prtime/365.25, precursor.states = "Tx",
+    new.state = "PR")
R> summary(bmtr)

Transitions:
     To
From   Tx   PR  RD  Records:  Events: Risk time:  Persons:
  Tx  577 1169 458      2204     1627       2439      2204
  PR    0  786 383      1169      383       3173      1169
  Sum 577 1955 841      3373     2010       5612      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 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, but now subdivided by state (Tx and PR).

When creating a new state with the cutLexis function, it is necessary to specify which of the existing states in the model are *precursor* states. Precursor states will be overridden by the new state in the newly created Lexis object. In this example, if a person originally exited the study in state Tx (a precursor state), then an entry to the new state (PR) will mean that the person exits in the new state, whereas if the person exited the study in state RD (a non-precursor state) the exit from the study will still be to this (original) state.

In order to understand which states are precursor states, it is helpful to draw a graph of the multi-state model, such as the examples in Figure 1. Most multi-state disease models are

progressive models, in which the disease can only increase in severity and never regresses. Such models are represented by directed acyclic graphs, which determine a partial ordering among the states. A state A is a precursor of state B if there is a path from A to B in the graph. Absorbing states, such as `RD` in Figure 1(b) are never precursor states. By default the newly created state is placed after the precursor states nominated, and before all non-precursor states in the ordering of the levels of the factor describing the states.

4.1. Duration as a timescale

So far we have only used one common timescale for follow-up, but often we are interested in the effect of the *duration* of platelet recovery as timescale. This timescale can be generated using the `new.scale` argument. If set to `TRUE` a new timescale will be generated, named `PR.dur`; alternatively we can set the argument to the name of the new timescale:

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

[1] "tft"  "tfPR"
```

This representation retains the total amount of risk time in the study; it merely adds an extra timescale to the follow-up time after transition to the `PR` state. Since this timescale is 0 at the entry of the new state this is sometimes referred to as the "clock back" approach.

Since we now have two timescales we can make a two-dimensional Lexis diagram, as shown in Figure 2. 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.

```
R> plot(bmtr, xlab = "Time since transplant",
+    ylab = "Time since platelet recovery")
```

From the plot we see that most platelet recoveries occur within the first year after transplant.

4.2. Subdividing states

We may also be interested in subdividing the *states* subsequent to `PR` according to whether this event has occurred or not; this is done with the argument `split.states`:

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

Transitions:
     To
From   Tx   PR  RD RD(PR)  Records:  Events: Risk time:  Persons:
  Tx  577 1169 458      0      2204     1627       2439      2204
  PR    0  786   0    383      1169      383       3173      1169
  Sum 577 1955 458    383      3373     2010       5612      2204
```

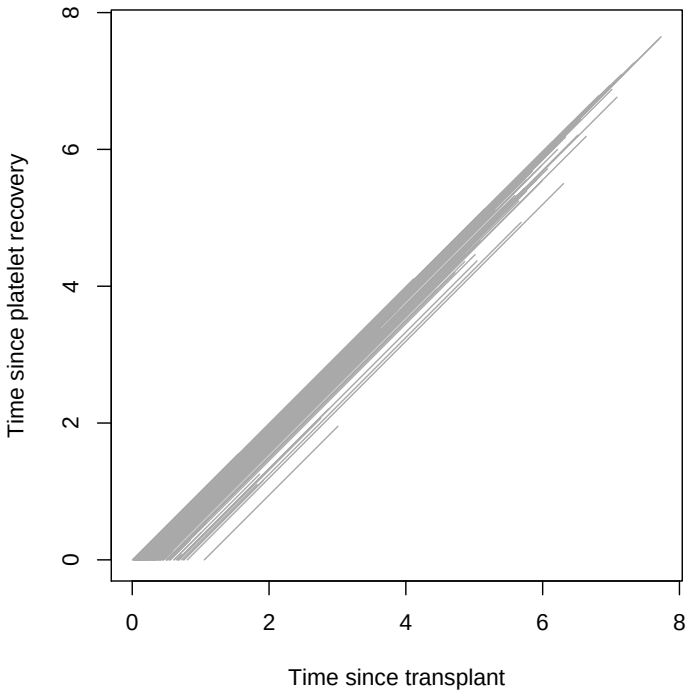

Figure 2: Two-dimensional Lexis diagram of the bone-marrow transplant data from `ebmt3`. Note that this diagram only shows follow-up *after* `PR`, as the timescale `tfPR` is undefined (=missing) for follow-up prior to `PR`.

```
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
```

This creates a `Lexis` object for the model represented in Figure 1(c).

Subdividing states is useful when drawing multi-state 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.

5. Illustrating the model structure

The model structure can be illustrated in a box diagram as shown in Figure 1 using the interactive function `boxes.Lexis`, which will prompt you for a click for the position of each of the states, and subsequently draws 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:

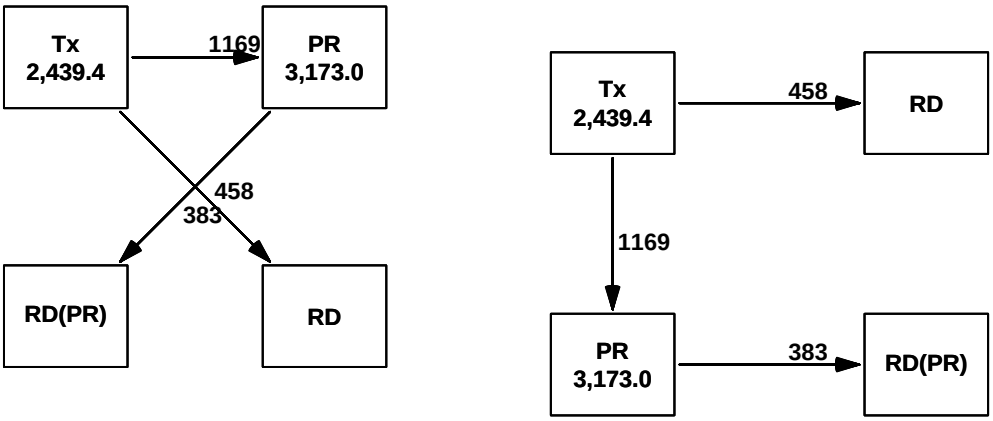

Figure 3: Illustration of the transitions between defined states for the `ebmt3` data, using the default layout (left) and a hand-tailored using point-and-click. The numbers on the arrows are the number of transitions, and the numbers in the boxes are person-years.

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

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 an output file that will contain code generating the display. The code in this file can then be modified.

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

This is the resulting code, where we modified the coordinates of the boxes to make them nicely aligned.

```
R> boxes.Lexis(bmtr, boxpos = list(x = c(20, 20, 80, 80),
+     y = c(80, 20, 80, 20)), cex = 1.5, wmult = 1.5, hmult = 2.25,
+     eq.wd = TRUE, eq.ht = TRUE, show.Y = TRUE, scale.Y = 1, digits.Y = 1,
+     show.D = TRUE, scale.D = FALSE, digits.D = 0)
```

The first argument to the `boxes` function is a `Lexis` object, and so the default 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 function has arguments for scaling the person-years and putting rates instead of number of transitions on the arrows.

The first argument to `boxes` may also be a transition matrix, in which case only the boxes and arrows are drawn, and requires an explicit call of the function `boxes.Lexis`.

6. Analysis of transition rates

Traditionally a Cox model is used for the analysis of rates. We can analyse transitions between boxes separately by selecting the appropriate rows and choosing the right event-type.

There is minimal difference in the code for the models for the different transitions; it is only the failure type (using `lex.Xst`) and the subset (using `lex.Cst`) that is different:

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

```
                          coef exp(coef) se(coef)      z Pr(>|z|)
dissubALL              -0.0436     0.957   0.0779 -0.560 5.76e-01
dissubCML              -0.2972     0.743   0.0680 -4.371 1.23e-05
age20-40               -0.1646     0.848   0.0791 -2.082 3.73e-02
age>40                 -0.0898     0.914   0.0865 -1.038 2.99e-01
drmatchGender mismatch  0.0458     1.047   0.0666  0.687 4.92e-01
tcdTCD                  0.4291     1.536   0.0804  5.335 9.57e-08
dissubALL               0.2559     1.292   0.1352  1.893 5.84e-02
dissubCML               0.0167     1.017   0.1084  0.155 8.77e-01
age20-40                0.2552     1.291   0.1510  1.689 9.11e-02
age>40                  0.5265     1.693   0.1579  3.334 8.55e-04
drmatchGender mismatch -0.0753     0.928   0.1103 -0.682 4.95e-01
tcdTCD                  0.2967     1.345   0.1501  1.977 4.80e-02
dissubALL               0.1366     1.146   0.1480  0.923 3.56e-01
dissubCML               0.2471     1.280   0.1169  2.115 3.44e-02
age20-40                0.0614     1.063   0.1534  0.400 6.89e-01
age>40                  0.5809     1.788   0.1601  3.627 2.86e-04
drmatchGender mismatch  0.1729     1.189   0.1145  1.510 1.31e-01
tcdTCD                  0.2007     1.222   0.1264  1.589 1.12e-01
```

These are exactly the same estimates as those obtained in Table III of Putter *et al.* (2007).

If some parameters are assumed to be common between (some of the) transitions, the analysis must be based on a dataset that is made by stacking the subsets of the data frame we used in the three separate analyses above. The stacked dataset must have a common failure indicator and a factor classifying the transitions. This dataset can be constructed by using the `stack.Lexis` function. The effect of stacking the dataset is illustrated by printing the records for person 2 from the `Lexis` object and the stacked object:

```
R> bmts <- stack(bmtr)
R> bmtr[bmtr$lex.id == 2, 1:6]
```

```
        tft tfPR lex.dur lex.Cst lex.Xst lex.id
2    0.0000   NA  0.0958      Tx      PR      2
2206 0.0958    0  0.8898      PR  RD(PR)      2
```

```
R> bmts[bmts$lex.id == 2, 1:8]
```

```
         tft tfPR lex.dur lex.Cst lex.Xst     lex.Tr lex.Fail lex.id
2     0.0000   NA  0.0958      Tx      PR     Tx->PR     TRUE      2
2206  0.0000   NA  0.0958      Tx      PR     Tx->RD    FALSE      2
22061 0.0958    0  0.8898      PR  RD(PR) PR->RD(PR)     TRUE      2
```

Two extra variables `lex.Tr` and `lex.Fail` have been added; `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 original meaning, the resulting object is not a `Lexis` object.

We will not be interested in modelling the `Tx->PR` transition jointly with the two transitions into the `RD` state. The main interest is a joint model for the two transitions to `RD`. We can reproduce the results from above for these two transitions 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:

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

The default model matrix set-up for interactions generates interactions assuming an intercept in the model. But there is none in a Cox model (it is relegated to the baseline hazard), so we get aliased parameters when all three levels of `dissub` have interaction columns with `lex.Tr` generated. It gives the correct model fit, 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:

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

6.1. Joint parameters between transitions

Of course it is not of much interest to make a joint model which gives the same results as the models fitted separately to each transition. 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 mortality rates, Tx->RD and PR->RD are proportional — or, in plain words, covariate effects are the same for both sets of mortality rates. This means that the Cox model should be changed so that the baseline hazard is the same for the two transitions, which just means that no strata argument is needed. However to avoid a model where we assume that the rates are *identical* (which would be nonsensical), we must introduce an indicator of being in state PR, but that is just the variable lex.Cst:

```
R> c2 <- coxph(Surv(tft, tft + lex.dur, lex.Fail) ~ dissub + age +
+    drmatch + tcd + I(lex.Cst == "PR"), data = bmts, 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 the model formula will be interpreted as (tcd + lex.Cst)=="PR", which will cause a crash.

We now get one set of rate ratio parameters as before, plus a rate-ratio between the two transitions to death, replacing the two separate baseline hazards with a pair of proportional ones with proportionality factor:

```
R> ci.lin(c2, subset = "PR", Exp = TRUE)

                         Estimate StdErr     z        P exp(Est.)  2.5% 97.5%
I(lex.Cst == "PR")TRUE   -0.314 0.0725 -4.33 1.51e-05     0.731 0.634 0.842
```

We cannot use the usual anova machinery to make a test of proportionality by comparing the two Cox models, because the likelihoods from the stacked datasets only are concerned with the regression parameters and the baseline is profiled out. So for this purpose we must resort to fully parametric models to make a formal likelihood ratio test for this.

7. Analyzing proportionality of rates by Poisson modeling

It is of course of interest to test whether the two transition rates actually are proportional or not, *i.e.* whether the dependency on time (since transplant) is the same pre and post platelet recovery. This is an interaction, and can be formulated as such in a Poisson model where we assume that rates are constant in (small) intervals, and then invoke a model for the rates in these intervals that assumes the rates are constant in each interval, but that the magnitude of the rates follow some smooth function. This is a slightly different approach from the one taken in the companion paper (Plummer and Carstensen 2011), where the functional form was assumed piecewise constant in larger intervals.

The **Epi** package provides the time-splitting function for Lexis objects that facilitates this operation:

```
R> bmtx <- splitLexis(bmtr, time.scale = "tft",
+    breaks = c(0:19/10, seq(2, 10, 1)))
R> summary(bmtx)

Transitions:
     To
```

```
From     Tx    PR  RD RD(PR)  Records:  Events: Risk time:  Persons:
  Tx  14628  1169 458     0     16255     1627       2439      2204
  PR      0 18873   0   383     19256      383       3173      1169
  Sum 14628 20042 458   383     35511     2010       5612      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
```

Note that the intervals that we use are not equally long. 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`.

As before we can stack the dataset using `stack.Lexis`. Note that we cannot use `splitLexis` on a stacked dataset because it is no longer a `Lexis` object. Hence stacking must be done *after* splitting data. Finally we are only interested in the two transitions to the relapse/dead (`RD`) state, and hence use `factor` to produce a factor with only those levels that actually do occur.

```
R> bmtxs <- stack(bmtx)
R> bmtxs <- bmtxs[grep("->RD", bmtxs$lex.Tr), ]
R> bmtxs$lex.Tr <- factor(bmtxs$lex.Tr)
```

We can now repeat the analysis of the two rates and also get a formal test for the proportionality of the two mortality rates as a simple likelihood-ratio test by fitting the model with and without a time×state interaction. There is also an intermediate model that in the Cox model literature is known as the "stratified model" where there is an interaction with time but not with other covariates.

We fit these three models and compare them by the usual asymptotic likelihood ratio test:

```
R> i.kn <- c(0.2, 0.5, 1, 1.5, 2, 4, 6)
R> b.kn <- c(0, 7)
R> mi <- glm(as.numeric(lex.Fail) ~ lex.Tr +
+    lex.Tr:ns(tft, knots = i.kn, Bo = b.kn) +
+    lex.Tr:(dissub + age + drmatch + tcd),
+    family = poisson, offset = log(lex.dur), data = bmtxs)
R> ms <- glm(as.numeric(lex.Fail) ~ lex.Tr +
+    lex.Tr:ns(tft, knots=i.kn, Bo=b.kn) + dissub + age + drmatch + tcd,
+    family = poisson, offset = log(lex.dur), data = bmtxs)
R> mp <- glm(as.numeric(lex.Fail) ~ lex.Tr +
+    ns(tft, knots = i.kn, Bo = b.kn) + dissub + age + drmatch + tcd,
+    family = poisson, offset = log(lex.dur), data = bmtxs)
R> anova(mi, ms, mp, mi, test = "Chisq")

Analysis of Deviance Table
```

```
Model 1: as.numeric(lex.Fail) ~ lex.Tr + lex.Tr:ns(tft, knots = i.kn,
    Bo = b.kn) + lex.Tr:(dissub + age + drmatch + tcd)
Model 2: as.numeric(lex.Fail) ~ lex.Tr + lex.Tr:ns(tft, knots = i.kn,
    Bo = b.kn) + dissub + age + drmatch + tcd
Model 3: as.numeric(lex.Fail) ~ lex.Tr + ns(tft, knots = i.kn, Bo = b.kn) +
    dissub + age + drmatch + tcd
Model 4: as.numeric(lex.Fail) ~ lex.Tr + lex.Tr:ns(tft, knots = i.kn,
    Bo = b.kn) + lex.Tr:(dissub + age + drmatch + tcd)
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1     35481       7439
2     35487       7450 -6    -10.4     0.109
3     35495       7468 -8    -18.4     0.018
4     35481       7439 14     28.8     0.011
```

There is no significant difference between the two first models, so we may assume that covariate effects are the same for both transitions, but the underlying hazards seem to have a different shape.

If we for a moment forget the interaction we can compare the estimated rate-ratio associated with platelet recovery as estimated from the Poisson model and the Cox model:

```
R> rbind(ci.lin(mp, subset = "lex.Tr", Exp = TRUE),
+    ci.lin(c2, subset = "lex.Cst", Exp = TRUE))
```

```
                       Estimate StdErr     z        P exp(Est.)  2.5% 97.5%
lex.TrPR->RD(PR)         -0.296 0.0727 -4.07 4.74e-05     0.744 0.645 0.858
I(lex.Cst == "PR")TRUE   -0.314 0.0725 -4.33 1.51e-05     0.731 0.634 0.842
```

We see that the difference in estimates is negligible (1/4 of the estimated s.e.). The advantage of the Poisson-modeling over the Cox model is that we get direct estimates of the baseline rates, and that visual comparison of them is quite straightforward, so that we can inspect the clinical relevance and not only the statistical significance of the interaction ("non-proportional rates").

In order to fish out the estimated log-rates from the model we must multiply the parameters from the `ns`-term in the model with a matrix so that the estimated log-rates at some pre-specified times are recovered.

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. We compare to the rates in the `Tx` state at 1 year after platelet recovery:

```
R> p.pt <- seq(0, 5, 0.02)
R> CM <- ns(p.pt, knots = i.kn, Bo = b.kn)
R> C1 <- ns(rep(1, length(p.pt)), knots = i.kn, Bo = b.kn)
```

Note that the matrix `C1` has all rows identical to the row in `CM` which corresponds to `p.pt == 1`:

```
R> round(cbind(p.pt, CM)[48:52,], 3)
```

```
     p.pt     1     2     3 4 5 6 7 8
[1,] 0.94 0.269 0.617 0.114 0 0 0 0 0
[2,] 0.96 0.242 0.628 0.130 0 0 0 0 0
[3,] 0.98 0.216 0.636 0.147 0 0 0 0 0
[4,] 1.00 0.192 0.641 0.167 0 0 0 0 0
[5,] 1.02 0.170 0.642 0.187 0 0 0 0 0
```

```
R> head(C1)
```

```
         1     2     3 4 5 6 7 8
[1,] 0.192 0.641 0.167 0 0 0 0 0
[2,] 0.192 0.641 0.167 0 0 0 0 0
[3,] 0.192 0.641 0.167 0 0 0 0 0
[4,] 0.192 0.641 0.167 0 0 0 0 0
[5,] 0.192 0.641 0.167 0 0 0 0 0
[6,] 0.192 0.641 0.167 0 0 0 0 0
```

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`, which both selects and sorts the parameters of interest. First we use `ci.lin` to show the total set of parameters:

```
R> ci.lin(ms)[, 1:2]
```

```
                                                    Estimate StdErr
(Intercept)                                          -1.3768 0.1486
lex.TrPR->RD(PR)                                     -1.2373 0.3848
dissubALL                                             0.2047 0.0997
dissubCML                                             0.1279 0.0797
age20-40                                              0.1589 0.1073
age>40                                                0.5419 0.1123
drmatchGender mismatch                                0.0359 0.0792
tcdTCD                                                0.2403 0.0967
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)1       -0.7007 0.2593
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)1    0.5402 0.3577
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)2       -0.3748 0.3551
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)2    0.8434 0.5199
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)3       -2.1909 0.3908
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)3   -0.8464 0.4617
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)4       -1.7264 0.6125
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)4   -0.8431 0.6550
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)5       -2.5504 0.9128
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)5   -1.3821 0.8260
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)6       -1.4513 1.1050
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)6   -1.1546 0.9026
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)7       -0.6390 1.4207
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)7    1.7567 1.1066
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)8       -1.9564 2.0909
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)8   -0.3233 1.1258
```

Then we can use it to show the subset and the ordering of this we get when using the `subset` argument:

```
R> ci.lin(ms, subset = c("PR", "Tx"))[, 1:2]
```

```
                                                 Estimate StdErr
lex.TrPR->RD(PR)                                   -1.237  0.385
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)1  0.540  0.358
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)2  0.843  0.520
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)3 -0.846  0.462
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)4 -0.843  0.655
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)5 -1.382  0.826
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)6 -1.155  0.903
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)7  1.757  1.107
lex.TrPR->RD(PR):ns(tft, knots = i.kn, Bo = b.kn)8 -0.323  1.126
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)1     -0.701  0.259
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)2     -0.375  0.355
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)3     -2.191  0.391
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)4     -1.726  0.612
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)5     -2.550  0.913
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)6     -1.451  1.105
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)7     -0.639  1.421
lex.TrTx->RD:ns(tft, knots = i.kn, Bo = b.kn)8     -1.956  2.091
```

For the state (`Tx`) we just use the spline parameters and subtract the value at the reference point, thus giving the rate-ratio (RR) relative to time `tft==1`:

```
R> rr.Tx <- ci.lin(ms, subset = "Tx", ctr.mat = CM - C1, Exp = TRUE)[, 5:7]
```

Note that we should formally also have included the intercept among the parameters, but in this case it would just cancel out.

The argument `Exp=TRUE` to `ci.lin` computes the exponential of the resulting parameter functions with 95% confidence intervals and puts them in columns 5 to 7 of the result:

```
R> head(rr.Tx)
```

```
     exp(Est.) 2.5% 97.5%
[1,]      2.10 1.43  3.08
[2,]      2.26 1.57  3.26
[3,]      2.44 1.72  3.46
[4,]      2.62 1.88  3.67
[5,]      2.81 2.03  3.90
[6,]      2.99 2.17  4.13
```

For the state `PR` level we want the RR relative to time `tft==1` for state `Tx`. Thus we want both the parameters for each of the transitions, but also the parameter giving the extra contribution for the `Pr->RD` transition relative to the `Tx->RD` transition: These must be multiplied by columns that give the RR relative to the rates from `Tx` at time 1:

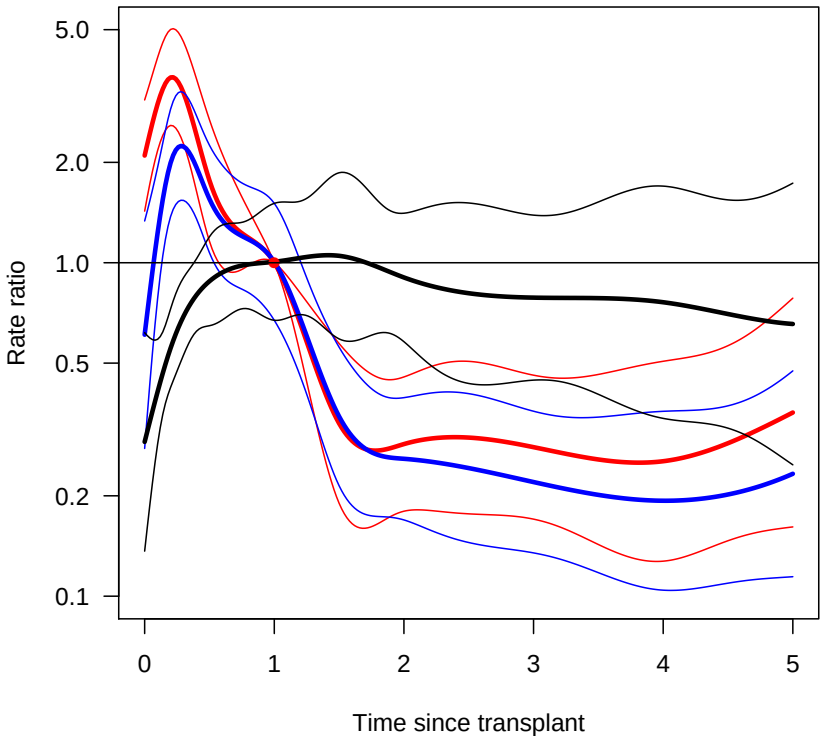

Figure 4: RR relative to year 1 after transplant for persons in states Tx (red) and PR (blue), as well as the RR between the mortality rates in PR and Tx (gray). Thin lines are estimated 95% confidence intervals.

```
R> rr.PR <- ci.lin(ms, subset = c("PR", "Tx"), ctr.mat = cbind(1, CM, -C1),
+    Exp = TRUE)[, 5:7]
```

The estimated rate-ratio between these two RRs is found by subtracting the two sets of parameter functions just derived. This means multiplying the `PR` parameters by `cbind(1, CM)`, and the `Tx` parameters by `-C1 - (CM - C1) = -CM`:

```
R> RR <- ci.lin(ms, subset = c("PR", "Tx"), ctr.mat = cbind(1, CM, -CM),
+    Exp = TRUE)[, 5:7]
```

We can now plot all three sets of curves (estimates with c.i.s):

```
R> matplot(p.pt, cbind(rr.Tx, rr.PR, RR), log = "y", ylim = c(0.1, 5),
+    type = "l", lty = 1, lwd = c(3, 1, 1), las = 1,
+    col = rep(c("red", "blue", "black"), each = 3),
+    xlab = "Time since transplant", ylab = "Rate ratio")
R> points(1, 1, pch = 16, col = "red")
R> abline(h = 1)
```

Clearly, the difference between the rates is confined to the first year, where the RR relative to year 1 in the Tx group is higher than in the PR group. After the first year there is no discernible non-proportionality of rates between the groups.

8. Connecting to the mstate package

The **Epi** package has a function that transforms a Lexis object to a msdata object which can be used in the **mstate** package (de Wreede *et al.* 2010, 2011). Essentially it is a renaming of the variables from a stacked.Lexis object as illustrated here:

```
R> bmtr[bmtr$id==13, 1:6]
```

```
         tft tfPR lex.dur lex.Cst lex.Xst lex.id
13    0.0000   NA  0.0903      Tx      PR     13
13100 0.0903    0  3.8166      PR      PR     13
```

```
R> subset(stack(bmtr), id == 1)
```

```
        tft tfPR lex.dur lex.Cst lex.Xst     lex.Tr lex.Fail lex.id id prtime
1     0.000   NA   0.063      Tx      PR     Tx->PR     TRUE      1  1     23
1893  0.000   NA   0.063      Tx      PR     Tx->RD    FALSE      1  1     23
18931 0.063    0   1.974      PR      PR PR->RD(PR)    FALSE      1  1     23
      prstat rfstime rfsstat dissub age         drmatch    tcd
1          1     744       0    CML >40 Gender mismatch No TCD
1893       1     744       0    CML >40 Gender mismatch No TCD
18931      1     744       0    CML >40 Gender mismatch No TCD
```

```
R> ms <- msdata(bmtr)
R> ms[ms$id == 13, 1:8]
```

```
       id from to trans Tstart  Tstop   time status
13     13    1  2     1 0.0000 0.0903 0.0903      1
13100  13    1  2     2 0.0000 0.0903 0.0903      0
131001 13    2  2     3 0.0903 3.9069 3.8166      0
```

9. Conclusion

The Lexis objects in the **Epi** package come with a range of facilities to make the modelling of multi-state data easier. Lexis objects simplify the initial exploration of the data structure by displaying the states and transitions, and by showing the follow-up time along different timescales. They also allow simplification and structuring of the code for analysis of rates that may be analysed separately or jointly.

References

Andersen PK, Keiding N (2002). "Multi-State Models for Event History Analysis." *Statistical Methods in Medical Research*, **11**, 91–115.

Carstensen B (2006). "Demography and Epidemiology: Practical Use of the Lexis Diagram in the Computer Age, or: Who Needs the Cox Model Anyway?" *Technical Report 06.2*, Department of Biostatistics, University of Copenhagen. URL http://biostat.ku.dk/reports/2006/rr-06-2.pdf.

Carstensen B, Plummer M, Laara E, Hills M (2010). ***Epi**: A Package for Statistical Analysis in Epidemiology*. R package version 1.1.20, URL http://CRAN.R-project.org/package=Epi.

de Wreede LC, Fiocco M, Putter H (2010). "The **mstate** Package for Estimation and Prediction in Non- and Semi-Parametric Multi-State and Competing Risks Models." *Computer Methods and Programs in Biomedicine*, **99**, 261–274.

de Wreede LC, Fiocco M, Putter H (2011). "**mstate**: an R Package for the Analysis of Competing Risks and Multi-State Models." *Journal of Statistical Software*, **38**(7), 1–30. URL http://www.jstatsoft.org/v38/i07/.

Plummer M, Carstensen B (2011). "Lexis: An R Class for Epidemiological Studies with Long-term Follow-up." *Journal of Statistical Software*, **38**(5), 1–12. URL http://www.jstatsoft.org/v38/i05/.

Putter H, Fiocco M, Geskus R (2007). "Tutorial in Biostatistics: Competing Risks and Multi-State Models." *Statistics in Medicine*, **26**, 2389–2430.

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Affiliation:

Bendix Carstensen
Steno Diabetes Center
Niels Steensens Vej 2
2820 Gentofte, Denmark
& Department of Biostatistics, University of Copenhagen
E-mail: bxc@steno.dk
Web-site: www.biostat.ku.dk/~bxc

Martyn Plummer
International Agency for Research on Cancer
150 Cours Albert-Thomas
69572 Lyon Cedex 08, France
E-mail: plummer@iarc.fr

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 38, Issue 6 *Submitted:* 2010-02-09
January 2011 *Accepted:* 2010-09-16