Occurrence rates, cumulative risks, competing risks, state probabilities with multiple states and time scales

with R and Epi::

Computer practicals

Baker HDI / Monash University
February 2023
http://bendixcarstensen.com/AdvCoh/courses/Melb-2023/
Version 3

Compiled Monday 20th February, 2023, 08:27
from: C:\Bendix\teach\AdvCoh\courses\Melb.2023\pracs/pracs.tex

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

Contents

Acknowledgement

I acknowledge the Traditional Owners of the land on which Baker Institute (Melbourne) resides, the Boon Wurrung peoples of the Yaluk-ut Weelam clan. I pay my respects to all Elders past, present and future.

0.0 Preface

This course draws to some extent on the content of the book "Epidemiology with R" [**?**], (http://bendixcarstensen.com/EwR), but in particular on the draft of book (which by no means is sure ever to appear as a book) "Practical multistate modeling with R and `Epi:Lexis`". The former is available through Oxford University Press, the latter as a draft (updated at unpredictable times) as http://bendixcarstensen.com/MSbook.pdf.

- The **target audience** is statisticians and epidemiologists working in epidemiological research

- The **prerequisites** are

  1. a basic knowledge of R,
  2. a working installation of R(`4.2.2`)
  3. a working installation of `Epi_2.47`
  4. a working installation of `popEpi_0.4.10`
  5. some epidemiological practice

- The **format** of the course will be short lectures closely aligned with the topics in the exercises. The exercises will be run in chunks between the short lectures.

Exercises are given including most of the solutions. You can get the exercise code chunks from the course website http://bendixcarstensen.com/AdvCoh/courses/Melb-2023

0.1 Program

Program

Wednesday 22 February 2023	
10:00–10:30	L: Introduction to multistate models
10:30–11:30	P: Survival analysis: rates
11:30–12:15	Lunch
12:15–13:15	L: Introduction to competing risks
13:15–16:00	P: Cause-specific rates and competing risks
Thursday 23 February 2023	
10:00–11:00	L: Multistate models in practice
11:00–12:30	P: Multistate models
12:30–13:15	Lunch
13:15–15:30	P: State probabilities
15:30–16:00	Q: Wrap-up and questions

Within each of the the topics (see the table of contents) there will be a short introductury lecture, introducing the practical.

Chapter 1

Survival and rates: `lung`

Paraphernalia

It is advisable to load all packages needed at the start:

```
> library(survival)
> library(Epi)
> library(popEpi)
> # popEpi::splitMulti returns a data.frame rather than a data.table
> options("popEpi.datatable" = FALSE)
> clear()
```

```
[1] R version 4.2.0 (2022-04-22 ucrt)
```

```
       Version Built
Epi    2.47    4.2.1
popEpi 0.4.9   4.2.0
```

1.1 Data and simple survival

1. Load the `lung` data from the `survival` package, and convert `sex` to a factor (*always* do that with categorical variables). Also we rescale time from days to months:

```
> data(lung)
> lung$sex <- factor(lung$sex,
+                    levels = 1:2,
+                    labels = c("M", "W"))
> lung$time <- round(lung$time / (365.25/12), 3)
> head(lung)
```

```
  inst   time status age sex ph.ecog ph.karno pat.karno meal.cal wt.loss
1    3 10.053      2  74   M       1       90       100     1175      NA
2    3 14.949      2  68   M       0       90        90     1225      15
3    3 33.183      1  56   M       0       90        90       NA      15
4    5  6.899      2  57   M       1       90        60     1150      11
5    1 29.010      2  60   M       0      100        90       NA       0
6   12 33.577      1  74   M       1       50        80      513       0
```

2. Use `survfit` to construct the Kaplan-Meier estimator of overall survival:

```
> ?Surv
> ?survfit

> km  <- survfit(Surv(time, status == 2) ~ 1, data = lung)
> km
Call: survfit(formula = Surv(time, status == 2) ~ 1, data = lung)

       n events median 0.95LCL 0.95UCL
[1,] 228    165   10.2    9.36    11.9
> # summary(km) # very long output
```

   The standard print method just prints the number of events and the median survival, while the `summary` prints the entire survival function estimate.

   We can plot the survival curve—this is the default plot for a `survfit` object:

```
> plot(km)
```

   What is the median survival? What does it mean?

3. Explore if survival patterns between men and women are different:

```
> kms <- survfit(Surv(time, status == 2) ~ sex, data = lung)
> kms
Call: survfit(formula = Surv(time, status == 2) ~ sex, data = lung)

        n events median 0.95LCL 0.95UCL
sex=M 138    112   8.87    6.96    10.2
sex=W  90     53  14.00   11.43    18.1
```

   We can plot the two resulting survival curves with confidence limits:

```
>  plot(kms, col = c("blue", "red"), lwd = 1, conf.int = TRUE)
> lines(kms, col = c("blue", "red"), lwd = 3)
```

   e see that men have worse survival than women, but they are also a bit older (`age` is age at diagnosis of lung cancer):

```
> with(lung, tapply(age, sex, mean))
       M        W
63.34058 61.07778
```

   Formally there is a significant difference in survival between men and women

```
> ?survdiff
> survdiff(Surv(time, status==2) ~ sex, data = lung)
```

   What is the null hypothesis tested here?

   Assumptions: We are actually testing whether the rate-ratio is 1, assuming proportional hazards (the log-rank test)

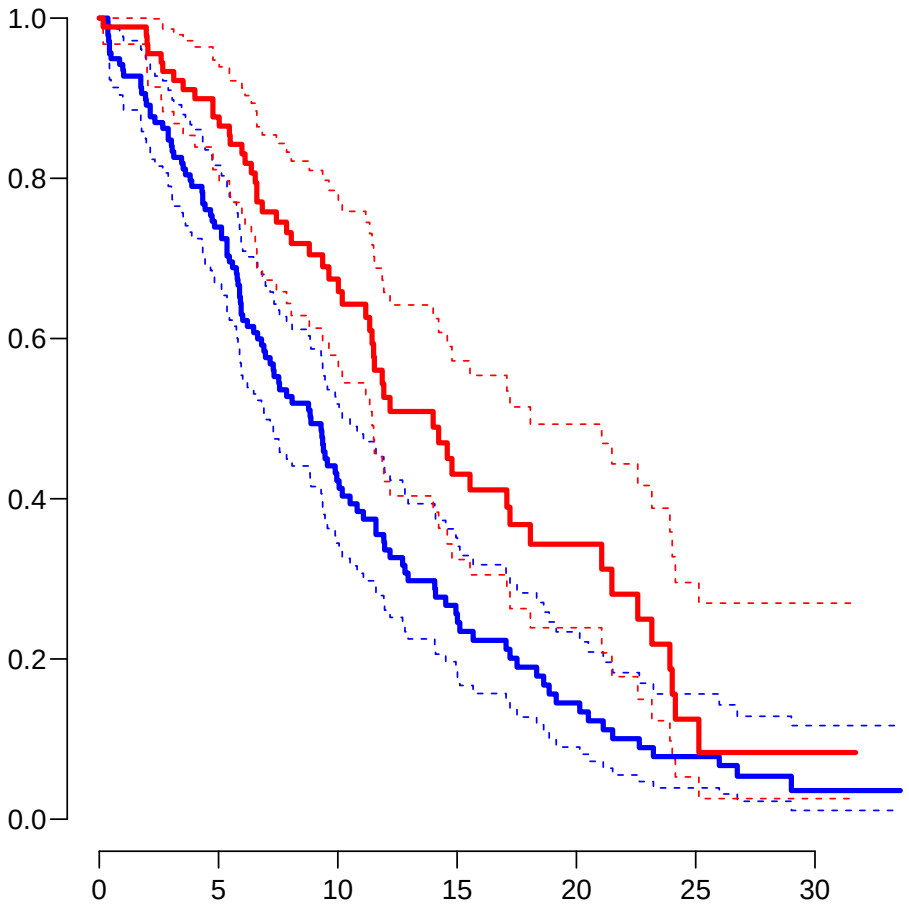


Figure 1.1: *Kaplan-Meier estimators of survival for men (blue) and women (red). W* `../graph/surv-kms`

1.2 Rates and rate-ratios: Simple Cox model

4. Now explore *how much* sex and age (at diagnosis) influence the mortality—note that we are now addressing the mortality *rate* and not the survival in a Cox-model:

```
> c0 <- coxph(Surv(time, status == 2) ~ sex             , data = lung)
> c1 <- coxph(Surv(time, status == 2) ~ sex + I(age/10), data = lung)
> summary(c1)

Call:
coxph(formula = Surv(time, status == 2) ~ sex + I(age/10), data = lung)

  n= 228, number of events= 165

               coef exp(coef) se(coef)      z Pr(>|z|)
sexW       -0.51322   0.59857  0.16746 -3.065  0.00218 **
I(age/10)   0.17045   1.18584  0.09223  1.848  0.06459 .
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

          exp(coef) exp(-coef) lower .95 upper .95
sexW         0.5986     1.6707    0.4311    0.8311
I(age/10)    1.1858     0.8433    0.9897    1.4208

Concordance= 0.603  (se = 0.025 )
Likelihood ratio test= 14.12  on 2 df,   p=9e-04
Wald test            = 13.47  on 2 df,   p=0.001
Score (logrank) test = 13.72  on 2 df,   p=0.001
```

```
> ci.exp(c0)
```

```
     exp(Est.)      2.5%     97.5%
sexW 0.5880028 0.4237178 0.8159848
```

```
> ci.exp(c1)
```

```
          exp(Est.)      2.5%     97.5%
sexW       0.598566 0.4310936 0.8310985
I(age/10)  1.185842 0.9897335 1.4208086
```

We see that there is not much confounding by age; the W/M mortality rate-ration (RR) (hazard ratio, HR, is another word for this) is slightly below 0.6 whether age is included or not.

The age effect is formally non-significant, the estimate corresponds to a 19% higher mortality rate per 10 years of age at diagnosis (mortality RR or hazard ratio of 1.1858).

5. We can check if the assumption of proportional hazards holds, `cox.zph` provides a test, and the plot method shows the Schoenfeld residuals and a smooth of them; interpretable as an estimate of the interaction effect; that is how the W/M (log) rate-ratio depends on time:

```
> ?cox.zph
```

```
> cox.zph(c0)
```

```
       chisq df     p
sex     2.86  1 0.091
GLOBAL  2.86  1 0.091
```

```
> (z1 <- cox.zph(c1))
```

```
          chisq df    p
sex       2.608  1 0.11
I(age/10) 0.209  1 0.65
GLOBAL    2.771  2 0.25
```

```
> par(mfrow = c(1, 2)) ; plot(z1)
```

If the proportional hazards model holds, then the resulting lines in he plots should be approximately horizontal. But we shall return to a more explicit way of addressing this.

6. We see that there is no systematic pattern for age, but an increase by sex. The `cox.zph` really gives a test for an *interaction* between each covariate and the time scale.

   We will keep that in mind so we can assess this through proper modeling of the interaction—the Cox model does not provide an estimate of the effect of `time` on mortality, and the by that token it is impossible to estimate any interactions with time either.

7. Above we showed the Kaplan-Meier estimator for each of the two sexes. We can also show the estimated survival curves for the two sexes as derived from the Cox-model. This requires a *prediction* data frame—a data frame with the same variables as in the Cox-model and values of these representing the persons for whom we want predictions:

   ```
   > prs <- survfit(c0, newdata = data.frame(sex = c("M","W")))
   > plot(prs, col = c("blue", "red"))
   ```

   How is the shape of the two curves relative to each other?

   How is the shape of the two curves relative to the two Kaplan-Meier curves?

8. Try to over-plot the Cox-prediction on the Kaplan-Meier curves:

   ```
   >  plot(prs, col = c("blue", "red"), lwd = 1, lty = 1, conf.int = FALSE)
   > lines(prs, col = c("blue", "red"), lty = 1, lwd=3)
   > lines(kms, col = c("blue", "red"), lty = "11", lwd = 2, lend = "butt")
   ```

   Do they agree? What does that mean?

So far we have only seen the rates through the glasses of survival curves—it is a little odd to try and judge hazards (rates) on a transformed sace, instaed of looking at the hazards directly.

1.3 Simple Poisson model

But we do not know how the mortality *per se* looks as a function of `time` (since diagnosis). That function is not available from the Cox-model or from the `survfit` object. To that end we must provide a model for the effect of time on mortality; the simplest is of course to assume that it is constant or a simple linear function of time.

9. For a start we assume that the mortality is constant over time. If it is so, then the likelihood for the model is equivalent to a Poisson likelihood, which can be fitted using the `poisreg` family from the `Epi` package—note in poarticular the the response is a two-column matrix (created with `cbind`) of events count and person-time (here `time`, because every one starts at 0):

```
> ?poisreg
```

```
> with(lung, cbind(status == 2, time)[1:10,])
```

```
          time
 [1,] 1 10.053
 [2,] 1 14.949
 [3,] 0 33.183
 [4,] 1  6.899
 [5,] 1 29.010
 [6,] 0 33.577
 [7,] 1 10.185
 [8,] 1 11.860
 [9,] 1  7.162
[10,] 1  5.454
```

```
> p1 <- glm(cbind(status == 2, time) ~ sex + I(age/10),
+           family = poisreg,
+             data = lung)
> ci.exp(p1) # estimates form Poisson
```

```
             exp(Est.)       2.5%     97.5%
(Intercept) 0.03255173 0.01029235 0.1029518
sexW        0.61820344 0.44555514 0.8577513
I(age/10)   1.16904426 0.97796555 1.3974567
```

```
> ci.exp(c1) # estimates from Cox
```

```
          exp(Est.)      2.5%     97.5%
sexW       0.598566 0.4310936 0.8310985
I(age/10)  1.185842 0.9897335 1.4208086
```

We see that the estimates of sex and age effects are quite close between the Poisson and the Cox models, but also that the Poisson model has an intercept term, the estimate of the (assumed) constant underlying mortality. Since we entered the risk time part of the response (second argument in the `cbind`) in units of months (remember we rescaled in the beginning?), the `(Intercept)` (taken from the `ci.exp`) is a rate per 1 person-month.

What age and sex does the `(Intercept)` refer to?

10. The syntax for `poisreg` is a bit different from that for `poisson`, which would be:

```
> px <- glm(status == 2  ~ sex + age + offset(log(time)),
+           family = poisson,
+             data = lung)
> ## or:
> px <- glm(status == 2  ~ sex + age,
+           offset = log(time),
+           family = poisson,
+             data = lung)
> ci.exp(px)
```

The formulation with the `offset` is the reason that papers use the description "... we fitted a Poisson model with log person years as offset".

The drawback of the `poisson` approach is that you need the (risk) `time` (person-years) as a variable in the prediction frame. This is not the case for `poisreg`, where you get the predicted rates per unit in which you entered the person years when specifying the model.

We shall return to prediction of `rates`.

1.4 Representation of follow-up: `Lexis` object

If we want to see how mortality varies by age we must split the follow-up of each person in small intervals of say, 30 days. This is most easily done using a `Lexis` object. That is basically just taking the `lung` dataset and adding a few features that defines times and states. The point is that it makes life a lot easier when things get more complex than just simple survival. It can be seen as an expansion of the `stset` facility in Stata.

11. First make a `Lexis` object—make sure you know what `Lexis` does:

```
> ?Lexis
```

```
> Ll <- Lexis(exit = list(tfl = time),
+             exit.status = factor(status,
+                                  levels = 1:2,
+                                  labels = c("Alive","Dead")),
+             data = lung)

NOTE: entry.status has been set to "Alive" for all.
NOTE: entry is assumed to be 0 on the tfl timescale.
```

```
> head(Ll)

 lex.id tfl lex.dur lex.Cst lex.Xst inst   time status age sex ph.ecog ph.karno pat.ka
      1   0   10.05   Alive    Dead    3 10.053      2  74   M       1       90
      2   0   14.95   Alive    Dead    3 14.949      2  68   M       0       90
      3   0   33.18   Alive   Alive    3 33.183      1  56   M       0       90
      4   0    6.90   Alive    Dead    5  6.899      2  57   M       1       90
      5   0   29.01   Alive    Dead    1 29.010      2  60   M       0      100
      6   0   33.58   Alive   Alive   12 33.577      1  74   M       1       50
 meal.cal wt.loss
```

```
1175      NA
1225      15
  NA      15
1150      11
  NA       0
 513       0
```

We see that 5 variables have been added to the dataset:

`lex.id`: a numerical id of each record in the dataset (normally this will be a person id). Can be set explicitly with the `id=` argument of `Lexis`.

`tfl`: time from lung cancer *at the time of entry*, therefore it is 0 for all persons; the entry time is 0 from the entry time. The name of this variable was taken from the `exit=` argument to `Lexis`.

`lex.dur`: the *length* of time a person is in state `lex.Cst`, here measured in months, because `time` is.

`lex.Cst`: Current state, the state in which the `lex.dur` time is spent.

`lex.Xst`: eXit state, the state to which the person moves after the `lex.dur` time in `lex.Cst`.

This seems a bit of an overkill for keeping track of time and death for the lung cancer patients, but the point here is that this generalizes to multistate data too.

It also gives a handy overview of the follow-up:

```
> summary(L1)
```

```
Transitions:
     To
From    Alive Dead  Records:  Events: Risk time:  Persons:
  Alive    63  165       228      165    2286.42       228
```

What is the average follow-up time for persons?

For a graphical representation, try:

```
> ?boxes
```

```
> boxes(L1, boxpos = TRUE)
```

Explain the numbers in the resulting graph. Redo the graph with risk time counted in years.

12. We can make the Cox-analysis using the `Lexis`-specific variables by:

```
> ?Surv
```

```
> cl <- coxph(Surv(tfl,
+                  tfl + lex.dur,
+                  lex.Xst == "Dead") ~ sex + age,
+             data = Ll)
```

but even simpler, by using the `Lexis` features:

```
> ?coxph.Lexis
```

```
> cL <- coxph.Lexis(Ll, tfl ~ sex + age)
```

```
survival::coxph analysis of Lexis object Ll:
Rates for the transition:
Alive->Dead
Baseline timescale: tfl
```

```
> ci.exp(cL)
```

```
     exp(Est.)      2.5%     97.5%
sexW  0.598566 0.4310936 0.8310985
age   1.017191 0.9989686 1.0357467
```

```
> ci.exp(cl)
```

```
     exp(Est.)      2.5%     97.5%
sexW  0.598566 0.4310936 0.8310985
age   1.017191 0.9989686 1.0357467
```

13. And we can make the Poisson-analysis by:

```
> pc <- glm(cbind(lex.Xst == "Dead", lex.dur) ~ sex + age,
+           family = poisreg,
+             data = Ll)
```

or even simpler, by using the `Lexis` features:

```
> pL <- glm.Lexis(Ll, ~ sex + age)
```

```
stats::glm Poisson analysis of Lexis object Ll with log link:
Rates for the transition:
Alive->Dead
```

```
> ci.exp(pL)
```

```
             exp(Est.)      2.5%     97.5%
(Intercept) 0.03255173 0.01029235 0.1029518
sexW        0.61820344 0.44555514 0.8577513
age         1.01574126 0.99777440 1.0340317
```

```
> ci.exp(pc)
```

```
             exp(Est.)      2.5%     97.5%
(Intercept) 0.03255173 0.01029235 0.1029518
sexW        0.61820344 0.44555514 0.8577513
age         1.01574126 0.99777440 1.0340317
```

Remember that the Poisson-model fitted is a pretty brutal approximation to the Cox-model—it assumes that the baseline hazard is constant, whereas the Cox-model allows the baseline hazard to vary arbitrarily by time.

1.5 Estimating the hazard function: splitting time

If we want a more detailed version of the baseline hazard we split follow-up time in small intervals, assume that the hazard is constant in each small interval, and assume the the *size* of the hazard varies smoothly with time, `tfl`:

14. We can subdivide the follow-up in small intervals by `survival:::survSplit`, `Epi:::splitLexis` or `popEpi:::splitMulti` (and possibly many more). The `splitMulti` is by far the easiest to use (and fastest as well). Recall we rescaled time to months, so we split in 1 month intervals (at times 0 to 36 months):

```
> Sl <- splitMulti(Ll, tfl = 0:36)
```

This will split the follow-up along the time-scale `tfl` at times 0, 1, . . . , 36 months; we see that the follow-up time is the same, but there are now about 10 times as many records:

```
> summary(Ll)

Transitions:
     To
From    Alive Dead  Records:  Events: Risk time:  Persons:
  Alive    63  165       228      165    2286.42       228

> summary(Sl)

Transitions:
     To
From    Alive Dead  Records:  Events: Risk time:  Persons:
  Alive  2234  165      2399      165    2286.42       228
```

We can see how the follow up for person, 10 say, is in the original and the split dataset:

```
> wh <- names(Ll)[1:10] # names of variables in some order
> subset(Ll, lex.id == 10)[,wh]

 lex.id tfl lex.dur lex.Cst lex.Xst inst  time status age sex
     10   0    5.45   Alive    Dead    7 5.454      2  61   M

> subset(Sl, lex.id == 10)[,wh]

 lex.id tfl lex.dur lex.Cst lex.Xst inst  time status age sex
     10   0    1.00   Alive   Alive    7 5.454      2  61   M
     10   1    1.00   Alive   Alive    7 5.454      2  61   M
     10   2    1.00   Alive   Alive    7 5.454      2  61   M
     10   3    1.00   Alive   Alive    7 5.454      2  61   M
     10   4    1.00   Alive   Alive    7 5.454      2  61   M
     10   5    0.45   Alive    Dead    7 5.454      2  61   M
```

In `Sl` each record now represents a small interval of follow-up for a person, so each person has many records. The main thing to note here is `tfl`, which represents the time since lung cancer diagnosis at the beginning of each interval, and `lex.dur` representing the risk time ("person-years", in months though)—the length of the interval. The reason we need to split follow-up in small pieces is that the modeling implicitly assumes that the mortality rate is constant within each interval. If intervals are too long, this assumption is a bad approximation.

15. We can now include a smooth effect of `tfl` in the Poisson-model allowing the baseline hazard to vary by time since lung cancer. That is done by natural splines, `Ns`:

```
> ps <- glm(cbind(lex.Xst == "Dead", lex.dur)
+           ~ Ns(tfl, knots = seq(0, 36, 12)) + sex + age,
+           family = poisreg,
+             data = Sl)
> ci.exp(ps)
```

```
                                   exp(Est.)        2.5%       97.5%
(Intercept)                      0.01898426 0.005700994  0.06321739
Ns(tfl, knots = seq(0, 36, 12))1 2.40380283 0.809425097  7.13873100
Ns(tfl, knots = seq(0, 36, 12))2 4.15015448 0.436289141 39.47790716
Ns(tfl, knots = seq(0, 36, 12))3 0.83994013 0.043932122 16.05885147
sexW                             0.59871596 0.431231819  0.83124850
age                              1.01658684 0.998376760  1.03512907
```

or even simpler:

```
> ?glm.Lexis
> ps <- glm.Lexis(Sl, ~ Ns(tfl, knots = seq(0, 36, 12)) + sex + age)
> ci.exp(ps)
```

16. Compare these to the regression estimates from the Cox-model and from the model with constant baseline:

```
> ests <- cbind(ci.exp(cl),
+               ci.exp(ps, subset = c("sex","age")),
+               ci.exp(pc, subset = c("sex","age")))
> colnames(ests)[1:3*3-2] <- c("Cox","Pois-spline","Pois-const")
> round(ests, 3)
```

```
       Cox  2.5% 97.5% Pois-spline  2.5% 97.5% Pois-const  2.5% 97.5%
sexW 0.599 0.431 0.831       0.599 0.431 0.831      0.618 0.446 0.858
age  1.017 0.999 1.036       1.017 0.998 1.035      1.016 0.998 1.034
```

We see that the smooth parametric Poisson model and the Cox model produce virtually the same estimates of age and sex-effects, whereas the Poisson model with constant hazard produce slightly different ones.

17. The proportional hazards assumption is the same for the Cox model and both the Poisson models: The M/W hazard ratio is the same at any time after diagnosis. What differs between the models is the assumed shape of the hazard (*not* a hazard ratio).

    The Cox model allows the baseline rate to change arbitrarily at every event time, not using the quantitative nature of time; the spilne Poisson model has a baseline that varies smoothly by time and the constant Poisson model has a baseline that is constant over time. The latter is clearly not tenable, whereas the smooth Poisson model and the Cox model give the same regression estimates.

18. So we now have a *parametric* model for the baseline hazard which means that we can show the estimated baseline hazard for, say, a 60 year old woman, by supplying a suitable prediction frame, i.e. a data frame where each row represents a set of covariate values (including the time) where we want the predicted mortality—her we use times from 0 to 30 months in steps of 0.2 months:

    ```
    > prf <- data.frame(tfl = seq(0, 30, 0.2),
    +                   sex = "W",
    +                   age = 60)
    ```

    The choice of prediction points are totally unrelated to the intervals in which we split the follow-up for analysis.

    We can over-plot with the predicted rates from the model where mortality rates are constant, the only change is the model (`pc` instead of `ps`):

    ```
    > matshade(prf$tfl, ci.pred(ps, prf), lwd = 3, plot = TRUE, log = "y")
    > matshade(prf$tfl, ci.pred(pc, prf), lwd = 3, lty = "21", lend = "butt")
    ```

    What we see from the plot is that mortality rates are increasing during the first 1.5 years after lung cancer and then leveling off.

    Put some sensible axis labels on the plot, and rescale the rates to rates per 1 person-year.

19. We can transform the hazard function, $\lambda(t)$, to a survival function, $S(t)$ using the relationship $S(t) = \exp(-\int_0^t \lambda(u)\,\mathrm{d}u)$. This integration exercise is implemented in the `ci.surv` function, which takes a model and a prediction data frame as arguments; the prediction data frame must correspond to a sequence of equidistant time points (makes life easier for the coder), so we can use `prf` for this purpose:

    ```
    > matshade(prf$tfl, ci.surv(ps, prf, intl = 0.2),
    +          plot = TRUE, ylim = 0:1, lwd = 3)
    ```

    This is the survival function for a 60 year old woman.

    We can expand this by overlaying the survival function from the model with constant hazard (also known as ”exponential(y distributed) survival”) and the KM-estimator

```
> matshade(prf$tfl, ci.surv(ps, prf, intl = 0.2),
+          plot = TRUE, ylim = 0:1, lwd = 3)
> lines(prf$tfl, ci.surv(pc, prf, intl = 0.2)[,1], lwd = 2, col = gray(0.5))
> lines(survfit(c1, newdata = data.frame(sex = "W", age = 60)),
+       lwd = 2, lty = 1)
```

We see that the survival function from the constant hazard model is quite a bit off, but also a good correspondence between the Cox-model based survival and the survival from the parametric hazard function.

We can bring the plots together in one graph:

```
> par(mfrow = c(1,2))
> # hazard scale
> matshade(prf$tfl, ci.pred(ps, prf),
+          plot = TRUE, log = "y", lwd = 3, xlim = c(0,30))
> matshade(prf$tfl, ci.pred(pc, prf), lty = 3, lwd = 3)
> #
> # survival
> matshade(prf$tfl, ci.surv(ps, prf, intl = 0.2),
+          plot = TRUE, ylim = 0:1, lwd = 3, xlim = c(0,30))
> matshade(prf$tfl, ci.surv(pc, prf, intl = 0.2),
+          lty = 3, alpha = 0, lwd = 3)
> lines(survfit(c1, newdata = data.frame(sex = "W", age = 60)),
+       col = "forestgreen", lwd = 3)
```

20. We have compared the predicted survival curve from a Poisson model with age and sex and time since lung cancer as covariates to that from a Cox-model with age and sex as covariates and time since lung cancer as underlying time scale.

    We now go back to the Kaplan-Meier estimator and compare that to the corresponding Poisson-model, which is one with time (`tfl`) as the only covariate:

```
> par(mfrow=c(1,2))
> pk <- glm(cbind(lex.Xst == "Dead",
+                 lex.dur) ~ Ns(tfl, knots = seq(0, 36, 12)),
+           family = poisreg,
+             data = Sl)
> #or
> pk <- glm.Lexis(Sl, ~ Ns(tfl, knots = seq(0, 36, 12)))
```

```
stats::glm Poisson analysis of Lexis object Sl with log link:
Rates for the transition:
Alive->Dead
```

```
> # hazard
> matshade(prf$tfl, ci.pred(pk, prf),
+          plot = TRUE, log = "y", lwd = 3, ylim = c(0.01,1))
> # survival from smooth model
> matshade(prf$tfl, ci.surv(pk, prf, intl = 0.2) ,
+          plot = TRUE, lwd = 3, ylim = 0:1)
> # K-M estimator
> lines(km, lwd = 2)
```

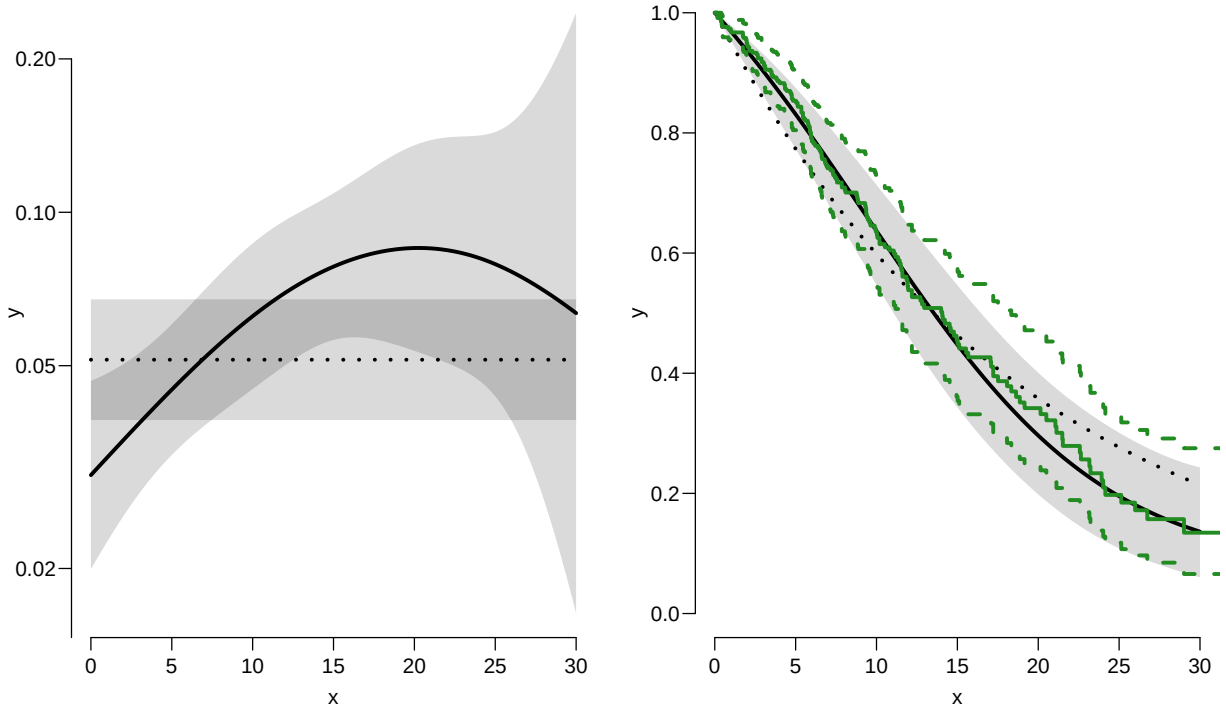


Figure 1.2: *Hazards (left) and survival (right) for 60 year old women. The left hand plot is unavailable from the Cox model.*

../graph/surv-ratesurv

21. We can explore how the tightness of the knots in the smooth model influence the underlying hazard and the resulting survival function. This is easiest done by setting up a function that does the analysis with different distance between of knots

```
> zz <-
+ function(dk)
+ {
+ kn <<- seq(0, 36, dk) # must be in global environment...
+ print(kn)
+ pk <<- glm.Lexis(Sl, ~ Ns(tfl, knots = kn))
+ matshade(prf$tfl, ci.pred(pk, prf),
+           plot = TRUE, log = "y", lwd = 2, ylim = c(0.01,1), xlim = c(0,30))
+ rug(kn, lwd=3)
+
+ plot(km, lwd = 2, col = "limegreen", xlim = c(0,30))
+ matshade(prf$tfl, ci.surv(pk, prf, intl = 0.2) ,
+           lwd = 2, ylim = 0:1, xlim = c(0,30))
+ }


> par(mfrow=c(1,2))
> zz(18)
> zz(12)
> zz(6)
```

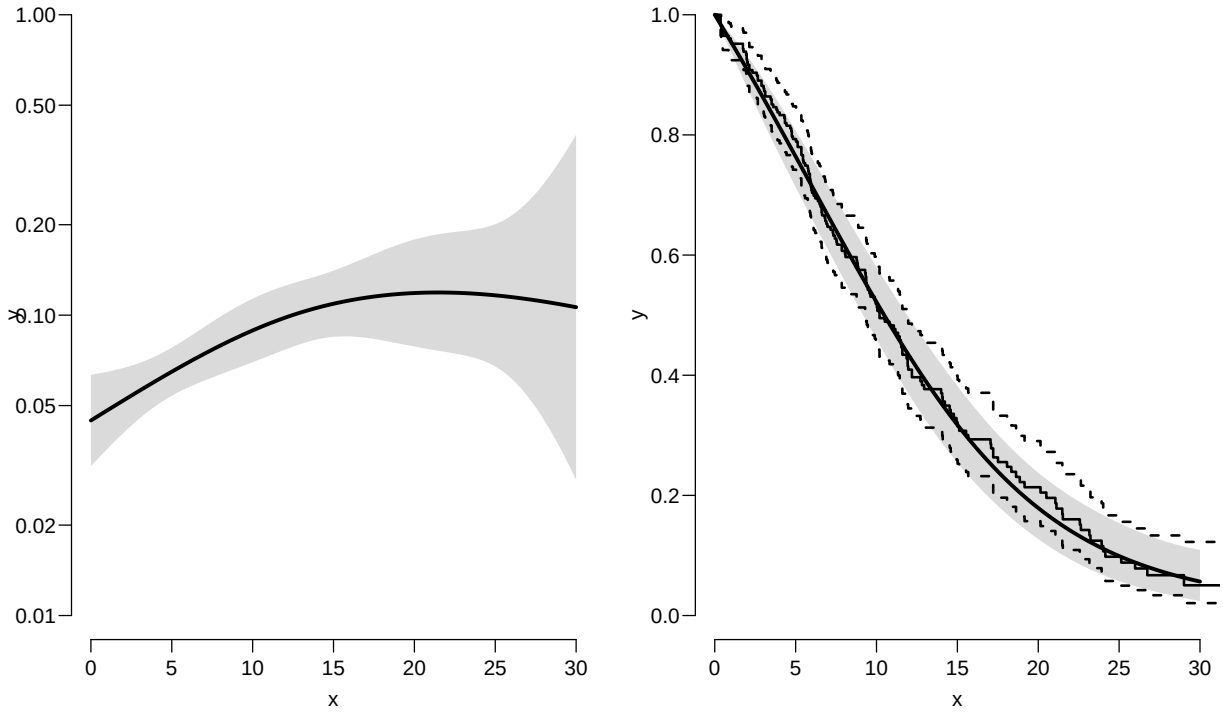


Figure 1.3: *Baseline hazard (left), and corresponding survival function from parametric model and Kaplan-Meier estimator.*

`../graph/surv-parkm`

```
> zz(4)
> zz(3)
```

```
> par(mfrow=c(5,2), mar = rep(1,4))
> for (nk in c(18, 12, 6, 4, 3)) zz(nk)

[1]  0 18 36
stats::glm Poisson analysis of Lexis object Sl with log link:
Rates for the transition:
Alive->Dead

[1]  0 12 24 36
stats::glm Poisson analysis of Lexis object Sl with log link:
Rates for the transition:
Alive->Dead

[1]  0  6 12 18 24 30 36
stats::glm Poisson analysis of Lexis object Sl with log link:
Rates for the transition:
Alive->Dead

 [1]  0  4  8 12 16 20 24 28 32 36
stats::glm Poisson analysis of Lexis object Sl with log link:
```

```
Rates for the transition:
Alive->Dead

 [1]  0  3  6  9 12 15 18 21 24 27 30 33 36
stats::glm Poisson analysis of Lexis object Sl with log link:
Rates for the transition:
Alive->Dead
```

You will see that the more knots you include, the closer the parametric estimate gets to the Kaplan-Meier estimator. But also that the estimated underlying hazard becomes increasingly silly. The ultimate silliness is of course achieved when we arrive at the Kaplan-Meier estimator.

Fortunately the baseline hazard underlying the Kaplan-Meier and the Breslow estimator is rarely shown.

1.6 Conclusion

- The Cox-model and the Poisson-model gives the same results for the regression parameters, provided the baseline is properly modeled.
- The Cox-model is a *partial* model, it only gives the HRs, not the rates. The baseline rates are missing.
- The Poisson-model is a *full* model for the rates. Any set of predicted rates (and derivatives thereof) can be derived from the Poisson model.
- AN extreme version of the Poisson model is identical to the Cox-model.

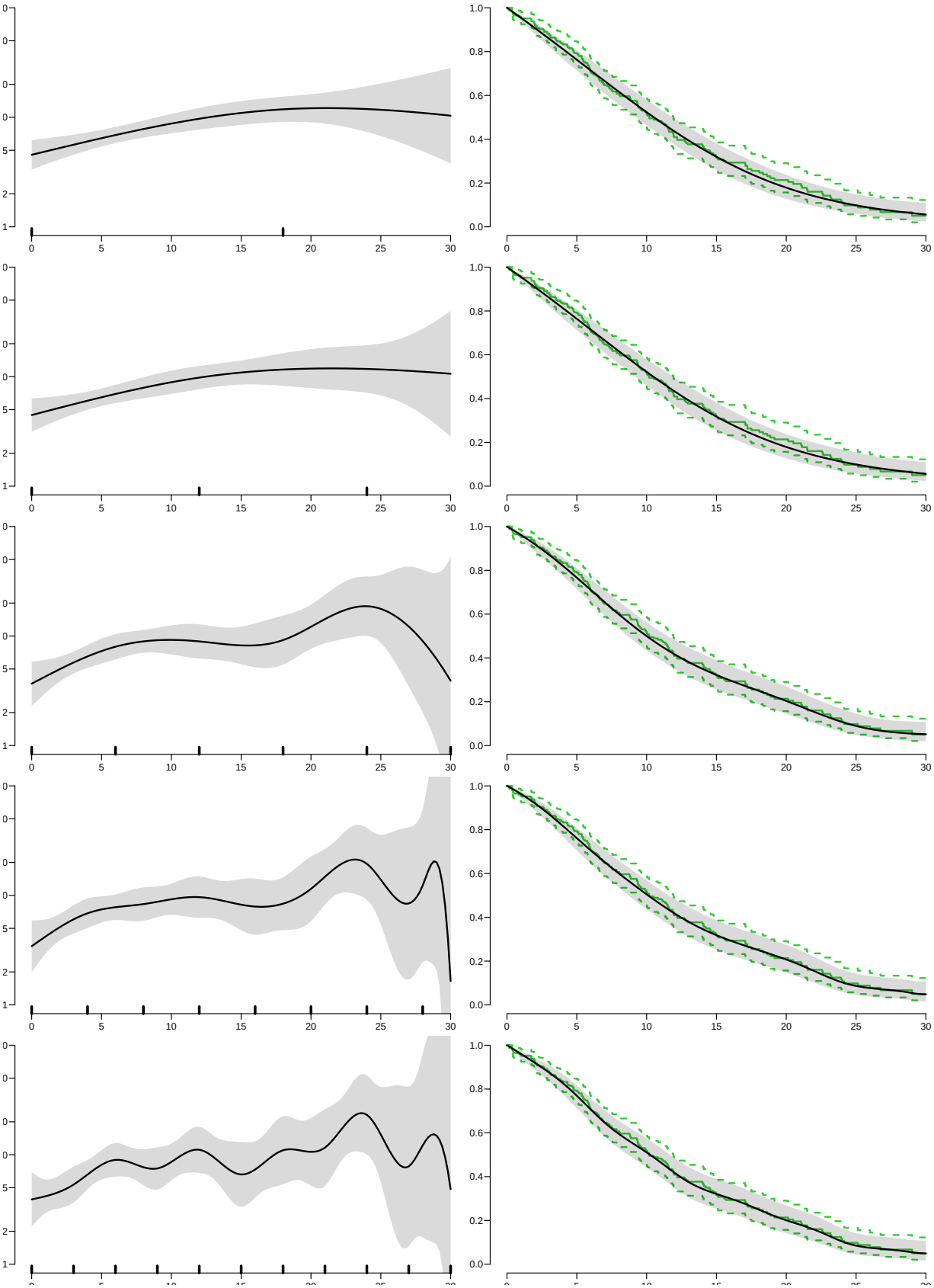


Figure 1.4: *Hazard (left) and survival (right) comparing a parametric model with different number of knots (black) and the Kaplan-Meier estimator (green curve).*

`../graph/surv-knots2`

Chapter 2

Competing risks: DMlate

Paraphernalia

It is advisable to load all packages needed at the start:

```
> library(survival)
> library(Epi)
> library(popEpi)
> # popEpi::splitMulti returns a data.frame rather than a data.table
> options("popEpi.datatable" = FALSE)
> library(tidyverse)
> clear()
```

2.1 Data

This exercise follows quite closely the section on competing risks in "Epidemiology with R", pp. 207 and 210 ff. With the major exception that we will use the function `ci.Crisk`, which was not available in the `Epi` package when the book was written.

We shall use the `DMlate` dataset which is a random sample of Danish diabetes patients, with dates of birth, diabetes, OAD start, insulin start and death, all diagnosed after 1996-1-1.

We want to look at the event "start of insulin use", which occurs at `doins`, while taking death as competing event into account. This means that we want to address the question of the probability of starting Ins, while taking death into account. Essentially estimating the probability of being in each of the states `DM`, `Ins` and `Dead`, where `Ins` means "started insulin and either alive or dead after this" and `Dead` means "dead without starting insulin".

1. Load the `DMlate` data from the `Epi` package (and for ease of calculation restrict to a random sample of 2000 persons while developing):

   ```
   > data(DMlate)
   > # set.seed(1952)
   > # DMlate <- DMlate[sample(1:nrow(DMlate), 2000),]
   > str(DMlate)
   ```

```
'data.frame':        10000 obs. of  7 variables:
 $ sex  : Factor w/ 2 levels "M","F": 2 1 2 2 1 2 1 1 2 1 ...
 $ dobth: num  1940 1939 1918 1965 1933 ...
 $ dodm : num  1999 2003 2005 2009 2009 ...
 $ dodth: num  NA NA NA NA NA ...
 $ dooad: num  NA 2007 NA NA NA ...
 $ doins: num  NA NA NA NA NA NA NA NA NA NA ...
 $ dox  : num  2010 2010 2010 2010 2010 ...
```

```
> head(DMlate)
```

```
       sex    dobth     dodm    dodth    dooad doins      dox
50185    F 1940.256 1998.917       NA       NA    NA 2009.997
307563   M 1939.218 2003.309       NA 2007.446    NA 2009.997
294104   F 1918.301 2004.552       NA       NA    NA 2009.997
336439   F 1965.225 2009.261       NA       NA    NA 2009.997
245651   M 1932.877 2008.653       NA       NA    NA 2009.997
216824   F 1927.870 2007.886 2009.923       NA    NA 2009.923
```

It is always wise to get en overview of the dates represented in the baseline dataset—preferably at a monthly level:

```
> par(mfrow=c(2,2))
> hist(DMlate$dobth, breaks = seq(1898, 2010, 1   ), col = "black")
> hist(DMlate$dodm , breaks = seq(1995, 2010, 1/12), col = "black")
>     abline(v = 1995:2020, col = "red")
> hist(DMlate$dodth, breaks = seq(1995, 2010, 1/12), col = "black")
>     abline(v = 1995:2020, col = "red")
> hist(DMlate$doins, breaks = seq(1995, 2010, 1/12), col = "black")
>     abline(v = 1995:2020, col = "red")
```

rom figure 2.1 we see that there is a tendency to fewer dates of diagnosis and insulin start in the summer (because of holydays) and more deaths in the winter, precisely as expected

2. First we define a `Lexis` object with the total follow up for each person:

```
> Ldm <- Lexis(entry = list(per = dodm,
+                           age = dodm - dobth,
+                           tfd = 0),
+              exit = list(per = dox),
+       exit.status = factor(!is.na(dodth),
+                            labels = c("DM", "Dead")),
+              data = DMlate)
```

```
NOTE: entry.status has been set to "DM" for all.
NOTE: Dropping  4  rows with duration of follow up < tol
```

```
> summary(Ldm)
```

```
Transitions:
     To
From   DM Dead  Records:  Events: Risk time:  Persons:
  DM 7497 2499      9996     2499   54273.27      9996
```

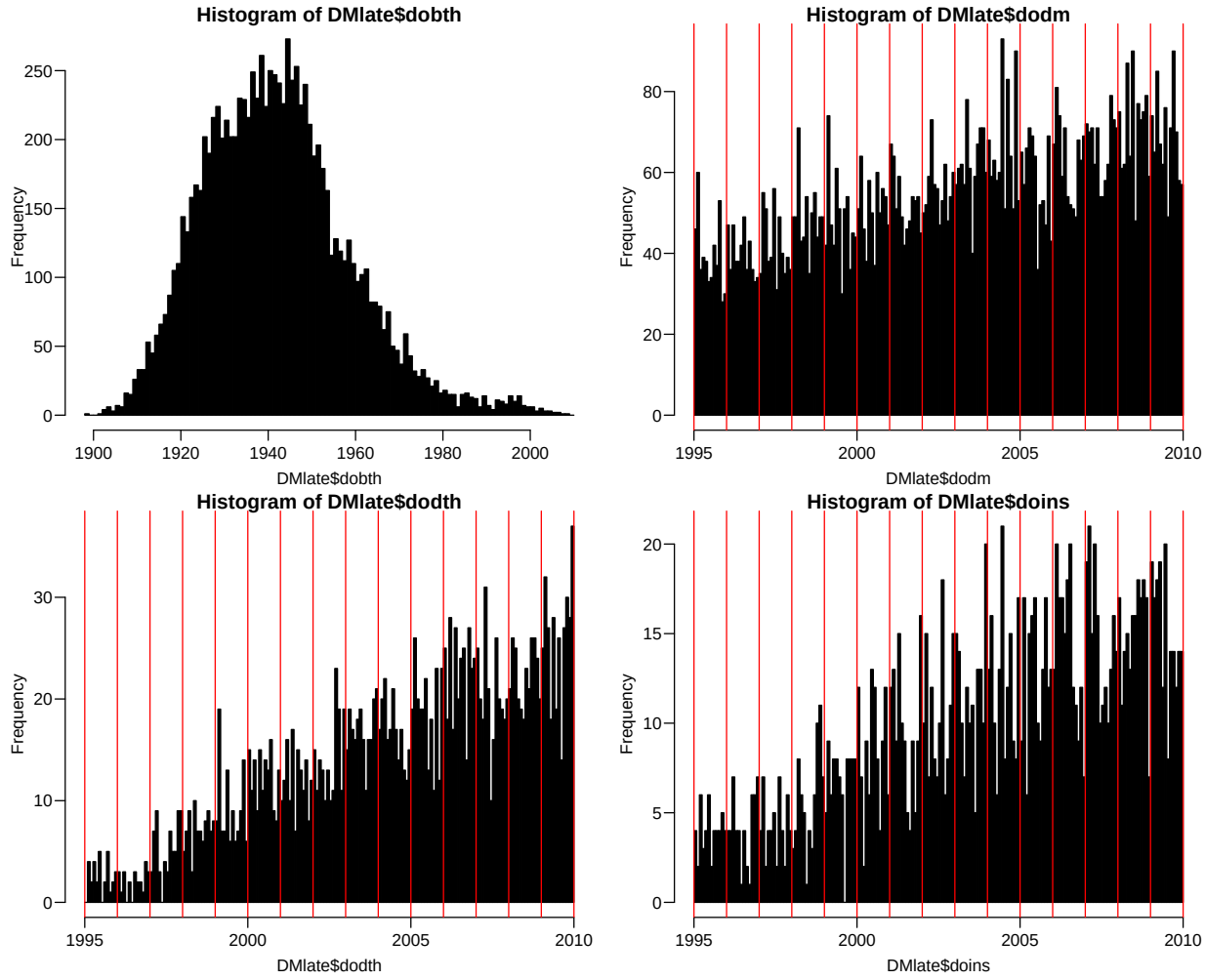


Figure 2.1: *Histograms of relevant dates. Vertical red lines are at each 1 January. F* `../graph/cmpr-DMhists`

Then subdivide the follow-up at the date of insuin, using `dooad`:

```
> Ldm <- sortLexis(Ldm)
> Cdm <- cutLexis(Ldm,
+                 cut = Ldm$doins,
+           timescale = "per",
+           new.state = "Ins")
> summary(Cdm)
```

```
Transitions:
     To
From    DM  Ins Dead  Records:  Events: Risk time:  Persons:
  DM  6157 1694 2048      9899      3742   45885.49      9899
  Ins    0 1340  451      1791       451    8387.77      1791
  Sum 6157 3034 2499     11690      4193   54273.27      9996
```

In this context we are not interested in what goes on after Ins so we only keep follow-up in state `DM`; we can use either `filter` or `subset`:

```
> Adm <- filter(Cdm, lex.Cst == "DM")
> Adm <- subset(Cdm, lex.Cst == "DM")
> summary(Adm)

Transitions:
     To
From   DM  Ins Dead  Records:  Events: Risk time:  Persons:
  DM 6157 1694 2048      9899     3742   45885.49      9899

> par(mfrow=c(1,2))
> boxes(Cdm, boxpos = TRUE, scale.R = 100, show.BE = TRUE)
> boxes(Adm, boxpos = TRUE, scale.R = 100, show.BE = TRUE)
```

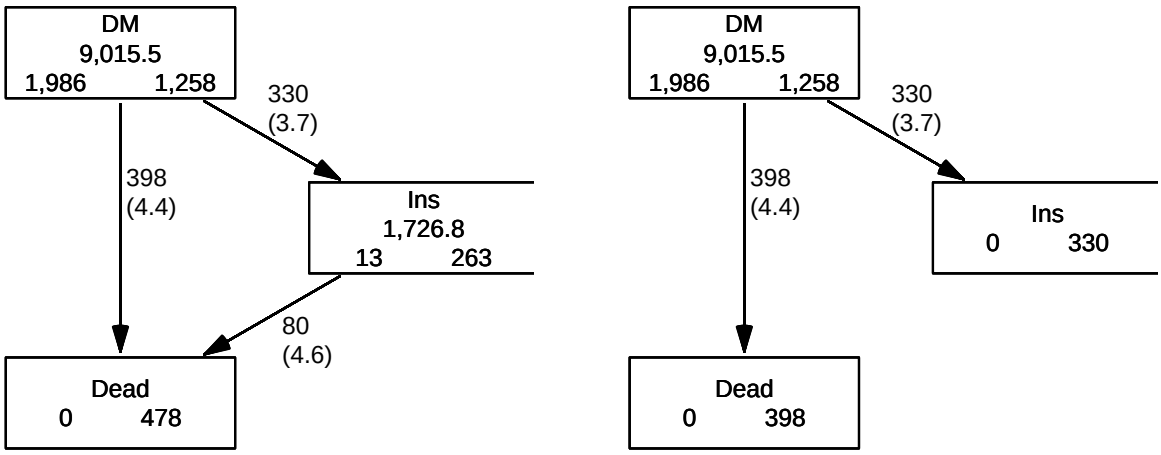


Figure 2.2: *Competing risks set-up for events* `Ins` *and* `Dead`.

`../graph/cmpr-boxCR`

As shown in figure 2.2 we now have a traditional competing risks set-up, with some 7500 DM patients starting without insulin, and where the quantity of interest is the probability of starting drug treatment, and the `Ins` state here means "having been on insulin treatment, disregarding subsequent death". The other event considered is `Dead` which here means "dead without initiating insulin treatment".

2.2 State probabilities

We can compute the (correct) counterpart of the survival function for this competing risks setup. The survival function we saw in the previous exercise gives the probability of being alive, and the complement is the probability of being dead.

3. `survfit` can do the corresponding calculation for the three states in the figure; the requirements are: 1) the third argument to the `Surv` function is a factor and 2) an `id` argument is given, pointing to an id variable that links together records belonging to the same person. The latter is superfluous in this case because there is only one record for each person, but even so it is required by the function `survfit`.

   Also note that the initial state (`DM`) must be the first level of the factor `lex.Xst`:

```
> levels(Adm$lex.Xst)

[1] "DM"   "Ins"  "Dead"

> m3 <- survfit(Surv(tfd,
+                    tfd + lex.dur,
+                    lex.Xst) ~ 1,
+                 id = lex.id,
+               data = Adm)
> names(m3)

 [1] "n"           "time"        "n.risk"      "n.event"     "n.censor"    "pstate"
 [7] "p0"          "cumhaz"      "std.err"     "sp0"         "logse"       "transition
[13] "lower"       "upper"       "conf.type"   "conf.int"    "states"      "type"
[19] "call"

> m3$states

[1] "(s0)" "Ins"  "Dead"

> head(cbind(time = m3$time, m3$pstate), 20)

             time
 [1,] 0.002737851 0.9988888 0.0003030609 0.0008081624
 [2,] 0.005475702 0.9982825 0.0005051424 0.0012123254
 [3,] 0.008213552 0.9972721 0.0011113869 0.0016164884
 [4,] 0.010951403 0.9955543 0.0024250496 0.0020206923
 [5,] 0.013689254 0.9939374 0.0038397633 0.0022227943
 [6,] 0.016427105 0.9916133 0.0057597319 0.0026269982
 [7,] 0.019164956 0.9883793 0.0087915703 0.0028291207
 [8,] 0.021902806 0.9858525 0.0108130026 0.0033344788
 [9,] 0.024640657 0.9820102 0.0140486211 0.0039411573
[10,] 0.027378508 0.9797855 0.0161722144 0.0040422808
[11,] 0.030116359 0.9774582 0.0182971236 0.0042446531
[12,] 0.032854209 0.9752321 0.0202196605 0.0045482115
[13,] 0.035592060 0.9734104 0.0215353535 0.0050542473
[14,] 0.038329911 0.9704742 0.0242690835 0.0052567458
[15,] 0.041067762 0.9686513 0.0256868690 0.0056618274
[16,] 0.043805613 0.9670301 0.0269027493 0.0060671208
[17,] 0.046543463 0.9653073 0.0280175399 0.0066751884
[18,] 0.049281314 0.9632802 0.0297405789 0.0069792541
[19,] 0.052019165 0.9615571 0.0308554865 0.0075873855
[20,] 0.054757016 0.9608476 0.0313622627 0.0077900960
```

   Because `lex.Xst` is a factor, `survfit` will compute the Aalen-Johansen estimator of being in a given state and place the probabilities in the matrix `m3$pstate`; the times

these refer to are in the vector `m3$time`. These are measured in years since diabetes, because `tfd` is in units of years,

Explore the object `m3`; start by using `names(m3)`.

Compare `m3$transitions` to `summary(Adm)`.

4. The `m3$pstate` contains the Aalen-Johansen probabilities of being in the `Alive`, having left to the `Ins`, resp. `Dead` state.

   Plot the three curves in the same graph (use for example `matplot`). Add the confidence limits.

5. These three curves have sum 1, so basically this is a way of distributing the probabilities across states at each time. It is therefore natural to stack the probabilities, which can be done by `stackedCIF`:

```
> par(mfrow = c(1, 2))
> matshade(m3$time, cbind(m3$pstate,
+                         m3$lower,
+                         m3$upper)[, c(1, 4, 7, 2, 5, 8, 3, 6, 9)],
+          plot = TRUE, lty = 1, lwd = 2,
+          col = clr <- c("ForestGreen","red","black"),
+          xlim=c(0,15), xaxs="i",
+          ylim = c(0,1), yaxs = "i")
> mat2pol(m3$pstate, perm = 3:1, x = m3$time, col = clr[3:1])
> text(rep(12, 3), c(0.8, 0.5, 0.2), levels(Cdm), col = "white")
```

6. What do you get if you replace "`~ 1`" by "`~ sex`" in the call to `survfit`?

2.3 What not to do

A very common error is to use a *partial* outcome such as `Ins`, when there is a competing type of event, in this case `Dead`. If that is ignored and a traditional survival analysis is made *as if* `Ins` were the only possible event, we will have a substantial *over*estimate of the cumulative probability of going on drug. Here is an illustration of this erroneous approach:

```
> m2 <- survfit(Surv(tfd,
+                    tfd + lex.dur,
+                    lex.Xst == "Ins" ) ~ 1,
+               data = Adm)
> M2 <- survfit(Surv(tfd,
+                    tfd + lex.dur,
+                    lex.Xst == "Dead") ~ 1,
+               data = Adm)
> par(mfrow = c(1,2))
> mat2pol(m3$pstate, c(2,3,1), x = m3$time,
+         col = c("red", "black", "transparent"),
+         xlim=c(0,15), xaxs="i",
+         yaxs = "i", xlab = "time since DM", ylab = "" )
```

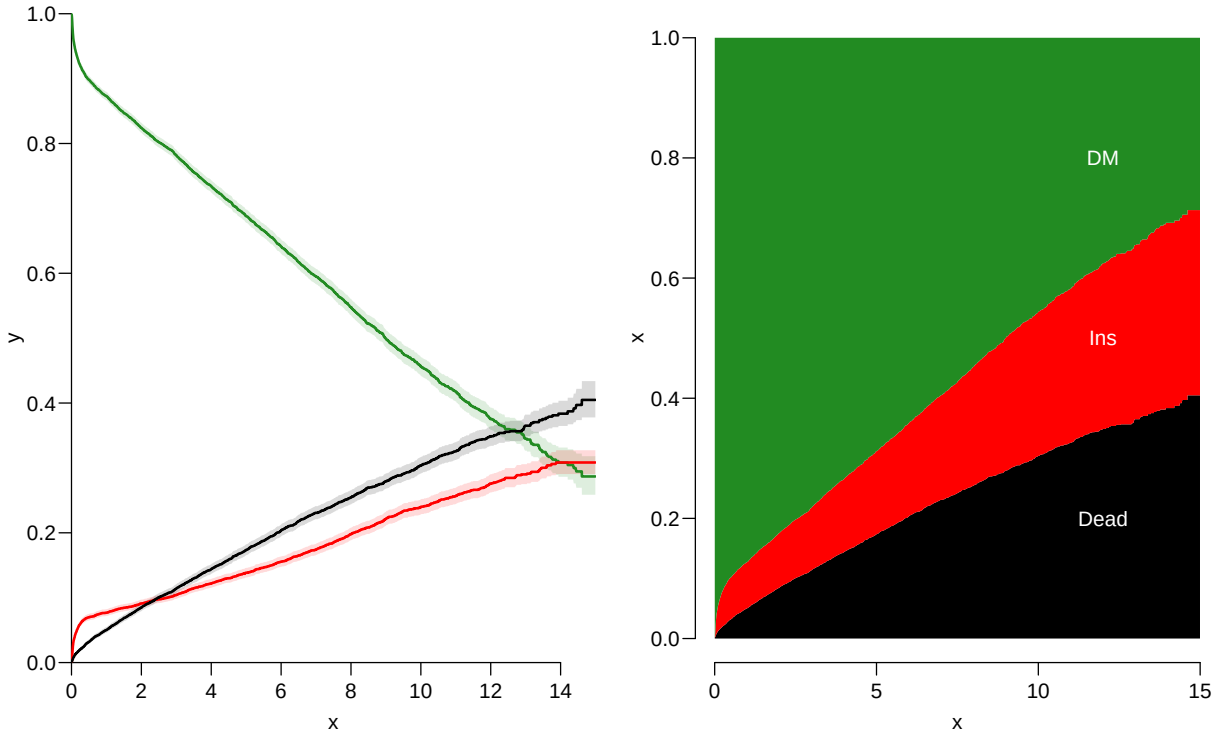


Figure 2.3: *Separate state probabilities (left) and stacked state probabilities (right). In the left panel,* `Alive` *is green,* `Ins` *is red and* `Dead` *is black.*

`../graph/cmpr-surv2`

```
>   lines(m2$time, 1 - m2$surv, lwd = 3, col = "red" )
> mat2pol(m3$pstate, c(3,2,1), x = m3$time, yaxs = "i",
+         col = c("black","red","transparent"),
+         xlim=c(0,15), xaxs="i",
+         yaxs = "i", xlab = "time since DM", ylab = "" )
>   lines(M2$time, 1 - M2$surv, lwd = 3, col = "black" )
```

The first two statements calculate the survival as if only `Ins`, respectively `Dead` were the only way of exiting the state `Alive`. The `mat2pol` (`mat`rix to `pol`ygon) takes the columns of state probabilities from the `survfit` object `m3` that contains the correctly modeled probabilities and plot them as coloured areas stacked; the second argument to `mat2pol` is the order in which they should be stacked. The `lines` plot the wrongly computed cumulative risks (from `m2` and `M2`) — in order to find these we fish out the `surv` component from the `survfit` objects.

A question frequently asked in competing risk situations is what the probability of being on `Ins` is, and it is often (wrongly) believed to be answered by the red curve in the left panel of figure 2.4. But a highly unrealistic assumption is often forgotten: How the rate of pharmaceutical treatment would look in the absence of death (or vice versa for that matter) cannot be deduced from data where both types of risk are present. It is essentially a theological question as no data is available concerning the situation.

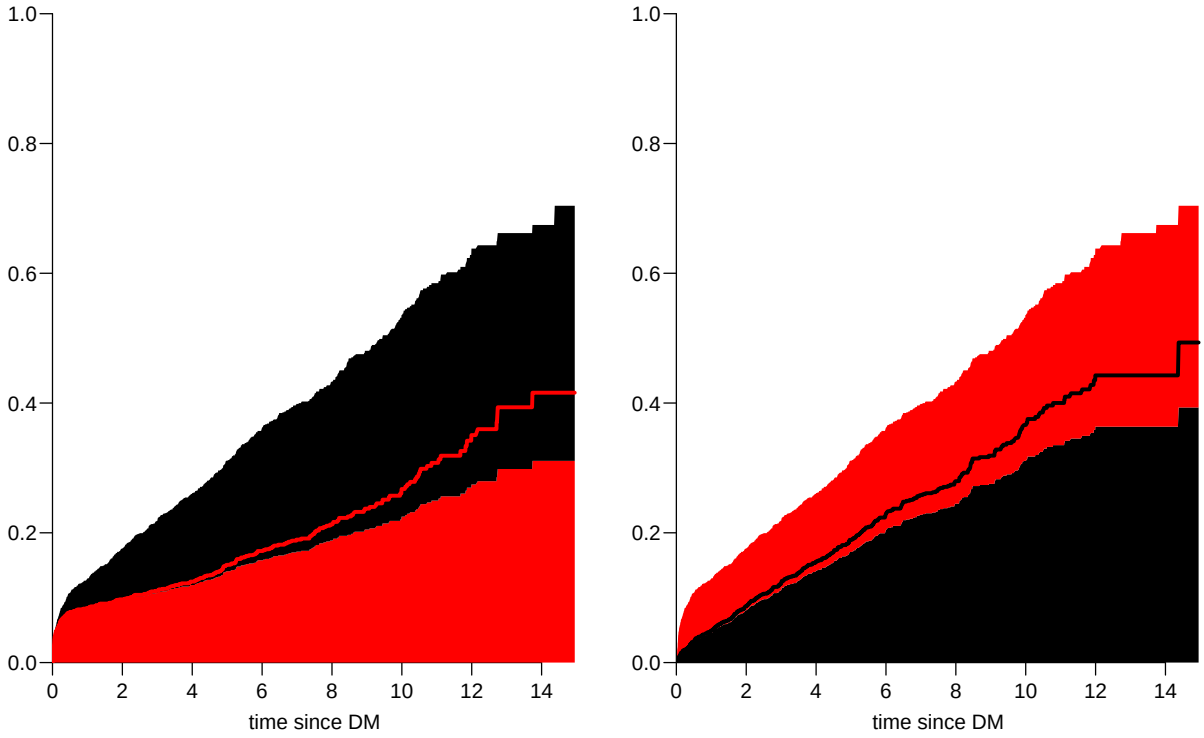


Figure 2.4: *Stacked state probabilities* `Alive` *is white,* `Ins` *is red and* `Dead` *is black. The red line in the left panel is the wrong (but often computed) "cumulative risk" of* `Ins`*, and the black line in the right panel is the wrong (but often computed) "cumulative risk" of* `Death`*. The black and the red areas in the two plots represent the correctly computed probabilities; they have the same size in both panels, only they are stacked differently. . . /graph/cmpr-surv3*

2.4 Modeling cause specific rates

There is nothing wrong with modeling the cause-specific event-rates, the problem lies in how you transform them into probabilities. The relevant model for a competing risks situation normally consists of separate models for each of the cause-specific rates. Not for technical or statistical reasons, but for *substantial* reasons; it is unlikely that rates of different types of event (Ins initiation and death, say) depend on time in the same way.

7. Now we model the two sets of rates by parametric models; this must be based on a time-split data set:

```
> Sdm <- splitMulti(Adm, tfd = seq(0, 20, 0.1))
> summary(Adm)

Transitions:
     To
From   DM  Ins Dead  Records:  Events: Risk time:  Persons:
  DM 6157 1694 2048      9899     3742   45885.49      9899

> summary(Sdm)
```

```
Transitions:
     To
From     DM  Ins Dead  Records:  Events: Risk time:  Persons:
  DM 460054 1694 2048    463796     3742   45885.49      9899
```

8. We will use natural splines for the effect of diabetes duration in a model using `glm`. The `Ns` requires a set of pre-specified knots for the time variable, where the specification should be (partially) guided by the location on the times of the events:

```
> round(cbind(
+ with(subset(Sdm, lex.Xst == "Ins" ), quantile(tfd + lex.dur, 0:10/10)),
+ with(subset(Sdm, lex.Xst == "Dead"), quantile(tfd + lex.dur, 0:10/10))),
+ 3)

       [,1]   [,2]
0%    0.003  0.003
10%   0.030  0.272
20%   0.066  0.769
30%   0.172  1.418
40%   0.454  2.133
50%   1.823  3.076
60%   3.288  4.047
70%   4.834  5.153
80%   6.638  6.523
90%   8.663  8.598
100% 13.878 14.609
```

We see that the `Ins` occur earlier than `Dead`, so we choose the knots a bit earlier:

```
> okn <- c(0,0.5,5,6)
> dkn <- c(0,2.0,5,9)
>  Ins.glm <- glm.Lexis(Sdm, ~ Ns(tfd, knots = okn), from = "DM", to = "Ins" )

stats::glm Poisson analysis of Lexis object Sdm with log link:
Rates for the transition:
DM->Ins

> Dead.glm <- glm.Lexis(Sdm, ~ Ns(tfd, knots = dkn), from = "DM", to = "Dead")

stats::glm Poisson analysis of Lexis object Sdm with log link:
Rates for the transition:
DM->Dead
```

9. With models for the two rates out of the `DM` state we can derive the estimated rates from the two models for rates by time by using a prediction frame, `nd`:

```
> int <- 0.01
> nd <- data.frame(tfd = seq(0, 15, int))
> l.glm <- ci.pred( Ins.glm, nd)
> m.glm <- ci.pred(Dead.glm, nd)
```

Now plot the estimated rates, in this case the `gam` models with dotted and `glm` models with full lines; mortality with black and Ins rates with red:

```
> matshade(nd$tfd,
+          cbind(l.glm, m.glm) * 100,
+          plot = TRUE,
+          log = "y", ylim = c(0.5, 50),
+          col = rep(c("red","black"), 2), lwd = 3,
+          xlab = "Time since DM (years)",
+          ylab = "Rates per 100 PY")
```

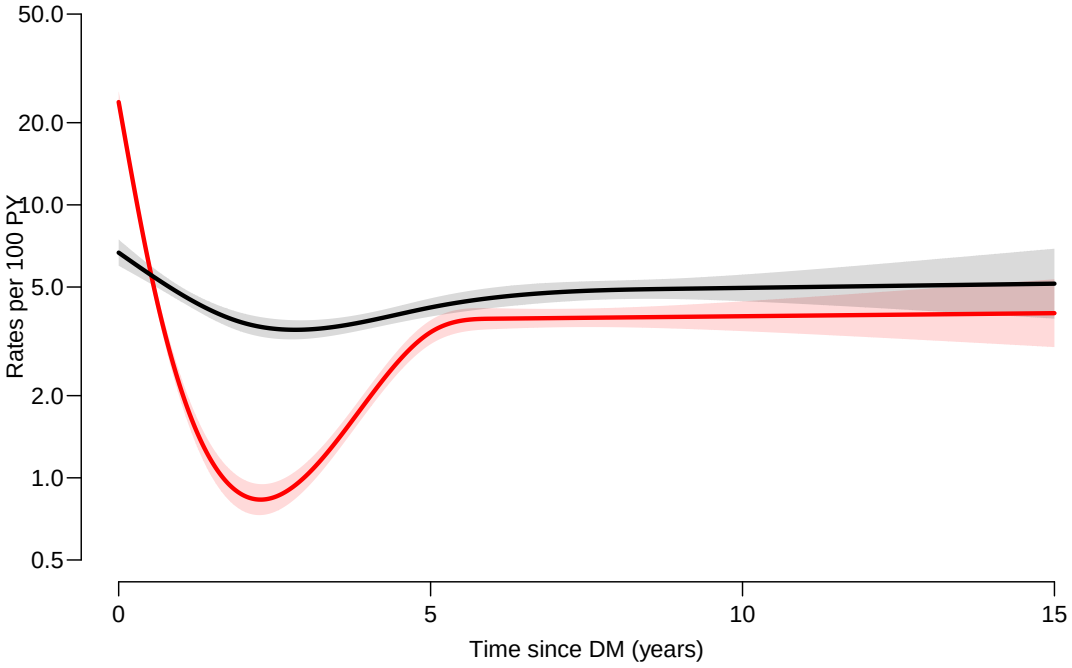


Figure 2.5: *Mortality rates (black) and rates of insukin start (red), from a* `glm` *model with natural splines.*

`../graph/cmpr-Ins-mort`

2.5 Integrals with R

Based on these parametric models we can estimate the cumulative risks of being in each of the states, but also the expected time time spent in each state. The theory of these involves calculation of integrals of the rate functions. Integrals looks scary to many people, but they are really just areas under curves. So here is a digression showing how to calculate integrals as areas under a curve.

The key is to understand how a curve is represented in R. A curve representing the function μ is just a set of two vectors, one vector of ts and one vector $y = \mu(t)$s. When we have a model such as the `gam` or `glm` above that estimates the mortality as a function of

time (`tfd`), we can get a representation of the mortality as a funtion of time by first choosing the timepoints, say from 0 to 15 years in steps of 0.01 year ($\approx$ 4 days). Then put this in a dataframe (`nd`, `newdata`) with the variable name from the model to get the function values at the chosen time points:

```
> t <- seq(0, 15, 0.01)
> nd <- data.frame(tfd = t)
> mu <- ci.pred(Dead.glm, nd)[,1]
> head(cbind(t, mu))
     t         mu
1 0.00 0.06681677
2 0.01 0.06657067
3 0.02 0.06632549
4 0.03 0.06608123
5 0.04 0.06583789
6 0.05 0.06559547
> plot(t, mu, type="l", lwd = 3,
+      xlim = c(0, 7), xaxs = "i",
+      ylim = c(0, max(mu)), yaxs = "i")
> polygon(t[c(1:501,501:1)], c(mu[1:501], rep(0, 501)),
+         col = "gray", border = "transparent")
```

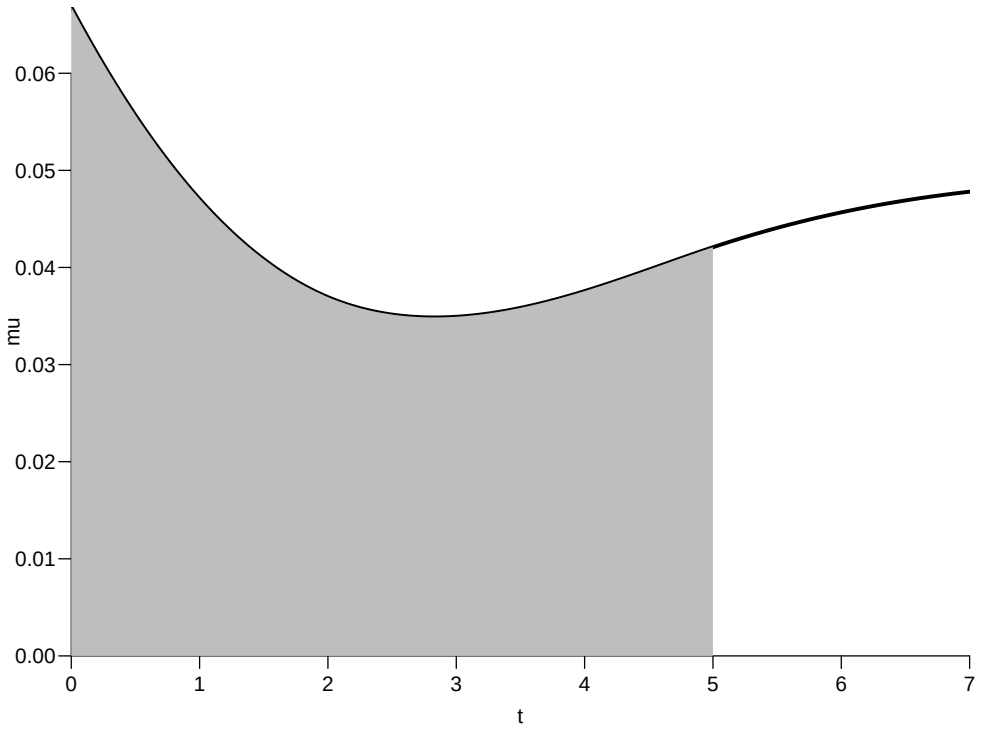


Figure 2.6: *Mortality function and integral from 0 to 5 years.*

`../graph/cmpr-int-ill`

This is a representaion of the points $(t, \mu(t))$; if we want the integral of mu over the interval $[0, 5]$, say, $M(5) = \int_0^5 \mu(s)\,\mathrm{d}s$, we are just asking for the area under the curve. Each t

represents an endpoint of an interval, but what we want in order to compute the area under the curve is the *width* of each interval, `diff(t)`, multiplied by the average of the function values at the ends of each interval (this goes under the name of the "trapezoidal formula"). So we need a small function to compute midpoints between successive values in a vector:

```
> mid <- function(x) x[-1] - diff(x) / 2
> (x <- c(1:5, 7, 10, 17))
[1]  1  2  3  4  5  7 10 17
> mid(x)
[1]  1.5  2.5  3.5  4.5  6.0  8.5 13.5
```

Note that `mid(x)` is a vector that is 1 shorter than the vector `x`, just as `diff(x)` is.

So if we want the integral over the period 0 to 5 years, we want the sum over the first 500 intervals, corresponding to the first 501 interval endpoints:

```
> sum(diff(t[1:501]) * mid(mu[1:501]))
[1] 0.2085188
```

So now we have computed $\int_0^5 \mu(s)\,\mathrm{d}(s)$. This is called the cumulative *rate* over the interval $[0, 5]$ years, even if it is not a rate—it is dimensionless (why?).

It is important to get the units right. In the modeling we entered the risk time ("person-years") in units of 1 year, so the unit of predicted mortality function, `mu`, is events per 1 person-year. Therefore, the units of `t` must be year too; otherwise we will introduce a scaling.

In pratice we will want the integral as *function* of μ, so for every t we want $M(t) = \int_0^t \mu(s)\,\mathrm{d}(s)$. This is easily accomplished by the function `cumsum`:

```
> Mu <- c(0, cumsum(diff(t) * mid(mu)))
> head(cbind(t, Mu))
     t           Mu
  0.00 0.0000000000
2 0.01 0.0006669372
3 0.02 0.0013314180
4 0.03 0.0019934516
5 0.04 0.0026530472
6 0.05 0.0033102141
```

Note that the first value is the integral from 0 to 0, so by definition 0.

2.6 Cumulative risks from parametric models

Here is the theory where we need integration: The cumulative risk of `Ins` at time t is:

$$R_{\texttt{Ins}}(t) = \int_0^t \lambda(u) S(u)\,\mathrm{d}u = \int_0^t \lambda(u) \exp\Big(-\int_0^u \lambda(s) + \mu(s)\,\mathrm{d}s\Big)\,\mathrm{d}u$$

where λ is the rate of `Ins` (`lam`), and μ the mortality rate (`mrt`). A similar formula is obtained for the cumulative risk of `Dead` (that is "dead without insulin use"), by exchanging λ and μ.

The practical calculation of these quantities are on pages 214–5 of "Epidemiology with R".

10. This means that if we have estimates of λ and μ as functions of time, we can derive the cumulative risks. In practice this will be by numerical integration; compute the rates at closely spaced intervals and evaluate the integrals as sums. This is easy, but what is not so easy is to come up with confidence intervals for the cumulative risks.

    Confidence intervals are most conveniently produced by simulation ("parametric bootstrap" as some say):

    (a) generate a random vector from the multivariate normal distribution with mean equal to the parameters of the model, and variance-covariance equal to the estimated variance-covariance of the parameter estimates (the Hessian as it is called).

    (b) use this to generate a simulated set of rates ($\lambda(t)$, $\mu(t)$), evaluated a closely spaced times.

    (c) use these in numerical integration to derive state probabilities at these times.

    (d) repeat 1000 times, say, to obtain 1000 sets of state probabilities at these times.

    (e) use these to derive confidence intervals for the state. probabilities as the 2.5 and 97.5 percentiles of the simulated state probabilities at each time point.

    This machinery is implemented in the function `ci.Crisk`

```
> cR <- ci.Crisk(mods = list(Ins =  Ins.glm,
+                            Dead = Dead.glm),
+                   nd = nd)

NOTE: Times are assumed to be in the column tfd at equal distances of 0.01

> str(cR)

List of 4
 $ Crisk: num [1:1501, 1:3, 1:3] 1 0.997 0.994 0.991 0.988 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ tfd  : chr [1:1501] "0" "0.01" "0.02" "0.03" ...
  .. ..$ cause: chr [1:3] "Surv" "Ins" "Dead"
  .. ..$      : chr [1:3] "50%" "2.5%" "97.5%"
 $ Srisk: num [1:1501, 1:2, 1:3] 0 0.000665 0.001326 0.001982 0.002633 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ tfd  : chr [1:1501] "0" "0.01" "0.02" "0.03" ...
  .. ..$ cause: chr [1:2] "Dead" "Dead+Ins"
  .. ..$      : chr [1:3] "50%" "2.5%" "97.5%"
 $ Stime: num [1:1501, 1:3, 1:3] 0 0.00998 0.01994 0.02987 0.03976 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ tfd  : chr [1:1501] "0" "0.01" "0.02" "0.03" ...
  .. ..$ cause: chr [1:3] "Surv" "Ins" "Dead"
  .. ..$      : chr [1:3] "50%" "2.5%" "97.5%"
 $ time : num [1:1501] 0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 ...
 - attr(*, "int")= num 0.01
```

    There are 4 components of the results, the three first are simply arrays with 2 or 3 functions of time with confidence intervals.

So now plot the cumulative *risks* of being in each of the states (the `Crisk` component):

```
> matshade(as.numeric(dimnames(cR$Crisk)[[1]]),
+          cbind(cR$Crisk[,1,],
+                cR$Crisk[,2,],
+                cR$Crisk[,3,]), plot = TRUE,
+          lwd = 2, col = c("limegreen","red","black"))
```

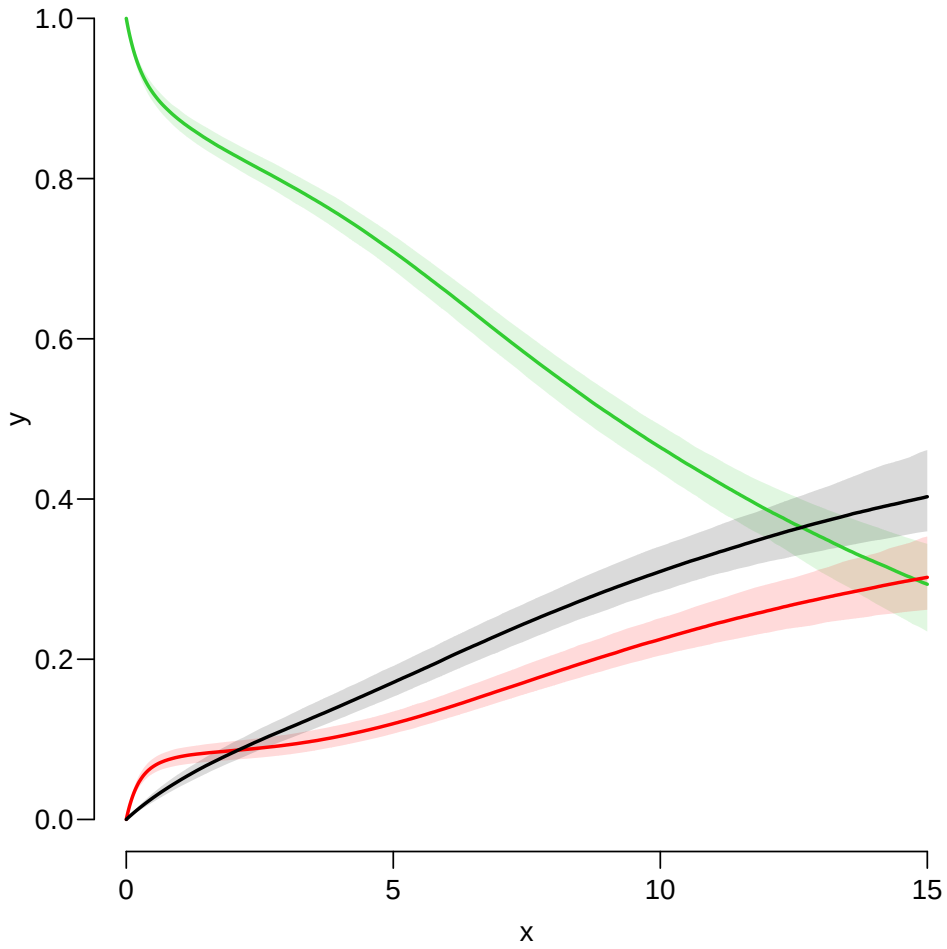


Figure 2.7: *Cumulative risks of being in each of the states* `DM` *(green),* `Ins` *(red) and* `Dead` *(black).*

`../graph/cmpr-crisk`

11. Plot the stacked probabilities (use for example `matrix` `2` `polygons`):

```
> str(cR$Crisk)
```

```
 num [1:1501, 1:3, 1:3] 1 0.997 0.994 0.991 0.988 ...
 - attr(*, "dimnames")=List of 3
```

```
  ..$ tfd  : chr [1:1501] "0" "0.01" "0.02" "0.03" ...
  ..$ cause: chr [1:3] "Surv" "Ins" "Dead"
  ..$      : chr [1:3] "50%" "2.5%" "97.5%"
```

```
> mat2pol(cR$Crisk[,3:1,1], col = c("forestgreen","red","black")[3:1])
```

The component `Srisk` has the confidence limits of the stacked probabilities, add these to the plot, for example by semi-transparent shades or dotted lines. If you are really entrepreneurial, devise a function that will take the `Srisk` component of `cR` and produce a stacked plot with shaded confidence limits; here is the stacked plot:

```
> matshade(as.numeric(dimnames(cR$Srisk)[[1]]),
+          cbind(cR$Srisk[,1,],
+                cR$Srisk[,2,]), plot = TRUE,
+          lwd = 2, col = c("black","red"),
+          ylim = 0:1, yaxs = "i")
```

You may want to look at `adjustcolor` or `rgb` to see how to make semi-transparent colours.

2.7 Expected life time: using simulated objects

12. It is not only the cumulative risks of being in different states that my be of interest, the *integrals* — area under the cumulative risk curves are of interest too. The cumulative risks are probabilities, so dimensionless, which means that integrals of these along the time-axis will have dimension time; they will represent the expected time spent in each of the states.

    The areas between the lines (up to say 10 years) are *expected sojourn times*, that is:

    - expected years alive without insulin
    - expected years lost to death without insulin
    - expected years after insulin, including years dead after insulin

    Not all of these are of direct relevance; actually only the first may be so. They are available (with simulation-based confidence intervals) in the component of `cR`, `Stime` (Sojourn `time`).

    A relevant quantity would be the expected time alive without insulin during the first 5, 10 and 15 years (remember that the first dimension of `Stime` is in units of 1/100 year):

```
> str(cR$Stime)
```

```
 num [1:1501, 1:3, 1:3] 0 0.00998 0.01994 0.02987 0.03976 ...
 - attr(*, "dimnames")=List of 3
  ..$ tfd  : chr [1:1501] "0" "0.01" "0.02" "0.03" ...
  ..$ cause: chr [1:3] "Surv" "Ins" "Dead"
  ..$      : chr [1:3] "50%" "2.5%" "97.5%"
```

```
> round(cR$Stime[1:3 * 500 + 1,"Surv",], 1)

tfd  50% 2.5% 97.5%
  5  4.1  4.0   4.1
  10 6.9  6.9   7.0
  15 8.8  8.7   8.9
```

13. We can also compute the expected fraction of the first 5, 10, 15 years spent alive without insulin therapy:

```
> (mY <- matrix(1:3 * 5, 3, 3)) # using the recycling rule

     [,1] [,2] [,3]
[1,]    5    5    5
[2,]   10   10   10
[3,]   15   15   15

> round(cR$Stime[1:3*500+1,"Surv",] / mY * 100, 1)

tfd   50% 2.5% 97.5%
  5  81.5 80.8  82.2
  10 69.4 68.7  70.3
  15 58.6 57.7  59.6
```

This can also be shown as a function of time; how large a fraction of the first t time can a person expect to be alive, for t ranging from 0 to 15 years:

```
> time <- as.numeric(dimnames(cR$Stime)[[1]])
> matshade(time, cR$Stime[,"Surv",] /
+                cbind(time,
+                      time,
+                      time) * 100,
+          plot=TRUE,
+          ylim = 0:1*100, yaxs = "i", xaxs = "i")
```

Amend the plot with proper axis labels.

Chapter 3

Multistate models: steno2

Paraphernalia

First we load the relevant packages and set some options:

```
> library(survival)
> library(Epi)
> library(popEpi)
> # popEpi::splitMulti returns a data.frame rather than a data.table
> options("popEpi.datatable" = FALSE)
> library(tidyverse)
> setwd("c:/bendix/teach/AdvCoh/courses/Melb.2023/pracs")
> getwd()
[1] "c:/bendix/teach/AdvCoh/courses/Melb.2023/pracs"
> clear()
```

3.1 Lexis object for steno2

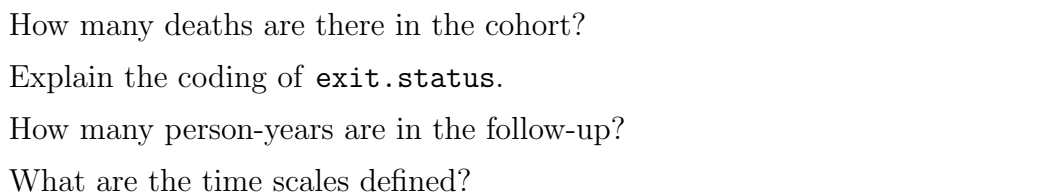
1. Bring in the steno2 dataset, and convert dates to cal.yr to get a natural unit of time (years—365.25 days, that is). Because of the way data were anonymized, the doEnd is not perfectly aligned to doDth, which we remedy on the fly by resetting doEnd if a doDth is known.

```
> data(steno2)
> steno2 <- cal.yr(steno2)
> steno2 <- transform(steno2,
+                     doEnd = pmin(doEnd, doDth, na.rm = TRUE))
> str(steno2)
'data.frame':        160 obs. of  14 variables:
 $ id      : num  1 2 3 4 5 6 7 8 9 10 ...
 $ allo    : Factor w/ 2 levels "Int","Conv": 1 1 2 2 2 2 2 1 1 1 ...
 $ sex     : Factor w/ 2 levels "F","M": 2 2 2 2 2 2 1 2 2 2 ...
 $ baseCVD : num  0 0 0 0 0 1 0 0 0 0 ...
 $ deathCVD: num  0 0 0 0 1 0 0 0 1 0 ...
 $ doBth   : 'cal.yr' num  1932 1947 1943 1945 1936 ...
```

```
 $ doDM    : 'cal.yr' num  1991 1982 1983 1977 1986 ...
 $ doBase  : 'cal.yr' num  1993 1993 1993 1993 1993 ...
 $ doCVD1  : 'cal.yr' num  2014 2009 2002 1995 1994 ...
 $ doCVD2  : 'cal.yr' num  NA 2009 NA 1997 1995 ...
 $ doCVD3  : 'cal.yr' num  NA 2010 NA 2003 1998 ...
 $ doESRD  : 'cal.yr' num  NaN NaN NaN NaN 1998 ...
 $ doEnd   : 'cal.yr' num  2015 2015 2002 2003 1998 ...
 $ doDth   : 'cal.yr' num  NA NA 2002 2003 1998 ...
```

2. Start by setting up a `Lexis` data frame for the entire observation time for each person; from entry (`doBase`, date of baseline) to exit, `doEnd`. Note that we call the initial state `Mic`(roalbuminuria), because all patients in the Steno2 study had this status at entry—it was one of the inclusion criteria:

```
> L2 <- Lexis(entry = list(per = doBase,
+                          age = doBase - doBth,
+                          tfi = 0),
+             exit = list(per = doEnd),
+       exit.status = factor(deathCVD + !is.na(doDth),
+                            labels=c("Mic","D(oth)","D(CVD)")),
+               id = id,
+             data = steno2)

NOTE: entry.status has been set to "Mic" for all.

> summary(L2, t = TRUE)

Transitions:
     To
From  Mic D(oth) D(CVD)  Records:  Events: Risk time:  Persons:
  Mic  67     55     38       160       93    2416.59       160

Timescales:
per age tfi
 ""  ""  ""

> boxes(L2, boxpos = TRUE, show.BE = TRUE, scale.R = 100)
```



How many deaths are there in the cohort?

Explain the coding of `exit.status`.

How many person-years are in the follow-up?

What are the time scales defined?

3. In this set-up we can study the CVD and the non-CVD mortality rates, a classical competing risks problem, but we want in particular to see how the mortality rates depend on albuminuria status.

   In order to allocate follow-up (person-time and events) to *current* albuminuria status we need to know when the persons change status; this is recorded in the data frame `st2alb`.

We will cut the follow-up at each date of albuminuria measurement allowing the patients to change between states Normoalbuminuria, Microalbuminuria and Macroalbuminuria at each of these dates, possibly several times per person. To this end we use the function rcutLexis (recurrent cuts), which requires a data frame of transitions with columns lex.id, cut and new.state — see ?rcutLexis.

We change the scale of the date of transition to year by cal.yr (to align with the per variable in L2), and in order to comply with the requirements of rcutLexis rename the id variable id to lex.id, the date variable doTr to cut and the state variable state to new.state:

```
> data(st2alb)
> cut2 <- cal.yr(st2alb)
> names(cut2)
[1] "id"    "doTr"  "state"
> names(cut2) <- c("lex.id", "cut", "new.state")
> str(cut2)
'data.frame':        563 obs. of  3 variables:
 $ lex.id   : num  1 1 1 1 1 2 2 2 2 2 ...
 $ cut      : 'cal.yr' num  1993 1995 2000 2002 2007 ...
 $ new.state: Factor w/ 3 levels "Norm","Mic","Mac": 2 1 2 1 2 1 2 3 2 2 ...
> head(cut2)
  lex.id      cut new.state
1      1 1993.444       Mic
2      1 1995.361      Norm
3      1 2000.067       Mic
4      1 2001.984      Norm
5      1 2007.317       Mic
6      2 1993.786      Norm
```

How many persons are in the cut2 data frame? We can do this in two different ways, illustrating the tidyverse philosophy

```
> addmargins(table(table(cut2$lex.id)))
  1   2   3   4   5 Sum
  4  25  40  46  41 156
> cut2$lex.id %>% table %>% table %>% addmargins
.
  1   2   3   4   5 Sum
  4  25  40  46  41 156
```

Explain the entries in this table.

4. Now cut the follow-up at intermediate transition times (note that rcutLexis assumes that values in the cut column refer to the first timescale by default, and the first of the timescales in L2 is per):

```
> L3 <- rcutLexis(L2, cut2)
> summary(L3)

Transitions:
     To
From   Mic Norm Mac D(oth) D(CVD)  Records:  Events: Risk time:  Persons:
  Mic  299   72  65     27     13       476      177    1381.57       160
  Norm  31   90   5     14      7       147       57     607.86        69
  Mac   20    3  44     14     18        99       55     427.16        64
  Sum  350  165 114     55     38       722      289    2416.59       160

> boxes(L3, boxpos = TRUE, cex = 0.8)
```

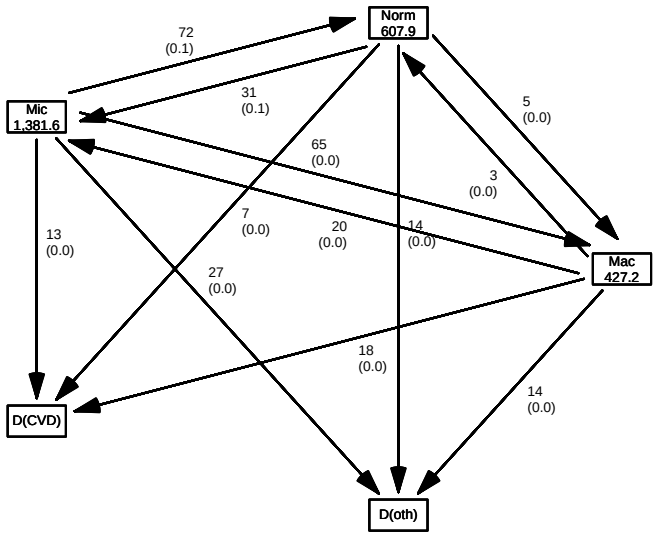


Figure 3.1: *The default lay-out of the 5 boxes placed on a circle, including the jumps directly between* `Norm` *and* `Mac`.

`../graph/ms-boxL3`

5. Note that there are transitions both ways between all three of `Norm`, `Mic` and `Mac`, which is a bit illogical since we have a natural ordering of states: `Norm` < `Mic` < `Mac`. Hence, transitions from `Norm` to `Mac` (and vice versa) should go through `Mic`.

   In order to remedy this anomaly we find all transitions `Norm` → `Mac` and provide a transition `Norm` → `Mic` in between, so that each transition `Norm` → `Mac` is replaced by two: `Norm` → `Mic` and `Mic` → `Mac`.

   And of course similarly for transitions `Mac` → `Norm`.

The relevant "jump" transitions are easily found:

```
> (jump <-
+ subset(L3, (lex.Cst == "Norm" & lex.Xst == "Mac") |
+             (lex.Xst == "Norm" & lex.Cst == "Mac"))[,
+        c("lex.id", "per", "lex.dur","lex.Cst", "lex.Xst")])
```

```
 lex.id     per lex.dur lex.Cst lex.Xst
     70 1999.49    2.67     Mac    Norm
     86 2001.76   12.82    Norm     Mac
    130 2000.91    1.88     Mac    Norm
    131 1997.76    4.24    Norm     Mac
    136 1997.21    0.47     Mac    Norm
    136 1997.69    4.24    Norm     Mac
    171 1996.39    5.34    Norm     Mac
    175 2004.58    9.88    Norm     Mac
```

6. What we need to do for each of these "jumps" is to provide an extra transition to `Mic` at a time during the stay in either `lex.Cst` (either `Norm` or `Mac`), i.e. somewhere between `per` and `per` + `lex.dur` in these records. Quite arbitrarily we choose a random time in the middle 80% between the dates:

```
> set.seed(1952)
> xcut <- select(transform(jump,
+                          cut = per + lex.dur * runif(per, 0.1, 0.9),
+                     new.state = "Mic"),
+                c(lex.id, cut, new.state))
> xcut
```

```
 lex.id      cut new.state
     70 2001.789       Mic
     86 2012.232       Mic
    130 2001.488       Mic
    131 2001.032       Mic
    136 1997.610       Mic
    136 2000.780       Mic
    171 1997.057       Mic
    175 2013.472       Mic
```

How many extra records will be generated when cutting the follow-up?

7. Now make extra cuts (transitions to `Mic`) at these dates using `rcutLexis` with `xcut` on the `L3` object, and order the levels sensibly:

```
> L4 <- rcutLexis(L3, xcut)
> L4 <- Relevel(L4, c("Norm","Mic","Mac","D(CVD)","D(oth)"))
> summary(L4)
```

```
Transitions:
     To
From   Norm Mic Mac D(CVD) D(oth)  Records:  Events: Risk time:  Persons:
  Norm   90  35   0      6     13       144       54     581.04        66
  Mic    72 312  65     14     30       493      181    1435.14       160
  Mac     0  22  41     18     12        93       52     400.41        60
  Sum   162 369 106     38     55       730      287    2416.59       160
```

We see that there are now no transitions directly between `Norm` and `Mac` in `L4`, so we can make a more intelligible plot of the transitions (remember to read the help page for `boxes.Lexis`):

```
> clr <- c("forestgreen","orange","red","blue",gray(0.3))
> boxes(L4, boxpos = list(x = c(20,20,20,80,80),
+                         y = c(10,50,90,75,25)),
+          show.BE = "nz",
+          scale.R = 100,
+         digits.R = 2,
+              cex = 0.9,
+          pos.arr = 0.3,
+           col.bg = clr,
+       col.border = clr,
+          col.txt = c("white","black")[c(1,2,1,1,1)])
```

Explain the arguments of `boxes`.

Explain the numbers in the graph.

Describe the overall effect of albuminuria on the two mortality rates.

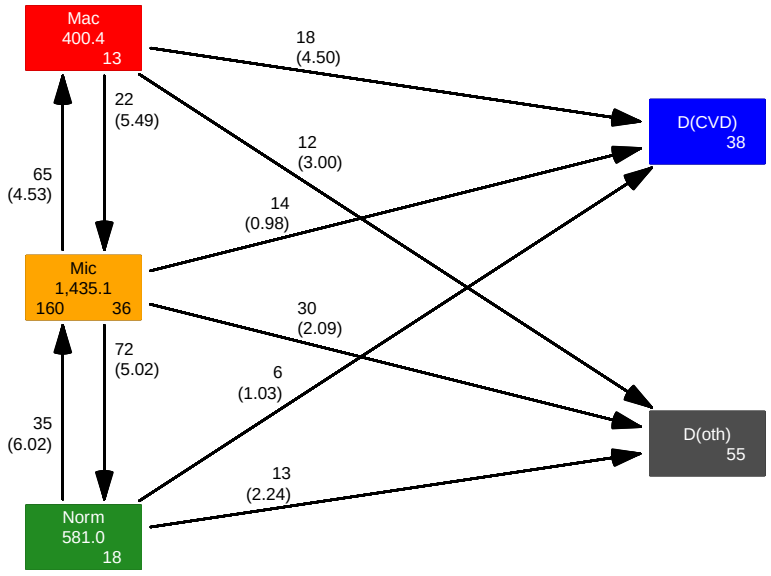


Figure 3.2: *Transitions between states in the Steno2 study.*

`../graph/ms-b4`

What we now have in the `Lexis` object `L4` is the follow-up of the 160 persons classified in the 5 states shown. So with this multistate model (well, there is no model yet) set up we can look at mortality rates and see how they depend on the current albuminuria state, or look at the transition rates between the different albuminuria states and assess how these depend on covariates.

3.2 Transition rates: multiple time scales

8. We will model the transition rates with parametric functions, so we need to split the dataset along some time scale; we will use 1 month intervals (they should be sufficiently small to accommodate an assumption of constant transition rates in each interval):

```
> S4 <- splitMulti(L4, tfi = seq(0, 25, 1/12))
> summary(L4)

Transitions:
     To
From   Norm Mic Mac D(CVD) D(oth)  Records:  Events: Risk time:  Persons:
  Norm   90  35   0      6     13       144       54     581.04        66
  Mic    72 312  65     14     30       493      181    1435.14       160
  Mac     0  22  41     18     12        93       52     400.41        60
  Sum   162 369 106     38     55       730      287    2416.59       160

> summary(S4)

Transitions:
     To
From   Norm   Mic  Mac D(CVD) D(oth)  Records:  Events: Risk time:  Persons:
  Norm 7061    35    0      6     13      7115       54     581.04        66
  Mic    72 17453   65     14     30     17634      181    1435.14       160
  Mac     0    22 4844     18     12      4896       52     400.41        60
  Sum  7133 17510 4909     38     55     29645      287    2416.59       160
```

We see that the number of events (transitions) and person-years are the same, in the two `Lexis` objects, but the number of records in `S4` is substantially larger than in `L4`.

9. We can now model the overall mortality rates as functions of age and duration (time since entry) using the defaults for `glm.Lexis` (this function call will trigger a warning):

```
> ma <- glm.Lexis(S4, ~ Ns(tfi, knots = seq( 0, 20, 5)) +
+                       Ns(age, knots = seq(50, 80, 10)) +
+                       lex.Cst)

NOTE:
Multiple transitions *from* state ' Mac', 'Mic', 'Norm ' - are you sure?
The analysis requested is effectively merging outcome states.
You may want analyses using a *stacked* dataset - see ?stack.Lexis
stats::glm Poisson analysis of Lexis object S4 with log link:
Rates for transitions:
Norm->D(CVD)
Mic->D(CVD)
Mac->D(CVD)
Norm->D(oth)
Mic->D(oth)
Mac->D(oth)
```

The warning triggered here just tells you that you are modeling the occurrence of any type of death, which amounts to modeling of the sum of CVD and non-CVD death rates—the overall mortality.

We can illustrate the actual model underlying this by collapsing the two causes for death using `Relevel`:

```
> clr <- c("forestgreen","orange","red",gray(0.3))
> summary(Relevel(L4, list("Dead" = 4:5), first = FALSE))

Transitions:
     To
From   Norm Mic Mac Dead  Records:  Events: Risk time:  Persons:
  Norm   90  35   0   19       144       54     581.04        66
  Mic    72 312  65   44       493      181    1435.14       160
  Mac     0  22  41   30        93       52     400.41        60
  Sum   162 369 106   93       730      287    2416.59       160

>   boxes(Relevel(L4, list("Dead" = 4:5), first = FALSE),
+          boxpos = list(x = c(20,20,20,80),
+                        y = c(10,50,90,50)),
+         show.BE = "nz",
+         scale.R = 100,
+        digits.R = 2,
+             cex = 0.9,
+         pos.arr = 0.3,
+          col.bg = clr,
+       col.border = clr,
+          col.txt = c("white","black")[c(1,2,1,1)])
```

The model structure with `lex.Cst` as an additive term is assuming that the overall mortality rates are proportional between states of albuminuria.

10. The default for `glm.Lexis` is to model all transitions to absorbing states which in this case are the two "dead" states, `D(oth)` and `D(CVD)`.

    The `glm.Lexis` above is just a convenience wrapper for:

```
> m1 <- glm(cbind(lex.Xst %in% c("D(oth)", "D(CVD)")
+                 & lex.Cst != lex.Xst,
+                 lex.dur)
+           ~ Ns(tfi, knots = seq( 0, 20, 5)) +
+             Ns(age, knots = seq(50, 80, 10)) +
+             lex.Cst,
+           family = poisreg,
+             data = subset(S4, lex.Cst %in% c("Norm","Mic","Mac")))
```

    . . . which will also give the same results as:

```
> m2 <- glm((lex.Xst %in% c("D(oth)", "D(CVD)")
+            & lex.Cst != lex.Xst)
+           ~ Ns(tfi, knots = seq( 0, 20, 5)) +
+             Ns(age, knots = seq(50, 80, 10)) +
```

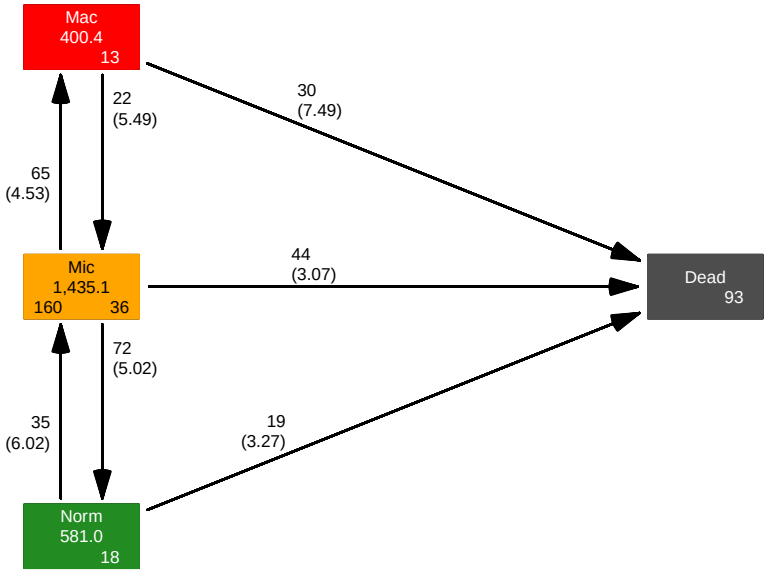


Figure 3.3: *Transitions between states in the Steno2 study, using only all-cause mortality.*

`../graph/ms-b4m`

```
+             lex.Cst,
+           offset = log(lex.dur),
+           family = poisson,
+             data = subset(S4, lex.Cst %in% c("Norm","Mic","Mac")))
```

—note the difference between the families `poisreg` and `poisson`: `poisreg` enters events and person-time more logically as part of the outcome, whereas `poisson` enters events as the response and person-years (`lex.dur`) via the `offset` argument.

11. The parameters from any of the formulations are on the log-scale so we want to see them exponentiated, so on the rate-scale:

```
> round(ci.exp(ma), 2)
```

```
                                  exp(Est.) 2.5%   97.5%
(Intercept)                            0.00 0.00    0.01
Ns(tfi, knots = seq(0, 20, 5))1        6.57 1.23   35.14
Ns(tfi, knots = seq(0, 20, 5))2        4.29 0.97   18.89
Ns(tfi, knots = seq(0, 20, 5))3       45.97 0.93 2265.04
Ns(tfi, knots = seq(0, 20, 5))4        0.52 0.17    1.58
Ns(age, knots = seq(50, 80, 10))1      3.26 1.33    8.00
Ns(age, knots = seq(50, 80, 10))2     11.00 1.34   90.24
Ns(age, knots = seq(50, 80, 10))3     12.41 5.56   27.71
lex.CstMic                             0.96 0.56    1.65
lex.CstMac                             1.71 0.95    3.06
```

What are the mortality rate-ratios (hazard ratios), what ratios do they refer to: rates of what between which groups?

We see there is a higher mortality in the `Mac` state but no discernible difference between the `Mic` and the `Norm` states.

12. It can be formally tested whether the three states carry the same mortality using a Wald test (testing whether the `Norm` and `Mac` parameters both are 0 on the log-scale):

```
> Wald(ma, subset = "lex.Cst")
```

```
     Chisq       d.f.          P
6.15752406 2.00000000 0.04601619
```

What is the meaning of this test (i.e. what is the null hypothesis)?

Do you like the formal 5% significance level? What about 4.5% instead?

13. Now do the same analysis for the two causes of death separately, using the `to` argument to `glm.Lexis`:

```
> # other causes of death
> mo <- glm.Lexis(S4, ~ Ns(tfi, knots = seq( 0, 20, 5)) +
+                       Ns(age, knots = seq(50, 80, 10)) +
+                       lex.Cst,
+                 to = "D(oth)")
```

```
stats::glm Poisson analysis of Lexis object S4 with log link:
Rates for transitions:
Norm->D(oth)
Mic->D(oth)
Mac->D(oth)
```

```
> round(ci.exp(mo), 3)
```

```
                                  exp(Est.)  2.5%        97.5%
(Intercept)                           0.000 0.000 7.000000e-03
Ns(tfi, knots = seq(0, 20, 5))1     161.560 2.938 8.882984e+03
Ns(tfi, knots = seq(0, 20, 5))2      37.189 1.461 9.467380e+02
Ns(tfi, knots = seq(0, 20, 5))3   39009.388 4.355 3.494245e+08
Ns(tfi, knots = seq(0, 20, 5))4       2.086 0.345 1.263200e+01
Ns(age, knots = seq(50, 80, 10))1     2.684 0.881 8.176000e+00
Ns(age, knots = seq(50, 80, 10))2     1.868 0.185 1.887100e+01
Ns(age, knots = seq(50, 80, 10))3    12.728 4.572 3.543300e+01
lex.CstMic                            1.004 0.520 1.937000e+00
lex.CstMac                            1.001 0.449 2.232000e+00
```

```
> #
> # CVD death
> mC <- glm.Lexis(S4, ~ Ns(tfi, knots = seq( 0, 20, 5)) +
+                       Ns(age, knots = seq(50, 80, 10)) +
+                       lex.Cst,
+                 to = "D(CVD)")
```

```
stats::glm Poisson analysis of Lexis object S4 with log link:
Rates for transitions:
Norm->D(CVD)
Mic->D(CVD)
Mac->D(CVD)
```

```
> round(ci.exp(mC), 3)
```

```
                                  exp(Est.)  2.5%      97.5%
(Intercept)                           0.001 0.000      0.014
Ns(tfi, knots = seq(0, 20, 5))1       1.171 0.164      8.344
Ns(tfi, knots = seq(0, 20, 5))2       2.008 0.303     13.333
Ns(tfi, knots = seq(0, 20, 5))3       1.493 0.018    126.727
Ns(tfi, knots = seq(0, 20, 5))4       0.144 0.019      1.072
Ns(age, knots = seq(50, 80, 10))1     6.833 1.078     43.327
Ns(age, knots = seq(50, 80, 10))2   486.871 1.658 142982.410
Ns(age, knots = seq(50, 80, 10))3    15.110 3.524     64.789
lex.CstMic                            0.919 0.351      2.410
lex.CstMac                            3.229 1.271      8.203
```

What is the conclusion w.r.t. the effect of albuminuria state on the two cause-specific mortality rates?

14. Now make formal tests of relevant hypotheses using `Wald`.

```
> Wald(mo, subset = "Cst")
```

```
       Chisq         d.f.            P
0.0001452741 2.0000000000 0.9999273656
```

```
> Wald(mC, subset = "Cst")
```

```
       Chisq         d.f.            P
13.785510275  2.000000000  0.001015113
```

What is the conclusion from these w.r.t. mortality dependence on albuminuria status?

15. We can show how fitted mortality rates look for persons currently in state `Mic` entering the study at a set of specific ages. The entry ages are in the vector `L2$age` (`L2` is the initial `Lexis` object with one record per person):

```
> summary(L2$age)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  37.39   48.52   56.60   55.13   61.06   67.50
```

Based on this we shall use ages (at entry) 45, 55 and 65, and show mortality rates for persons entering at these ages. We will show the rates as functions of their current age. We need a prediction data frame, with values for all variables in the model, (current) `age` and time from entry, `tfi`. Here `expand.grid` is our friend; note we are using `ain` for age at entry:

```
> expand.grid(tfi = c(NA, seq(0, 20, 5)),
+             ain = c(45, 55, 65))
```

```
   tfi ain
1   NA  45
2    0  45
3    5  45
4   10  45
5   15  45
6   20  45
7   NA  55
8    0  55
9    5  55
10  10  55
11  15  55
12  20  55
13  NA  65
14   0  65
15   5  65
16  10  65
17  15  65
18  20  65
```

—it will give all combinations of the values in the vectors supplied as a `data.frame`. The `NA`s are there for plotting purposes— we get a break in plotted curves if there is an `NA` in the data. We want the `tfi` points to be closer than in the illustrative example:

```
> prf <- transform(expand.grid(tfi = c(NA, seq(0, 19, 0.5)),
+                              ain = c(45, 55, 65))[-1,],
+                  age = ain + tfi,
+              lex.Cst = "Mic")
> prf[ 1: 5,]
```

```
  tfi ain  age lex.Cst
2 0.0  45 45.0     Mic
3 0.5  45 45.5     Mic
4 1.0  45 46.0     Mic
5 1.5  45 46.5     Mic
6 2.0  45 47.0     Mic
```

```
> prf[40:44,]
```

```
   tfi ain  age lex.Cst
41  NA  55   NA     Mic
42 0.0  55 55.0     Mic
43 0.5  55 55.5     Mic
44 1.0  55 56.0     Mic
45 1.5  55 56.5     Mic
```

```
> matshade(prf$age, cbind(ci.pred(mo, prf),
+                         ci.pred(mC, prf)) * 100,
+          lwd = 3, col = c("black","blue"),
+          log = "y", ylim = c(0.02,20), plot = TRUE,
+          xlab = "Age at follow-up (years)",
+          ylab = "Mortality rate per 100 PY")
> abline(v = c(45, 55, 65), lty = 3)
```

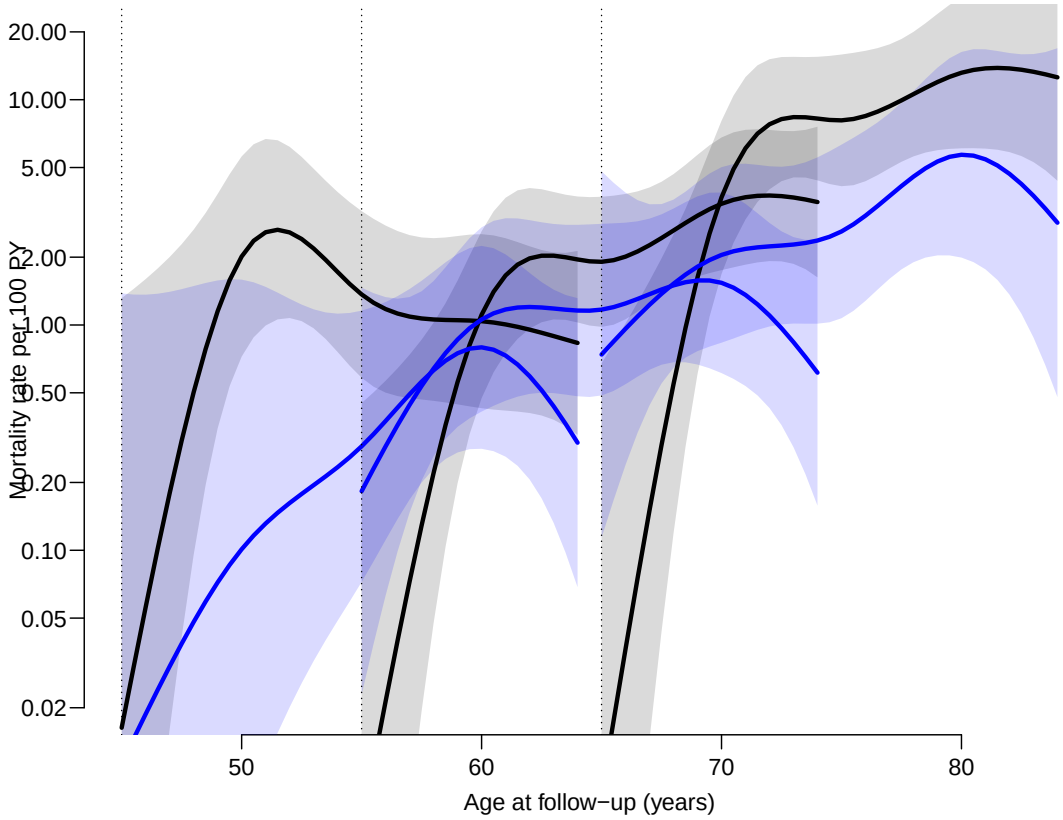


Figure 3.4: *CVD mortality rates (blue) and non-CVD mortality rates (black), with 95% confidence intervals as shades. Curve represent persons entering the study at ages 45, 55 and 65 respectively. T* `../graph/ms-mort1`

he rates of death from other causes is very small at the beginning and increases steeply over the first 5 years of follow-up, while the CVD mortality rates are pretty stable with a foreseeable increase by age.

Give an informal description of the curves, and a possible reason for the shape of the curves.

16. We can show the impact of albuminuria state on the mortality rates in a 3-panel layout:

```
> par(mfrow=c(1,3))
> for(st in c("Norm","Mic","Mac"))
+    {
+ matshade(prf$age, pmin(pmax(
+                    cbind(ci.pred(mo, transform(prf, lex.Cst = st)),
+                          ci.pred(mC, transform(prf, lex.Cst = st))) * 100,
+                              0.05), 60),
+          lwd = 3, col = c("black","blue"),
+          log = "y", ylim = c(0.1,50), plot = TRUE)
+ text(60, 50, st, adj = 0)
+    }
```

How are the curves in the three panels related?

Describe the effect of albuminuria status on the two types of mortality.

How can you see this from the model parameters of the models `mo` and `mC`?

3.3 State probabilities

We would like to see how the probabilities of being in each of the states in figure 3.2 look as a function of time since entry, and we will in particular be interested in how this depends on `allo`, the allocation to intensified or standard treatment.

3.3.1 Models for transition rates

Thus we will need models for 1) the cause-specific mortality rates and 2) transition rates between albuminuria states. And of course models which all include the effect of `allo` (treatment allocation).

We already fitted models for the mortality rates, but here we refit them in a slightly different guise, namely including the treatment allocation (`allo`) as covariate too.

3.3.1.1 Mortality rates

17. We first model the mortality rates using a proportional hazards model, but allowing different levels of mortality between the two allocation groups (in `allo`), and the three albuminuria states (in `lex.Cst`):

```
> mix <- glm.Lexis(S4, ~ Ns(tfi, knots = seq( 0, 20, 5)) +
+                        Ns(age, knots = seq(50, 80, 10)) +
+                        lex.Cst * allo,
+                  to = "D(oth)")

stats::glm Poisson analysis of Lexis object S4 with log link:
Rates for transitions:
Norm->D(oth)
Mic->D(oth)
Mac->D(oth)

> round(ci.exp(mix), 3)

                                  exp(Est.)  2.5%        97.5%
(Intercept)                           0.000 0.000 5.000000e-03
Ns(tfi, knots = seq(0, 20, 5))1     195.021 3.367 1.129679e+04
Ns(tfi, knots = seq(0, 20, 5))2      43.667 1.644 1.159968e+03
Ns(tfi, knots = seq(0, 20, 5))3   59106.199 5.928 5.893319e+08
Ns(tfi, knots = seq(0, 20, 5))4       2.617 0.429 1.595000e+01
Ns(age, knots = seq(50, 80, 10))1     2.673 0.884 8.081000e+00
Ns(age, knots = seq(50, 80, 10))2     1.479 0.141 1.552500e+01
Ns(age, knots = seq(50, 80, 10))3    11.747 4.215 3.274000e+01
lex.CstMic                            0.967 0.361 2.593000e+00
lex.CstMac                            1.630 0.532 5.000000e+00
alloConv                              1.797 0.591 5.463000e+00
```

```
lex.CstMic:alloConv                    1.072 0.281 4.092000e+00
lex.CstMac:alloConv                    0.362 0.072 1.821000e+00
```

We would however like to see the allocation effect on mortality separately for each albuminuria state; this is done by the "/" operator in the model formula (pronounced: `allo` effect *within* `lex.Cst`):

```
> mox <- glm.Lexis(S4, ~ Ns(tfi, knots = seq( 0, 20, 5)) +
+                        Ns(age, knots = seq(50, 80, 10)) +
+                        lex.Cst / allo,
+                  to = "D(oth)")

stats::glm Poisson analysis of Lexis object S4 with log link:
Rates for transitions:
Norm->D(oth)
Mic->D(oth)
Mac->D(oth)

> round(ci.exp(mox), 3)

                                  exp(Est.)  2.5%        97.5%
(Intercept)                           0.000 0.000 5.000000e-03
Ns(tfi, knots = seq(0, 20, 5))1     195.021 3.367 1.129679e+04
Ns(tfi, knots = seq(0, 20, 5))2      43.667 1.644 1.159968e+03
Ns(tfi, knots = seq(0, 20, 5))3   59106.199 5.928 5.893319e+08
Ns(tfi, knots = seq(0, 20, 5))4       2.617 0.429 1.595000e+01
Ns(age, knots = seq(50, 80, 10))1     2.673 0.884 8.081000e+00
Ns(age, knots = seq(50, 80, 10))2     1.479 0.141 1.552500e+01
Ns(age, knots = seq(50, 80, 10))3    11.747 4.215 3.274000e+01
lex.CstMic                            0.967 0.361 2.593000e+00
lex.CstMac                            1.630 0.532 5.000000e+00
lex.CstNorm:alloConv                  1.797 0.591 5.463000e+00
lex.CstMic:alloConv                   1.927 0.925 4.013000e+00
lex.CstMac:alloConv                   0.651 0.205 2.071000e+00

> c(deviance(mox), deviance(mix))

[1] 734.0855 734.0855
```

The use of the deviance gives a good indication that the models fitted actually *are* the same model, just differently parametrized.

What is the meaning of the parameters for the `allo` effect?

18. We also need a similar model for the CVD-mortality:

```
> mCx <- glm.Lexis(S4, ~ Ns(tfi, knots = seq( 0, 20, 5)) +
+                        Ns(age, knots = seq(50, 80, 10)) +
+                        lex.Cst / allo,
+                  to = "D(CVD)")

stats::glm Poisson analysis of Lexis object S4 with log link:
Rates for transitions:
Norm->D(CVD)
Mic->D(CVD)
Mac->D(CVD)
```

```
> round(ci.exp(mCx), 3)

                                  exp(Est.)  2.5%      97.5%
(Intercept)                           0.000 0.000      0.013
Ns(tfi, knots = seq(0, 20, 5))1       1.004 0.140      7.179
Ns(tfi, knots = seq(0, 20, 5))2       2.291 0.354     14.834
Ns(tfi, knots = seq(0, 20, 5))3       1.361 0.016    115.800
Ns(tfi, knots = seq(0, 20, 5))4       0.125 0.016      0.988
Ns(age, knots = seq(50, 80, 10))1     7.289 1.122     47.353
Ns(age, knots = seq(50, 80, 10))2   641.288 1.868 220199.762
Ns(age, knots = seq(50, 80, 10))3    20.847 4.730     91.871
lex.CstMic                            0.808 0.199      3.275
lex.CstMac                            1.250 0.245      6.373
lex.CstNorm:alloConv                  1.399 0.278      7.047
lex.CstMic:alloConv                   1.682 0.579      4.889
lex.CstMac:alloConv                   4.856 1.367     17.250
```

What is the conclusion for the intervention effect on CVD mortality rates?

3.3.1.2 Albuminuria state rates

For a complete description of transitions in the model we also need models for the transitions between albuminuria states (the vertical arrows in figure 3.2):

19. We will use different models for `det`erioration (arrows up) and `imp`rovement (arrows down) in albuminuria in figure 3.2). Again the model specification is a simplified by `glm.Lexis`, but now requires specification of both `from` and `to` arguments:

```
> det <- glm.Lexis(S4, ~ Ns(tfi, knots = seq( 0, 20, 5)) +
+                        Ns(age, knots = seq(50, 80, 10)) +
+                        lex.Cst / allo,
+                  from = c("Norm","Mic"),
+                    to = c("Mic","Mac"))

stats::glm Poisson analysis of Lexis object S4 with log link:
Rates for transitions:
Norm->Mic
Mic->Mac

> imp <- glm.Lexis(S4, ~ Ns(tfi, knots = seq( 0, 20, 5)) +
+                        Ns(age, knots = seq(50, 80, 10)) +
+                        lex.Cst / allo,
+                  from = c("Mac","Mic"),
+                    to = c("Mic","Norm"))

stats::glm Poisson analysis of Lexis object S4 with log link:
Rates for transitions:
Mac->Mic
Mic->Norm

> round(ci.exp(det), 3)
```

```
                                  exp(Est.)  2.5%  97.5%
(Intercept)                           0.071 0.032  0.158
Ns(tfi, knots = seq(0, 20, 5))1       0.648 0.248  1.692
Ns(tfi, knots = seq(0, 20, 5))2       0.276 0.079  0.961
Ns(tfi, knots = seq(0, 20, 5))3       0.281 0.043  1.823
Ns(tfi, knots = seq(0, 20, 5))4       0.218 0.065  0.738
Ns(age, knots = seq(50, 80, 10))1     1.977 0.839  4.656
Ns(age, knots = seq(50, 80, 10))2     3.588 0.948 13.574
Ns(age, knots = seq(50, 80, 10))3     2.728 0.795  9.358
lex.CstMic                            0.393 0.223  0.695
lex.CstNorm:alloConv                  0.490 0.222  1.082
lex.CstMic:alloConv                   1.965 1.178  3.278
```

```
> round(ci.exp(imp), 3)
```

```
                                  exp(Est.)  2.5% 97.5%
(Intercept)                           0.208 0.128 0.339
Ns(tfi, knots = seq(0, 20, 5))1       0.239 0.075 0.759
Ns(tfi, knots = seq(0, 20, 5))2       0.069 0.011 0.420
Ns(tfi, knots = seq(0, 20, 5))3       0.046 0.008 0.277
Ns(tfi, knots = seq(0, 20, 5))4       0.172 0.034 0.859
Ns(age, knots = seq(50, 80, 10))1     0.844 0.292 2.444
Ns(age, knots = seq(50, 80, 10))2     0.342 0.069 1.690
Ns(age, knots = seq(50, 80, 10))3     0.580 0.073 4.624
lex.CstMac                            1.055 0.465 2.393
lex.CstMic:alloConv                   0.526 0.324 0.855
lex.CstMac:alloConv                   1.341 0.545 3.303
```

```
> round(ci.exp(det, subset="al"), 1)
```

```
                     exp(Est.) 2.5% 97.5%
lex.CstNorm:alloConv       0.5  0.2   1.1
lex.CstMic:alloConv        2.0  1.2   3.3
```

```
> round(ci.exp(imp, subset="al"), 1)
```

```
                     exp(Est.) 2.5% 97.5%
lex.CstMic:alloConv        0.5  0.3   0.9
lex.CstMac:alloConv        1.3  0.5   3.3
```

What was the meaning of "different models for `det` and `imp`"?

What do the parameters in the models represent?

What are the assumptions in the models?

Label the transitions in figure 3.2 with the models for each of the transitions.

3.3.2 Simulation of state probabilities

We now have statistical models for all transitions, two models for the cause specific mortality rates, and two models for transitions between albuminuria states.

The state probabilities that in principle can be derived from these are not trivial to compute, essentially they can only be computed by simulation[1].

[1] A detailed description of the use of `simLexis` is available in the vignette in the `Epi` package, also available as http://bendixcarstensen.com/Epi/simLexis.pdf

20. First we need an explicit specification of what transitions are decribed by what the model, since the simulated transitions will be using predictions from these models. This is specified in a list of lists (remember what a list is??).

    There must be one element in the list for each transient state (of which we have 3):

```
> Tr <- list(Norm = list("Mic" = det,
+                        "D(oth)" = mox,
+                        "D(CVD)" = mCx),
+             Mic = list("Mac" = det,
+                        "Norm" = imp,
+                        "D(oth)" = mox,
+                        "D(CVD)" = mCx),
+             Mac = list("Mic" = imp,
+                        "D(oth)" = mox,
+                        "D(CVD)" = mCx))
> lapply(Tr, names)

$Norm
[1] "Mic"    "D(oth)" "D(CVD)"

$Mic
[1] "Mac"    "Norm"   "D(oth)" "D(CVD)"

$Mac
[1] "Mic"    "D(oth)" "D(CVD)"
```

    For example, the object `Tr$Norm$Mic` is the model `det`; the model for the transition rate `Norm` → `Mic`; we see that there are 10 entries in the specification of `Tr`, corresponding to each of the 10 transitions in the diagram in figure 3.2. Some of the entries in `Tr` point to the same model. All the models fitted were actually joint models for more than one transition, but with specific parameters representing differences between the specific transition rates modeled.

21. We can use the estimated rates to simulate the transition between states in a group of people with a given set of covariates—the initial cohort.

    The simulated data can the be used to assess the probability of being in each of the states at a given time after entry to the study, by simply counting how many of the persons from the initial cohort are simulated to be in each state.

    These probabilities depend on the covariates used for modeling of transition rates, that is any of the covariates in `mox`, `mCx`, `det` and `imp`.

    We can choose our initial cohort in (at least) two different ways:

    - Use a population with the same covariate distribution as the entire study population at entry ("population-averaged")
    - Use a population with a prespecified set of covariates at entry ("conditional").

    Either way, what is needed is a data frame of persons indicating their initial status. `simLexis` will then simulate their individual trajectories through states (what

transition takes place when) and produce a simulated cohort of persons in the form of a `Lexis` object. The initial (baseline) data frame should also be a `Lexis` object, but the values of `lex.Xst` and `lex.dur` need not be given, since these will be simulated.

3.3.2.1 Study population cohort

22. First construct a cohort with the same covariate distribution as the entire study for each of the allocation groups:

```
> ini <- L2[,c("per", "age", "tfi")]
> ini <- rbind(transform(ini, lex.Cst = factor("Mic"), allo = factor("Int")),
+              transform(ini, lex.Cst = factor("Mic"), allo = factor("Conv")))
> str(ini)
```

```
Classes 'Lexis' and 'data.frame':        320 obs. of  5 variables:
 $ per    : 'cal.yr' num  1993 1993 1993 1993 1993 ...
 $ age    : 'cal.yr' num  61.1 46.6 49.9 48.5 57.3 ...
 $ tfi    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ lex.Cst: Factor w/ 1 level "Mic": 1 1 1 1 1 1 1 1 1 1 ...
 $ allo   : Factor w/ 2 levels "Int","Conv": 1 1 1 1 1 1 1 1 1 1 ...
 - attr(*, "breaks")=List of 3
  ..$ per: NULL
  ..$ age: NULL
  ..$ tfi: NULL
 - attr(*, "time.scales")= chr [1:3] "per" "age" "tfi"
 - attr(*, "time.since")= chr [1:3] "" "" ""
```

This will be the initial values in the cohort we follow through states—we have the starting state in `lex.Cst` and the covariates (at start): timescales (`per`, `age`, `tfi`) and the other covariates `allo`

23. First we simulate transitions from a large cohort that looks like the study population, say 10 copies of each person in the original data set (see `?simLexis`):

```
> set.seed(1952)
> system.time(
+ Sorg <- simLexis(Tr = Tr,  # models for each transition
+                 init = ini, # cohort of straters
+                    N = 100, # how many copies of each person in ini
+              t.range = 21,  # how long should we simulate before censoring
+                n.int = 200))# how many intervals for evaluating rates
```

```
   user  system elapsed
 200.53   36.29  240.66
```

There is no guaranteed order of the states in the `Sorg` object, so we explicitly reorder the states:

```
> Sorg <- Relevel(Sorg, c("Norm", "Mic", "Mac", "D(CVD)", "D(oth)"))
> summary(Sorg, by = "allo")
```

```
$Int

Transitions:
     To
From    Norm   Mic  Mac D(CVD) D(oth)  Records:  Events: Risk time:  Persons:
  Norm  2293  5472    0    648   1324      9737     7444   76184.55      8601
  Mic   9737  4202 4916   1258   2590     22703    18501  151493.64     16000
  Mac      0  1231 1523    619   1543      4916     3393   33953.90      4674
  Sum  12030 10905 6439   2525   5457     37356    29338  261632.09     16000

$Conv

Transitions:
     To
From   Norm  Mic  Mac D(CVD) D(oth)  Records:  Events: Risk time:  Persons:
  Norm 1460 1574    0    597   1457      5088     3628   47906.47      4927
  Mic  5088 2237 8302   1479   3169     20275    18038  129297.70     16000
  Mac     0 2701 1439   3154   1008      8302     6863   45188.18      7404
  Sum  6548 6512 9741   5230   5634     33665    28529  222392.35     16000
```

```
> subset(Sorg, lex.id %in% 28:32)
```

```
 lex.id     per   age   tfi lex.dur lex.Cst lex.Xst allo     cens
     28 1993.33 61.05  0.00    1.16     Mic    Norm  Int 2014.326
     28 1994.49 62.21  1.16    2.31    Norm     Mic  Int 2014.326
     28 1996.79 64.52  3.47    3.13     Mic  D(oth)  Int 2014.326
     29 1993.33 61.05  0.00    3.17     Mic    Norm  Int 2014.326
     29 1996.50 64.22  3.17   14.87    Norm     Mic  Int 2014.326
     29 2011.37 79.10 18.04    2.96     Mic     Mic  Int 2014.326
     30 1993.33 61.05  0.00    8.20     Mic  D(CVD)  Int 2014.326
     31 1993.33 61.05  0.00    9.08     Mic    Norm  Int 2014.326
     31 2002.40 70.13  9.08    3.40    Norm     Mic  Int 2014.326
     31 2005.80 73.53 12.48    1.90     Mic  D(CVD)  Int 2014.326
     32 1993.33 61.05  0.00    3.15     Mic     Mac  Int 2014.326
     32 1996.48 64.20  3.15    0.84     Mac     Mic  Int 2014.326
     32 1997.32 65.05  4.00   11.54     Mic  D(CVD)  Int 2014.326
```

24. Describe in words how the simulated data looks, and what each record represents. What is it really we simulated?

```
> addmargins(table(table(Sorg$lex.id)))
```

```
    1     2     3     4     5     6     7     8     9   Sum
 8416 13680  5765  3125   708   248    44    12     2 32000
```

What does this table mean?

25. Now we can just count how many of the original 1600 persons are in each of the states at each of a set of times; this is done by the function `nState`:

```
> system.time(
+ Nst <- nState(Sorg,
+                  at = seq(0, 20, 0.2),
+                from = 0,
+          time.scale = "tfi"))
```

```
   user  system elapsed
  13.62    0.32   14.02

> str(Nst)

 'table' int [1:101, 1:5] 0 873 1625 2346 2985 3547 4056 4585 5055 5477 ...
 - attr(*, "dimnames")=List of 2
  ..$ when : chr [1:101] "0" "0.2" "0.4" "0.6" ...
  ..$ State: chr [1:5] "Norm" "Mic" "Mac" "D(CVD)" ...

> head(Nst)

     State
when   Norm   Mic   Mac D(CVD) D(oth)
  0       0 32000     0      0      0
  0.2   873 30754   350     23      0
  0.4  1625 29711   611     53      0
  0.6  2346 28678   889     84      3
  0.8  2985 27739  1160    113      3
  1    3547 26903  1408    136      6
```

This is however not necessarily a relevant summary; we would be interested in seeing how things look in each of the allocation groups, `Int` and `Conv`.

```
> Nint <- nState(subset(Sorg, allo == "Int"),
+                at = seq(0, 20.2, 0.1),
+              from = 0,
+       time.scale = "tfi")
> Nconv<- nState(subset(Sorg, allo == "Conv"),
+                at = seq(0, 20.2, 0.1),
+              from = 0,
+       time.scale = "tfi")
> cbind(
+ head(Nint), NA,
+ head(Nconv))

    Norm   Mic Mac D(CVD) D(oth)    Norm   Mic Mac D(CVD) D(oth)
0      0 16000   0      0      0 NA    0 16000   0      0      0
0.1  294 15629  71      6      0 NA  157 15723 115      5      0
0.2  575 15287 129      9      0 NA  298 15467 221     14      0
0.3  820 14995 170     15      0 NA  405 15270 304     21      0
0.4 1091 14656 232     21      0 NA  534 15055 379     32      0
0.5 1327 14364 278     29      2 NA  676 14793 491     39      1
```

If we divide each of these by the number of persons, we get the probabilities of being in each if the states at the different times since entry.

26. If we want the state probabilities cumulated over states we can derive these by `pState`, that yields a matrix with the cumulative state probabilities.

```
> Pint  <- pState(Nint )
> Pconv <- pState(Nconv)
>  str(Pint)
```

```
 'pState' num [1:203, 1:5] 0 0.0184 0.0359 0.0512 0.0682 ...
 - attr(*, "dimnames")=List of 2
  ..$ when : chr [1:203] "0" "0.1" "0.2" "0.3" ...
  ..$ State: chr [1:5] "Norm" "Mic" "Mac" "D(CVD)" ...
```

```
> head(Pint)
```

```
     State
when       Norm       Mic       Mac   D(CVD) D(oth)
  0   0.0000000 1.0000000 1.0000000 1.000000      1
  0.1 0.0183750 0.9951875 0.9996250 1.000000      1
  0.2 0.0359375 0.9913750 0.9994375 1.000000      1
  0.3 0.0512500 0.9884375 0.9990625 1.000000      1
  0.4 0.0681875 0.9841875 0.9986875 1.000000      1
  0.5 0.0829375 0.9806875 0.9980625 0.999875      1
```

Describe the structure of `Pint`.

27. There is a standard plotting method for a `pState` object; it will plot the stacked state probabilities stacked in the order given by the `perm` argument (not used here because they are already in the order we want):

```
> clr <- c("forestgreen", "orange", "red", "blue", gray(0.4))
> par(mfrow = c(1,2), mar=c(3,3,2,2))
> plot(Pint , col = clr, xlim = c(0, 20))
> plot(Pconv, col = clr, xlim = c(20, 0))
```

. . . and slightly more sophisticated:

```
> clr <- c("forestgreen", "orange", "red", "blue", gray(0.4))
> par(mfrow = c(1,2), mar=c(3,3,2,2))
> plot(Pint, col = clr, xlim = c(0, 20))
> # the survival curve
> lines(as.numeric(rownames(Pint)), Pint[,"Mac"], lwd = 3, col = "black")
> lines(as.numeric(rownames(Pint)), Pint[,"Mac"], lwd = 1, col = "white")
> text(rownames(Pint)[150],
+      Pint[150,] - diff(c(0, Pint[150,]))/2,
+      colnames(Pint), col = "white", cex = 0.8)
> plot(Pconv, col = clr, xlim = c(20, 0))
> # the survival curve
> lines(as.numeric(rownames(Pconv)), Pconv[,"Mac"], lwd = 3, col = "black")
> lines(as.numeric(rownames(Pconv)), Pconv[,"Mac"], lwd = 1, col = "white")
> text(rownames(Pconv)[150],
+      Pconv[150,] - diff(c(0, Pconv[150,]))/2,
+      colnames(Pint), col = "white", cex = 0.8)
> mtext(c("Intensive care","Conventional care"),
+       side = 3, at = c(1,3)/4, outer = TRUE, line = -2)
```

Redo the plot with proper labeling of axes, including units where needed.

28. Describe the results and conclude on the probabilities shown.

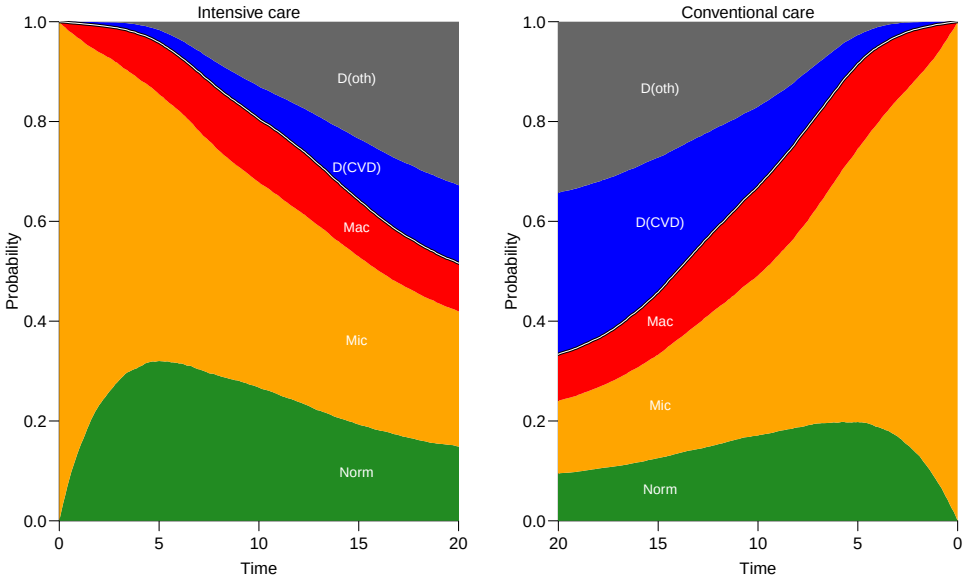


Figure 3.5: *State probabilities for the two intervention groups, for populations of the same structure as the original* total *Steno2 population.*

`../graph/ms-pStates`

29. The plot 3.5 may be of limited interest; the probabilities here are really "the probability that a randomly chosen person from the Steno 2 study…". So we are referring to a universe that is not generalizable, the reference is to a particular distribution of ages at entry etc. into the study. The plot is only partially relevant for showing the intervention effect, the absolute sizes of the state probabilities are strictly speaking irrelevant.

3.3.2.2 Initiation cohort with predefined variables

30. Even if we take the modeling background deeply serious and accept that occurrence rates depend only on current age (`age`), time since entry (`tfi`) and treatment allocation (`allo`), the assumption of age-distribution as in the Steno 2 study is quite absurd; who wants to refer to this? Often this is disguised in terms such as "population averaged".

    Therefore, it would be more relevant to show the results for a homogeneous population of persons at select ages at entry. This would just require a different `init` data frame. But note that it must be a `Lexis` object, easiest obtained by copying from `S4`:

```
> ini <- S4[1:10,c("lex.id", "per", "age", "tfi", "lex.Cst", "allo")]
> ini[,"lex.id"]  <- 1:10
> ini[,"per"]     <- 1993 # not used but it is a time scale in S4
> ini[,"age"]     <-
+ ini[,"ain"]     <- rep(seq(45,65,5), 2)
> ini[,"tfi"]     <- 0
```

```
> ini[,"lex.Cst"] <- factor("Mic",
+                           levels = c("Norm","Mic","Mac","D(CVD)","D(oth)"))
> ini[,"allo"]    <- factor(rep(c("Int","Conv"), each = 5))
> ini
```

```
 lex.id  per age tfi lex.Cst allo ain
      1 1993  45   0     Mic  Int  45
      2 1993  50   0     Mic  Int  50
      3 1993  55   0     Mic  Int  55
      4 1993  60   0     Mic  Int  60
      5 1993  65   0     Mic  Int  65
      6 1993  45   0     Mic Conv  45
      7 1993  50   0     Mic Conv  50
      8 1993  55   0     Mic Conv  55
      9 1993  60   0     Mic Conv  60
     10 1993  65   0     Mic Conv  65
```

```
> str(ini)
```

```
Classes 'Lexis' and 'data.frame':        10 obs. of  7 variables:
 $ lex.id : int  1 2 3 4 5 6 7 8 9 10
 $ per    : num  1993 1993 1993 1993 1993 ...
 $ age    : num  45 50 55 60 65 45 50 55 60 65
 $ tfi    : num  0 0 0 0 0 0 0 0 0 0
 $ lex.Cst: Factor w/ 5 levels "Norm","Mic","Mac",..: 2 2 2 2 2 2 2 2 2 2
 $ allo   : Factor w/ 2 levels "Conv","Int": 2 2 2 2 2 1 1 1 1 1
 $ ain    : num  45 50 55 60 65 45 50 55 60 65
 - attr(*, "time.scales")= chr [1:3] "per" "age" "tfi"
 - attr(*, "time.since")= chr [1:3] "" "" ""
 - attr(*, "breaks")=List of 3
  ..$ per: NULL
  ..$ age: NULL
  ..$ tfi: num [1:301] 0 0.0833 0.1667 0.25 0.3333 ...
```

Note that it is important that we enter the variable `lex.Cst` as a factor with the same levels as in the `Lexis` object `S4`, in the order we want the states when reporting results, because we will be using `ini` as basis for predictions of rates needed to do the simulations. Further, `allo` must also be entered as a factor, otherwise it is not possible to compute predictions from the models, because `allo` were included as a factor in the models.

31. For each of these combinations of age (at entry) and treatment allocation we will simulate 100 persons (note that we are using the same transition rates, the models in `Tr`):

```
> system.time(
+ Sdef <- simLexis(Tr = Tr,
+                  init = ini,
+                     N = 1000,
+               t.range = 21,
+                 n.int = 200))
```

```
   user  system elapsed
  58.27    7.86   66.22
```

```
> summary(Sdef, by = "allo")

$Conv

Transitions:
     To
From   Norm  Mic  Mac D(CVD) D(oth)  Records:  Events: Risk time:  Persons:
  Norm  455  496    0    175    426       1552     1097   14534.05      1498
  Mic  1552  726 2591    432   1021       6322     5596   41195.79      5000
  Mac     0  826  520    941    304       2591     2071   14578.57      2316
  Sum  2007 2048 3111   1548   1751      10465     8764   70308.41      5000

$Int

Transitions:
     To
From   Norm  Mic  Mac D(CVD) D(oth)  Records:  Events: Risk time:  Persons:
  Norm  706 1648    0    216    396       2966     2260   22922.78      2652
  Mic  2966 1362 1553    381    775       7037     5675   48019.31      5000
  Mac     0  389  493    186    485       1553     1060   11131.40      1488
  Sum  3672 3399 2046    783   1656      11556     8995   82073.49      5000

> subset(Sdef, lex.id < 5)

 lex.id     per   age  tfi lex.dur lex.Cst lex.Xst allo ain cens
      1 1993.00 45.00 0.00    2.76     Mic    Norm  Int  45 2014
      1 1995.76 47.76 2.76    0.29    Norm  D(oth)  Int  45 2014
      2 1993.00 45.00 0.00    4.50     Mic     Mac  Int  45 2014
      2 1997.50 49.50 4.50    2.33     Mac  D(oth)  Int  45 2014
      3 1993.00 45.00 0.00    2.51     Mic    Norm  Int  45 2014
      3 1995.51 47.51 2.51   18.49    Norm    Norm  Int  45 2014
      4 1993.00 45.00 0.00    7.19     Mic    Norm  Int  45 2014
      4 2000.19 52.19 7.19   13.81    Norm    Norm  Int  45 2014
```

In real applications we would use 5000 or 10,000 replicates of each to minimize the simulation error.

32. Now we will repeat the graph above, but for the 10 combinations of age at enrollment (`ain`), and allocation; we start with the 45 year old allocated to `Int`:

```
> P45i <- nState(subset(Sdef, ain == 45 & allo == "Int"),
+                at = seq(0, 20, 0.1),
+              from = 0,
+         time.scale = "tfi")
> head(P45i)

     State
when  Norm  Mic  Mac D(CVD) D(oth)
  0      0 1000    0      0      0
  0.1   20  980    0      0      0
  0.2   44  953    3      0      0
  0.3   57  939    4      0      0
  0.4   78  915    7      0      0
  0.5   93  900    7      0      0
```

```
> head(pState(P45i))

     State
when   Norm   Mic Mac D(CVD) D(oth)
  0   0.000 1.000   1      1      1
  0.1 0.020 1.000   1      1      1
  0.2 0.044 0.997   1      1      1
  0.3 0.057 0.996   1      1      1
  0.4 0.078 0.993   1      1      1
  0.5 0.093 0.993   1      1      1
```

This should then be repeated for 4 other ages at enrollment and the two allocations, plus we will only store the state probabilities:

```
> P45c <- pState(nState(subset(Sdef, ain == 45 & allo == "Conv"),
+                at = seq(0, 21, 0.1),
+              from = 0,
+        time.scale = "tfi"))
> P45i <- pState(nState(subset(Sdef, ain == 45 & allo == "Int"),
+                at = seq(0, 21, 0.1),
+              from = 0,
+        time.scale = "tfi"))
> P50c <- pState(nState(subset(Sdef, ain == 55 & allo == "Conv"),
+                at = seq(0, 21, 0.1),
+              from = 0,
+        time.scale = "tfi"))
> P50i <- pState(nState(subset(Sdef, ain == 55 & allo == "Int"),
+                at = seq(0, 21, 0.1),
+              from = 0,
+        time.scale = "tfi"))
> P55c <- pState(nState(subset(Sdef, ain == 55 & allo == "Conv"),
+                at = seq(0, 21, 0.1),
+              from = 0,
+        time.scale = "tfi"))
> P55i <- pState(nState(subset(Sdef, ain == 55 & allo == "Int"),
+                at = seq(0, 21, 0.1),
+              from = 0,
+        time.scale = "tfi"))
> P60c <- pState(nState(subset(Sdef, ain == 55 & allo == "Conv"),
+                at = seq(0, 21, 0.1),
+              from = 0,
+        time.scale = "tfi"))
> P60i <- pState(nState(subset(Sdef, ain == 55 & allo == "Int"),
+                at = seq(0, 21, 0.1),
+              from = 0,
+        time.scale = "tfi"))
> P65c <- pState(nState(subset(Sdef, ain == 65 & allo == "Conv"),
+                at = seq(0, 21, 0.1),
+              from = 0,
+        time.scale = "tfi"))
> P65i <- pState(nState(subset(Sdef, ain == 65 & allo == "Int"),
+                at = seq(0, 21, 0.1),
+              from = 0,
+        time.scale = "tfi"))
```

33. Then we can plot these in a multiple lay-out:

```
> par(mfrow = c(5,2), mar = c(1,1,0,0),
+       oma = c(3,3,1,0), mgp=c(3,1,0)/1.6)
> plot(P45i, col = clr, xlim = c(0,20))
> plot(P45c, col = clr, xlim = c(20,0))
> plot(P50i, col = clr, xlim = c(0,20))
> plot(P50c, col = clr, xlim = c(20,0))
> plot(P55i, col = clr, xlim = c(0,20))
> plot(P55c, col = clr, xlim = c(20,0))
> plot(P60i, col = clr, xlim = c(0,20))
> plot(P60c, col = clr, xlim = c(20,0))
> plot(P65i, col = clr, xlim = c(0,20))
> plot(P65c, col = clr, xlim = c(20,0))
> mtext(c("Int","Conv"), side = 3, at = c(1,3)/4, outer = TRUE, line = 0)
> mtext(paste(seq(45,65,5)), side = 2, at = (5:1*2-1)/10,
+       outer = TRUE, line = 0)
```

e see that the curves are quite ragged; this is due to the simulation errors, it would be nicer if we simulated 1000 copies of each instead of only 100.

34. *Digression*: The previous is a lot of hard-coding, we would like to be able to easily get a plot with only a subset of the ages. To this end it is more convenient to collect the state probabilities in an array:

```
> (ain <- seq(45, 65, 5))
```

```
[1] 45 50 55 60 65
```

```
> (alo <- levels(S4$allo))
```

```
[1] "Int"  "Conv"
```

```
> pdef <- NArray(c(list(ain = ain,
+                       allo = alo),
+                  dimnames(P45i)))
> str(pdef)
```

```
 logi [1:5, 1:2, 1:211, 1:5] NA NA NA NA NA NA ...
 - attr(*, "dimnames")=List of 4
  ..$ ain  : chr [1:5] "45" "50" "55" "60" ...
  ..$ allo : chr [1:2] "Int" "Conv"
  ..$ when : chr [1:211] "0" "0.1" "0.2" "0.3" ...
  ..$ State: chr [1:5] "Norm" "Mic" "Mac" "D(CVD)" ...
```

But when we stick the results in an array we lose the `pState` class of the results: so we resort to the `mat2pol` function that stacks probabilities and plots them, so we simply take the result from `nState` and divide by the number in the initial state (`Mic`) using `sweep`:

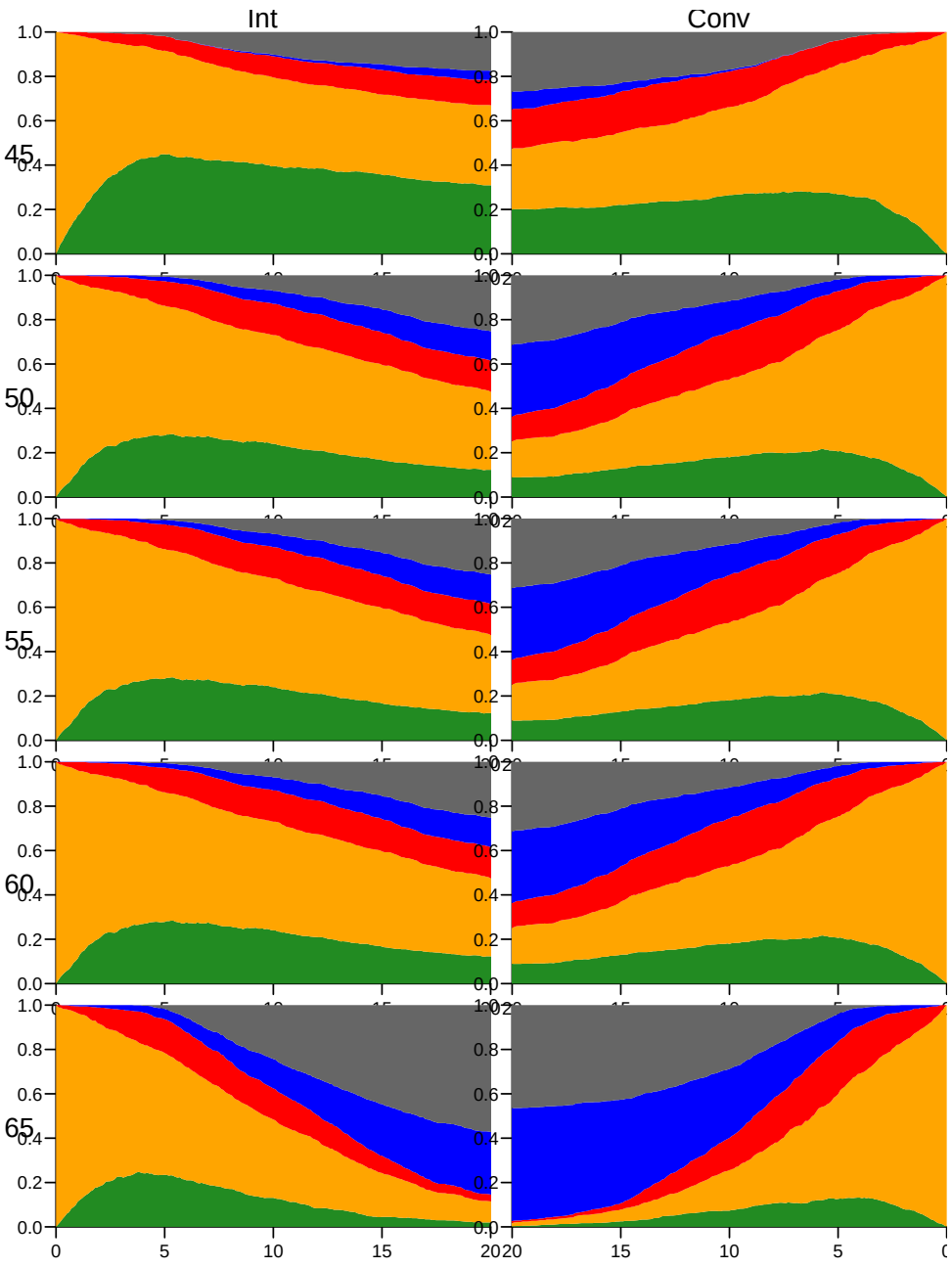


Figure 3.6: *Predicted probabilities of being in each of the states for persons aged 45, 50, 55, 60 and 65 at entry, separately for the two intervention groups. W* `../graph/ms-panel5`

```
> for(aa in ain)
+ for(gg in alo)
+    pdef[paste(aa), gg, ,] <-
+    nState(subset(Sdef, ain == aa & allo == gg),
+           at = as.numeric(dimnames(pdef)[["when"]]),
+         from = 0,
```

```
+   time.scale = "tfi")
> pdef <- sweep(pdef, 1:2, pdef[,,1,"Mic"], "/")
> str(pdef)

 num [1:5, 1:2, 1:211, 1:5] 0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, "dimnames")=List of 4
  ..$ ain  : chr [1:5] "45" "50" "55" "60" ...
  ..$ allo : chr [1:2] "Int" "Conv"
  ..$ when : chr [1:211] "0" "0.1" "0.2" "0.3" ...
  ..$ State: chr [1:5] "Norm" "Mic" "Mac" "D(CVD)" ...
```

Then we have the state probabilities in the array `pdef`

```
> ain <- seq(45, 65, 10)
> par(mfrow = c(length(ain),2),
+       mar = c(1,2,1,2)/5,
+       oma = c(2,4,2,3),
+       mgp = c(3,1,0) / 1.6)
> for(aa in ain)
+    {
+ mat2pol(pdef[paste(aa),"Int" ,,], col = clr, xlim = c(0,20),
+         xaxs = "i", yaxs = "i", xaxt = "n", yaxt = "n")
+ axis(side = 2)
+ axis(side = 4, labels = NA)
+ mat2pol(pdef[paste(aa),"Conv",,], col = clr, xlim = c(20,0),
+         xaxs = "i", yaxs = "i", xaxt = "n", yaxt = "n")
+ axis(side = 2, labels = NA)
+ axis(side = 4)
+    }
> mtext(c("Int","Conv"), side = 3, at = c(1,3)/4, outer = TRUE, line = 0)
> mtext(ain, side = 2, at = (length(ain):1 * 2 - 1) / (length(ain) * 2),
+       outer = TRUE, line = 2)
```

3.4 State probabilities using the Aalen-Johansen approach from `survival`

The `survival` package allows estimation of state probabilities by the Aalen-Johansen estimator similar to what we did in competing risks. The Aalen-Johansen estimator is the multi-state counterpart of the Kaplan-Meier estimator of survival probability in a 2-state (alive/dead) model.

As mentioned under competing risks, the results will refer to a population of the same structure as the study population, and so the absolute sizes of the state probabilities will not be generalizable to other populations. The results here correspond to the results we derived using the original Steno2 population cohort in section 3.3.2.1 on page 54 ff.

The estimates of state probabilities in section 3.3.2.1 are based on parametric models for the transition probabilities, where some of the transition rates depend on age and duration in the same way. The estimates from the Aalen-Johansen approach is non-parametric in the sense that the transition rates can have any shape; the down side is that they cannot

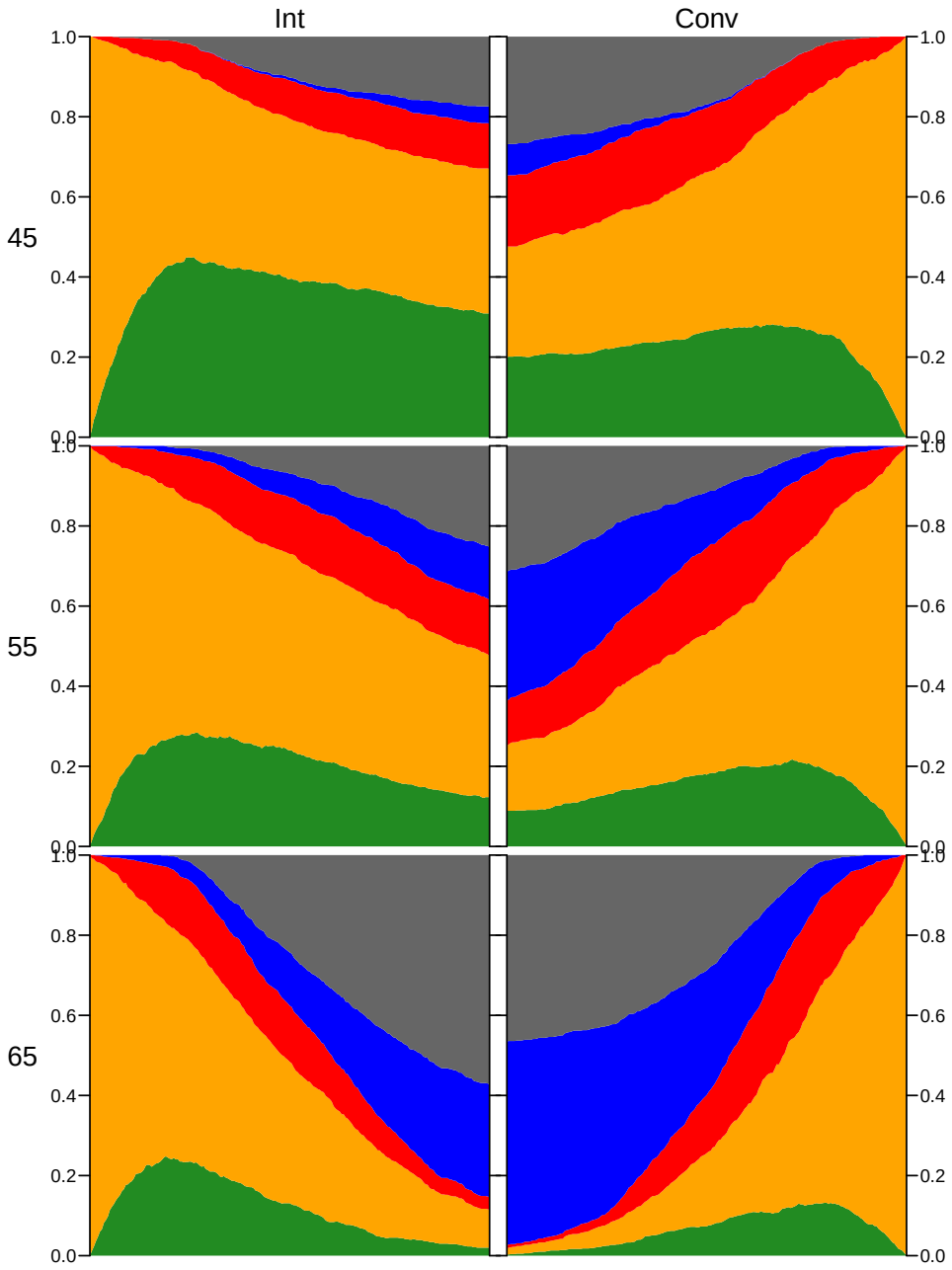


Figure 3.7: *Predicted probabilities of being in each of the states for persons aged 45, 55 and 65 at entry, separately for the two intervention groups.*

`../graph/ms-panel3`

depend on more than one time scale (here, sensibly time since entry) and the shape and size of them are not easily retrievable.

35. In `Epi` package there is a function, `AaJ.Lexis`, that does the hard work of reshaping a `Lexis` object for use by `survfit`, and calls `survfit` to return an object of class

`survfitms`.

```
> AaJepi <- AaJ.Lexis(L4, timeScale = "tfi")
NOTE: Timescale is tfi
> class(AaJepi)
[1] "survfitms" "survfit"
> AaJepi
Call: survfit(formula = form, data = Lx, id = lex.id, istate = lex.Cst)

         n nevent   rmean*
Norm   730     72 3.718799
Mic    730     57 9.173179
Mac    730     65 2.574825
D(CVD) 730     38 2.855333
D(oth) 730     55 3.580671
   *restricted mean time in state (max time = 21.90281 )
```

We find the same number of transitions in the `transitions` slot in the result from `AaJ.Lexis` as from the `summary.Lexis` (as we should):

```
> AaJepi$transitions
        to
from     Norm Mic Mac D(CVD) D(oth) (censored)
  Norm     72  35   0      6     13         18
  Mic      72 276  65     14     30         36
  Mac       0  22  28     18     12         13
  D(CVD)    0   0   0      0      0          0
  D(oth)    0   0   0      0      0          0
> summary(L4, simp = FALSE)
Transitions:
     To
From     Norm Mic Mac D(CVD) D(oth)  Records:  Events: Risk time:  Persons:
  Norm     90  35   0      6     13       144       54     581.04        66
  Mic      72 312  65     14     30       493      181    1435.14       160
  Mac       0  22  41     18     12        93       52     400.41        60
  D(CVD)    0   0   0      0      0         0        0         NA        NA
  D(oth)    0   0   0      0      0         0        0         NA        NA
  Sum     162 369 106     38     55       730      287    2416.59       160
```

36. The predicted state probabilities are in the slot called `pstate`, and the confidence intervals in the corresponding slots `lower` and `upper`.

```
> names(AaJepi)
 [1] "n"          "time"       "n.risk"     "n.event"    "n.censor"   "pstate"
 [7] "p0"         "cumhaz"     "std.err"    "sp0"        "logse"      "transition
[13] "lower"      "upper"      "conf.type"  "conf.int"   "states"     "type"
[19] "call"
```

```
> AaJepi$states

[1] "Norm"   "Mic"    "Mac"    "D(CVD)" "D(oth)"

> head(AaJepi$pstate)

        [,1]    [,2]    [,3] [,4] [,5]
[1,] 0.00000 0.99375 0.00625    0    0
[2,] 0.00625 0.98750 0.00625    0    0
[3,] 0.00625 0.98750 0.00625    0    0
[4,] 0.01250 0.98125 0.00625    0    0
[5,] 0.01250 0.98125 0.00625    0    0
[6,] 0.01250 0.98125 0.00625    0    0

> head(AaJepi$lower)

             [,1]      [,2]         [,3] [,4] [,5]
[1,]           NA 0.9816133 0.0008858142   NA   NA
[2,] 0.0008858142 0.9704340 0.0008858142   NA   NA
[3,] 0.0008858142 0.9704340 0.0008858142   NA   NA
[4,] 0.0031535032 0.9604561 0.0008858142   NA   NA
[5,] 0.0031535032 0.9604561 0.0008858142   NA   NA
[6,] 0.0031535032 0.9604561 0.0008858142   NA   NA

> head(AaJepi$upper)

           [,1] [,2]       [,3] [,4] [,5]
[1,]         NA    1 0.04409785   NA   NA
[2,] 0.04409785    1 0.04409785   NA   NA
[3,] 0.04409785    1 0.04409785   NA   NA
[4,] 0.04954807    1 0.04409785   NA   NA
[5,] 0.04954807    1 0.04409785   NA   NA
[6,] 0.04954807    1 0.04409785   NA   NA
```

We can now show the Aalen-Johansen estimator of the state probabilities:

```
> par(mfrow = c(1,1))
> mat2pol(AaJepi$pstate, perm = c(1:3,5,4), x = AaJepi$time, col = clr)
> lines(AaJepi$time, apply(AaJepi$pstate[,1:3], 1, sum), lwd = 5)
```

37. But as above, we are interested in seeing the results from each of the allocation groups, so we do the calculation for each:

```
> AaJallo <- AaJ.Lexis(L4, ~ allo, timeScale = "tfi")

NOTE: Timescale is tfi

> AaJallo

Call: survfit(formula = form, data = Lx, id = lex.id, istate = lex.Cst)

                    n nevent   rmean*
allo=Int, Norm    384     47 4.794785
allo=Conv, Norm   346     25 2.640688
allo=Int, Mic     384     34 9.933409
```

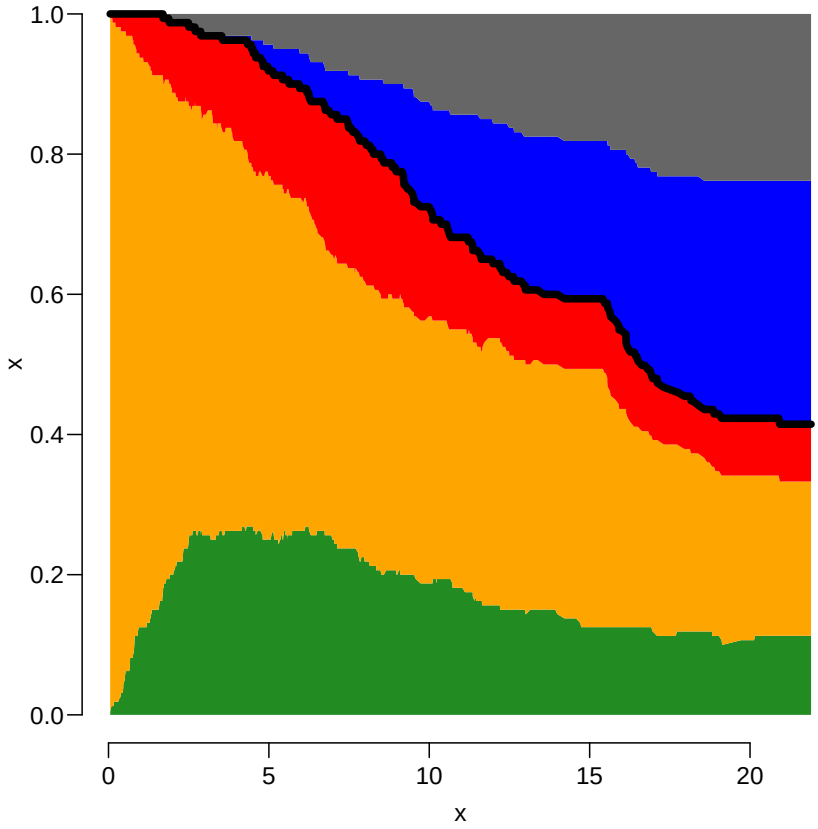


Figure 3.8: *Overall state probabilities from the Aalen-Johansen model*

`../graph/ms-AaJ`

```
allo=Conv, Mic    346     23 8.401154
allo=Int, Mac     384     23 2.073443
allo=Conv, Mac    346     42 3.077278
allo=Int, D(CVD)  384     12 2.024196
allo=Conv, D(CVD) 346     26 3.691880
allo=Int, D(oth)  384     26 3.076973
allo=Conv, D(oth) 346     29 4.091806
   *restricted mean time in state (max time = 21.90281 )
```

The result in the `AaJallo` object is in a long vector of `time` and `pstate`, the two parts corresponding to `Int` and `Conv` put after one another, with the length of each part in `strata`.

```
> AaJallo$states
[1] "Norm"    "Mic"     "Mac"     "D(CVD)" "D(oth)"
> AaJallo$strata
 allo=Int allo=Conv
      375       337
```

```
> wh <- rep(substr(names(AaJallo$strata), 6, 9), AaJallo$strata)
> table(wh)

wh
Conv  Int
 337  375
```

So we just make the plots for the two subsets and place them next to each other as before:

```
> par(mfrow = c(1,2), mar=c(3,3,2,0), oma = c(0,0,0,3), las = 1, bty = "n")
> mat2pol(AaJallo$pstate[wh=="Int",],
+         x = AaJallo$time[wh=="Int"],
+         col = clr, xlim = c(0,21), xaxs = "i", yaxs = "i")
> lines(AaJallo$time[wh=="Int"],
+       apply(AaJallo$pstate[,1:3], 1, sum)[wh=="Int"], lwd = 4)
> mat2pol(AaJallo$pstate[wh=="Conv",],
+         x = AaJallo$time[wh=="Conv"],
+         col = clr, xlim = c(21,0), xaxs = "i", yaxs = "i")
> lines(AaJallo$time[wh=="Conv"],
+       apply(AaJallo$pstate[,1:3], 1, sum)[wh=="Conv"], lwd = 4)
> axis(side = 4)
> mtext(c("Int","Conv"), side = 3, at = c(1,3)/4, outer = TRUE, line = -2)
```

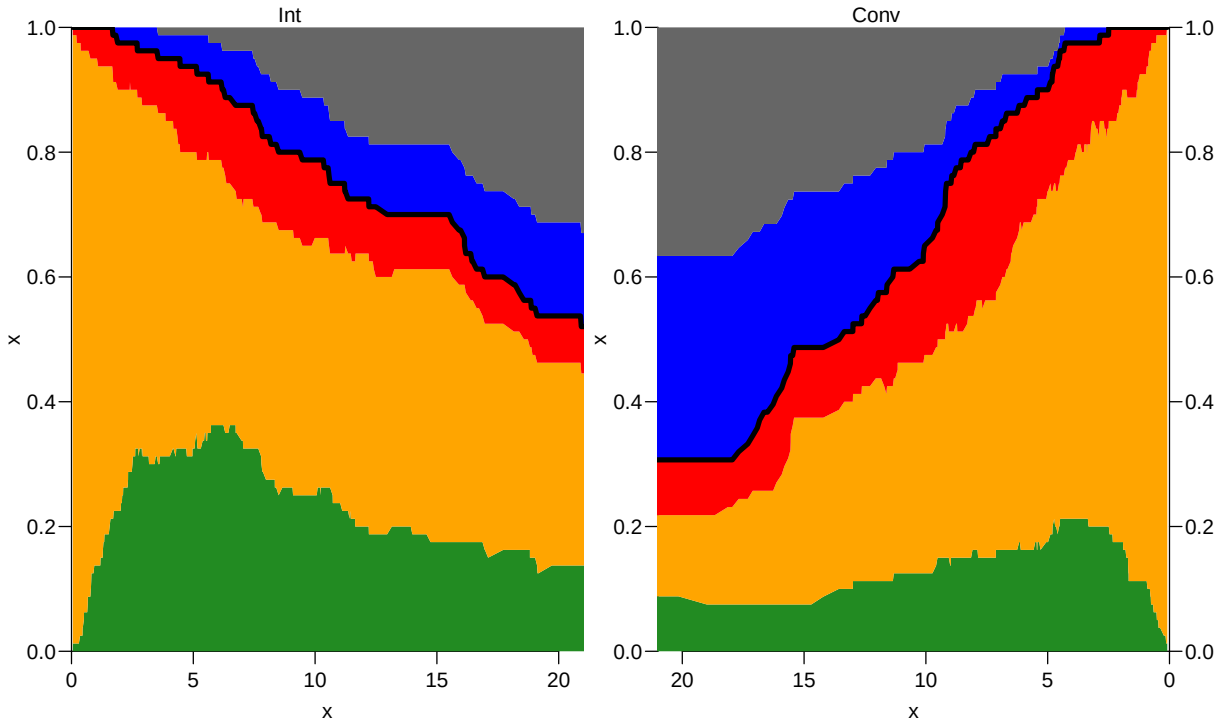


Figure 3.9: *Aalen-Johansen estimator of the state probabilities for the two intervention groups, for the original Steno2 data, subdivided by intervention allocation.*

`../graph/ms-AaJstates`

38. The estimator of state probabilities in figure 3.9 can be considered the empirical counterpart of figure 3.5; the state probabilities for a population as the one in the study. But then, not quite so; the models underlying figure 3.5 are proportional hazards in the sense that the effects of age and time since enrollment are proportional between state by allocation (6 groups for mortality, 4 groups for albuminuria state), whereas the figures in 3.9 are based on separate models for each transition and allocation, but only with time since enrollment as timescale (no age-effect assumed).

39. We have confidence intervals for each of the state probabilities in the slots `lower` and `upper`, but not for the sums of these. And it is (cumulative) sums of state probabilities we have shown in the graph.

    Moreover we would also want confidence intervals for areas under the curves. Neither are available from the Aalen-Johansen nor from the simulation approach. The simulation approach does not even give confidence intervals for each of the state probabilities.

3.5 Time spent in albuminuria states

Besides the state probabilities at different times after entry for groups of patients, we may also want to assess the time spent in each state, during, say, the first 15 or 20 years after entry.

40. We may want to compare groups by the expected time spent in the albuminuria states during the first, say, 20 years. The expected time in a state is simply the time-integral of the probabilities, so we could compute it from `pdef`; each probability represents an interval of length 0.1, so we just take the midpoint of the probabilities from `pstate` at the ends of each interval.

    This is however a bit like bringing coal to Newcastle, we have the simulated cohort `Sdef`, where we have simulated sojourn times in each state; so we can just sum these and use as estimates of time spent in each state.

```
> tLive <- xtabs(lex.dur ~ ain + allo + lex.Cst, data = Sdef) /
+          nid(Sdef) * 10
> str(mtLive <- addmargins(tLive, 3))
```

```
 'table' num [1:5, 1:2, 1:6] 4.69 3.56 3.05 1.94 1.3 ...
 - attr(*, "dimnames")=List of 3
  ..$ ain    : chr [1:5] "45" "50" "55" "60" ...
  ..$ allo   : chr [1:2] "Conv" "Int"
  ..$ lex.Cst: chr [1:6] "Norm" "Mic" "Mac" "D(CVD)" ...
```

```
> round(ftable(mtLive          , col.vars = c(3,2)), 1)
```

```
    lex.Cst Norm       Mic       Mac      D(CVD)      D(oth)       Sum
    allo    Conv  Int Conv  Int Conv  Int   Conv  Int   Conv  Int Conv  Int
ain
45           4.7  7.5  9.7  9.5  3.0  1.8    0.0  0.0    0.0  0.0 17.4 18.7
```

```
50           3.6  5.7  9.8 10.5  3.4  2.3    0.0  0.0    0.0  0.0 16.8 18.4
55           3.0  4.1  8.5 10.9  3.2  2.5    0.0  0.0    0.0  0.0 14.8 17.6
60           1.9  3.2  7.4  9.2  2.7  2.6    0.0  0.0    0.0  0.0 12.0 15.0
65           1.3  2.4  5.8  8.0  2.3  2.0    0.0  0.0    0.0  0.0  9.4 12.4
```

```
> round(ftable(mtLive[,,-(4:5)], col.vars = c(3,2)), 1)
```

```
    lex.Cst Norm       Mic       Mac       Sum
    allo    Conv  Int Conv  Int Conv  Int Conv  Int
ain
45           4.7  7.5  9.7  9.5  3.0  1.8 17.4 18.7
50           3.6  5.7  9.8 10.5  3.4  2.3 16.8 18.4
55           3.0  4.1  8.5 10.9  3.2  2.5 14.8 17.6
60           1.9  3.2  7.4  9.2  2.7  2.6 12.0 15.0
65           1.3  2.4  5.8  8.0  2.3  2.0  9.4 12.4
```

With the results in an array we can also easily show the difference between the intervention and the control arms of the trial:

```
> round((mtLive[,"Int",-(4:5)] - mtLive[,"Conv",-(4:5)]), 1)
```

```
    lex.Cst
ain  Norm  Mic  Mac  Sum
  45  2.8 -0.2 -1.2  1.3
  50  2.2  0.6 -1.1  1.7
  55  1.1  2.4 -0.7  2.8
  60  1.3  1.8 -0.1  3.0
  65  1.1  2.2 -0.3  3.0
```

41. We might also want to know the lifetime lost (during the first 20 years) to each of the two causes of death. This timespan is not directly available in `Sdef`, it is the time from death till 20 years (the time-frame we have chosen). For persons who die, the time of death is `tfi` + `lex.dur` from the record with `lex.Xst` $\in$ (`D(oth)`, `D(CVD)`), so quite easily evaluated with `xtabs` too:

```
> tDead <- xtabs((20 - tfi - lex.dur) ~ ain + allo + lex.Xst,
+          data = subset(Sdef, lex.Xst %in% c("D(oth)", "D(CVD)"))) /
+          nid(Sdef) * 10
> str(mtDead <- addmargins(tDead[,,4:5], 3))
```

```
 'table' num [1:5, 1:2, 1:3] 0.439 1.767 3.069 4.327 5.738 ...
 - attr(*, "dimnames")=List of 3
  ..$ ain    : chr [1:5] "45" "50" "55" "60" ...
  ..$ allo   : chr [1:2] "Conv" "Int"
  ..$ lex.Xst: chr [1:3] "D(CVD)" "D(oth)" "Sum"
```

```
> round(ftable(mtDead          , col.vars = c(3,2)), 1)
```

```
    lex.Xst D(CVD)      D(oth)       Sum
    allo      Conv  Int   Conv  Int Conv  Int
ain
45             0.4  0.3    2.8  1.8  3.3  2.1
50             1.8  0.8    2.0  1.4  3.8  2.3
55             3.1  1.2    2.5  1.8  5.6  3.0
60             4.3  2.1    3.8  3.3  8.1  5.4
65             5.7  2.8    4.9  5.0 10.6  7.8
```

```
> round(ftable(mtDead[,,-(4:5)], col.vars = c(3,2)), 1)

    lex.Xst D(CVD)      D(oth)       Sum
    allo      Conv  Int   Conv  Int Conv  Int
ain
45             0.4  0.3    2.8  1.8  3.3  2.1
50             1.8  0.8    2.0  1.4  3.8  2.3
55             3.1  1.2    2.5  1.8  5.6  3.0
60             4.3  2.1    3.8  3.3  8.1  5.4
65             5.7  2.8    4.9  5.0 10.6  7.8
```

The difference between these numbers between intervention and control groups are the years gained from the intervention, subdivided by cause of death:

```
> round((mtDead["Int",] - mtDead["Conv",]), 1)

    lex.Xst
ain  D(CVD) D(oth)  Sum
  45   -0.2   -1.0 -1.2
  50   -0.9   -0.6 -1.5
  55   -1.8   -0.7 -2.5
  60   -2.3   -0.5 -2.8
  65   -3.0    0.1 -2.9
```

Draw a conclusion from these numbers.

3.6 Clinical variables

So far we have only considered covariates that we know the value of at any time point, including future time points, that is the allocation status and timescales such as age and time since inclusion.

42. In the dataset st2clin are clinical measurements taken at different dates, up to six different occasions per person:

```
> data(st2clin)
>  str(st2clin)

'data.frame':        750 obs. of  5 variables:
 $ id  : num  1 2 3 4 5 6 7 8 9 10 ...
 $ doV : Date, format: "1993-05-07" "1993-05-05" ...
 $ a1c : num  87.3 66.5 73 61.2 102.7 ...
 $ chol: num  3.9 6.6 5.6 5.2 6 4.8 8.6 5.1 4.2 5.4 ...
 $ crea: num  83 83 68 97 149 55 56 78 123 79 ...

> st2clin <- cal.yr(st2clin)
> names(st2clin)

[1] "id"   "doV"  "a1c"  "chol" "crea"

> names(st2clin)[1:2] <- c("lex.id","per")
> summary(st2clin)
```

```
     lex.id            per             a1c              chol             crea
 Min.   :  1.00   Min.   :1993   Min.   : 32.80   Min.   : 2.200   Min.   :  28.00
 1st Qu.: 39.00   1st Qu.:1995   1st Qu.: 54.80   1st Qu.: 4.000   1st Qu.:  67.00
 Median : 84.50   Median :1997   Median : 66.35   Median : 4.800   Median :  88.00
 Mean   : 85.81   Mean   :2000   Mean   : 68.22   Mean   : 4.941   Mean   :  99.16
 3rd Qu.:131.00   3rd Qu.:2002   3rd Qu.: 79.38   3rd Qu.: 5.700   3rd Qu.: 115.25
 Max.   :176.00   Max.   :2015   Max.   :147.60   Max.   :14.000   Max.   :1067.00
                                 NA's   :4        NA's   :3        NA's   :2
```

```
> addmargins(table(table(st2clin$lex.id)))

  1   2   3   4   5   6 Sum
  2   6  23  38  31  60 160
```

Explain the contents of the table.

43. We can use `addCov.Lexis` to amend the follow-up data with the clinical measurements:

```
> S5 <- addCov.Lexis(S4, st2clin, "per")
> tt <- table(st2clin$lex.id)
> (who <- names(tt[tt == 3])[1])

[1] "5"

> subset(st2clin, lex.id == who)

    lex.id      per   a1c chol crea
5        5 1993.151 102.7  6.0  149
165      5 1995.511  54.7  8.8  140
321      5 1997.496  41.9  5.8  141

> subset(S5,
+        lex.id == who,
+        select = c(lex.id,per,tfi,tfc,exnam,a1c,chol,crea))

 lex.id     per  tfi  tfc exnam   a1c chol crea
      5 1993.22 0.00 0.07   ex1 102.7  6.0  149
      5 1993.31 0.08 0.15   ex1 102.7  6.0  149
      5 1993.39 0.17 0.24   ex1 102.7  6.0  149
      5 1993.47 0.25 0.32   ex1 102.7  6.0  149
      5 1993.56 0.33 0.40   ex1 102.7  6.0  149
      5 1993.64 0.42 0.49   ex1 102.7  6.0  149
      5 1993.72 0.50 0.57   ex1 102.7  6.0  149
      5 1993.77 0.55 0.62   ex1 102.7  6.0  149
      5 1993.81 0.58 0.65   ex1 102.7  6.0  149
      5 1993.89 0.67 0.74   ex1 102.7  6.0  149
      5 1993.97 0.75 0.82   ex1 102.7  6.0  149
      5 1994.06 0.83 0.90   ex1 102.7  6.0  149
      5 1994.14 0.92 0.99   ex1 102.7  6.0  149
      5 1994.22 1.00 1.07   ex1 102.7  6.0  149
      5 1994.31 1.08 1.15   ex1 102.7  6.0  149
      5 1994.39 1.17 1.24   ex1 102.7  6.0  149
      5 1994.47 1.25 1.32   ex1 102.7  6.0  149
      5 1994.56 1.33 1.40   ex1 102.7  6.0  149
```

```
5 1994.64 1.42 1.49   ex1 102.7  6.0  149
5 1994.72 1.50 1.57   ex1 102.7  6.0  149
5 1994.81 1.58 1.65   ex1 102.7  6.0  149
5 1994.89 1.67 1.74   ex1 102.7  6.0  149
5 1994.97 1.75 1.82   ex1 102.7  6.0  149
5 1995.06 1.83 1.90   ex1 102.7  6.0  149
5 1995.14 1.92 1.99   ex1 102.7  6.0  149
5 1995.22 2.00 2.07   ex1 102.7  6.0  149
5 1995.31 2.08 2.15   ex1 102.7  6.0  149
5 1995.39 2.17 2.24   ex1 102.7  6.0  149
5 1995.47 2.25 2.32   ex1 102.7  6.0  149
5 1995.51 2.29 0.00   ex2  54.7  8.8  140
5 1995.56 2.33 0.04   ex2  54.7  8.8  140
5 1995.64 2.42 0.13   ex2  54.7  8.8  140
5 1995.72 2.50 0.21   ex2  54.7  8.8  140
5 1995.81 2.58 0.29   ex2  54.7  8.8  140
5 1995.89 2.67 0.38   ex2  54.7  8.8  140
5 1995.97 2.75 0.46   ex2  54.7  8.8  140
5 1996.06 2.83 0.54   ex2  54.7  8.8  140
5 1996.14 2.92 0.63   ex2  54.7  8.8  140
5 1996.22 3.00 0.71   ex2  54.7  8.8  140
5 1996.31 3.08 0.79   ex2  54.7  8.8  140
5 1996.39 3.17 0.88   ex2  54.7  8.8  140
5 1996.47 3.25 0.96   ex2  54.7  8.8  140
5 1996.56 3.33 1.04   ex2  54.7  8.8  140
5 1996.64 3.42 1.13   ex2  54.7  8.8  140
5 1996.72 3.50 1.21   ex2  54.7  8.8  140
5 1996.81 3.58 1.29   ex2  54.7  8.8  140
5 1996.89 3.67 1.38   ex2  54.7  8.8  140
5 1996.97 3.75 1.46   ex2  54.7  8.8  140
5 1997.06 3.83 1.54   ex2  54.7  8.8  140
5 1997.07 3.85 1.56   ex2  54.7  8.8  140
5 1997.14 3.92 1.63   ex2  54.7  8.8  140
5 1997.22 4.00 1.71   ex2  54.7  8.8  140
5 1997.31 4.08 1.79   ex2  54.7  8.8  140
5 1997.39 4.17 1.88   ex2  54.7  8.8  140
5 1997.47 4.25 1.96   ex2  54.7  8.8  140
5 1997.50 4.27 0.00   ex3  41.9  5.8  141
5 1997.56 4.33 0.06   ex3  41.9  5.8  141
5 1997.64 4.42 0.14   ex3  41.9  5.8  141
5 1997.72 4.50 0.23   ex3  41.9  5.8  141
5 1997.81 4.58 0.31   ex3  41.9  5.8  141
5 1997.89 4.67 0.39   ex3  41.9  5.8  141
5 1997.97 4.75 0.48   ex3  41.9  5.8  141
```

```
> timeScales(S5)

[1] "per" "age" "tfi" "tfc"

> timeSince(S5)

per age tfi tfc
 ""  ""  "" "X"
```

We see that `tfc` is included as a time scale, but it is a not a proper time scale; it is reset to 0 at every clinical visit, and it also has some missing values, as do the clinical

variables. The missing values are where there is follow-up before the earliest clinical measurement for a person.

But `tfc` needs to be a time scale in the `Lexis` object in order to be properly handled when subsequently cutting and splitting a `Lexis` object.

44. The values of the clinical measurements in `st2clin` are added to the follow-up data: extra cut points at the measurement dates are added, and the values of the clinical variables are propagated as LOCF (Last Observation Carried Forward), so it is possible to model the effect of these clinical variables on transition rates—creatinine is traditionally modeled on a log-scale, here we use the base 2 logarithm.

```
> detc <- glm.Lexis(S5, ~ Ns(tfi, knots = seq( 0, 20, 5)) +
+                         Ns(age, knots = seq(50, 80, 10)) +
+                         lex.Cst / allo +
+                         a1c + chol + log2(crea),
+                   from = c("Norm","Mic"),
+                     to = c("Mic","Mac"))

stats::glm Poisson analysis of Lexis object S5 with log link:
Rates for transitions:
Norm->Mic
Mic->Mac

> impc <- glm.Lexis(S5, ~ Ns(tfi, knots = seq( 0, 20, 5)) +
+                         Ns(age, knots = seq(50, 80, 10)) +
+                         lex.Cst / allo +
+                         a1c + chol + log2(crea),
+                     to = c("Norm","Mic"),
+                   from = c("Mic","Mac"))

stats::glm Poisson analysis of Lexis object S5 with log link:
Rates for transitions:
Mic->Norm
Mac->Mic

> round(ci.exp(detc), 3)

                                  exp(Est.)  2.5%  97.5%
(Intercept)                           0.080 0.005  1.337
Ns(tfi, knots = seq(0, 20, 5))1       0.714 0.271  1.880
Ns(tfi, knots = seq(0, 20, 5))2       0.300 0.084  1.067
Ns(tfi, knots = seq(0, 20, 5))3       0.297 0.045  1.967
Ns(tfi, knots = seq(0, 20, 5))4       0.226 0.066  0.776
Ns(age, knots = seq(50, 80, 10))1     2.032 0.862  4.789
Ns(age, knots = seq(50, 80, 10))2     4.453 1.131 17.535
Ns(age, knots = seq(50, 80, 10))3     3.387 0.948 12.106
lex.CstMic                            0.390 0.220  0.692
a1c                                   1.005 0.993  1.018
chol                                  1.093 0.912  1.311
log2(crea)                            0.864 0.582  1.281
lex.CstNorm:alloConv                  0.433 0.193  0.975
lex.CstMic:alloConv                   1.698 0.974  2.959
```

We do not really need to see the first 8 parameter estimates so we omit them:

```
> round(ci.exp(detc, subset = -(1:8), pval = T), 3)

                     exp(Est.)  2.5% 97.5%     P
lex.CstMic               0.390 0.220 0.692 0.001
a1c                      1.005 0.993 1.018 0.398
chol                     1.093 0.912 1.311 0.336
log2(crea)               0.864 0.582 1.281 0.466
lex.CstNorm:alloConv     0.433 0.193 0.975 0.043
lex.CstMic:alloConv      1.698 0.974 2.959 0.062
```

```
> round(ci.exp(impc, subset = -(1:8), pval = T), 3)

                    exp(Est.)  2.5% 97.5%     P
lex.CstMac              1.053 0.465 2.382 0.902
a1c                     0.990 0.978 1.003 0.129
chol                    0.964 0.804 1.157 0.697
log2(crea)              0.869 0.578 1.308 0.501
lex.CstMic:alloConv     0.598 0.359 0.996 0.048
lex.CstMac:alloConv     1.525 0.611 3.804 0.366
```

We also look at the effect of the clinical variables on the mortality rates:

```
> moc <- glm.Lexis(S5, ~ Ns(tfi, knots = seq( 0, 20, 5)) +
+                        Ns(age, knots = seq(50, 80, 10)) +
+                        lex.Cst / allo +
+                        a1c + chol + log2(crea),
+                    to = "D(oth)")

stats::glm Poisson analysis of Lexis object S5 with log link:
Rates for transitions:
Norm->D(oth)
Mic->D(oth)
Mac->D(oth)
```

```
> mCc <- glm.Lexis(S5, ~ Ns(tfi, knots = seq( 0, 20, 5)) +
+                        Ns(age, knots = seq(50, 80, 10)) +
+                        lex.Cst / allo +
+                        a1c + chol + log2(crea),
+                    to = "D(CVD)")

stats::glm Poisson analysis of Lexis object S5 with log link:
Rates for transitions:
Norm->D(CVD)
Mic->D(CVD)
Mac->D(CVD)
```

```
> round(ci.exp(moc, subset = -(1:8), pval = T), 3)

                     exp(Est.)  2.5% 97.5%     P
lex.CstMic               0.972 0.361 2.615 0.955
lex.CstMac               1.304 0.411 4.135 0.652
a1c                      1.005 0.987 1.024 0.582
chol                     0.839 0.630 1.117 0.229
log2(crea)               1.860 1.145 3.019 0.012
lex.CstNorm:alloConv     1.900 0.606 5.962 0.271
lex.CstMic:alloConv      1.944 0.878 4.306 0.101
lex.CstMac:alloConv      0.777 0.235 2.575 0.680
```

```
> round(ci.exp(mCc, subset = -(1:8), pval = T), 3)

                      exp(Est.)  2.5%  97.5%     P
lex.CstMic                0.806 0.199  3.274 0.763
lex.CstMac                1.144 0.221  5.916 0.873
a1c                       0.999 0.980  1.019 0.951
chol                      1.002 0.734  1.367 0.990
log2(crea)                1.351 0.757  2.409 0.309
lex.CstNorm:alloConv      1.397 0.272  7.177 0.689
lex.CstMic:alloConv       1.680 0.552  5.113 0.361
lex.CstMac:alloConv       5.067 1.387 18.513 0.014
```

Only `crea` has any effect; a doubling of creatinine is associated with a 1.86 times higher mortality rate from other (non-CVD) causes. Confidence interval is (1.14,3.02), so not terribly precisely determined.

There are limitations in using clinical measurements as time-dependent variables without a model for the clinical variables. In order to simulate events based on models for transition rates we must know all covariates at all (future) times, so models with non-deterministicly varying are not usable. Timescales are time-varying covariate, but they vary `deterministicly`, so their value for each person will be known at any time of future follow-up.

So the models with effects of clinical variables as presented here *cannot* be used for prediction of state probabilities—that would require a model for the change of the clinical variables over time as well.

3.7 Several transitions from one state: `stack`

So far, we have only jointly modeled transitions that originated in *different* states, for example:

`Mic` → `Mac` and `Norm` → `Mic`;

`Norm` → `D(CVD)`, `Mic` → `D(CVD)` and `Mac` → `D(CVD)`.

As long as the different rates modeled are originating in *different* states, the likelihood will have at most one contribution from each record in the `Lexis` follow-up data set.

But if we want to create a joint model for more than one rate originating in a given state we must repeat some of risk time in different contributions to the likelihood. This means that the modeling cannot be based on (subsets of) a `Lexis` object, we must repeat some records. This is detailed in section on Competing Risks in the chapter on multistate likelihood in the PMM (Practical Multistate Modeling, http://bendixcarstensen.com/MSbook.pdf).

This behaviour can be achieved by the `stack.Lexis` function:

```
> St4 <- stack(S4)
> c(nrow(S4), nrow(St4))
[1]  29645 106569
> table(S4$lex.Cst)
```

```
  Norm    Mic    Mac D(CVD) D(oth)
  7115  17634   4896      0      0
> table(St4$lex.Tr, St4$lex.Cst)
               Norm   Mic   Mac D(CVD) D(oth)
  Mac->D(CVD)     0     0  4896      0      0
  Mac->D(oth)     0     0  4896      0      0
  Mac->Mic        0     0  4896      0      0
  Mic->D(CVD)     0 17634     0      0      0
  Mic->D(oth)     0 17634     0      0      0
  Mic->Mac        0 17634     0      0      0
  Mic->Norm       0 17634     0      0      0
  Norm->D(CVD)  7115     0     0      0      0
  Norm->D(oth)  7115     0     0      0      0
  Norm->Mic     7115     0     0      0      0
> ftable(St4$lex.Tr, St4$lex.Xst, St4$lex.Fail, col.vars = 2:3)
              Norm         Mic         Mac       D(CVD)       D(oth)
             FALSE  TRUE FALSE  TRUE FALSE  TRUE  FALSE  TRUE  FALSE  TRUE

Mac->D(CVD)      0     0    22     0  4844     0      0    18     12     0
Mac->D(oth)      0     0    22     0  4844     0     18     0      0    12
Mac->Mic         0     0     0    22  4844     0     18     0     12     0
Mic->D(CVD)     72     0 17453     0    65     0      0    14     30     0
Mic->D(oth)     72     0 17453     0    65     0     14     0      0    30
Mic->Mac        72     0 17453     0     0    65     14     0     30     0
Mic->Norm        0    72 17453     0    65     0     14     0     30     0
Norm->D(CVD)  7061     0    35     0     0     0      0     6     13     0
Norm->D(oth)  7061     0    35     0     0     0      6     0      0    13
Norm->Mic     7061     0     0    35     0     0      6     0     13     0
```

We see that the `lex.Fail` is only `TRUE` where `lex.Xst` is equal to the second part if the `lex.Tr`.

The two ways of representing the data for person 102 are quite different:

```
> subset(S4 , lex.id == 102)[,1:8]
> subset(St4, lex.id == 102)[,1:9]
```

Suppose we wanted to fit a model for the two types of mortality assuming that, say, the effect of sex was the same.

Since some of the transitions we put in the same model originate from the same state we need the stacked data representation where each record corresponds to a likelihood term.

```
> cbind(with(subset(St4, grepl("D", lex.Tr)), table(lex.Tr)))
              [,1]
Mac->D(CVD)   4896
Mac->D(oth)   4896
Mac->Mic         0
Mic->D(CVD)  17634
Mic->D(oth)  17634
Mic->Mac         0
Mic->Norm        0
Norm->D(CVD)  7115
Norm->D(oth)  7115
Norm->Mic        0
```

We can then fit a model with common effect of `sex` for the two types mortality:

```
> stD <- glm(cbind(lex.Fail, lex.dur)
+              ~ Ns(tfi, knots = seq( 0, 20,  5)) * lex.Tr +
+                Ns(age, knots = seq(50, 80, 10)) * lex.Tr +
+                lex.Tr / allo + sex,
+            family = poisreg,
+              data = subset(St4, grepl("D", lex.Tr)))
> round(ci.exp(stD)[,1,drop=F],3)
```

	exp(Est.)
(Intercept)	0.000000e+00
Ns(tfi, knots = seq(0, 20, 5))1	2.069800e+01
Ns(tfi, knots = seq(0, 20, 5))2	2.756800e+01
Ns(tfi, knots = seq(0, 20, 5))3	7.291900e+01
Ns(tfi, knots = seq(0, 20, 5))4	5.860000e-01
lex.TrMac->D(oth)	0.000000e+00
lex.TrMic->D(CVD)	7.322000e+00
lex.TrMic->D(oth)	9.000000e-03
lex.TrNorm->D(CVD)	0.000000e+00
lex.TrNorm->D(oth)	1.350600e+01
Ns(age, knots = seq(50, 80, 10))1	7.378000e+00
Ns(age, knots = seq(50, 80, 10))2	2.669160e+02
Ns(age, knots = seq(50, 80, 10))3	6.845600e+01
sexM	1.440000e+00
Ns(tfi, knots = seq(0, 20, 5))1:lex.TrMac->D(oth)	5.334862e+72
Ns(tfi, knots = seq(0, 20, 5))2:lex.TrMac->D(oth)	3.913890e+51
Ns(tfi, knots = seq(0, 20, 5))3:lex.TrMac->D(oth)	1.417514e+141
Ns(tfi, knots = seq(0, 20, 5))4:lex.TrMac->D(oth)	2.902424e+30
Ns(tfi, knots = seq(0, 20, 5))1:lex.TrMic->D(CVD)	5.000000e-03
Ns(tfi, knots = seq(0, 20, 5))2:lex.TrMic->D(CVD)	7.600000e-02
Ns(tfi, knots = seq(0, 20, 5))3:lex.TrMic->D(CVD)	3.100000e-02
Ns(tfi, knots = seq(0, 20, 5))4:lex.TrMic->D(CVD)	1.690000e-01
Ns(tfi, knots = seq(0, 20, 5))1:lex.TrMic->D(oth)	2.015738e+03
Ns(tfi, knots = seq(0, 20, 5))2:lex.TrMic->D(oth)	8.679800e+01
Ns(tfi, knots = seq(0, 20, 5))3:lex.TrMic->D(oth)	3.163383e+07
Ns(tfi, knots = seq(0, 20, 5))4:lex.TrMic->D(oth)	3.372500e+01
Ns(tfi, knots = seq(0, 20, 5))1:lex.TrNorm->D(CVD)	4.469074e+04
Ns(tfi, knots = seq(0, 20, 5))2:lex.TrNorm->D(CVD)	5.744664e+04
Ns(tfi, knots = seq(0, 20, 5))3:lex.TrNorm->D(CVD)	1.297133e+11
Ns(tfi, knots = seq(0, 20, 5))4:lex.TrNorm->D(CVD)	1.000000e-03
Ns(tfi, knots = seq(0, 20, 5))1:lex.TrNorm->D(oth)	2.480000e-01
Ns(tfi, knots = seq(0, 20, 5))2:lex.TrNorm->D(oth)	1.960000e-01
Ns(tfi, knots = seq(0, 20, 5))3:lex.TrNorm->D(oth)	1.109700e+01
Ns(tfi, knots = seq(0, 20, 5))4:lex.TrNorm->D(oth)	2.950000e-01
lex.TrMac->D(oth):Ns(age, knots = seq(50, 80, 10))1	2.060000e-01
lex.TrMic->D(CVD):Ns(age, knots = seq(50, 80, 10))1	2.002000e+00
lex.TrMic->D(oth):Ns(age, knots = seq(50, 80, 10))1	2.490000e-01
lex.TrNorm->D(CVD):Ns(age, knots = seq(50, 80, 10))1	1.990000e+00
lex.TrNorm->D(oth):Ns(age, knots = seq(50, 80, 10))1	1.107000e+00
lex.TrMac->D(oth):Ns(age, knots = seq(50, 80, 10))2	1.100000e-02
lex.TrMic->D(CVD):Ns(age, knots = seq(50, 80, 10))2	6.593000e+00
lex.TrMic->D(oth):Ns(age, knots = seq(50, 80, 10))2	5.000000e-03
lex.TrNorm->D(CVD):Ns(age, knots = seq(50, 80, 10))2	1.000000e-03
lex.TrNorm->D(oth):Ns(age, knots = seq(50, 80, 10))2	1.200000e-02
lex.TrMac->D(oth):Ns(age, knots = seq(50, 80, 10))3	1.490000e-01

```
lex.TrMic->D(CVD):Ns(age, knots = seq(50, 80, 10))3   2.070000e-01
lex.TrMic->D(oth):Ns(age, knots = seq(50, 80, 10))3   1.650000e-01
lex.TrNorm->D(CVD):Ns(age, knots = seq(50, 80, 10))3  0.000000e+00
lex.TrNorm->D(oth):Ns(age, knots = seq(50, 80, 10))3  2.940000e-01
lex.TrMac->D(CVD):alloConv                            8.706000e+00
lex.TrMac->D(oth):alloConv                            6.270000e-01
lex.TrMic->D(CVD):alloConv                            1.688000e+00
lex.TrMic->D(oth):alloConv                            2.096000e+00
lex.TrNorm->D(CVD):alloConv                           2.054000e+00
lex.TrNorm->D(oth):alloConv                           1.800000e+00
```

So under the assumption that the sex-effect is the same for all 6 mortality rates in figure 3.2 the M/W rate ratio is 1.46.

But it is only rarely that we want to model different rates `out` of the same state, so use of `stack`(`.Lexis`) is seldom needed.