Practical Multistate Modeling with R and Epi::Lexis

SDCC
April 2023
http://bendixcarstensen.com/PMM
Version 18

Compiled Friday 31st March, 2023, 10:08
from: C:\Bendix\teach\MSbook\chapters/MSbook.tex

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

List of Figures . . . v
Preface . . . 1

1 Introduction **2**
1.1 A simple multistate example . . . 2
1.1.1 Time scales . . . 4
1.1.2 Transient and absorbing states . . . 4
1.2 Quantities of interest in multistate models . . . 4
1.3 Traditional approaches to rates . . . 5
1.4 Components of multistate models . . . 5
1.5 Terminology . . . 6

2 Probability models for multiple states **7**
2.1 Markov models . . . 7
2.1.1 Transition matrix . . . 7
2.1.2 Using the transition matrix . . . 9
2.1.3 Multiple transitions in the same time interval . . . 10
2.1.4 The analytic solution . . . 11
2.1.5 States by time . . . 13
2.2 Time varying rates: inhomogeneous Markov model . . . 14
2.3 Semi-Markov models . . . 15
2.4 Multiple time scales for rates . . . 15
2.5 Summary of rate models for transitions . . . 16

3 Multistate likelihood: rates **17**
3.1 Survival model . . . 17
3.1.1 Likelihood for a single rate . . . 17
3.1.2 Poisson likelihood . . . 19
3.1.3 Implications for analysis . . . 19
3.1.3.1 Time as covariate . . . 19
3.1.3.2 Likelihood, not model . . . 19
The offset . . . 20
Identity link model . . . 20
3.1.3.3 Real Poisson data . . . 20
3.2 Multistate model . . . 20
3.2.1 One transition . . . 20
3.2.1.1 Time varying rates . . . 20
3.2.1.2 Time as covariate (when) . . . 21
3.2.1.3 Time as response (how long) . . . 21
3.2.2 Competing risks . . . 21
3.2.2.1 Stacked data for joint likelihood . . . 22
3.2.2.2 A small illustration of stacking data . . . 22
3.2.3 Multiple transitions . . . 23
3.2.4 Total log-likelihood for a multistate model . . . 23

4 Lexis: data representation and analysis of rates **25**
4.1 Using the Lexis function . . . 25
4.1.1 Defining a Lexis object . . . 26
4.1.2 Printing a Lexis object . . . 26
4.2 Modeling mortality . . . 27
4.2.1 Effect of age at diagnosis . . . 27
4.2.2 Splitting follow up for analysis by current age . . . 28
4.3 Survival analysis . . . 31
4.4 Multiple states: cutting follow-up . . . 32
4.4.1 Cutting follow-up at one intermediate event . . . 32
4.4.1.1 Graphical representation of states and transitions . . . 35
Lexis version . . . 35
cohorttools version . . . 35
4.4.2 Cutting follow-up at multiple intermediate events . . . 36
4.5 Joint model for several transition rates . . . 38
4.5.1 Two time scales . . . 41
4.5.2 Proportional hazards? . . . 43
4.6 Recurrent multiple intermediate events . . . 45
4.7 Adding time-varying covariate information . . . 45
4.7.1 Adding clinical measurements: addCov.Lexis . . . 45
4.7.2 Adding drug exposure: addDrug.Lexis . . . 45
4.8 Recurrent events and time dependent covariates: Steno2 trial . . . 46
4.8.1 A Lexis object . . . 46
4.8.2 Definition of intermediate states . . . 47
4.8.2.1 Defining transition times: interval censoring . . . 50
4.8.3 Adding clinical measurements . . . 50
4.9 Summary of multistate data representation . . . 52
4.9.1 Prerequisites and construction of a Lexis object . . . 53
4.10 Summary of analysis of transition rates . . . 54

5 Recurrent events **55**
5.1 Example: Hospitalized hypoglycemia (HH) . . . 56
5.1.1 Data set of recurrences . . . 56
5.1.1.1 Removing presumed duplicated HH events . . . 58
5.1.2 Constructing a Lexis object . . . 58
5.1.2.1 Initial Lexis data frame . . . 58
5.1.2.2 Time since last HH as time scale . . . 61
5.1.3 Simple modeling of crude rates . . . 64
5.1.4 Proper analysis of HH rates . . . 66
5.1.4.1 Splitting the follow-up . . . 66
5.1.5 More realistic models for rates . . . 68
5.1.5.1 Choice of reference . . . 73
5.2 Random effects: frailty . . . 74
5.2.0.1 Cox-models . . . 75
5.2.0.2 Cox or Poisson model(s)? . . . 77
5.2.0.3 coxme: Cox model with mixed effects . . . 78
5.2.0.4 The other Cox model . . . 79

5.2.0.5 Individual random effects . . . 80
5.2.1 RR by no. of HH . . . 83
5.2.1.1 RRs between successive states . . . 84
5.3 Unknown history of events before entry . . . 86
5.3.1 Comparison with the full data . . . 88
5.4 Summary of recurrent events . . . 90

6 Competing risks in practice **92**
6.1 Example data . . . 93
6.2 Models for rates . . . 94
6.3 Cumulative rates and risks . . . 97
6.4 Confidence intervals for cumulative risks . . . 97
6.5 Joint models for several transitions . . . 98
6.6 Simulation based confidence intervals . . . 99
6.6.1 Keeping the bootstrap samples . . . 100
6.6.2 Rates . . . 101
6.6.3 Cumulative risks . . . 101
6.6.4 Stacked cumulative risks . . . 102
6.6.5 Sojourn times . . . 103
6.7 Aalen-Johansen approach . . . 103
6.8 Interpretation . . . 106

7 Estimation from multistate models **107**
7.1 Simulated multistate objects . . . 107
7.1.1 Simulation machinery . . . 108
7.1.1.1 Alternative simulation scheme . . . 108
7.2 Using simulated multistate data . . . 108
7.3 A worked example of using `simLexis` . . . 108
7.3.1 `Lexis` object for the `steno2` data frame . . . 108
7.3.2 Simulation of state probabilities . . . 116
7.3.3 Using the Steno2 population . . . 117
7.3.4 Age-stratified initial data . . . 121
7.3.5 Time spent in albuminuria states . . . 125
7.3.5.1 Using estimated probabilities . . . 127
7.3.5.2 Using simulated sojourn times . . . 128
Discrepancy to numerical integrals . . . 129
Time spent in absorbing states . . . 129
7.4 State probabilities from the Aalen-Johansen estimator in `survival` . . . 129
7.4.1 Relation to the simulation approach by `simLexis` . . . 131

8 Multistate packages **133**
8.1 The `msm` package . . . 133
8.1.1 Relationship between `Lexis` and `msm` . . . 133
8.1.2 Exact transition times? . . . 133
8.2 Data transformation from `Lexis` to `msm` . . . 133
8.3 Analysis of rates . . . 136
8.3.1 Follow-up analysis using `glm.Lexis` . . . 136

8.3.2 Markov analysis by msm . . . 137

9 Advice and dogma **140**
9.1 Practical advice for multistate analysis . . . 140
9.2 Dogma for multistate analysis . . . 141

10 Useful R functions and packages **142**
10.1 Lexis functions . . . 142
10.2 Modeling and reporting functions . . . 143
10.3 R Packages . . . 144

11 Appendices **145**
11.1 Comparing mcut and rcut . . . 145
11.1.1 mcutLexis: Cutting at dates of transitions, keeping history . . . 146
11.1.1.1 If state sequence matters . . . 146
11.1.1.2 If state sequence does not matter . . . 147
11.1.2 rcutLexis: Recurrent events — no memory . . . 147
11.2 Adding time-dependent variables . . . 151
11.2.1 addCov.Lexis . . . 151
11.2.1.1 Rationale . . . 151
11.2.1.2 The example data . . . 151
11.2.1.3 A Lexis object . . . 152
11.2.1.4 Factor or character lex.id? . . . 152
11.2.1.5 Clinical measurements . . . 153
11.2.1.6 Adding clinical data . . . 153
11.2.1.7 Exchanging split and add . . . 154
11.2.1.8 Filling the NAs . . . 156
11.2.2 addDrug.Lexis . . . 157
11.2.2.1 The help example . . . 157
11.2.2.2 A more realistic example with timing . . . 165
Follow-up data: DMlate . . . 165
Artificial prescription data . . . 166
Using addDrug . . . 167
1000 and 500 persons . . . 167
Fewer prescription records . . . 168
Fewer prescription types . . . 169
Too many records –coarse.Lexis . . . 169
Records to be kept . . . 170

References **170**

Index **172**

Back cover **177**

List of Figures

1.1 Simple multistate model . . . 2
1.2 Multistate model for beginning of insulin use and death . . . 3

2.1 State probabilities for `DM` and `Ins` from a time-homogenaous Markov model. 14

4.1 Two models for mortality in DM patients. . . . 30
4.2 Mortality of Danish diabetes patients as a function of diabetes duration. . . . 32
4.3 Parametric and non-parametric survival curves. . . . 33
4.4 Multistate model for beginning of insulin use and death . . . 35
4.5 `graphviz` version of multiple states graph. . . . 36
4.6 States and transitions for the `DMlate` data . . . 39
4.7 `graphviz` version of the multiple states. . . . 40
4.8 Mortality by current age and duration of diabetes for persons in states `DM` and `Ins`. . . . 42
4.9 Mortality by current age and duration of diabetes for persons in all states. . . . 44
4.10 Transitions between states of albuminuria and death in the Steno2 study. . . . 49

5.1 Entry and exit dates in the hypoglycemia study. . . . 57
5.2 Distribution of within-person gaps between successive HH events; shown in one-day bins. A person may contribute more than one gap in the histogram, but only persons with at least two recorded HH dates contribute. . . . 59
5.3 Transitions by recurrences of hypoglycemia. . . . 62
5.4 Rate-ratios of death and next HH relative to the `0.HH` state. . . . 67
5.5 Effects of age (left) and time since last HH (right) at diabetes duration 20 years. The horizontal dotted lines correspond to the rates for men aged 50 in the left panel, as illustrated by the blob. . . . 71
5.6 Effects of age, sex and time since last HH from a model with no interaction between HH number of HH and time since latest HH. . . . 74
5.7 Estimated frailties (on the log-rate-scale) for the Cox-models with duration respectively age as baseline timescale. . . . 81
5.8 Posterior frailties (log-rate scale) for the two Cox-models, plotted against the number of HH events for each person. . . . 82
5.9 Hazard ratios of next HH event relative to the rate of the first event, respectively previous event. . . . 86
5.10 Estimates from two different definitions of recurrence number. . . . 90

6.1 The transitions in the multistate model with and without follow-up extended after beginning of first drug exposure. . . . 95
6.2 Estimated rates of OAD, Ins and Dead from the `DM` state. . . . 96

6.3 Probabilities of being in the 4 different states as a function of time since diagnosis. . . . 98
6.4 Estimated rates (log-scale) from the DM state. . . . 102
6.5 Cumulative risks for the three types of events, with 95% bootstrap-based confidence intervals as shades. . . . 103
6.6 Probabilities of being in the 4 different states as a function of time since diagnosis. . . . 104
6.7 Cumulative risks for the three types of events based on parametric models, compared to the Nelson-Aalen estimator . . . 105

7.1 Transitions between albuminuria and death states in the Steno2 study. . . . 112
7.2 State probabilities for the two intervention groups, for populations of the same structure w.r.t. age as the original *total* Steno2 population. . . . 121
7.3 Predicted probabilities of being in each of the states for persons aged 45, 55 and 65 at entry . . . 124
7.4 Predicted probabilities of being in each of the states for persons aged 45, 55 and 65 at entry, separately for the two intervention groups. . . . 126
7.5 Aalen-Johansen estimator of state probabilities for the total Steno2 study population, disregarding intervention allocation. . . . 130
7.6 Aalen-Johansen estimator of the state probabilities for the two intervention groups . . . 132

8.1 Simple artificial example of a 5-state model. The numbers in the middle of each box is the risk time spent in each state (in years), and teh numbers at the bottom are the number of persons beginning, resp. ending follow-up in the state. The number of transitions and the empirical rates are given (per 1000 PY) on the transition arrows. . . . 135

11.1 States and transitions in the three different cases. . . . 150

. . . now input from `preface`

Preface

The first three words of the title of this book are meant literally. It has a particular emphasis on data-representation and -manipulation. Which also means that it is presented in terms of computer code for all analyses—in this case using R.

In order to analyze and report results from a multistate set-up you must have a proper overview of the raw data. That is provided by the `Lexis` machinery from the `Epi` package.

The other main focus is the outcome(s) of statistical analysis, i.e. the quantities that are desirable to derive from the data base, and how best to convey these to the audience.

The probability and statistical considerations in between the two are technicalities of lesser primary interest. But they are of crucial importance in getting the analysis right, and they should take the structure of the data into account. And hence they are carefully explained in relation to the structure of the basic data.

This is transmitted by worked examples that explain the rationale of analyses as well as the practical details from definition to final reporting.

Methodologically, the main focus will be on parametric models for intensity rates, as these are not only biologically credible but also the simplest approach to derivation of the quantities of interest.

. . . now input from `intro`

Chapter 1

Introduction

1.1 A simple multistate example

In figure 1.1 is an example where persons with diabetes (DM, diabetes mellitus) in Denmark have been followed from date of diagnosis of DM for initiation of insulin treatment and death in the period 1996–2009. Persons begin their follow-up either as persons with diabetes not on insulin (state `DM`) or as persons with DM currently treated with insulin (`Ins`).

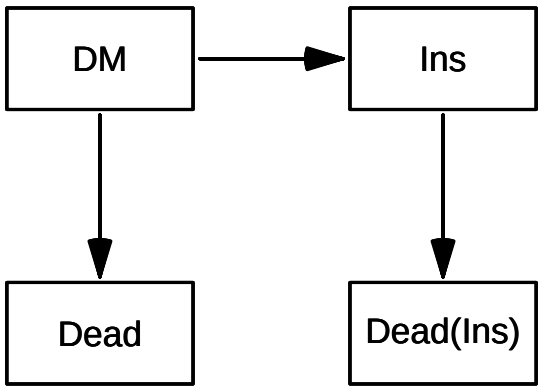


Figure 1.1: *Simple multistate model for beginning of insulin use and death among diabetes patients.* `../graph/intro-box1`

Those beginning without insulin can move to the insulin state some time during their follow-up. From either state persons can die; we keep track of whether persons were on insulin or not at the time of death by having two different death states. Some persons are still alive (in DM or in Ins) when follow-up is ended; we say that the follow-up of these persons is *censored* alive.

In figure 1.2 the model has been annotated with numbers derived from the database, showing:

- The total sojourn time (time spent) in state `DM` is 45,885.5 years
- The total sojourn time (time spent) in state `Ins` is 8,387.8 years
- 9,899 persons started the follow-up in the state `DM`, that is as persons with diabetes *not* using insulin. Out of these

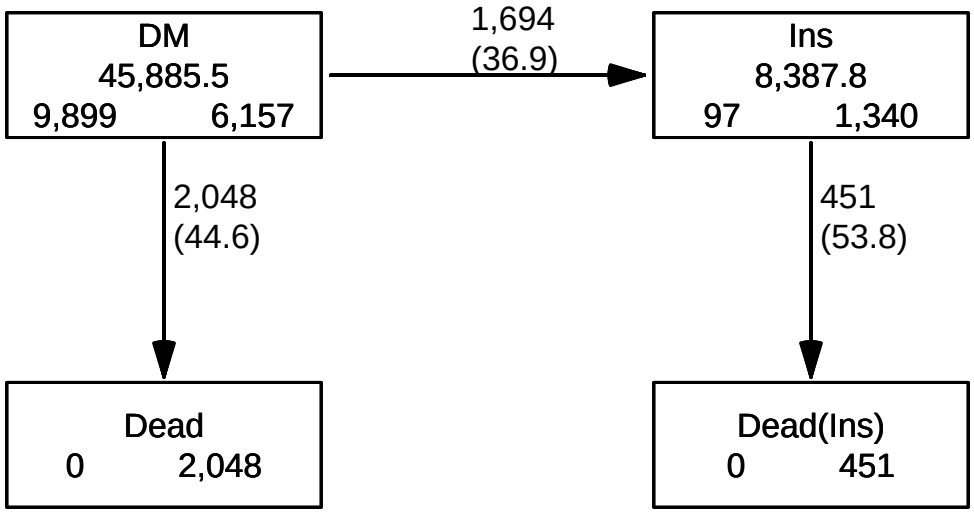


Figure 1.2: *Multistate model for beginning of insulin use and death among diabetes patients. The number in the center of each box is the total follow-up time. The numbers at the bottom of each state are the number of persons beginning, resp. ending their follow-up in the state. The numbers on the arrows are the number of transitions and in brackets the transition rates per 1000 person-years.* `../graph/intro-box2`

  - 2,048 died without ever using insulin—corresponding to a mortality rate of $2,048/45,885.5\text{PY} = 44.6$ per 1000 PY.
  - 1,694 went on to use insulin—corresponding to a rate of beginning insulin of $1,694/45,885.5\text{PY} = 36.9$ per 1000 PY.
  - 6,157 reached end of follow-up (2009-31-12) alive while not using insulin
- 97 persons were already using insulin at the inclusion in the study
- There were $97 + 1.694 = 1791$ ever on insulin. Out of these
  - 451 died on insulin—corresponding to a mortality rate of $451/8,387.8\text{PY} = 53.8$ per 1000 PY.
  - 1,340 reached end of follow-up (2009-31-12) alive while using insulin
- The mortality rates among persons on insulin is almost 20% higher than among those not on insulin (53.8 vs. 44.6 per 1000 PY).

The data base from which these numbers were derived is merely a set of *dates* for each person (some of which may be missing), namely:

- date of entry to the study
- date of first insulin use (which may be before or at the date of entry)
- date of death
- date of end of follow-up

So we see that we from information on 4 dates for each person can determine quite a number of characteristics of the follow-up of the DM population studied.

The information needed for a multistate model for each person is a set of pairs of (date in, date out, state); the date the person is first, respectively last seen in the state. We will also need to know the date where the person is last seen alive (the censoring data).In most practical clinical settings, the dates and corresponding states will also be the way data on persons are initially recorded.

1.1.1 Time scales

When modeling transition rates we may be interested in the effect of time-scales such as age and disease duration, which will require that the origins of these are recorded too, i.e. dates of birth, disease onset etc.

In general it is advisable to keep dates of all relevant events, it will facilitate calculation of other time-related quantities such as age and disease duration at given dates.

1.1.2 Transient and absorbing states

Two types of states are normally distinguished: transient and absorbing. Transient states are states from which it is possible to exit, absorbing states are states from which it is impossible to exit, typically death. In figure 1.2, the states `DM` and `Ins` are transient states, while `Dead` and `Dead(Ins)` are absorbing states.

1.2 Quantities of interest in multistate models

From the follow-up for the persons through the multiple states we can answer questions in three realms, quantities that all will be functions of one or more time scales:

rates, measured in time^{-1}; this is the same scale as the observed data

- What are the mortality rates among persons on insulin and among persons not on insulin?
- What is the rate of beginning insulin treatment?
- How do the two mortality rates compare (rate-ratio or rate difference)?

probabilities, dimensionless, but requires an *origin*:

- As a function of time since some origin:
  - What is the *survival* of diabetes patients since diagnosis (ignoring use of insulin)?
  - What is the *survival* of diabetes patients since start of insulin?
  - What is the *cumulative risk* of beginning insulin since the diagnosis of DM.
  - What is the *probability of ever visiting* the insulin state?

  Note that the origin may be defined arbitrarily; we could ask for *conditional* survival given survival to some fixed time point. This is also sometimes called landmarking.
- As a function of some time t:
  - What is the probability that a person who is in state `A` at time t is in state `B` at a later time t_x for some fixed $t_x (> t)$.
    An example would be asking what the probability of survival till, say, 5 years (t_x) after diagnosis is, given that you already survived 1, 2,... years (t) after diagnosis

times, measured in units of time, requiring an *interval* (origin and endpoint)

- What is the *expected lifetime* after diagnosis of diabetes
- What is the *expected lifetime* spent without insulin?
- What is the *expected lifetime* spent using insulin?
- How much *lifetime* is *lost* between persons not on insulin and persons currently on insulin.
- How much time will a person spend in, say, state `A`, between times t_e and t_x. This is sometimes called the *sojourn time* in state `A`.

Mathematically speaking, the first group (the rates) is at the same level as the basic observations—we observe data in the form of events and risk time, interpret able as rates (events per time, with dimension time^{-1}). The second group (probabilities) are (transformations of the) integrals of the rates w.r.t. time, so of dimension time×time^{-1}, i.e. dimensionless. The third group of measures are integrals of the probabilities w.r.t. time, so of dimension time.

In practice we will also want to know how these quantities vary by age, sex calendar time etc. One way of doing this is modeling the rates as functions of these covariates. Note that so far we have not said anything about *estimation* of the rates, let alone the other quantities mentioned.

The rest of this book will show how to handle and analyze multistate data, in order to be able to answer questions as those above—after posing them in a precise (that is, answerable) form. Thus, as in most realms of statistics and data analysis it is not about what questions you would *like* to answer, but about the questions that are *possible* to answer from the available data. Listing the questions that are possible to answer tends to fool you to think that these are the only relevant ones. We shall however also touch on questions that *cannot* be answered in this framework.

The basis for answering the questions will be knowledge of the transition rates. So we should estimate them from data as accurately as possible. In real applications we will of course allow rates to depend on sex, age at diagnosis, current age etc., and thus let the derived measures depend on these factors as well.

1.3 Traditional approaches to rates

The prevailing approach to modeling of multistate event history data is via semi-parametric (some call it non-parametric) models for the transition rates. Which in most practical instances boils down to fitting Cox-models for transition rates, and using some variant of the Aalen-Johansen estimator to compute state occupancy probabilities.

Even if these are models for the rates, in practice only estimates of cumulative risk or survival are derived together with hazard ratios, but hardly ever are actual rates presented, see *e.g.* [7]. This is largely a consequence of the model structure that does not allow extraction of the baseline hazard—some may formulate it by saying that the hazard is not part of the model at all.

Royston & Lambert [10] and Crowther & Lambert [4] have reasonably argued that a more natural and biologically plausible model for transition rates would be a model that assumes some smooth shape of the rates as function of time.

Both of these approaches mainly use one timescale to describe occurrence rates, presumably because it is a requirement for the Aalen-Johansen machinery and its cousins to work. Incidentally this has also had the consequence that most example datasets only provide information of one time scale of interest. But in most practical applications the single time scale seems to be an over-simplification; I would personally have a hard time convincing my diabetes clinical colleagues that mortality among diabetes patients depend only on current age and not on diabetes duration, or vice versa. They would likely also maintain that age *at* diagnosis influence the mortality rates [5].

1.4 Components of multistate models

The machinery chosen for estimation of rates is alien to the discussion of what measures are of interest in description of the multistate model. It is therefore useful to separate the different components of modeling and reporting of measures of interest.

- model specification: what are the states of interest and relevant transitions between them.
- rate specification: what are the relevant covariates in the models for the rates, particularly what time scales to include in the description of the rates.
- specification of measures of interest: rates, rate-ratios, survival, cumulative risks, sojourn times, lifetime lost.
- practical derivation of the desired quantities from the estimated rates.

The first two points are prerequisites for constructing a data frame representing the follow-up through the multiple states, a so-called `Lexis` object.

The last point is in principle just an application of standard probability theory for stochastic processes using the estimated transition rates as building blocks to construct the quantities of interest. This is occasionally termed the "plug in" approach; estimates of the rates are plugged into the formulae linking the rates to the quantity of interest.

There are statistical methods around that do not rely on estimating rates.

1.5 Terminology

In the survival, multistate and demographic modeling literature there is a plethora of overlapping terms that mean the same, and also instances of the same term used about different concepts. Here are some groups of synonyms, the first one mentioned in each group being the one (mainly) used in this book.

cumulative intensity, integrated intensity, cumulative rate. Note that this is neither a probability nor a rate. Dimensionless.

cumulative risk, probability, risk; a probability. Dimensionless.

event, outcome—indicator of transition to a new state; the variable assumes only a finite number of values (namely the names of states *to* which transitions are considered events). Sometimes it is a count, occasionally restricted to $0, 1$.

follow-up, history, observation, outcome, response—a bivariate structure consisting of (event indicator, risk time).

interaction, non-proportional hazards, see below under "time scale interaction".

rate, intensity, transition intensity, hazard, hazard rate, occurrence rate, mortality rate, morbidity rate, incidence density. Measured in units of time^{-1}.

rate ratio, hazard ratio. Ratio of two rates. Sometimes wrongly termed relative risk or risk ratio. Dimensionless.

risk ratio, relative risk. Ratio of two probabilities. Dimensionless.

risk time, time, exposure time, follow-up time, exposure: time since a person entered observation—measured in units of time. It is always the *difference* between two points on a(ny) time scale. Measured in units of time.

sojourn time, time spent in a state, state time, expected life time in a state. Measured in units of time.

state probability, state occupancy probability Dimensionless.

time scale, time. Measured in units of time *since* some origin (such as date of birth and diagnosis, 1900-1-1, ...) Measured in units of time, an origin is however required.

time scale interaction, non-proportional hazards, violation of proportionality assumption, interaction between a time scale and some other variable(s).

transition probability dimensionless, a probability.

... now input from `probs`

Chapter 2

Probability models for multiple states

This section requires a basic knowledge of matrix algebra.

There is a large mathematical theory about multistate models for transitions as shown in figure 1.1, though mostly for models with constant transition rates—Markov models. In the biomedical literature the term "Markov model" is frequently used as a term for just any multistate models. Here we stick to the more formal definition of a Markov model.

2.1 Markov models

A Markov[1] model is a multistate model where the transition probabilities only depend on the current state, not on the history of the person and neither on how long time a person has been in a given state. This implies that the transition rates are constant over time. Probabilists call this a *homogeneous* Markov model or *time-homogeneous* Markov model.

If you can leave a state, the state is called a *transient* state, if you cannot, it is called an *absorbing* state. Absorbing states would normally be states like "Dead" or "Dead from cancer", but if we were only interested in whether a person had a diagnosis of cardiovascular disease (CVD) and we had no interest in what happened after that, we might define "CVD" as an absorbing state, meaning "CVD and whatever follows".

2.1.1 Transition matrix

For a multistate model we can define the transition matrix (really a matrix with functions as entries). Rows and columns are both labeled by the possible states. The entry in row labeled `A` and column labeled `B` is the probability of being in state `A` at time s and in state `B` at time t, more precisely:

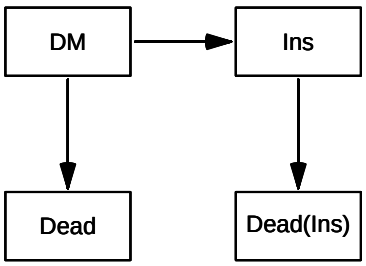
$$P_{\texttt{AB}}(s,t) = \mathrm{P}\{\text{state } \texttt{B} \text{ at time } t \mid \text{state } \texttt{A} \text{ at time } s\}$$

Loosely speaking, the probability of moving from `A` to `B` sometime between s and t, but note there is no requirement only to consider moves *directly* from `A` to `B`. In a time-homogeneous Markov model $P_{\texttt{AB}}(s,t)$ only depend on s and t through $t-s$.

For the simple model shown in figure 1.1 the transition matrix would (at first glance) be a 4×4 matrix (we are omitting the dependence of s and t for notational convenience):

[1] After the Russian mathematician Andrey Andreyevich Markov (1856–1922)

	to			
from	DM	Ins	Dead	Dead(Ins)
DM	$1 - p_{DI} - p_{DD}$	p_{DI}	p_{DD}	0
Ins	0	$1 - p_{ID}$	0	p_{ID}
Dead	0	0	1	0
Dead(Ins)	0	0	0	1



where the rows represent states *from* which transitions occur and the columns represent states *to* which transitions occur.

This transition matrix has a 0 entry for the transition DM → Dead(Ins) which actually *can* occur in a given interval of time, namely by first DM → Ins and then Ins → Dead(Ins). But if the time interval is sufficiently small the matrix above may be a decent approximation, because the probability of two transitions in a single very small interval will be ignorably small.

If we assume a Markov model, where the transition *rates* are constant over time, the transition probabilities would only depend on the difference between the two times, $t - s$. If we take the rates shown in figure 1.2 as constant, the one-month transition probabilities will be very close to the rates multiplied by 1/12 year, so the one-month transition matrix will be (using: 0.0369/12=0.003076; 0.0446/12=0.003719; 0.0538/12=0.004481, adjusting the diagonal so that the row sums of the matrix are 1):

	to			
from	DM	Ins	Dead	Dead(Ins)
DM	0.993205	0.003076	0.003719	0
Ins	0	0.995519	0	0.004481
Dead	0	0	1	0
Dead(Ins)	0	0	0	1

This matrix tells that for a person with diabetes without insulin use after one month the probability of being in the same state is 99.32%, and the probability of being on insulin is 0.31% and the probability of being dead is 0.37%.

During the next month, the 99.32% of persons in the DM state will move to states Dead and Ins with probabilities 0.31% and 0.37%, and the 0.31% in Ins at the end of the first month will move to Dead(Ins) with probability 0.45%.

These operations can be recognized as matrix multiplications, where we start with the vector of state occupancy probabilities at the beginning of the first month as a 1×4 matrix:

```
> (p0 <- c(DM=1, Ins=0, Dead=0, "Dead(Ins)"=0))
      DM       Ins      Dead Dead(Ins)
       1         0         0         0
```

This just means that everyone is in the state DM at time 0. Formally it is the probability distribution of states at time 0, the sum of entries in such vectors will always be 1.

The transition matrix is defined by the transition probabilities and the probabilities of remaining in a state as 1 minus the sum of the probabilities of leaving the state:

```
> Tm <- matrix(0, 4, 4)
> rownames(Tm) <- colnames(Tm) <- names(p0)
> Tm["DM","Ins"] <- 0.003076
> Tm["DM","Dead"] <- 0.003719
> Tm["Ins","Dead(Ins)"] <- 0.004481
> diag(Tm) <- 1 - apply(Tm, 1, sum)
> Tm
```

probs

```
               DM      Ins     Dead Dead(Ins)
DM       0.993205 0.003076 0.003719  0.000000
Ins      0.000000 0.995519 0.000000  0.004481
Dead     0.000000 0.000000 1.000000  0.000000
Dead(Ins) 0.000000 0.000000 0.000000  1.000000
```

CODE EXPLAINED: `Tm` is first defined as a 4 by 4 matrix of 0s, and assigned the state names both for row and columns. The transition probabilities are then assigned to the relevant entries in `Tm`. Finally `apply` is used so compute the row sums of `Tm`, and the diagonal is then filled filled with 1 minus the row-sums, so that row-sums of the resuting matrix `Tm` are all 1.

2.1.2 Using the transition matrix

We can now compute the state occupancy probabilities after one month by matrix multiplication—this is when you should consult your knowledge of how to compute the matrix product of the 1×4 matrix `p0` with the 4×4 matrix `Tm` to form the vector (really a 1×4 matrix) of probabilities of being in each of the four states after one month. In R matrix multiplication is `%*%`:

```
> (p1 <- p0 %*% Tm)
          DM      Ins     Dead Dead(Ins)
[1,] 0.993205 0.003076 0.003719         0
```

These are the probabilities of being in each of the states after one month, assuming you start in state `DM`. You will recognize `p1` as the first row of `Tm`.

The handy thing is that we can compute the state occupancy probabilities after two, three etc. months the same way:

```
> (p2 <- p1 %*% Tm)
            DM          Ins        Dead    Dead(Ins)
[1,] 0.9864562 0.006117315 0.007412729 1.378356e-05
> (p3 <- p2 %*% Tm)
            DM          Ins       Dead    Dead(Ins)
[1,] 0.9797532 0.009124243 0.01108136 4.119524e-05
```

We can see that after two months there is a positive, albeit small, probability of being dead after initiating insulin. That was not the case after one month, which is not quite true, it was merely a consequence of the assumption that two transitions do not occur during a single one month period.

If we want the probabilities at 60 months (=5 years) we can use a loop:

```
> px <- p0
> for(m in 1:60) px <- px %*% Tm
> px
            DM       Ins      Dead Dead(Ins)
[1,] 0.6642535 0.1323121 0.1837588 0.0196756
```

Here we see that the 5-year transition probability from `DM` to `Dead(Ins)` is about 2% and from `DM` to `Dead` about 18%.

The 1-month transition probabilities are very well approximated by the rates multiplied by the length of the interval, *e.g.* $\lambda\ell$, but formally the probability of remaining in the `DM` state for a duration of ℓ is $\exp(-(\lambda+\mu)\ell)$, the probability of being in the state `Dead` after a period of ℓ is $\lambda/(\lambda+\mu) \times (1-\exp(-(\lambda+\mu)\ell))$ and the probability of being in the state `Ins` is $\mu/(\lambda+\mu) \times (1-\exp(-(\lambda+\mu)\ell))$. We therefore use these formulae to populate the matrix of 1 month transition probabilities, starting with the rates:

```
> lambda <- 1694 / 45885.5
>    muD <- 2048 / 45885.5
>    muI <-  451 /  8387.8
> SDM <- exp( -(lambda + muD) / 12)
> SIn <- exp( -          muI  / 12)
> pDI <- lambda / (lambda + muD) * (1-SDM)
> pDD <-    muD / (lambda + muD) * (1-SDM)
> pID <- 1 - SIn
```

CODE EXPLAINED: We first compute the rates (per 1 PY) from the numbers in figure 1.2, then the 1 month survival probabilities for the two transient states, and finally the 1 month transition probabilities from the transient states.

If we use these numbers in the matrix `Tm` instead we get a slightly different transition matrix:

```
> Tm["DM","Ins"] <- pDI
> Tm["DM","Dead"] <- pDD
> Tm["Ins","Dead(Ins)"] <- pID
> diag(Tm) <- 0
> diag(Tm) <- 1 - apply(Tm, 1, sum)
> Tm
                DM         Ins        Dead   Dead(Ins)
DM        0.9932271 0.003066068 0.003706793 0.000000000
Ins       0.0000000 0.995529309 0.000000000 0.004470691
Dead      0.0000000 0.000000000 1.000000000 0.000000000
Dead(Ins) 0.0000000 0.000000000 0.000000000 1.000000000
```

We can then inspect how the state occupancy probabilities at 60 months are influenced by this different calculation:

```
> pm <- p0
> for(i in 1:60) pm <- pm %*% Tm
> round(cbind(appr = as.vector(px),
+             theor = as.vector(pm),
+              diff = as.vector(pm-px)), 5)
        appr   theor     diff
[1,] 0.66425 0.66514  0.00089
[2,] 0.13231 0.13201 -0.00030
[3,] 0.18376 0.18327 -0.00049
[4,] 0.01968 0.01958 -0.00010
```

We see that the difference between the two approaches is negligible; less than 1/1000 on the probability scale, so for small intervals, or more precisely for small transition probabilities we get a reasonable approximation.

2.1.3 Multiple transitions in the same time interval

In the above, we have made the formally incorrect assumption that a person cannot within the same month both move to the `Ins` state and then further to the `Dead(Ins)` state. This is most likely a reasonable approximation for a 1 month interval, but we can empirically investigate how other interval lengths perform relative to this.

First we do the calculations by 3-month intervals—stored in `p3`:

```
> SDM <- exp( -(lambda + muD) / 4)
> SIn <- exp( -          muI  / 4)
> pDI <- lambda / (lambda + muD) * (1-SDM)
> pDD <-    muD / (lambda + muD) * (1-SDM)
> pID <- 1 - SIn
> T3 <- Tm * 0
> T3["DM","Ins"] <- pDI
> T3["DM","Dead"] <- pDD
> T3["Ins","Dead(Ins)"] <- pID
> diag(T3) <- 1 - apply(T3, 1, sum)
> p3 <- p0
> for(i in 1:(5*3)) p3 <- p3 %*% T3
```

Then the same by 1-week intervals (1/52 year)—stored in `pw`:

probs

```
> SDM <- exp( -(lambda + muD) / 52)
> SIn <- exp( -          muI  / 52)
> pDI <- lambda / (lambda + muD) * (1-SDM)
> pDD <-    muD / (lambda + muD) * (1-SDM)
> pID <- 1 - SIn
> Tw <- Tm * 0
> Tw["DM","Ins"] <- pDI
> Tw["DM","Dead"] <- pDD
> Tw["Ins","Dead(Ins)"] <- pID
> diag(Tw) <- 1 - apply(Tw, 1, sum)
> pw <- p0
> for(i in 1:(5*52)) pw <- pw %*% Tw
```

Finally we do the same by 1-day intervals (1/365 year)—stored in `pd`:

```
> SDM <- exp( -(lambda + muD) / 365)
> SIn <- exp( -          muI  / 365)
> pDI <- lambda / (lambda + muD) * (1-SDM)
> pDD <-    muD / (lambda + muD) * (1-SDM)
> pID <- 1 - SIn
> Td <- Tm * 0
> Td["DM","Ins"] <- pDI
> Td["DM","Dead"] <- pDD
> Td["Ins","Dead(Ins)"] <- pID
> diag(Td) <- 1 - apply(Td, 1, sum)
> pd <- p0
> for(i in 1:(5*365)) pd <- pd %*% Td
```

We have now made the calculations using intervals of length 3 and 1 months, 1 week and 1 day. Clearly, the calculation using 1 day intervals must be the most accurate because the assumption about no two transitions during 1 day is the least restrictive.

We can now compare the estimated 5-year state probabilities from the different approximations; as well as the difference to the 1-day approximation:

```
> five <- cbind("3mon" = as.vector(p3),
+               "1mon" = as.vector(pm),
+               "week" = as.vector(pw),
+                "day" = as.vector(pd),
+                "3-d" = as.vector(p3-pd),
+                "m-d" = as.vector(pm-pd),
+                "w-d" = as.vector(pw-pd))
> rownames(five) <- names(p0)
> round(five, 6)
              3mon     1mon     week      day       3-d       m-d      w-d
DM        0.736522 0.665142 0.665142 0.665142  0.071380  0.000000  0.0e+00
Ins       0.108193 0.132011 0.131783 0.131725 -0.023532  0.000286  5.8e-05
Dead      0.144202 0.183268 0.183268 0.183268 -0.039066  0.000000  0.0e+00
Dead(Ins) 0.011083 0.019579 0.019806 0.019865 -0.008782 -0.000286 -5.8e-05
```

So we see that the approximation when using 3 month is really bad, whereas the difference between the results from using a month, a week or a day as interval are quite small. Inspection of the transition matrices `Tm`, `Tw` and `Td` would indicate that it is wise to make the intervals so small that we keep transition probabilities *to* transient states to 1/1000 or less.

2.1.4 The analytic solution

Of course there is a mathematical solution corresponding to making the intervals smaller and smaller *ad infinitum*. This is based on the *intensity* matrix that has the rates (transition intensities) in the off-diagonal entries and the diagonal adjusted so that the row-sums are 0:

```
> Q <- Tm * 0
> Q["DM","Ins"] <- lambda
> Q["DM","Dead"] <- muD
> Q["Ins","Dead(Ins)"] <- muI
> diag(Q) <- -apply(Q, 1, sum)
> Q
```

```
                  DM         Ins       Dead  Dead(Ins)
DM        -0.08155082  0.03691798 0.04463284 0.00000000
Ins        0.00000000 -0.05376857 0.00000000 0.05376857
Dead       0.00000000  0.00000000 0.00000000 0.00000000
Dead(Ins)  0.00000000  0.00000000 0.00000000 0.00000000
```

The mathematically accurate transition probability matrix for say 5 years is then $\text{Exp}(Q \times 5)$ years, where Exp here is the matrix exponential function[2]. Of course it does not make any sense to have a scaled quantity such as the rates in the matrix `Q`, the entries we put there were really the 1-year integrated intensities (dimensionless), and the expression $Q \times 5$ is the matrix of 5-year integrated intensities. As always it is important to keep track of the units in which quantities are measured.

Calculation of the matrix exponential is implemented in the function `MatrixExp` in the `msm` package. .

The 5-year transition matrix based on 1-day transition probabilities with an assumption of no two transitions in one day is:

```
> T5 <- Td
> for(i in 2:(365*5)) T5 <- T5 %*% Td
> T5
                 DM       Ins      Dead  Dead(Ins)
DM        0.6651424 0.1317249 0.1832679 0.01986481
Ins       0.0000000 0.7642634 0.0000000 0.23573664
Dead      0.0000000 0.0000000 1.0000000 0.00000000
Dead(Ins) 0.0000000 0.0000000 0.0000000 1.00000000
```

CODE EXPLAINED: The transition matrix for a period of two days is merely the matrix product of `Td` by itself. For three days we should multiply by `Td` once more. And for 5 years a total of 365×5 times.

The mathematically accurate calculation is:

```
> library(msm)
> (Q5 <- MatrixExp(Q * 5))
                 DM       Ins      Dead  Dead(Ins)
DM        0.6651424 0.1317152 0.1832679 0.01987451
Ins       0.0000000 0.7642634 0.0000000 0.23573664
Dead      0.0000000 0.0000000 1.0000000 0.00000000
Dead(Ins) 0.0000000 0.0000000 0.0000000 1.00000000
```

```
> Q5 - T5
```

```
                   DM           Ins          Dead    Dead(Ins)
DM        -1.44329e-15 -9.702158e-06 1.720846e-15 9.702158e-06
Ins        0.00000e+00 -7.083223e-14 0.000000e+00 7.152612e-14
Dead       0.00000e+00  0.000000e+00 0.000000e+00 0.000000e+00
Dead(Ins)  0.00000e+00  0.000000e+00 0.000000e+00 0.000000e+00
```

So we see that there is no practically noticeable difference between the two transition matrices.

Finally note that the first row of `Q5` is practically identical the 5-year state probabilities computed using the 1-day transition probabilities:

```
> cbind(Q = Q5[1,], pD = as.vector(pd))
                  Q         pD
DM        0.66514243 0.66514243
Ins       0.13171520 0.13172491
Dead      0.18326785 0.18326785
Dead(Ins) 0.01987451 0.01986481
```

[2]For a quadratic matrix A the matrix exponential of A, $\text{Exp}(A)$, is defined as the infinite sum $\sum_{i=0}^{\infty} A^i/i! = I + A + AA/2! + AAA/3! + \cdots$

probs

2.1.5 States by time

In the loops above we computed the probability of being in each state at a certain time, specifically at 5 years after entry. We can redo the calculations while at the same time collecting the state probabilities and then plot the state probabilities as a function of time (we make the calculation to 10 years this time):

```
> pt <- NArray(list(month = 0:120,
+                   state = names(p0)))
> pt["0",] <- p0
> for(i in 1:120) pt[i+1,] <- pt[i,] %*% Tm
> pt[1:5,]
     state
month        DM         Ins        Dead    Dead(Ins)
    0 1.0000000 0.000000000 0.000000000 0.000000e+00
    1 0.9932271 0.003066068 0.003706793 0.000000e+00
    2 0.9865001 0.006097663 0.007388481 1.370744e-05
    3 0.9798187 0.009095079 0.011045233 4.096821e-05
    4 0.9731825 0.012058609 0.014677218 8.162949e-05
```

CODE EXPLAINED: `pt` will be an array of dimension 121 by 4, where the rows refer to the times 0,1,2,...,120 months, and each column represents one of the 4 states. The first row (labeled "0") is assigned the initial state distribution (1 in `DM`, 0 in the rest), and the remaining are computed from the previous distribution by multiplying by the transition matrix.

We can now plot these 4 probabilities as functions of time by stacking them—remember at any time after entry, the sum of these 4 probabilities is 1:

```
> par(mar=c(3,3,1,2))
> m2p <- mat2pol(pt, perm = c(1,2,4,3), x=0:120 / 12,
+                col = c(cl <- c("forestgreen", "orange"),
+                        adjustcolor(cl[2:1], 0.5)),
+               xaxs = "i", xlab = "Time since DM (years)",
+               yaxs = "i", ylab = "State probability")
> lines(0:120/12, m2p[,3], lwd=2)
> mid <- function(x) x[-1] - diff(x)/2
> text(rep(9, 4),
+      mid(m2p[9*12,]),
+      colnames(m2p)[-1],
+      col = gray(c(1,1,0,0)))
> axis(side = 2, at = 0:10 / 10, labels = NA, tcl=-0.3)
> axis(side = 4, at = 0:10 / 10)
> axis(side = 4, at = 0:20 / 20, labels = NA, tcl=-0.3)
> axis(side = 4, at = 0:100/100, labels = NA, tcl=-0.2)
> axis(side = 4, pos = 5, col = "white", at = 0:10 / 10, labels = NA)
> axis(side = 4, pos = 5, col = "white", at = 0:20 / 20, labels = NA, tcl=-0.3)
> axis(side = 4, pos = 5, col = "white", at = 0:100/100, labels = NA, tcl=-0.2)
```

CODE EXPLAINED: `mat2pol` (`mat`rix `2` `pol`ygons) takes the matrix of state probabilities (`pt`) as input and show them as stacked coloured polygons, ordered as indicated by `perm=`. The `lines` draws the survival curve in black (3rd column of the `m2p`; where `m2p` is the result from `met2pol`).

Finally, the midpoint of the polygons at 9 years is computed (via the ad-hoc function `mid`) and the state names put on the plot.

In this simple instance with constant rates we could have made the calculations using the facility in `MatrixExp` to compute the transition matrix from time 0 to a range of times, and then take the first row as representing the state probabilities of a person beginning in the state `DM`:

```
> qm <- MatrixExp(Q, 0:120/12)
> str(qm)
 num [1:4, 1:4, 1:121] 1 0 0 0 0 1 0 0 0 0 ...
> qm <- t(qm[1,,])
> str(qm)
 num [1:121, 1:4] 1 0.993 0.987 0.98 0.973 ...
```

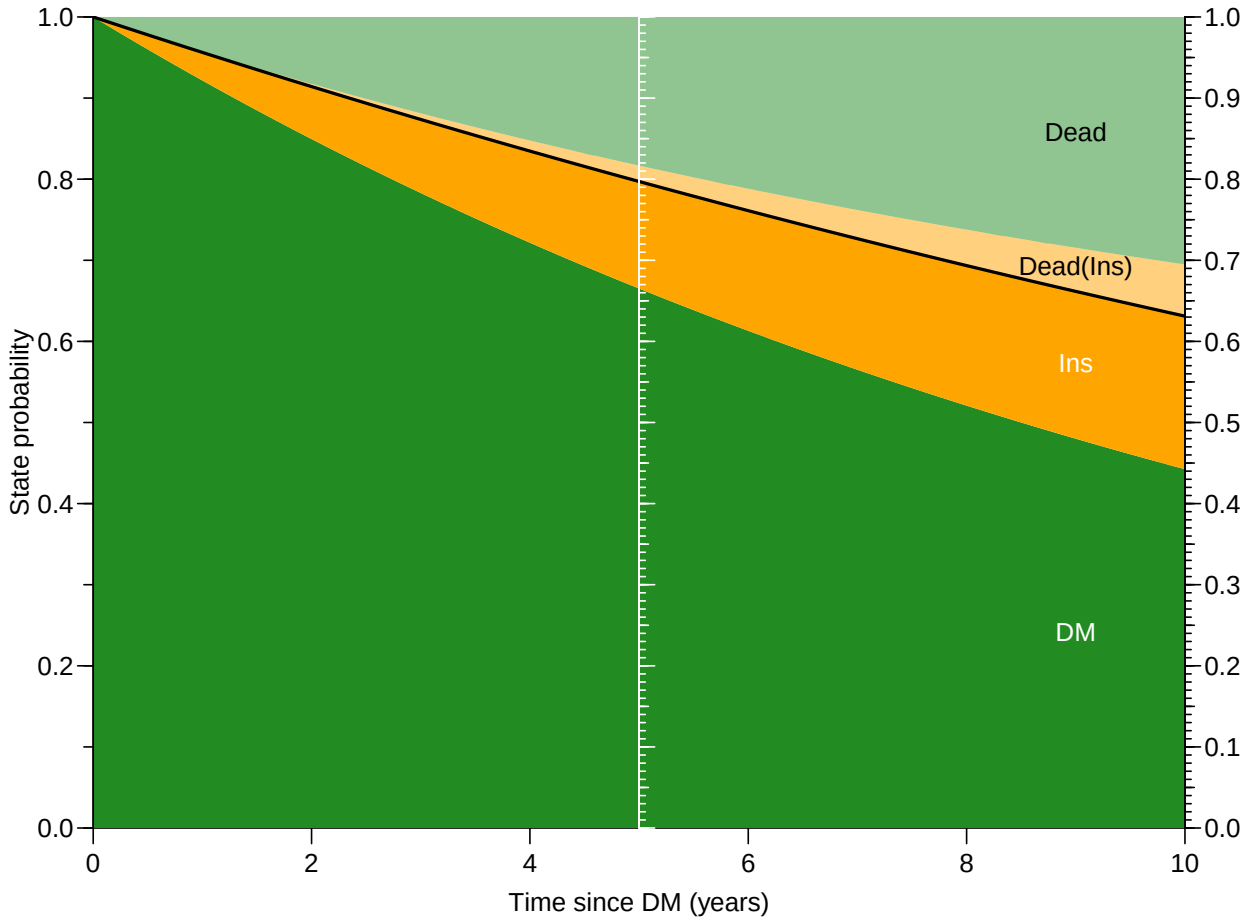


Figure 2.1: *State probabilities for* `DM` *and* `Ins` *from a time-homogenaous Markov model. This is the model shown in figure 1.2 with constant transition rates between states. The green area is the* `DM` *state, the orange is the* `Ins` *state, and the corresponding pale areas are the two* `Dead` *states as annotated. The vertical extent of each color at a given time represents the state probability at that time, conditional on being in the* `DM` *(green) state at time 0 (which is the date of DM diagnosis).*

Using the scale on the r.h.s. of the plot we can for example see that some 25% of patients have started insulin during the first 10 years $(69 - -44\%)$*, and that 31% have died without using insulin* $(100 - 69\%)$*. Using the white scale located at 5 years, we can see that some 15% started insulin during the first 5 years.* `../graph/probs-stDMIns`

CODE EXPLAINED: We supply a 4×4 intensity matrix as the first argument to `MatrixExp` and 121 time points from 0 to 10 years in 1-month intervals as the second. The results will be a $4 \times 4 \times 121$ array with each of the 121 slices equal to the transition matrix from time 0 to the corresponding time.

Taking the first row of each of these matrices (`qm[1,,]`) is the same as pre-multiplying with the state distribution at time 0, (1,0,0,0). Therefore, the first row of `qm` represents the same state probabilities as `pt` that we computed using the 1-day approximation above.

2.2 Time varying rates: inhomogeneous Markov model

The calculations above are easy to program because we have the same transition matrix for every month (*i.e.* time interval)—it is the same transition matrix every month. This is also the background for the

applicability of the matrix exponential. If rates were not constant, the matrix exponential machinery would not work.

But the transition matrix need not be the same for all months, we could have different transition matrices for each month; as long as all the transition rates are constrained to vary along the *same* time scale, we can be sure that all persons in any state at any given time have the same transition rates out of the state. Or more simply put, we can be sure that the persons in any one state at any one time are identical w.r.t. transition rates.

A Markov model (or more precisely, a Markov process) with transition intensities varying by time is called a (time-)*in*homogeneous Markov model.

In the example above we would have transition rates to depend in some functional form on time since entry, in this case time since diagnosis, in other words on duration of diabetes.

A representation of the time-varying transition rates would then have to be in a form of an array of dimensions states×states×times, in the case of 1-month transition probabilities over 5 years of dimension $4 \times 4 \times 120$, and the calculation would change from:

```
> for(i in 1:120) pt[i+1,] <- pt[i,] %*% Tm
```

to:

```
> for(i in 1:120) pt[i+1,] <- pt[i,] %*% Tm[,,i]
```

So the matrix machinery would only change slightly, but there would be considerably more effort defining `Tm`; in the Markov case it is a 4×4 matrix, in the latter case it is a $4 \times 4 \times 120$ array, requiring knowledge of the transition probabilities for 120 different intervals. The core assumption here is that *all* the transition rates vary along the *same* time scale; in this example it would be time since DM diagnosis.

2.3 Semi-Markov models

A semi-Markov model is a multistate model where the transition intensity varies by time since entry to each state.

This type of model will not lend itself to the type of calculations by simple matrix algebra as above. This is because "time" means different things in different states; in state `DM` it is time since DM diagnosis, in state `Ins` it is time since beginning of insulin therapy. If we in the state `Ins` refer to time since DM diagnosis, we will be facing the fact the persons at the same time since DM will have different times since beginning of insulin therapy and hence different transition rates to `Dead`. And in the `DM` state we cannot refer to time since beginning of insulin therapy.

A calculation where we successively multiply the state probability vector with the current transition probability matrix rests on the assumption that all persons in any given state at a given time have the same transition probabilities to other states, so the semi-Markov assumption violates this. Thus the calculation of state probabilities in semi-Markov models is not possible with the matrix machinery outlined above.

The semi-Markov model is also occasionally called the "clock-back" model—time starts at 0 at each transition to a new state.

2.4 Multiple time scales for rates

Suppose transition rates out of a state depend on another time scale such as age, *as well* as time since entry to the state. If persons have arrived at different ages, the state will at any given age comprise persons with different time in the state and hence with different transition probabilities. Conversely, at a given time after entry the state will comprise persons at different ages and hence with different transition probabilities. So facing the same problems as in the semi-Markov case.

But this type of model is really the most realistic for transition rates; one time scale throughout for all transition rates from all states is really not a very realistic assumption in real world applications. With the possible exception of calculation of classical demographic measures using age as the only time scale.

If we want to entertain this type of models (or semi-Markov models) for transition rates and compute state probabilities we must resort to simulation: Begin with one person in the initial state, simulate a time and state of transition using the rates out of the initial state. Then, for the time and the state the persons

arrived in, simulate the next transition etc. That gives transition times and -states for this person. If this is repeated for, say, 1000 persons, we will have a simulated dataset where we know who is in what state at any time, and the relevant probabilities can then be computed as the empirical fractions of the initial 1000 persons known to be in each state at a given time.

This machinery is frequently referred to as micro-simulation; we will return to this in chapter **??**.

2.5 Summary of rate models for transitions

We have four levels of complexity in models for transition intensities; these represent decreasingly strict assumptions about the transition rates, which translates to increasing complexity in computing state probabilities. But also to increasing biological credibility of the models.

The naming of the models (except for item 4) comes from probability theory:

1. **Homogeneous Markov:** All transition rates are constant over time. This allows calculation of state probabilities using the matrix exponential on the transition intensity matrix. This can be closely approximated by successive matrix multiplication of the state probability vector by the transition probability matrix.
2. **Inhomogeneous Markov:** Transition rates vary over time but all transition rates vary along the *same* time scale. This allows calculation of state probabilities by successive multiplication of the state probability vector by the *time-specific* transition probability matrices. The use of the matrix exponential is not possible.
3. **Semi Markov:** Transition rates from different states vary by time since entry to the state, so along *different* time scales in different states. This makes calculations of state probabilities using transition probability matrices impossible. In fact a transition matrix is not even well-defined because transitions from a given state is different between the persons in the state—they may have arrived in the state at different times.
4. **Multiple timescales:** Transition rates depend on more than one time scale, such as current age and current duration of diabetes. It is by far the most realistic in practical settings, but it requires micro-simulation to compute state probabilities for a given initial state configuration.

The first (homogeneous Markov) model is rarely used, except for exploratory analyses to give a broad overview of the magnitude of transition rates, as for example the ones given on the arrows in figure 1.2.

The second (inhomogeneous Markov) model is by far the the most used in practice because it allows calculations of state probabilities using non-parametric models for the time effects. This is the realm of the Kaplan-Meier and Aalen-Johansen estimators of state proabilities. The drawback of the non-parametric estimators is that you hardly ever get to see estimated *rates*. Some will even claim that the rates are not part of these models. But the main problem is that the estimated probabilities refer to the study population at hand—hardly ever generalizable.

It is only for the two cases of Markov models (types 1 and 2) that the general probability theory for Markov models is applicable in practise, for the last two the general theory using the transition matrices does not apply. When dealing with the more realistic types of models (types 3 and 4), parametric models for the transition intensities with timescales as covariates are preferable, particularly because this type of model facilitates micro-simulation of data. The concept of micro-simulation is covered in chapter 7.

. . . now input from `likelihood`

Chapter 3

Multistate likelihood: rates

When fitting statistical models to observed data, the standard procedure is to use maximum likelihood, which means that we find the values of model parameters that maximizes the probability of the observed data. Therefore we must be able to compute the probability of observing a given set of follow-up data from a multistate model as a function of the model parameters. This is what is derived in this chapter.

But you will not find a full exposition of the theory of maximum likelihood here. It suffices to know that the likelihood machinery allows calculation of uncertainty of parameters based on the likelihood function alone, so we need to establish the likelihood for multistate observations, in order to come up with parameters and associated uncertainties.

Outcome data from a multistate model consists of *risk time* in each state and the (type of) *event* (if any) at the end of this, often several observations per person. Normally a number of covariates will be available for each of these observations, including one or more time scales.

In practice the available data for a person is a sequence of dates: date of entry to the study, dates of each event, and date of end of study—often date of death will be both an event time as well as end of study time. If date of birth is available we will also have age available as a covariate; similarly for other dates: date of diagnosis will allow calculation of the disease duration, etc.

The point here is that we observe transition *rates* as a bivariate observation of time at risk and no. of events (0 or 1) of each possible type.

3.1 Survival model

The simplest multistate model is a survival model with states `Alive` and `Dead` and one possible transition; that from `Alive` to `Dead`. From this model we can estimate 1) the mortality *rate* (typically as a function of age or time since entry), 2) as a transformation of the integral of this, the *survival* function and 3) as the integral of the survival, the *expected lifetime*.

So even in the simplest possible case we have all three types of derived measures mentioned in section available.

The basic observation for each person is the (empirical) rate in the form (d, y), where d is the event count (0 or 1) and y is the risk time, *i.e.* the time the person has been at risk of dying. Some formulate this as the survival time T and the indicator δ of whether the person died or was censored at time T.

In order to derive the probability of this as a function of the occurrence rate we need a precise definition of a *theoretical* mortality rate. This is defined as the probability of death in small interval, *relative* to the length of the interval (h). The rate is often called λ (the Greek letter "lambda"):

$$\lambda(t) = \mathrm{P}\{\text{death in } (t, t+h] \,|\, \text{alive at } t\}\,/h$$

Formally, the rate is the limit of this quantity as $h \to 0$.

3.1.1 Likelihood for a single rate

To derive the likelihood we must compute the probability of an observation from a single person as a function of the rate. Suppose a person is alive from time t_e (entry) to t_x (exit) and that the person's status

at t_x is d, where $d = 0$ means alive and $d = 1$ means dead. If we choose, say, two time points, t_1, t_2 between t_e and t_x, standard use of conditional probability (formally, repeated use of Bayes' formula) gives

$$\begin{aligned} \mathrm{P}\{d \text{ at } t_x \mid \text{entry at } t_e\} = &\mathrm{P}\{\text{survive } (t_e, t_1] \mid \text{alive at } t_e\} \times \\ &\mathrm{P}\{\text{survive } (t_1, t_2] \mid \text{alive at } t_1\} \times \\ &\mathrm{P}\{\text{survive } (t_2, t_x] \mid \text{alive at } t_2\} \times \\ &\mathrm{P}\{d \text{ at } t_x \mid \text{alive just before } t_x\} \end{aligned} \tag{3.1}$$

By choosing more intermediate time points we can make the intervals arbitrarily small, so we just need to derive the probability of surviving a small piece of time, as a function of the mortality rate.

For a start assume that the mortality is constant over time $\lambda(t) = \lambda$. From the definition of a rate we have (conditional on being alive at t):

$$\begin{aligned} \mathrm{P}\{\text{death during } (t, t+h]\} &\approx \lambda h \\ \Rightarrow \mathrm{P}\{\text{survive } (t, t+h]\} &\approx 1 - \lambda h \end{aligned} \tag{3.2}$$

where the approximation gets better the smaller h is. Suppose we have survival for a time span $y = t_x - t_e$ and that this is subdivided in N intervals, each of length $h = y/N$, then the survival probability for the entire span from t_e to t_x is the product of probabilities of surviving each of the small intervals, conditional on being alive at the beginning of each interval:

$$\mathrm{P}\{\text{survive } t_e \text{ to } t_x\} \approx (1 - \lambda h)^N = \left(1 - \frac{\lambda y}{N}\right)^N$$

From mathematics it is known that $(1 + x/n)^n \to \exp(x)$ as $n \to \infty$ (some define $\exp(x)$ this way). So if we divide the time span y in successively smaller pieces we will have that $N \to \infty$, and hence that:

$$\mathrm{P}\{\text{survive } t_e \text{ to } t_x\} \approx \left(1 - \frac{\lambda y}{N}\right)^N \to \exp(-\lambda y), \quad N \to \infty \tag{3.3}$$

Therefore the contribution to the likelihood from a person observed for a time span of length y is $\exp(-\lambda y)$, and the contribution to the log-likelihood is therefore $-\lambda y$.

If we observe a person dying at the end of the last interval, the contribution to the likelihood from the last interval will be the probability surviving till just before the end of the interval, multiplied by the probability of dying in the last tiny instant (of length ϵ) of the interval (the last term in (3.1)). This probability is by (3.2) $\lambda\epsilon$, and hence the log-likelihood contribution from this last instant is $\log(\lambda\epsilon) = \log(\lambda) + \log(\epsilon)$.

The total likelihood for one person is the product of all these terms from the follow-up intervals (i) for the person; and the log-likelihood (ℓ) is therefore:

$$\begin{aligned} \ell(\lambda) &= -\lambda \sum_i y_i + \sum_i d_i \log(\lambda) + \sum_i d_i \log(\epsilon) \\ &= \sum_i \left(d_i \log(\lambda) - \lambda y_i\right) + \sum_i d_i \log(\epsilon) \end{aligned}$$

where y_i is observation time (risk time or person-years) in the i^{th} interval and d_i the death indicator (0/1) for the i^{th} interval (which is 0 for all intervals, except possibly for the last). We want to estimate λ, so terms that does not involve λ can be ignored—such as the last term. Very convenient, we then do not have to decide precisely how small ϵ is.

Thus we have established that the contribution to the log-likelihood from a single person's follow-up is the sum of a number of terms of the form $d_i \log(\lambda) - \lambda y_i$ where d_i is the event indicator (0/1) and y_i the length of the i^{th} interval.

If we subdivide the follow-up of a person across several records, each representing y_i of follow-up, and keep track of the deaths in each interval, d_i (which is 0 for all intervals, except possibly for the last if the person dies), then each record (d_i, y_i) will represent a contribution to the log-likelihood. This will be exploited in the practical modeling of rates.

By the assumption that the rate λ is constant over time, the log-likelihood contribution from a person with d deaths (0 or 1) at the end of a follow-up period of y is $d \log(\lambda) - \lambda y$. But once we have subdivided

follow-up we do not need the assumption of a constant rate across the intervals, we can allow separate rates (λ_is) in different intervals, relaxing the assumption to only constant rates within each of the very small intervals. These rates can of course not be estimated based on the observation from a single person. A model for the λ_is must be specified in terms of covariates associated with each interval. Among these will normally be the value of one or more time scales at the beginning of each interval.

3.1.2 Poisson likelihood

The Poisson distribution with mean μ is a distribution on the non-negative integers ($x = 0, 1, 2, 3, \ldots$), where:

$$\mathrm{P}\{X = x\} = \frac{\mu^x \exp(-\mu)}{x!}$$

Thus, the log-likelihood from observation of a Poisson variate x with mean μ is $x \log(\mu) - \mu - \log(x!)$. The last term does not depend on the parameter μ so it can be omitted when maximizing the log-likelihood over values of μ.

Changing the notation, the log-likelihood contribution from a Poisson-variate d with mean λy is $d \log(\lambda y) - \lambda y = d \log(\lambda) - \lambda y + d \log(y)$, which is the same as the likelihood for d events during y follow-up time with rate λ, except for the term $d \log(y)$. But this term does not depend on the parameter λ and therefore can be ignored when maximizing the log-likelihood.

This means that maximum-likelihood estimation for observations from follow up, (d, y), can be done using a function that can maximize the likelihood for independent Poisson variables. We just need to pretend that the ds are independent Poisson variates with means λy.

In R we have the functions `glm` and `gam` to do this, depending on how we will model the covariate effects.

3.1.3 Implications for analysis

The derivation in equation 3.1 (page 18) has the implication that we can split the follow-up for a person in many small intervals and use a model that assigns *different* (albeit constant) rates to each interval. The intervals just need to be sufficiently small to entertain the assumption of a constant rate within each interval as reasonable. The log-likelihood for a person's follow-up will then be the sum of the log-likelihood contributions from each interval (record in the dataset).

3.1.3.1 Time as covariate

If we keep track of the value of the time *scales* (e.g. calendar time or age) at the beginning of each interval we can compute the value of other time scales at the beginning of each interval. These values can then be used as covariates in parametric models estimating the effect of different time scales on the rates. Note that we are not necessarily assigning a separate parameter to each interval, we will normally use the time scales as quantitative (metric) variables with a smoothly varying effect on the rates.

This is called time-varying rates; it corresponds to the time *in*homogeneous Markov models mentioned in chapter 2, p. 16. If time since state entry is used as covariate, we have a semi-Markov model, if more time scales are used we have a model with multiple time scales, which, formally speaking is not a Markov model.

3.1.3.2 Likelihood, not model

It is important to note that the central feature we use is that the likelihood from a single person is a *product* of terms; there is no assumption about the contributions being independent. It is the product form of (3.1) that makes the likelihood for one person *look like* the likelihood for many independent Poisson variates (the d_is).

There is namely not a one-to-one correspondence between likelihoods and models; here we have two different models:

1. the model for follow-up of a person assuming constant rate in each of a sequence of small intervals; outcome data for interval i is (d_i, y_i), parameter is λ_i.

2. the model for independent Poisson variates; outcome data for interval i is d_i, parameter is $\lambda_i y_i$.

The (log-)likelihood from both these models is the same. Since practical estimation only uses the likelihood and not the model, we can fit the follow-up model by telling the program that we have independent Poisson variates d_i with mean $\lambda_i y_i$—even if they are neither Poisson nor independent.

The offset Since the mean parameter in the Poisson-model corresponding the rate model is $\lambda_i y_i$, then if we use the logarithm as link function, the model for the linear predictor will be $\log(\lambda_i y_i) = \log(\lambda_i) + \log(y_i)$. Therefore, if we want a linear model for $\log(\lambda_i)$, we must accommodate the term $\log(y_i)$ as a separate term in the linear predictor with a fixed regression coefficient of 1, a so-called offset.

This is the background for the phrase normally seen in description of analysis of rates: "... we fitted a Poisson model for the number of deaths with the log of the person years as offset..." or something similar. Some even write that they assumed the number of deaths to be Poisson-distributed. But they are just fitting a model for piecewise constant rates by maximum likelihood; the Poisson machinery is purely a computational technicality.

If the occurrence rate is constant, then the observed survival times will follow an exponential distribution, so occasionally the machinery will be described as "exponential model", "piecewise exponential model" or "model with exponentil life times", but it is still the same machinery.

Note that the offset is closely linked to the log-link, so if we instead want a model with identity link the risk time (y) must be accommodated differently. We shall return to this in section **??**.

Identity link model If we want to fit an additive model for the rates, we still need to specify a model for the link if the mean, but with identity link we are facing the specification of a linear function of $\lambda_i y_i$, which is not so straight forward as for the logarithmic link.

There are a number of work-arounds that will alleviate this; they are described in the chapter on additive rate models, **??** on page **??**.

3.1.3.3 Real Poisson data

A theorem in probability theory states that the number of events (deaths) observed over a *fixed* period of time will be Poisson-distributed, if we at each death replace the dead person with an identical living person, and include this person in the follow up.

So it is actually possible to get Poisson-distributed counts in follow-up studies, but the study logistics will be complicated The scenario is somewhat more realistic if we consider testing life-time of light-bulbs and not people.

3.2 Multistate model

3.2.1 One transition

For a possible transition from state `A` to state `B`, say, the likelihood *conditional* on being in state `A` and available for transition to `B` depends on the rate-parameter λ for the transition as well as on the observed data, which for a single person is (d, y), the number of transitions d (0 or 1) and the risk time, y, spent in `A`.

The rate parameter for the transition `A` to `B` is defined as:

$$\lambda_{\texttt{AB}}(t) = \lim_{h \to 0} \mathrm{P}\{\texttt{A} \to \texttt{B} \text{ during } (t, t+h] \,|\, \text{in } \texttt{A} \text{ at } t\} / h$$

For convenience we will leave out the subscript `AB` for λ in the following.

If the rate is assumed constant over time ($\lambda(t) = \lambda$), then the log-likelihood contribution from (d, y) is $d \log(\lambda) - \lambda y$, where d is the sum of the d_ts and y the sum of the y_ts. As noted above, this is also the log-likelihood from a Poisson variate d with mean λy.

3.2.1.1 Time varying rates

If the rate is allowed to vary by time, we can accommodate this by subdividing the observation (d, y) in several small intervals each with a separate (d_t, y_t) where $\sum d_t = d$ and $\sum y_t = y$—note that all the d_ts

from one person will be 0 except possibly the last one which will be 1 if a transition occurs, so the sum is just the last d_t from each person.

This subdivision of data allows us to make the less restrictive assumption that the rate is only constant in each of the small intervals, and the log-likelihood can be expressed as $\sum_t \big(d_t \log(\lambda_t) - \lambda_t y_t\big)$. This is because the likelihood contribution from interval t is the conditional probability of the observation (d_t, y_t) given that the person is alive and at risk of transition to `B` at the beginning of the interval. So the total likelihood for a person is a product of conditional probabilities, one for each small interval the person lives through.

3.2.1.2 Time as covariate (when)

In the context of time-varying rates t is a *covariate*; a variable upon which the the outcome, the rate λ_t, depends. By that token, λ_t could be modeled as a smooth function of t, we do not necessarily need a separate rate parameter for each t, we can use t as a *quantitative* (metric) variate. Note that t has not been specified as age or calendar time or disease duration; it could be any of these.

In fact it could be all three of them, in the sense that λ could be a smooth function of all of these three; the data requirement is just that they be evaluated at the beginning of each of the small intervals. This is referred to as models with *multiple time scales*.

In this context time (t) refers to *when* a person is at risk, where "when" could refer to more than one time scale.

3.2.1.3 Time as response (how long)

We saw above that the log-likelihood contribution from one person is the same as the log-likelihood contribution from a set of independent Poisson variates d_t with mean $\lambda_t y_t$. Therefore practical modeling of the transition rate $\texttt{A} \to \texttt{B}$ can proceed *as if* observations were independent Poisson variates. This is straightforward with the usual `glm` machinery in `R`, once the split (subdivided) records (d_t, y_t) are generated; in this context (d_t, y_t) is the response (outcome). Thus the risk time y_t is part of the response, often referred to as *risk time* or *person years*.

In this context time (y) refers to *how long* a person is at risk, where "how long" will be the same on any time scale.

3.2.2 Competing risks

If more than one type of event can occur, we have what is called competing risks—several risks compete for the exit from the current state.

Specifically, suppose that in addition to the transition $\texttt{A} \to \texttt{B}$ a transition from `A` to some other state `C` were also possible, with rate parameter μ where:

$$\mu_{\texttt{AC}} = \lim_{h \to 0} \mathrm{P}\{\texttt{A} \to \texttt{C} \text{ during } (t, t+h] \,|\, \text{in } \texttt{A} \text{ at } t\}\,/h$$

Let the observation of this transition be $k_t \in (0, 1)$, say, so formally we have the observation (d_t, k_t, y_t) where at most one of d_t and k_t are 1—events are mutually exclusive; they are "competing". Note that the y_ts (the risk time) are identical for the $\texttt{A} \to \texttt{B}$ and $\texttt{A} \to \texttt{C}$ transitions—it is the time the person is at risk (in state `A`) of any of the two transitions.

In the derivation of the likelihood for a single rate the likelihood contribution from surviving an interval of length h, say, was $\exp(-\lambda h)$; but in this case the person has to escape both the rate $\texttt{A} \to \texttt{B}$ and $\texttt{A} \to \texttt{C}$, so the likelihood contribution will be $\exp(-(\lambda + \mu)h)$. If we observe a transition to `B` ($d_t = 1, k_t = 0$) we will have a final contribution of $\lambda\epsilon$ and if a transition to `C` ($d_t = 0, k_t = 1$) a contribution of $\mu\epsilon$. The log-likelihood contribution from the observation of (d_t, k_t, y_t) is therefore:

$$\begin{aligned} \ell(\lambda, \mu) &= -(\lambda + \mu) y_t + d_t \log(\lambda\epsilon) + k_t \log(\mu\epsilon) \\ &= \big(d_t \log(\lambda) - \lambda y_t\big) + \big(k_t \log(\mu) - \mu y_t\big) + (d_t + k_t)\log(\epsilon) \end{aligned} \tag{3.4}$$

where we can disregard the term $(d_t + k_t)\log(\epsilon)$ that does not depend on the parameters λ and μ.

This is precisely the same as the sum of contributions from each of the transitions *as if* they each were the only possible transition. This means that we can model the transitions separately, but we will be forced to assume that λ and μ are specified without common parameters.

3.2.2.1 Stacked data for joint likelihood

If the models for two transition rates do have common parameters we can still construct the log-likelihood contributions from each transition separately. And since the total log-likelihood is a sum of the two, we can treat all of the contributions as contributions from independent Poisson variates in a joint model—as described above in section 3.2.2 on competing risks. But in order to get a representation of all the likelihood contributions each record of follow-up time must appear twice; one time with the transition to **B** as event (the d_i) and one time with the transition to **C** as event (the k_i).

In practice this is done by "stacking" the data sets relating to each of the transitions from `A`: take d_t and the y_t as variables `D` and `Y`, along with a variable `Tr` with value `A2B`, and put this dataset head-to-foot with a dataset with k_t and the y_t as variables `D` and `Y`, along with the variable `Tr` with value `A2C`. Other covariates of interest will be carried over as well, and the values will thus appear twice in the stacked dataset. In the resulting dataset each interval for a person contributes two records in the calculation of the log-likelihood, one for (d_t, y_t) and one for (k_t, y_t)—with the same y_t component and the same set of covariates.

Any parameter specified in a model using the resulting stacked data set will be common for the two transitions. But if an interaction with the `Tr` variable is specified, then we will have a parameter value for each level of `Tr`. Thus the stacked dataset allows models where parameters are common for the two rates, but also, via interactions with the `Tr` variable, effects which differ between the two sets of rates.

Note however that the stacked dataset is a representation of the components of the log-likelihood, not of the follow-up.

If more than two transitions from `A` are possible the same setup applies, with one contribution for each type of possible transition from the state.

3.2.2.2 A small illustration of stacking data

Suppose we have 3 persons in state `A`, at risk for events `B` and `C`, one seeing event `B`, one seeing event `C` and one seeing neither (censored alive). The representation of the follow up would be, with `ptime` being the risk time and `x` and `z` being some covariates:

```
  id event ptime    x    z
1  1     B  12.3 5.23  170
2  2     C   8.7 0.52 1757
3  3  cens  14.1 4.22 3803
```

The dataset for analysis of the rates of `B` would be:

```
  id d    y    x    z
1  1 1 12.3 5.23  170
2  2 0  8.7 0.52 1757
3  3 0 14.1 4.22 3803
```

Using `(d,y)` as the data (outcome) and the rate $\lambda_{\mathtt{B}}$ as parameter, each record in this dataset would correspond to a log-likelihood contribution for a model for the rate of `B` occurrence, $\mathtt{d}\log(\lambda_{\mathtt{B}}) - \lambda_{\mathtt{B}}\mathtt{y}$.

The analysis dataset for analysis of the rates of `C` would be:

```
  id d    y    x    z
1  1 0 12.3 5.23  170
2  2 1  8.7 0.52 1757
3  3 0 14.1 4.22 3803
```

Each record in this dataset corresponds to a log-likelihood contribution for a model for the rate of `C` occurrence, $\mathtt{d}\log(\lambda_{\mathtt{C}}) - \lambda_{\mathtt{C}}\mathtt{y}$. The only difference between the two analysis datasets is the coding of `d`, the event indicator; in the first case it is the indicator of an `B` outcome, in the second it is the indicator of a `C` outcome.

If we were to model occurrence rates of `B` and `C` together, with, say, identical effects of `x` on the two transition rates, but different effect of `z` on the two rates, we would need to combine the two datasets, in order to get all contributions to the likelihood as in equation (3.4)

```
  id d    y    x    z   Tr
1  1 1 12.3 5.23  170 A->B
2  2 0  8.7 0.52 1757 A->B
3  3 0 14.1 4.22 3803 A->B
4  1 0 12.3 5.23  170 A->C
5  2 1  8.7 0.52 1757 A->C
6  3 0 14.1 4.22 3803 A->C
```

or equivalently:

```
  id d    y    x    z   Tr
1  1 1 12.3 5.23  170 A->B
4  1 0 12.3 5.23  170 A->C
2  2 0  8.7 0.52 1757 A->B
5  2 1  8.7 0.52 1757 A->C
3  3 0 14.1 4.22 3803 A->B
6  3 0 14.1 4.22 3803 A->C
```

Note that we have added the variable `Tr` indicating which type of transition each record refers to; for persons with an event of either type the two records differ on the event indicator `d`, but for the person that sees no event, the two records are identical, except for the `Tr`. We see that in the stacked dataset there is at most one of the `d` that is 1 within each person.

A model with identical `x` effect on both rates would just have `x` as a main effect, while different effects of `z` for the two transitions would require an interaction between `z` and `Tr`.

In the `Epi` package there is a function `stack.Lexis` that does this type of data expansion for rate data represented as `Lexis` objects, see section **??**.

However, it is not very realistic to encounter two different transition *out* of a single state that share parameter values. For example, an assumption that the male/female mortality rate-ratio were the same for death from cancer and from cardiovascular disease is not very likely to be of any practical relevance.

3.2.3 Multiple transitions

If we have two completely separate transitions $\texttt{A} \rightarrow \texttt{c}$ and $\texttt{B} \rightarrow \texttt{E}$, say, then the likelihood contribution from the two transitions would be completely independent, and the log-likelihood from the two transitions would just be the sum of the two separate likelihoods from each of the transitions.

Now, suppose it was possible to make a further transition from `C` to `D`. The likelihood contribution for this from a person would be the conditional probability of transition to `D`, *given* that the person were available in `C` for a $\texttt{C} \rightarrow \texttt{D}$ transition. So by Bayes' theorem the likelihood for $\texttt{A} \rightarrow \texttt{C}$ *and* $\texttt{C} \rightarrow \texttt{D}$ would be the probability of the first ($\texttt{A} \rightarrow \texttt{C}$) multiplied by the conditional probability of the second ($\texttt{C} \rightarrow \texttt{D}$) given that the first ($\texttt{A} \rightarrow \texttt{C}$) had occurred.

In this case, some persons would contribute two terms to the likelihood and two data records $(d = 1, y_{\texttt{A}})$ and $(d, y_{\texttt{C}})$, other persons would contribute only one $(d = 0, y_{\texttt{A}})$, where $y_{\texttt{A}}$ and $y_{\texttt{C}}$ are risk times contributed in states `A` and `C` respectively. But again, the likelihood contribution from a single persons would be a product of terms, each one looking like a Poisson likelihood term.

3.2.4 Total log-likelihood for a multistate model

We have seen that the log-likelihood for *all* transitions in the multistate model is the sum of the separate log-likelihoods for each of the transitions, where we treat each transition as if it were the only possible transition from the origin state.

This implies that for states *from* which transition to more than one other state is possible, the same risk time will be represented multiple times, one for each possible transition. It will also imply that the sum of risk time over all records will be larger that the total risk time in the study.

The `Lexis` machinery (chapter 4) represents follow-up, not log-likelihood contributions, so if more than one transition from a given state is modeled in the same model, the `Lexis` representation cannot be used to evaluate the log-likelihood.

In principle we can set up one large model for all transitions where we have the possibility of having common parameters between different transitions.

Alternatively we could model each transition rate separately if we assumed that no two transitions had common parameters. The latter will only require a subset of the `Lexis` representation to construct the log-likelihood. Actually, we can model several transitions jointy from records in a `Lexis` object provided only that no two transitions are *from* the same state.

. . . now input from `datarep`

`datarep`

Chapter 4

Lexis: data representation and analysis of rates

In chapter 2 on probability models we looked at a simple multistate model for mortality and insulin use among diabetes patients. This chapter illustrates how to construct a `Lexis` object to represent and manipulate the multistate information, and gives some examples of analysis of transition rates using the Poisson likelihood derived in chapter 3

The key to understanding the purpose of the `Lexis` machinery is the likelihood of a rate as derived in chapter 3 on multistate likelihood. In particular the formula 3.1 on page 18, which tells us that we can split the follow-up in many small pieces and analytically treat event indicators (0/1) in these as if they were independent Poisson variates, and still arrive at the correct likelihood.

In `Lexis` objects (data frames) we represent person follow-up with one record per interval lived, along with information about the interval:

- the position of the beginning of the interval on each time scale of interest (age, calendar time, disease duration, ...)
- the time spent in the interval (length of interval)
- the state in which the time is spent
- the state to which the person moves next (if any)
- the current status of the person with respect to other covariates of interest, such as sex, marital status, blood pressure or other clinical variables, drug exposure etc.

This chapter shows how to manipulate information on follow-up and covariates into a `Lexis` object that allows splitting in many small intervals suitable for statistical analysis of transition rates.

... now input from `DM1ate`

4.1 Using the `Lexis` function

The following example is illustrating the basic set-up of follow-up data as a `Lexis`[1] object for multistate data with multiple time scales, allowing simple summarizing, graphics and analysis. A `Lexis` object is just a data frame where each record represents an interval of follow-up, with some extra variables and attributes that define features of the follow-up. `Lexis` objects allow definition of multistate models with multiple states, and different ways of looking at these.

[1] Named after the German demographer and economist Wilhelm Lexis (17 July 1837 – 24 August 1914), who in his book "Einleitung in die Theorie der Bevölkerungsstatistik" [Introduction to the theory of population statistics] (1875) devised a diagram showing follow-up of persons on multiple time scales. The diagram is nowadays called a Lexis diagram, even if it is usually drawn in a slightly different manner than that Lexis used in his book.

Ultimately the Lexis representation is going to be used for modeling of rates, so a few simple modeling examples are included in this section too.

4.1.1 Defining a Lexis object

First we define the total follow-up for all persons in DMlate; persons enter the follow-up at date of diabetes, dodm[2], and end follow up at date of exit, dox. A person who ends his follow-up as dead has a non-missing dodth. The dates are all coded in units of years (365.25 days) with 1970-01-01 represented as 1970.000; for example, 1996.5 represents 2nd July 1996.

```
> data(DMlate)
> dmL <- Lexis(entry = list(Per = dodm,
+                           Age = dodm - dobth,
+                         DMdur = 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
```

CODE EXPLAINED: The Lexis function defines time-scales of follow-up at the time of study entry on each time scale, in this case the time scales Per—calendar time ("Period"), Age—current age and DMdur—duration of diabetes. The variables are given in a *named* list, the names will be the names of the time scales, which will appear as new variables in the resulting data frame.

The follow-up time (risk time) is defined via the time of exit on one of the time scales; exit is also a named list, but only one element is required since risk time is the same measured on any time scale, in this case we use Per=dox, so the risk time is computed as dox − dodm.

Finally we must define the exit *status*, this should be a factor—the standard way of representation of a variable that only takes a finite no. of values. Here we use the indicator of a valid death date as variable and turn it into a factor with levels DM and Dead. The expression !is.na(dodth) returns a logical, and for logicals, FALSE<TRUE, so if !is.na(dodth) is FALSE (meaning date of death is missing) the exit.status will be DM, if it is TRUE it will be Dead, because DM is mentioned before Dead in labels=.

The 4 dropped persons are persons with date of diabetes equal to date of death.

We can summarize the follow-up as defined in the Lexis object dmL in tabular form:

```
> summary(dmL)
Transitions:
     To
From   DM Dead  Records:  Events: Risk time:  Persons:
  DM 7497 2499      9996     2499   54273.27      9996
```

CODE EXPLAINED: summary (really summary.Lexis because dmL is a Lexis object) gives an overview of the follow-up from entry to death or censoring. We see that 9,996 persons contribute some 54,000 person-years, and 2499 deaths.

4.1.2 Printing a Lexis object

The result from Lexis is a data frame of class Lexis, basically the original data frame, DMlate, amended with the time scales and some extra variables (with prefix lex.):

```
> dmL[1:5,1:10]
```

[2]The DM/dm used in the code refers to "diabetes mellitus" which in turn refer to the symptoms of the disease, from the Greek diabainein, meaning "to pass through", describing the copious urination, and mellitus, from the Latin meaning "sweetened with honey", referring to sugar in the urine.

```
lex.id     Per   Age DMdur lex.dur lex.Cst lex.Xst sex    dobth     dodm
     1 1998.92 58.66     0   11.08      DM      DM   F 1940.256 1998.917
     2 2003.31 64.09     0    6.69      DM      DM   M 1939.218 2003.309
     3 2004.55 86.25     0    5.45      DM      DM   F 1918.301 2004.552
     4 2009.26 44.04     0    0.74      DM      DM   F 1965.225 2009.261
     5 2008.65 75.78     0    1.34      DM      DM   M 1932.877 2008.653
```

CODE EXPLAINED: When the name of a Lexis object is given, R will invoke the function print.Lexis if it exists (and it does exist) otherwise the print.default function. print.Lexis function will print the columns of the Lexis object in order with id and the time scales first, followed by the state indicators. Time scales and lex.dur rounded to 2 decimals, and other variables rounded to 3 decimals. This latter behaviour can be changed by calling print.Lexis directly, see ?print.Lexis.

We list only the first five rows and the first 10 columns of dmL; this is sufficient for illustration.

The function Lexis has added variables with the three defined time scales Per, Age and DMdur; they represent the value of the time scale at the *beginning* of the follow-up interval of length lex.dur. lex.Cst (Current state) is the state in which the follow-up takes place, whereas lex.Xst (eXit state) is the next state the person occupies after lex.dur time in lex.Cst. If lex.Xst is the same as lex.Cst it means that the person is censored in the state; no transition is recorded. lex.id is the person identifier, if not explicitly set (via the id= argument) when calling Lexis it will be a simple consecutive numbering of the records in the input data.

Thus, each record in the Lexis object represents follow-up for a period of lex.dur in the state recorded in lex.Cst, starting at calendar time Per in age Age and diabetes duration DMdur, after which the person continues in the state recorded in lex.Xst.

The default behaviour of Lexis is to assume the each record in the data represents the follow-up of one person, but a person may enter with multiple records, for example if the person has been emigrated for a period. In this case the argument id should be used, but Lexis provides no check of consistency between records with the same id.

4.2 Modeling mortality

4.2.1 Effect of age at diagnosis

We can model mortality as a function of Age, it will be a model with the age at entry (diagnosis of diabetes) as explanatory variable, assuming that mortality is constant over the follow-up. This is not very realistic, but the only possibility with this dataset. A Poisson model can be fitted (using the Lexis features of dmL):

```
> library(mgcv)
> m0 <- gam.Lexis(dmL, ~ s(Age))
mgcv::gam Poisson analysis of Lexis object dmL with log link:
Rates for the transition:
DM->Dead
```

CODE EXPLAINED: gam.Lexis fits a rate model (Poisson likelihood) where the mortality is assumed constant during follow-up of each person (i.e. the follow-up represented in each record of dmL), but where the magnitude of this constant mortality is determined by smooth function of Age, which in dmL is the age at diagnosis of diabetes). This is because dmL is a data frame with only one record per person, each record representing the entire follow-up of the person. So not a very interesting model.

The default for gam.Lexis is to model all transitions *to* any absorbing state, but in this case there is only one absorbing state, namely Dead, and only one transition to it, namely DM→Dead. In full detail we would have written:

```
> m0 <- gam.Lexis(dmL, ~ s(Age), from = "DM", to = "Dead")
```

Further, gam.Lexis is merely a wrapper for gam; in this case the call of gam.Lexis is tantamount to:

```
> gam(cbind(lex.Xst %in% "Dead" & lex.Xst != lex.Cst,
+           lex.dur) ~ s(Age),
+     family = poisreg,
+       data = subset(dmL, lex.Cst %in% "DM"))
```

Note the logical clause `lex.Xst != lex.Cst` which is necessary when handling transitions *to* transient states. So `gam.Lexis` makes life a bit easier.

The family `poisreg` uses the likelihood derived for follow-up data in chapter 3; it is an improvement of the `poisson` family that would have provided the same fit by:

```
> gam((lex.Xst %in% "Dead" & lex.Xst != lex.Cst) ~ s(Age),
+      offset = log(lex.dur),
+      family = poisson,
+        data = subset(dmL, lex.Cst %in% "DM"))
```

A similar wrapper, `glm.Lexis`, exists for `glm`.

4.2.2 Splitting follow up for analysis by current age

It is of considerably more interest to quantify how the *current* age (also called *attained* age) determines the mortality. In order to do this we must split the follow-up in intervals so small that we can safely assume that mortality rate is constant within each interval and then use the current age for the interval. This done by either `splitLexis` from the `Epi` package or `splitMulti` from the `popEpi` package—the latter slightly faster and with a simpler syntax:

```
> library(popEpi)
> options("popEpi.datatable" = FALSE)
> system.time(dmS <- splitLexis(dmL, breaks = seq(0, 120, 0.25),
+                                    time.scale = "Age"))
   user  system elapsed
   2.32    0.30    2.63
> system.time(dmS <- splitMulti(dmL, Age = seq(0, 120, 0.25)))
   user  system elapsed
   1.47    0.26    1.61
> summary(dmL)
Transitions:
     To
From   DM Dead  Records:  Events: Risk time:  Persons:
  DM 7497 2499      9996     2499   54273.27      9996
> summary(dmS)
Transitions:
     To
From     DM Dead  Records:  Events: Risk time:  Persons:
  DM 224557 2499    227056     2499   54273.27      9996
> (who <- subset(dmL, lex.Xst == "Dead")[4,'lex.id'])
[1] 24
> subset(dmL, lex.id == who)[,1:10]
 lex.id     Per   Age DMdur lex.dur lex.Cst lex.Xst sex    dobth     dodm
     24 1996.89 63.16     0    1.43      DM    Dead   M 1933.723 1996.886
> subset(dmS, lex.id == who)[,1:10]
 lex.id     Per   Age DMdur lex.dur lex.Cst lex.Xst sex    dobth     dodm
     24 1996.89 63.16  0.00    0.09      DM      DM   M 1933.723 1996.886
     24 1996.97 63.25  0.09    0.25      DM      DM   M 1933.723 1996.886
     24 1997.22 63.50  0.34    0.25      DM      DM   M 1933.723 1996.886
     24 1997.47 63.75  0.59    0.25      DM      DM   M 1933.723 1996.886
     24 1997.72 64.00  0.84    0.25      DM      DM   M 1933.723 1996.886
     24 1997.97 64.25  1.09    0.25      DM      DM   M 1933.723 1996.886
     24 1998.22 64.50  1.34    0.09      DM    Dead   M 1933.723 1996.886
```

CODE EXPLAINED: We split the follow-up in 3-month intervals (0.25 years) with `splitLexis` and `splitMulti`, showing the time difference using `system.time`, but with the same result, except that `splitMulti` returns a `data.table`, unless told to not to by setting the option `popEpi.datatable` to `FALSE`. So `dmS` is of class `data.frame` in this case.

The `summary` shows that the number of deaths and amount of follow-up is the same in `dmS` as in `dmL`, but `dmS` has about 23 times as many records.

We extract the `lex.id` for a person—here the 4$^{\text{th}}$ person (who has `lex.id` 24) that ends follow-up in `Dead`, and show the person's records in the two datasets.

We see that person 24 in the split dataset `dmS` only have one transition to `Dead` as in `dmL`, and also that the sum of the `lex.dur` is the same in the two datasets. But the values for `Per`, `Age` and `DMdur` vary between records, the values of these refer to the time scale values at the beginning of each of the intervals. Therefore, if we model mortality by `Age` using `dmS` we will be modeling the effect of `Age` at the beginning of each of these small intervals, so effectively modeling the effect of *current* age:

```
> m1  <- gam.Lexis(dmS, ~ s(Age))
mgcv::gam Poisson analysis of Lexis object dmS with log link:
Rates for the transition:
DM->Dead
```

The only difference between `m0` and `m1` is the dataset used; using `dmL` assumes the mortality constant over the entire follow-up for each person using the age at entry to the study as variable, using `dmS` we assume mortality constant only in 3-month intervals using the age at the beginning of each interval as explanatory variable.

Thus the first model (`m0`) is showing mortality as function of age at diagnosis (assuming that mortality does not depend on current age or disease duration), the latter (`m1`) shows mortality as function of current age (assuming that mortality does not depend on age at onset or disease duration).

We can compare the two estimated age-effects models by making predictions for ages 30 through 90, say, from both models. Strictly speaking the graph is nonsense, one of the curves has x-axis as age at entry, the other x-axis as current age; the deception lies in the sloppiness of using the word "age" without further specification.

```
> nd <- data.frame(Age = 30:90)
> matshade(nd$Age, cbind(ci.pred(m0, nd),
+                        ci.pred(m1, nd)) * 100,
+          plot = TRUE,
+          xlab = "Age", ylab = "Mortality per 100 PY",
+          col = c('green','orange'), lwd = 2, log = "y")
> axis(side = 1, at = seq(65,75,5), labels = NA, pos = 5, tcl =  0.3)
> axis(side = 1, at = seq(65,75,1), labels = NA, pos = 5, tcl =  0.2)
> axis(side = 1, at = seq(65,75,5), labels = NA, pos = 5, tcl = -0.3)
> axis(side = 1, at = seq(65,75,1), labels = NA, pos = 5, tcl = -0.2)
```

CODE EXPLAINED: `nd` is a data frame with values of the explanatory variable(s) in the model(s), in this case only `Age`. `ci.pred` computes the predicted mortality rates from the models, `cbind` puts the columns side-by-side and finally `matshade` plots these with shaded confidence intervals, green for the model with age at diagnosis, and orange for the model with current age. The `log="y"` argument makes the y-axis logarithmic.

The `axis` statements each draw a horizontal axis without labels, at ages 65 to 75 years, located at $y = 5$ (`pos=5`). The `tcl` argument determines the size and direction of the tick marks; default value is -0.5, here we use ± 0.2 and ± 0.3

Both estimates of mortality show an exponentially increasing mortality (straight lines when using a log-scale), almost parallel. The orange curve shows the effect of *current* age, i.e. each of the many 3-month interval for a person has its own mortality, whereas the green curve refer to the age at entry (age at diabetes diagnosis) where the entire follow-up for a person has the same mortality. As seen from the small axis in the middle of the display, the curves are offset horizontally about 4 years. The average follow-up time (`lex.dur`) in the two data sets in the intervals where death occur are:

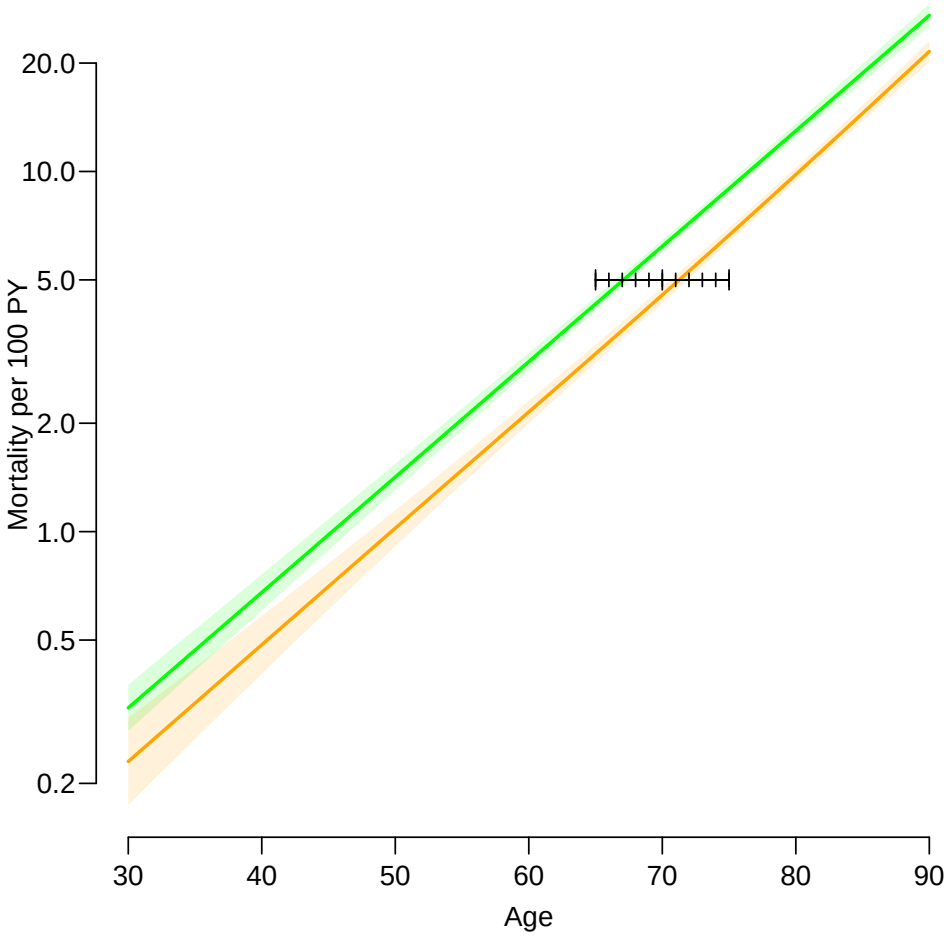


Figure 4.1: *Two models for mortality in DM patients. The green curve is based on estimates from data with one record per person (in which case* `Age` *represents age at DM diagnosis), the orange on data with 3-month intervals of follow-up (which means that* `Age` *represents current age).* `../graph/DMlate-mort0`

```
> with(subset(dmL, lex.Xst=="Dead"), mean(lex.dur))
[1] 4.12682
> with(subset(dmS, lex.Xst=="Dead"), mean(lex.dur))
[1] 0.1180175
```

We see that the difference of these means is about 4 years; so the age at onset is on average 4 years earlier than the left endpoint of the intervals representing current age. This is what is reflected in the horizontal distance of 4 years between the two curves. This is formally an inaccurate statement, there is nothing in the code that restricts the two curves to be parallel, they just happen to be so approximately.

The purpose of this example is to show that the modeling of rates as function of current age (or any other time scale) requires data subdivided so finely that the assumption of constant rates in the intervals is reasonable, but also that the left endpoint should be a reasonable representation of the age at risk (specifically the value of the time scale). These are credible assumptions for 3 month intervals, but hardly for intervals of 4 years or more.

Note that this is an analysis that targets the *rates* so this analysis does not have a counterpart in the Cox-model, that bypasses the baseline hazard (rate) and only yields hazard ratios (rate ratios).

`DMlate`

4.3 Survival analysis

The analyses presented in figure 4.1 shows the mortality as a function of age at diagnosis respectively current age. We might instead postulate that the mortality primarily depends on time since diagnosis of diabetes. All persons in our dataset are at risk from time 0 on this time scale, which is certainly not the case for the age as time scale.

With a common origin of time it makes sense to ask for the *survival* probabilities, the probability that a person with diabetes survives, say, 5 years from diagnosis. We can fit a model describing the mortality as a function of time since diagnosis using the time-split dataset `dmS`; the modeling looks very similar to the model with age as time scale:

```
> m2  <- gam.Lexis(dmS, ~ s(DMdur))
mgcv::gam Poisson analysis of Lexis object dmS with log link:
Rates for the transition:
DM->Dead
```

CODE EXPLAINED: The dataset `dmS` was split along the age-scale in intervals of 0.25 years, but the other time scales were calculated at these split times as well, see p. 28. Therefore we can just use the split times on the `DMdur` scale from in split dataset `dmS` when modeling the effect of diabetes duration.

And as for the mortality as function of age we can show the mortality as a function of time since diagnosis:

```
> nd <- data.frame(DMdur = seq(0, 15, 0.2))
> matshade(nd$DMdur, ci.pred(m2, nd) * 100,
+          plot = TRUE,
+           lwd = 2,
+          ylim = 0:1*10,
+          xaxs = "i",
+          yaxs = "i",
+          xlab = "Time since DM diagnosis (years)",
+          ylab = "Mortality per 100 PY")
```

CODE EXPLAINED: The data frame `nd` contains the times for `DMdur` where we want the mortality evaluated, used as 2nd argument to `ci.pred` to get the predicted rates with confidence intervals.

We are not using a log-scale for the rates because we will be using the *cumulative* rates, that is the integral—the area under the curve from 0 up to a given time (since diagnosis). By the same token we also show the origins of the axes at exactly 0 (`xaxs="i"` and `yaxs="i"`); this way the area shown in the plot corresponds to the cumulative rate.

We see from figure 4.2 that the mortality declines from time of diagnosis for the first 2 years, and only then start to increase slightly. The figure shows how the *mortality* looks as a function of time since diagnosis, but we wanted to see the *survival*, the probability of being alive at time t after diagnosis. The survival function is related to the mortality rate by:

$$S(t) = \exp\left(-\int_0^t \mu(s)\,\mathrm{d}s\right)$$

where $\mu(s)$ is the function in figure 4.2. Since the survival is a closed form function of the mortality rate, so is the (approximate) confidence intervals for the survival a function of the standard error of the estimated (log-)mortality, albeit a complicated one. This is conveniently wrapped in the function `ci.surv` which takes a model and a prediction data frame as arguments, assuming the records in the prediction frame correspond to equidistant times at which we want the survival probability computed:

```
> matshade(nd$DMdur, ci.surv(m2, nd, intl = 0.2),
+          plot = TRUE,
+           lwd = 2,
+          xaxs = "i",
+          yaxs = "i",
+          xlab = "Time since DM diagnosis (years)",
+          ylab = "Survival probabiliy")
> lines(survfit(Surv(dmL$lex.dur,
+                    dmL$lex.Xst == "Dead") ~ 1),
+       col = "red")
```

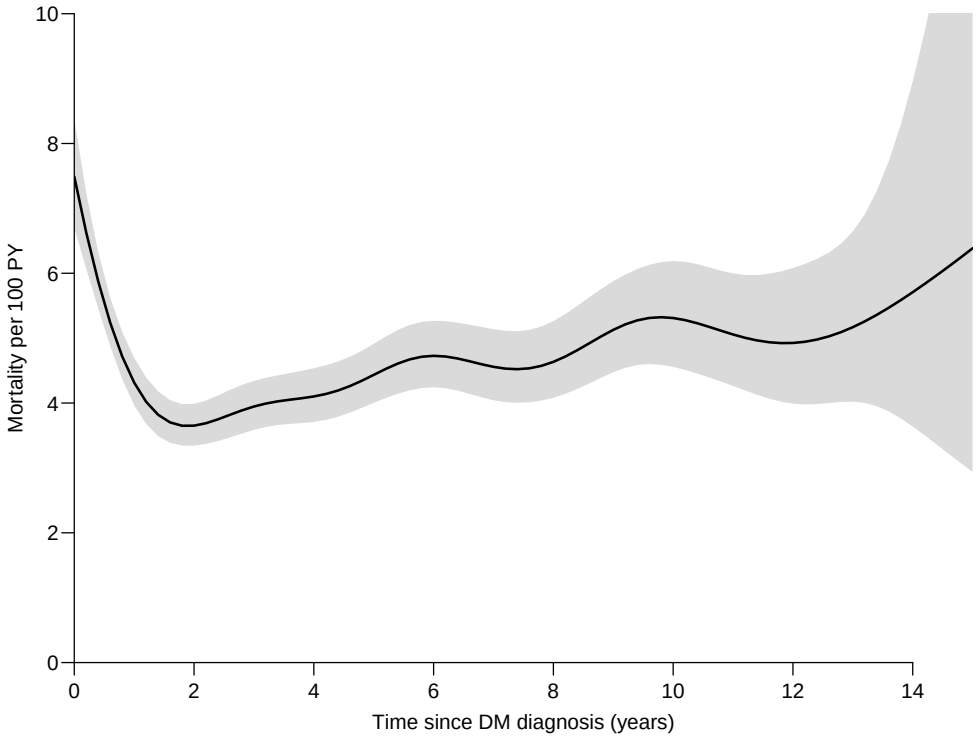


Figure 4.2: *Mortality of Danish diabetes patients as a function of diabetes duration.* `../graph/DMlate-mort2`

CODE EXPLAINED: The function `ci.surv` takes the estimated mortalities from the model `m2` at the points given in the data frame `nd`. Since the calculation of the survival requires numerical integration we must also supply the length of the intervals between points in `nd`, this is in the argument `intl`. `ci.surv` then computes the survival function at the points given in the second argument (called `ctr.mat`), and returns a 3-column matrix with estimated survival as the first column, and the confidence limits as the next two.

For comparison we also show the Kaplan-Meier curve derived by `survfit`.

From figure 4.3 we see that there is no practical difference between the two approaches to computing the survival function; the traditional Kaplan-Meier approach has some notches at the far end of the curve that are not biologically credible. But it is only the parametric approach that produces an estimate of the mortality function as in figure 4.2.

4.4 Multiple states: cutting follow-up

When splitting the follow-up we use a set of predefined (deterministic) points on a time scale, see page 28. A different type of subdividing the follow-up is *cutting* at times of (random) events. This will typically be at the time a person moves to a new state, such as "Insulin treated".

Note the terminology we have introduced here: *Splitting* is reserved for division of follow-up at pre-determined points on a time-scale (typically many `breaks`); *cutting* is reserved for division of follow-up at time of some intermediate event, typically constituting a transition to a new state.

These two endeavours are handled by different functions, splitting by `splitLexis` or `splitMulti` (the latter from the `popEpi` package), cutting by `cutLexis` and its cousins `mcutLexis` and `rcutLexis`.

4.4.1 Cutting follow-up at one intermediate event

To accommodate the beginning of insulin therapy, which starts at `doins`, follow-up must be cut in two for persons that begin insulin therapy, one piece before (that is, not on insulin) and one after (that is,

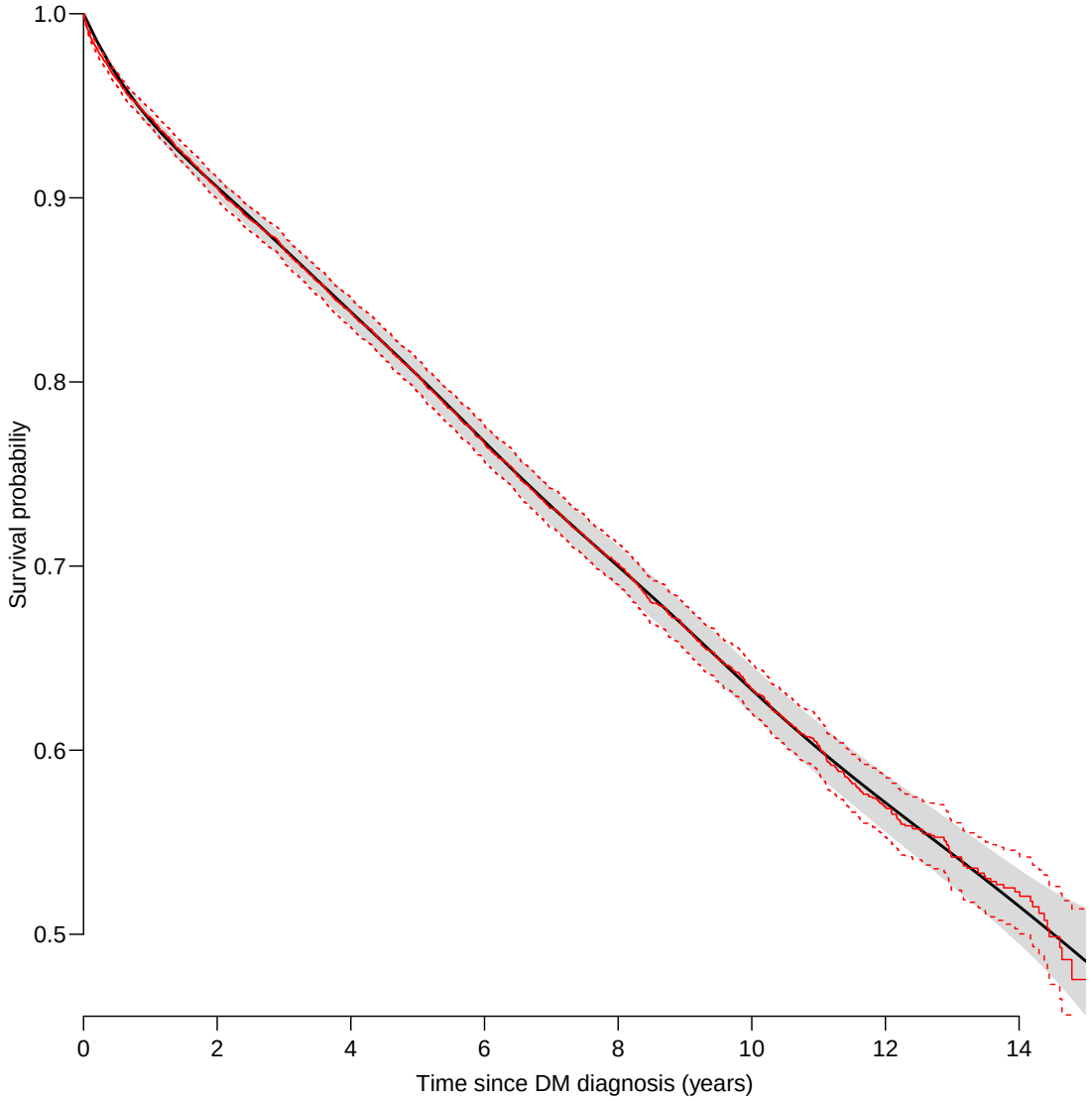


Figure 4.3: *Parametric and non-parametric survival curves. For clarity the y-axis does not start in 0. The smooth black curve and the shaded confidence interval are based on a parametric model for the mortality rates; the ragged red curve with broken lines as confidence interval is the Kaplan-Meier estimator of the survival function.* ../graph/DMlate-DMsurv

currently on insulin, or more precisely, ever on insulin). This is done by the function cutLexis:

```
> dmL <- sortLexis(dmL)
> dmI <- cutLexis(dmL,
+                 cut = dmL$doins,
+           timescale = "Per",
+           new.state = "Ins",
+           new.scale = "tfIns",
+         split.states = TRUE,
+     precursor.states = "DM")
```

CODE EXPLAINED: The cut argument is a vector of transition times (possibly with missing values) of length equal to nrow(dmL)—in this case a column in dmL, but it need not be. The timescale argument indicates which of the time scales the values of the cut argument refer to. new.state is the name of the state the person transitions to, and new.scale is the name of the time scale defined as time since entry to

the new state. `split.states=TRUE` indicates that the `Dead` state should be subdivided according to whether the entry to `Dead` is from state `Ins` or state `DM`.

Finally, `precursor.states` names those states that will be overridden by the new state; if a person that begins insulin previously ended in `DM` (that is, censored in `DM`), the person should now end in `Ins` (that is, still censored, but now in `Ins`), so `DM` is a precursor state. If a person previously ended in `Dead` the person should still end as dead after beginning insulin, so `Dead` is not a precursor state. The default value for `precursor.states` is `transient(dmL)`, the set of transient states—in this case only `DM`. It is rarely relevant to use anything but the default, so this argument is usually omitted.

```
> summary(dmI, timeScales = TRUE)
Transitions:
     To
From    DM  Ins Dead Dead(Ins)  Records:  Events: Risk time:  Persons:
  DM  6157 1694 2048         0      9899     3742   45885.49      9899
  Ins    0 1340    0       451      1791      451    8387.77      1791
  Sum 6157 3034 2048       451     11690     4193   54273.27      9996

Timescales:
  Per   Age DMdur tfIns
   ""    ""    "" "Ins"
```

CODE EXPLAINED: We see from the `summary` that the total number of persons and deaths is the same in `dmI` as in `dmL`. But the number of records is larger.

Asking for the time scales in `dmI` shows the four time scale variables, and that `tfIns` is defined as time since entry to the `Ins` state, whereas none of the other time scales are defined as time since entry

We can illustrate the result of cutting time by listing records from a few selected persons both from `dmL` and `dmI`. We select one person for each combination of missing/non-missing of `dodth` and `doins`

```
> (who <- c(
+ subset(dmL, !is.na(dodth) & !is.na(doins))[1,"lex.id"],
+ subset(dmL,  is.na(dodth) & !is.na(doins))[1,"lex.id"],
+ subset(dmL, !is.na(dodth) &  is.na(doins))[1,"lex.id"],
+ subset(dmL,  is.na(dodth) &  is.na(doins))[1,"lex.id"]))
[1] 28 15  6  1
> subset(dmL, lex.id %in% who)[,c(1:7,13)]
 lex.id     Per   Age DMdur lex.dur lex.Cst lex.Xst    doins
      1 1998.92 58.66     0   11.08      DM      DM       NA
      6 2007.89 80.02     0    2.04      DM    Dead       NA
     15 2002.55 58.13     0    7.45      DM      DM 2005.354
     28 1998.52 73.72     0    9.68      DM    Dead 2007.221
> subset(dmI, lex.id %in% who)[,1:8]
 lex.id     Per   Age DMdur tfIns lex.dur lex.Cst   lex.Xst
      1 1998.92 58.66   0.0    NA   11.08      DM        DM
      6 2007.89 80.02   0.0    NA    2.04      DM      Dead
     15 2002.55 58.13   0.0    NA    2.80      DM       Ins
     15 2005.35 60.93   2.8     0    4.64     Ins       Ins
     28 1998.52 73.72   0.0    NA    8.70      DM       Ins
     28 2007.22 82.43   8.7     0    0.98     Ins Dead(Ins)
```

CODE EXPLAINED: We select one person (the first, by "`[1,`") among those who have or have not a valid date of death, respectively date of insulin, and list the relevant variables for the four chosen persons from `dmL` and `dmI`.

Persons 1 and 6 do not have a date of insulin so they have one record each in both data sets, with `lex.Cst` equal to `DM`. Persons 15 and 28 both have a `doins`, so their follow-up is cut in two pieces at `doins`. Person 15 remains in state `Ins` (is censored there), while person 28 dies. So `cutLexis` keeps track of the absorbing

`DMlate`

states (in this case Dead), and makes sure that transitions to absorbing states are maintained, as is the case for person 28.

Note that the total lex.dur is the same within persons in the two data sets, but in the data frame dmI persons 15 and 28 spend their time in two different states, as indicated in lex.Cst.

The time scale tfIns (time from Insulin) is NA for the records representing follow up in state DM, and 0 for the new records—at the beginning of insulin the time since insulin is of course 0, but the variable tfIns is now defined as a time scale and will therefore be properly updated when we split follow-up.

4.4.1.1 Graphical representation of states and transitions

Lexis version A graphical version of the summary is provided by the boxes.Lexis function, that shows each state as a box and each transition as an arrow, optionally annotated by the summaries:

```
> boxes(dmI, boxpos = list(x = c(15,85,15,85),
+                          y = c(85,85,15,15)),
+           scale.R = 1000,
+           show.BE = TRUE)
```

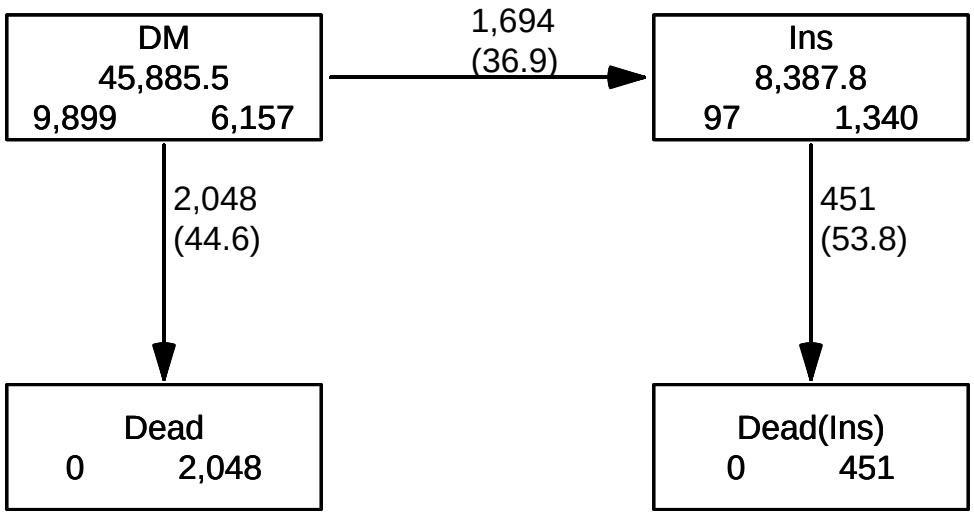


Figure 4.4: *Multistate model for beginning of insulin use and death among diabetes patients. The number in the center of each box is the total follow-up time. The numbers at the bottom of each box are the number of persons beginning, resp. ending their follow-up in the state. The numbers on the arrows are the number of transitions and in brackets the transition rates per 1000 person-years.* ../graph/DMlate-box3

CODE EXPLAINED: The boxes derives the states from the Lexis object dmI, the argument boxpos places the boxes in a coordinate system that ranges from 0 to 100 in both directions; the x and y are the coordinates of the center of the boxes. scale.R scales the rates that are printed along the transition arrows, in this case to rates per 1000 PY. show.BE prints the number of persons that begin resp. end their follow-up in each state.

The resulting plot is the same as the one in the introduction in chapter 1.

cohorttools version There is a function in the cohorttools package that automatically puts the states in an order attempting to avoid crossing arrows. It gives access to a lot of features from the graphviz program devised for drawing of graphs:

```
> library(cohorttools)
> kk <- boxesLx(dmI, layout = "dot",
+                    rankdir = "LR", # Left to Right
+                    show.gr = FALSE,
+                    scale.Y = 100,
+               show.persons = TRUE,
+              prop.penwidth = TRUE)
> gv2image(kk, file = "../graph/DMlate-gvboxes", type = "pdf")
```

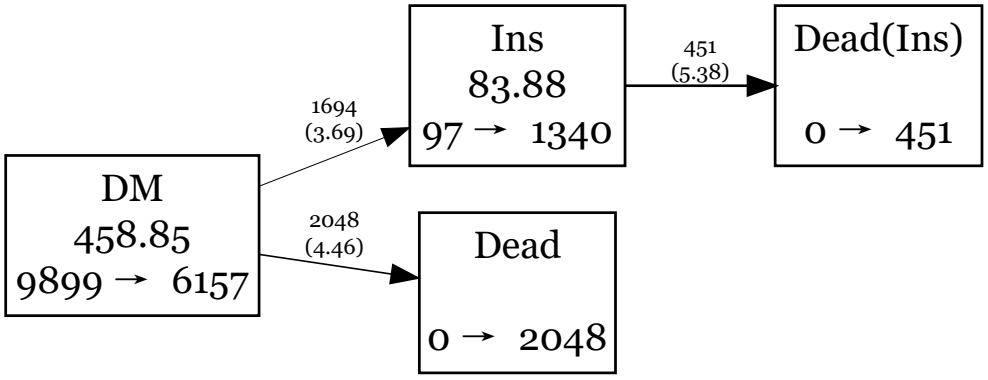


Figure 4.5: `graphviz` *version of multiple states graph. The* `prop.penwidth` *makes the widths of the arrows porportional to the log-rates.* `../graph/DMlate-gvboxes`

The funny filename of the graph is chosen to comply with those automatically generated by `Sweave` for the other graphs in this book.

The beauty of the result of `boxesLx` can be debated, but `graphviz` has a lot of possible parameters available that can be accessed from `boxesLx`, see `?boxesLx` for links etc.

4.4.2 Cutting follow-up at multiple intermediate events

When multiple intermediate events can occur, `A` and `B`, say, it will matter which order you cut your follow-up; if the time sequence for a person is `B` then `A`, and we cut data first at occurrence of `A` and then at `B`, we might accidentally override the state `A` by `B`. Handling this is really clumsy; the solution is basically to cut in order of occurrence of the events for each person, but it will involve a lot of subsetting, cutting and stacking.

Therefore, there is a function `mcutLexis` that does this properly and keeps track of the history; here we cut at the date of beginning insulin therapy (`doins`) and date of starting oral anti-diabetic drugs (`dooad`)—note that we are starting all over again from the `dmL` object with no cuts:

```
> dmI0 <- mcutLexis(dmL,
+                   wh = c("dooad", "doins"),
+            timescale = "Per",
+            new.states = c("OAD", "Ins"),
+            new.scales = c("t.OAD", "t.Ins"),
+          ties.resolve = 1/365.25)
NOTE: Precursor states set to DM
NOTE: 15 records with tied events times resolved (adding 0.002737851 random uniform),
      so results are only reproducible if the random number seed was set.
```

CODE EXPLAINED: `mcutLexis` assumes that the times of the transitions are stored as variables in `dmL`, named in the argument `wh`—so a bit more restrictive data requirements than for `cutLexis`. `mcutLexis` requires that there is at most one record per person per state. Thus transition to a state (`lex.Xst`) can occur at most once for each person.

The names of the new states assumed at the times given in the variables named by `wh` are given in the argument `new.states`. There is also the possibility of defining new time scales defined as time since entry

into each of the two states, via the new.scales. Finally, where the two dates dooad and doins are identical, a decision must be made as to which occurred first. If you do not resolve that by hand prior to calling mcutLexis, ties.resolve=TRUE will add a bit of random noise to the identical dates so ties are broken. If ties.resolve=FALSE, the function will fail if any two transition dates for a person are recorded as equal.

To see what happens with mcutLexis we can list the result for two select persons:

```
> wh1 <- subset(dmIO, lex.Cst == "OAD-Ins")$lex.id[1]
> wh2 <- subset(dmIO, lex.Cst == "Ins-OAD")$lex.id[1]
> who <- c(wh1, wh2)
> subset(dmIO, lex.id %in% who)[,1:9]
 lex.id     Per   Age DMdur t.OAD t.Ins lex.dur lex.Cst lex.Xst
     18 1996.75 61.72  0.00    NA    NA    1.17      DM     OAD
     18 1997.92 62.89  1.17  0.00    NA    8.08     OAD OAD-Ins
     18 2005.99 70.97  9.25  8.08  0.00    4.00 OAD-Ins OAD-Ins
     38 2008.37 63.93  0.00    NA    NA    0.09      DM     Ins
     38 2008.46 64.02  0.09    NA  0.00    0.21     Ins Ins-OAD
     38 2008.67 64.24  0.31  0.00  0.21    1.33 Ins-OAD    Dead
```

CODE EXPLAINED: First we extract the ids of persons who has been in states Ins-OAD, or OAD-Ins, and take the first of each of these. We then print the records from the cut data frame dmIO for these persons. Note the new time scales (t.OAD and t.OAD); they are NA until the state is assumed, at which point they are 0, the time since, say, OAD at the time of beginning OAD.

```
> summary(dmL)

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

> dmIO <- Relevel(dmIO, c("DM","Ins","Ins-OAD",
+                         "OAD","OAD-Ins","Dead"))
> summary(dmIO)

Transitions:
     To
From        DM  Ins Ins-OAD  OAD OAD-Ins Dead  Records:  Events: Risk time:  Persons:
  DM      2830  688       0 2958       0 1056      7532     4702   22920.27      7532
  Ins        0  462     171    0       0  152       785      323    3883.07       785
  Ins-OAD    0    0     137    0       0   34       171       34     715.24       171
  OAD        0    0       0 3327    1006  992      5325     1998   22965.25      5325
  OAD-Ins    0    0       0    0     741  265      1006      265    3789.45      1006
  Sum     2830 1150     308 6285    1747 2499     14819     7322   54273.27      9996
```

CODE EXPLAINED: When comparing to dmL we see that the number of records in dmIO is some 50% larger because of the extra records for persons that see an intermediate event.

The ordering of states in the resulting Lexis object depends on the dataset, so you will often want to reorder states "by hand", using Relevel in order to get a meaningful ordering of states in the summary.

From the summary we see that mcutLexis generates not only states OAD and Ins, but also the states OAD-Ins and Ins-OAD according to the order in which states were visited. There is an argument to mcutLexis, seq.states=FALSE that will cause the two latter states to be merged to the state Ins+OAD (ordering of state names is alphabetic).

When there are many states are in the model, a graphical representation is particularly important; here we show the multistate data with and without distinguishing between the sequence of OAD and Ins:

```
> par(mfrow = c(2,1))
> boxes(dmIO,
+       boxpos = list(x = c(15,50,85,50,85,50),
+                     y = c(50,90,70,15,30,50)),
+       scale.R = 1000,
+       show.BE = TRUE)
> boxes(Relevel(dmIO,
+               list("DM","OAD","Ins+OAD" = c(3,5),"Ins","Dead"),
+               first = FALSE),
+       boxpos = list(x = c(15,50,85,50,50),
+                     y = c(50,90,50,15,50)),
+       scale.R = 1000,
+       show.BE = TRUE)
```

CODE EXPLAINED: We reordered the levels of `dmIO`, and so the ordering of the components of the `boxpos` list elements codex and codey will refer to this ordering. The `scale.R` scales the rates shown on the arrows and `show.BE = TRUE` shows the number of persons who begin and end follow-up in each state.

We can also use `boxesLx` that automatically places boxes and draws non-interfering arrows, but it only allows arrows in one direction (either horizontal or vertical):

```
> kk <- boxesLx(dmIO, layout = "dot",
+                     rankdir = "LR", # Left to Right
+                     show.gr = FALSE,
+                     scale.Y = 100,
+                show.persons = TRUE,
+               prop.penwidth = TRUE)
> gv2image(kk, file="../graph/DMlate-xxboxes", type="pdf")
```

4.5 Joint model for several transition rates

We can use the `Lexis` object `dmIO` as basis for modeling the 5 different mortality rates, it is however apparent from figure 4.6 that the transition `Ins-OAD` → `Dead` is poorly determined with only 33 transitions so we will merge the two states `Ins-OAD` and `OAD-Ins`, so we only model the 4 mortality rates in the lower panel of figure 4.6.

The reason that we can model the 4 mortality rates in one model based on the `Lexis` data frame `dmIO` is that the four rates originate from 4 different states, so the joint likelihood only include each piece of risk time once—see the section on likelihood for multiple transitions, p. 23.

But first we must split the follow-up in small intervals in order to be able to interpret effects of age and diabetes duration as current age and current duration. We use 3 month intervals as before:

```
> sIO <- splitMulti(Relevel(dmIO,
+                           list("Ins+OAD" = c("Ins-OAD", "OAD-Ins"),
+                                "Dead" = "Dead"),
+                           first = FALSE),
+                   Age = seq(0, 120, 1/4))
> summary(sIO)
Transitions:
     To
From          DM    Ins   OAD Ins+OAD Dead  Records:  Events: Risk time:  Persons:
  DM       94476   688  2958       0 1056     99178     4702   22920.27      7532
  Ins          0 15996     0     171  152     16319      323    3883.07       785
  OAD          0     0 95193    1006  992     97191     1998   22965.25      5325
  Ins+OAD      0     0     0   18890  299     19189      299    4504.69      1177
  Sum      94476 16684 98151   20067 2499    231877     7322   54273.27      9996
```

CODE EXPLAINED: Before we split the follow-up, we use `Relevel` to merge the states `Ins-OAD` and `OAD-Ins` to one called `Ins+OAD`. Note that we redundantly mention `Dead` in `Relevel`; it has the effects of putting `Dead` as the last level, giving a nicer summary, while the `first = FALSE` puts the newly created levels as the last ones.

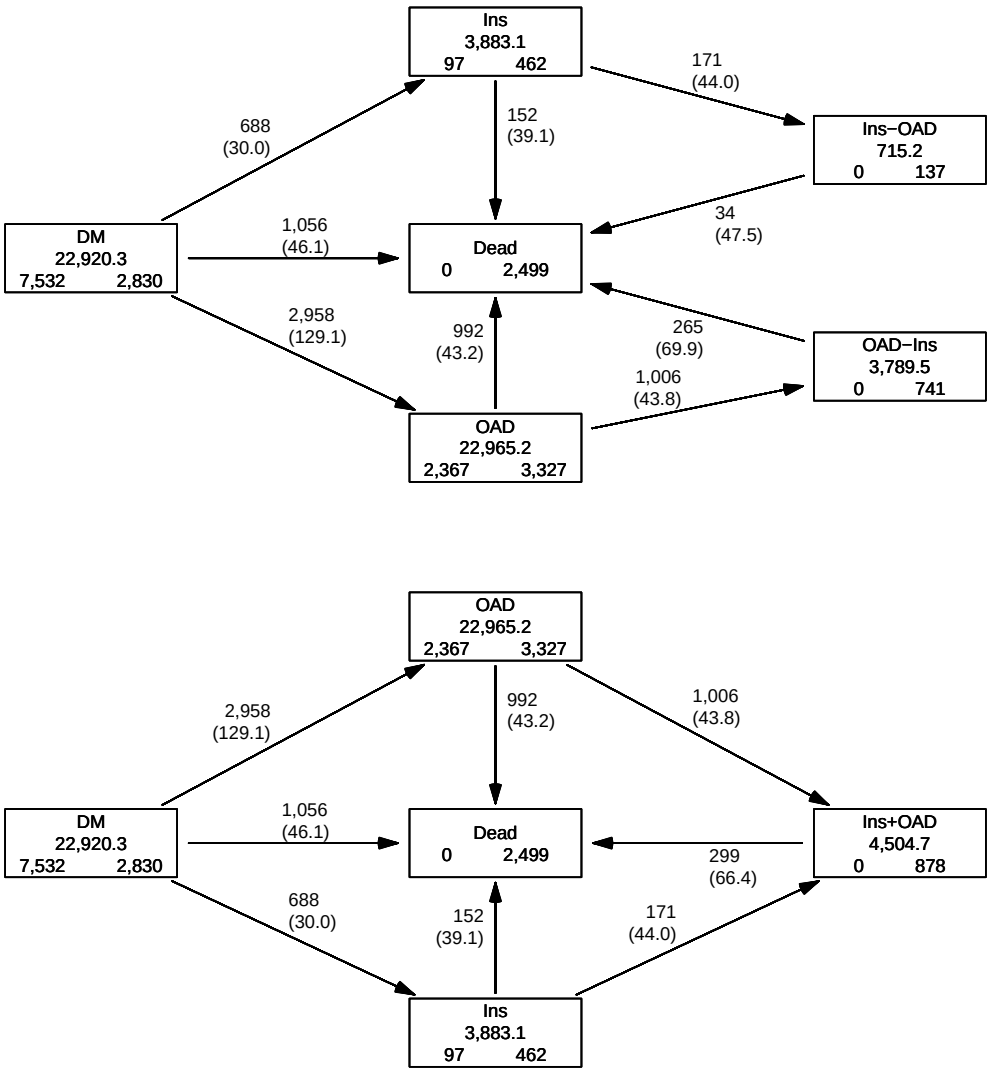


Figure 4.6: *States and transitions for the* `DMlate` *data with and without distinguishing between the sequence of* `OAD` *and* `Ins`. `../graph/DMlate-mbox2`

We can model the mortality rates in one model, where we take the rates to be proportional between the states before death, simply by including `lex.Cst` as a covariate. Because we still have only one absorbing state and we model all transitions into this it is particularly simple to specify the model for all mortality rates taking current age and diabetes duration into account:

```
> mph <- gam.Lexis(sIO, ~ s(Age) + s(DMdur) + lex.Cst)
```

Formally we would use:

```
> system.time(
+ mph <- gam.Lexis(sIO, ~ s(Age) + s(DMdur) + lex.Cst,
+                   to = "Dead"))
mgcv::gam Poisson analysis of Lexis object sIO with log link:
Rates for transitions:
DM->Dead
Ins->Dead
OAD->Dead
```

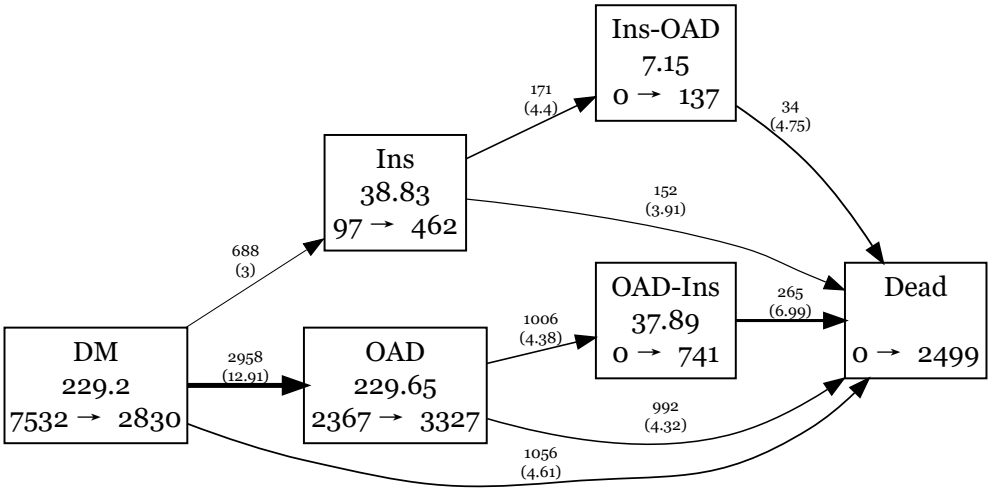


Figure 4.7: ***graphviz*** *version of the multiple states. The graph is drawn with arrow-widths proportional to the corresponding rates.* `../graph/DMlate-xxboxes`

```
Ins+OAD->Dead

   user  system elapsed
  25.23    2.14   27.40
```

CODE EXPLAINED: If neither `from` or `to` are specified as arguments in `gam.Lexis`, the default is to model all transitions into any absorbing state. In this case the only absorbing state is `Dead`, so the default is to model all 4 mortality rates together in the same model.

If only `to` is specified, all rates into the states indicated in `to` are modeled. If only `from` is specified, all rates out of the states indicated in `from` are modeled.

We model the mortality rates as depending on both age (`Age`) and duration of diabetes (`DMdur`), and also include the factor `lex.Cst` in the model; this will introduce different levels of mortality for the four different states. But we assume the same (non-linear) shape of effects of age and diabetes duration for all 4 mortality rates; a proportional hazards model, that is.

```
> summary(mph)
Family: poisson
Link function: log

Formula:
cbind(trt(Lx$lex.Cst, Lx$lex.Xst) %in% trnam, Lx$lex.dur) ~ s(Age) +
    s(DMdur) + lex.Cst

Parametric coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)    -3.795076   0.040907 -92.774   <2e-16 ***
lex.CstIns      0.991018   0.088113  11.247   <2e-16 ***
lex.CstOAD     -0.008453   0.044946  -0.188    0.851
lex.CstIns+OAD  0.659929   0.070357   9.380   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
           edf Ref.df Chi.sq p-value
s(Age)   1.006  1.011 1987.5  <2e-16 ***
s(DMdur) 7.523  8.453  136.3  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`DMlate`

```
R-sq.(adj) =  0.00212   Deviance explained = 9.31%
UBRE = -0.89139  Scale est. = 1         n = 231877
> round(ci.exp(mph, subset = "lex"), 2)
               exp(Est.) 2.5% 97.5%
lex.CstIns          2.69 2.27  3.20
lex.CstOAD          0.99 0.91  1.08
lex.CstIns+OAD      1.93 1.69  2.22
```

CODE EXPLAINED: `summary` gives a summary of the model, including the parameters associated with `lex.Cst`, which are the log rate ratios of mortality relative to the `DM` state. The section on smooth terms reveals the the effective degrees of freedom (`edf`) for the `Age` term is just about 1, meaning that the age term is almost linear. The p-value is not a test of non-linearity, it is a test of no effect.

`ci.exp` extracts those parameters with a name where "`lex`" appear, and exponentiates them, so we get the mortality rate-ratios relative to the mortality in the `DM` state.

We see that the persons in `OAD` alone do not have a different mortality from those not treated (`DM`), the reference level. Those on `Ins` alone have a mortality RR of 2.7 and those on both a RR of 1.9, significantly different since the confidence intervals do not overlap.

The model `mph` is a proportional hazards model since we assume that the 4 mortality rates are proportional; the mortality rates depend on age and disease duration in the same way: the RR between persons in any two of the states is assumed to be the same regardless of age and duration of diabetes.

4.5.1 Two time scales

The model `mph` is a proportional hazards model and we can get an impression of how mortality rates looks for different combinations of age and duration, by defining a prediction data frame; a dataset with select values for the covariates in the model `mph`:

```
> # traditional approach
> nd <- transform(data.frame(expand.grid(DMdur = c(NA, seq(0, 15, 0.2)),
+                                        AgeDM = seq(40, 70, 10)),
+                            lex.Cst = "DM"),
+                 Age = AgeDM + DMdur)
> # tidyverse approach
> nd <- expand.grid(DMdur = c(NA, seq(0, 15, 0.2)),
+                   AgeDM = seq(40, 70, 10)) %>%
+       mutate(lex.Cst = "DM",
+                  Age = AgeDM + DMdur)
> head(nd)
  DMdur AgeDM lex.Cst  Age
1    NA    40      DM   NA
2   0.0    40      DM 40.0
3   0.2    40      DM 40.2
4   0.4    40      DM 40.4
5   0.6    40      DM 40.6
6   0.8    40      DM 40.8
```

```
> matshade(nd$Age, ci.pred(mph, nd) * 100,
+          plot = TRUE,
+           log = "y",
+           lwd = 2,
+          ylim = c(0.25, 50),
+          xlab = "Current age (years)",
+          ylab = "Mortality per 100 PY")
> xa <- seq(40, 70, 10)
> ya <- exp(seq(log(8), log(50),, 4))
> for (i in 1:4)
+     {
+ axis(side = 3, at = 0:3*5+ xa[i], labels = 0:3*5, pos = ya[i])
+ axis(side = 3, at = 0:15 + xa[i], labels = NA   , pos = ya[i],
```

```
+                                                  tcl = -0.3 )
+     }
> matshade(nd$Age,
+          ci.pred(mph,
+                  transform(nd, lex.Cst = "Ins")) * 100,
+          lwd = 3,
+          lty = "22")
```

CODE EXPLAINED: We want a prediction data frame with variables `Age` and `DMdur`, but for combinations of (a few) ages *at diagnosis* and a range of values of diabetes duration. `expand.grid` returns a data frame with all combinations of the arguments, the first argument varying fastest. The model does not have `AgeDM` (`Age` at `DM` diagnosis) as predictor but rather `Age` (current age), so this is computed in `mutate`. WE have shown the data frame manipulation if two different ways; it gives the same result.

Note we inserted an `NA` at the beginning of the `DMdur` values; this is in order to break the predicted line at each new age at diagnosis; otherwise the curves would have been connected. In general if there is an `NA` in either the x or the y coordinate when a vector plotted as a line, then the two points on either side of the `NA` will not be connected.

To illustrate the joint effects of age and duration we also included horizontal axes showing the duration of diabetes for the four curves—`xa` are the left end points of the axes, `ya` are the vertical position the axes used in the `pos` argument.

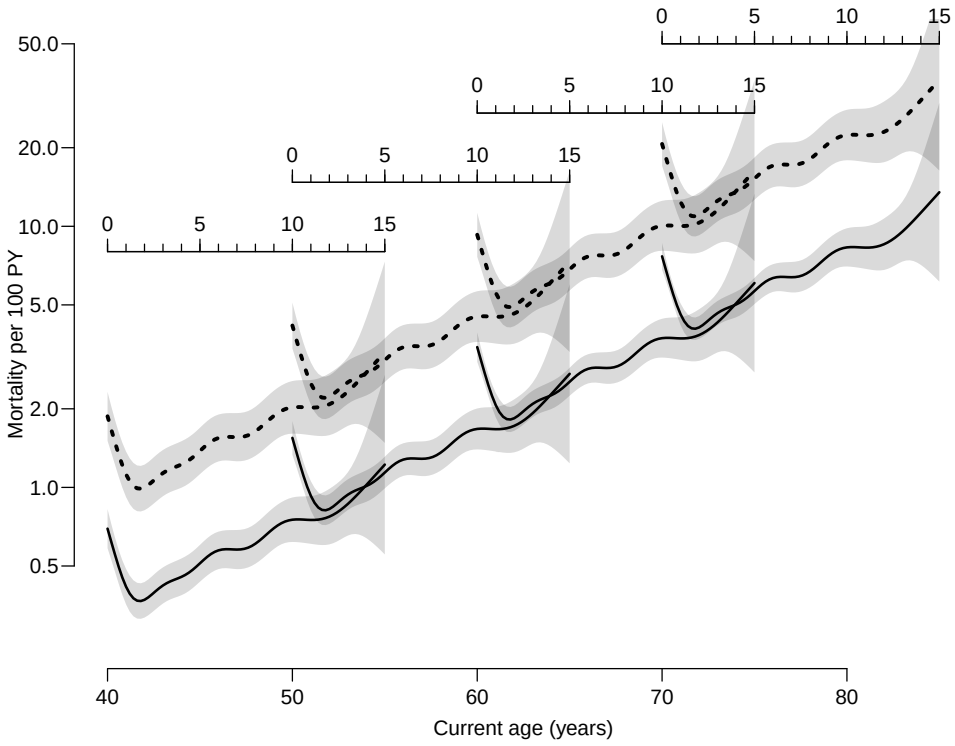


Figure 4.8: *Mortality by current age and duration of diabetes for persons in states* `DM` *and* `Ins`*. Full lines are for persons in state* ***DM****, broken lines for persons in state* `Ins`*. There is one curve for each age at diagnosis: 40, 50, 60 and 70, and an axis indicating the diabetes duration for each of the four curves.*
Note that mortality rates actually decline *the first two (well 1.7) years after diagnosis.*
`../graph/DMlate-age-dur`

Figure 4.8 shows the mortality among persons in the `DM` state as a function of current age and diabetes duration, for persons diagnosed with DM at ages 40, 50, 60 and 70.

`DMlate`

We see that the mortality is high at time of diagnosis, but decreases during the two first years, and subsequently increase exponentially by age, with no detectable difference between different ages at diagnosis. Since there is no interaction between Age, DMdur and lex.Cst, we have a proportional hazards model; the hazard ratio (mortality rate-ratio) between the 4 states is assumed to be constant on both of the two time scales.

Note however that the 8 curves in figure 4.8 are not identical even though they look so in this case; they consist of a duration curve over the span 1–15 years (identical for all 8 curves) but combined by the effect of current age over the span 40–85. Thus the curves in full lines and broken lines are pairwise proportional. But the 4 curves in each line type are slightly different; they look deceptively similar because the age-effect in this case is almost linear.

The initial peak (at duration 0) in mortality is presumably a clinical artifact: some of the newly diagnosed persons are diagnosed in connection with some other severe disease, such as for example cardiovascular disease. Therefore there is an over-representation of quite sick people among the newly diagnosed. They contribute to the initial high mortality, but as this part of the newly diagnosed die out, the mortality among the remaining reflect the mortality among "normal" diabetes patients.

4.5.2 Proportional hazards?

In normal statistical terms this would simply be called a main-effects model, but "proportional hazards" has become the standard terminology when analyzing occurrence rates. Which is unfortunate since "test for non-proportionality" has become a term used for test of interactions with time scales. In this instance a test of non-proportionality would be to test against a model with interactions between the time scales and lex.Cst:

```
> mnph <- gam.Lexis(sI0, ~ s(Age  , by = lex.Cst) +
+                          s(DMdur, by = lex.Cst) +
+                          lex.Cst)
> anova(mnph, mph, test = "Chisq")
```

We may also use a more elaborate interaction where we have an interaction between age and duration, here via a non-linear term in the difference, and fitted by glm using natural splines. There are many different ways of specifying an interaction, but the code to graph the estimated mortality rates will be the same if the interactions are specified in terms of the original variables (and no special variables are constructed):

```
> mpph <- glm.Lexis(sI0, ~ Ns(Age  , knots = 1:4*20) * lex.Cst +
+                          Ns(DMdur, knots = c(0,1,5,10)) * lex.Cst +
+                          Ns(Age -
+                             DMdur, knots = 5:8*10) * lex.Cst +
+                          lex.Cst)
stats::glm Poisson analysis of Lexis object sI0 with log link:
Rates for transitions:
DM->Dead
Ins->Dead
OAD->Dead
Ins+OAD->Dead
```

But besides the test for interaction we will of course want to know *how* the interaction looks; in this case we have estimated 4 different curves for age-effect and 4 different curves for the effect of diabetes duration. For each state they can of course be combined as in the previous section; the graph is produced by the same code, only the model is changed:

```
> clr <- c("forestgreen","darkorange")
> matshade(nd$Age, ci.pred(mpph,
+                          transform(nd, lex.Cst = "DM")) * 100,
+          plot = TRUE,
+           log = "y",
+           lwd = 2, lty = 1, col = clr[1], alpha = 0.10,
+          ylim = c(0.25, 50),
+          xlab = "Current age (years)",
+          ylab = "Mortality per 100 PY")
> xa <- seq(40, 70, 10)
```

```
> ya <- exp(seq(log(8), log(50),, 4))
> for (i in 1:4)
+     {
+ axis(side = 3, at = 0:3*5+ xa[i], labels = 0:3*5, pos = ya[i])
+ axis(side = 3, at = 0:15 + xa[i], labels = NA   , pos = ya[i],
+                                                   tcl = -0.3 )
+     }
> matshade(nd$Age,
+          ci.pred(mpph,
+                  transform(nd, lex.Cst = "Ins")) * 100,
+          lwd = 2, lty = "22", col = clr[1], alpha = 0.10)
> matshade(nd$Age,
+          ci.pred(mpph,
+                  transform(nd, lex.Cst = "OAD")) * 100,
+          lwd = 2, lty = 1, col = clr[2], alpha = 0.10)
> matshade(nd$Age,
+          ci.pred(mpph,
+                  transform(nd, lex.Cst = "Ins+OAD")) * 100,
+          lwd = 2, lty = "22", col = clr[2], alpha = 0.10)
```

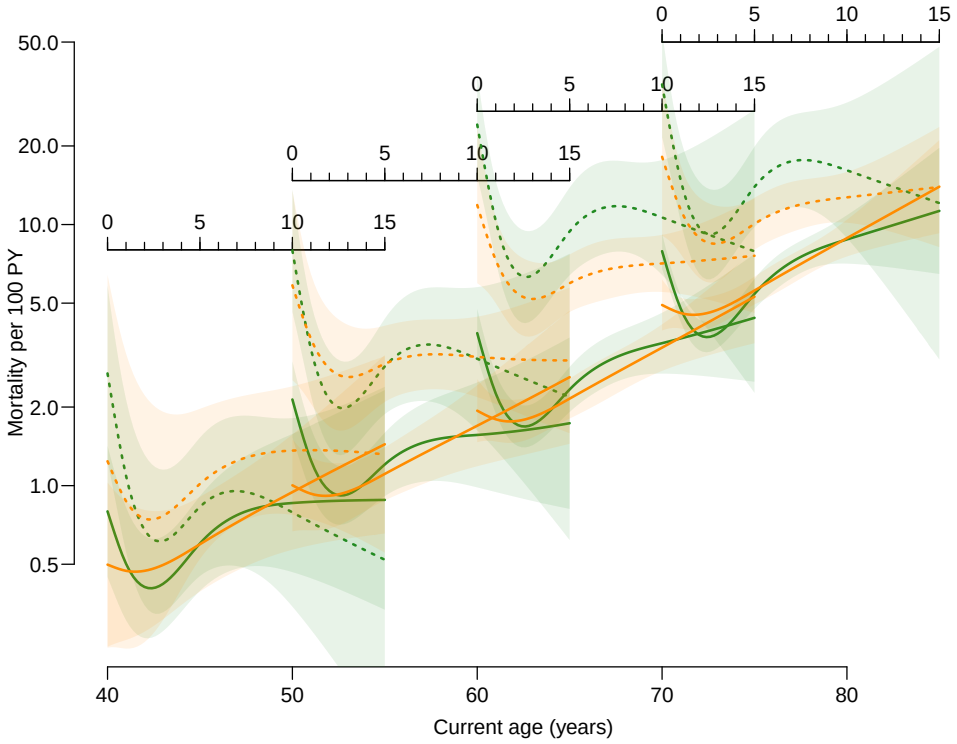


Figure 4.9: *Mortality by current age and duration of diabetes for persons in all states. Full lines are for persons in states not including insulin (*DM,OAD*), dotted lines for persons in* Ins *or* Ins+OAD*. Green line are persons not on OAD, orange for persons on OAD. There is one curve for of each type and color for each age at diagnosis: 40, 50, 60 and 70, and an axis indicating the diabetes duration for each of the four curves.* ../graph/DMlate-age-dur-int

The model we are using has four transient states, defined by cross-classification of the presence or absence of insulin and of oral antidiabetic drugs. The mortality rates from different states are shown in dotted lines if the originating state is an insulin state and full lines if not. And they are colored orange if the state includes oral antidiabetic drugs and green if not. This is a specific feature for this particular model: the states correspond to a cross-classification of two factors, and the colours and line types correspond to the two factors (insulin respectively OAD) that defines the 4 states. On top of this we have curves for select values of age at diagnosis, but those are clear from the location of the curves.

. . . now input from datarep

datarep

4.6 Recurrent multiple intermediate events

In cases of recurrent (for example curable or regressing) disease states we may not be interested in keeping track of the entire history of visited states of a person, but only care about the current (*i.e.* latest visited) state. One example would be kidney disease as assessed by albuminuria, where urine albumin levels < 30 mmol/l is considered normal and levels above 30 and 300 mmol/l are called micro-, respectively macro-albuminuria (this is not the correct clinical diagnosis, though). In this case we would only be interested in the state a person is currently in, not the (sequence of) states previously visited.

The construction of `Lexis` objects that correspond to this type of multistate model is handled by the function `rcutLexis`, which besides a `Lexis` object takes as input times and (names of) states entered at these times for each person.

This is illustrated in detail in section 4.8, "A real world example" below. The situation where one type of event is repeatedly observed in the same person is illustrated in chapter 5

4.7 Adding time-varying covariate information

So far we have only been concerned with time-dependent transitions between states, but we will also want to assess the effect of covariates measured at irregular time points on the transition rates.

When adding time-varying covariate information, we will gain the possibility to assess how the value of these influence the transition rates between states. But we will lose the possibility of predicting state probabilities—we shall return to this later.

There are (at least) two substantially different types of added time-varying covariate information:

1. Information obtained at a given date is carried unchanged forward until the next piece of information becomes available. This is normally called "last observation carried forward" (LOCF). This will typically be used for clinical measurements taken at different follow-up times, such as cholesterol, blood pressure etc.

2. Interpolation between adjacent measurements is used to assign covariate values at given times between the two measurement times.
   This will typically be used to construct covariates such as cumulative amounts of drug consumed or other exposure measurements in pharmaco-epidemiological studies. Since we are doing some sort of interpolation from the raw data, we can in principle compute the covariate values at any time of follow-up, so this will also require decisions about the times at which we evaluate the exposure.

4.7.1 Adding clinical measurements: `addCov.Lexis`

Persons may attend clinical visits or there may be administrative updates performed at (possibly irregular) times. In that case we would like to amend the follow-up in a `Lexis` object with values of the most recent measurements from the clinical visits, but without defining a new state in the model.

One way to do this is to introduce a cut of the follow-up at the date of each clinical visit, in order to add the clinical variables measured at the visit to records representing follow-up after the clinical visit (until the next). This is known as last observation carried forward (LOCF). This is also the way that `splitLexis`, `cutLexis`, `mcutLexis` and `rcutLexis` operates; when the follow-up in a record is cut in two, both resulting records will have the same values as for all variables in the original record.

Furthermore, we might want to keep track of the time since the most recent visit. These features are available in the function `addCov.Lexis`. This is further detailed in the appendix, section 11.2.1, where the function is described and contrasted to `addDrug.Lexis`, and also illustrated in section 4.8 "A real world example" below in this chapter

4.7.2 Adding drug exposure: `addDrug.Lexis`

In pharmaco-epidemiology we want to add information on drug exposure to the `Lexis` object; we want the current drug exposure status computed at each time represented in the `Lexis` object. The data on drug exposure will come from either records of drug purchases (filled prescriptions) or records of issued prescriptions.

In both cases we will for each person have date of drug (purchase or prescription), amount of drug and possibly a prescribed dosage (how much to take per day).

We want the `Lexis` object amended with the current drug exposure at the beginning of each interval represented in the `Lexis` object.

There are a many different ways to define current drug exposure such as: 1) is the person currently on drug, 2) time since first exposure, 3) cumulative time on the drug or 4) cumulative dose of the drug. The calculation of these variables for each record is implemented in `addDrug.Lexis`, with a choice of method for defining the 4 mentioned variables from the purchase / prescription records. .

Unlike adding clinical measurements by `addCov.Lexis`, the use of `addDrug.Lexis` *must* be the last operation on the `Lexis` object. This is because `addDrug.Lexis` computes exposure variables at each time point in the `Lexis` object by interpolation. Thus any further subdivision of time using LOCF will generate time points with the variable values copied from previous time points, and thus not computed by interpolation as desired.

This aspect is further detailed in the appendix, section 11.2, where the function `addDrug.Lexis` is described and contrasted to `addCov.Lexis`.

. . . now input from `steno2`

4.8 Recurrent events and time dependent covariates: Steno2 trial

We shall illustrate the facilities of `Lexis`, `rcutLexis`, `addCov.Lexis` and `splitMulti` using the Steno2 data (read the documentation using `?steno2`).

Briefly, the data is a simulated version of the original Steno 2 data; no persons can be identified; but the results will be broadly similar to analysis based on the original Steno 2 data. The study randomized T2 diabetes patients with micro-albuminuria to either standard treatment or intensive treatment. Enrollment were from beginning of 1993 through mid 1994. There were 80 patients in each group; the intensive care turned out to be superior, so in 2001 it was decided to put all on the intensive care arm. The dataset `steno2` contains follow-up till the end of 2014.

The dates in `steno2` are stored in `Date` format so we convert all of them to `cal.yr` format using the function `cal.yr`:

```
> data(steno2)
> steno2 <- cal.yr(steno2)
```

CODE EXPLAINED: The data frame `steno2` has all dates represented as `Date` variables, so by `cal.yr` argument) we convert them all to `cal.yr` format, so risk time will be counted in years (defined as 365.25 days).

4.8.1 A `Lexis` object

We then devise a `Lexis` object with the follow-up from baseline of the study, `doBase`, to end of follow-up, `doEnd`:

```
> # ------------------------------
> # to be fixed in the data: a few deaths are a few days after doEnd
> steno2$doEnd <- pmax(steno2$doEnd, steno2$doDth, na.rm = TRUE)
> # ------------------------------
> L2 <- Lexis(entry = list(per = doBase,
+                          age = doBase - doBth),
+             exit = list(per = doEnd),
+       exit.status = factor((!is.na(doDth)) + deathCVD,
+                            labels=c("Mic", "D(oth)", "D(CVD)")),
+               id = id,
+             data = steno2)
NOTE: entry.status has been set to "Mic" for all.
```

`steno2`

```
> summary(L2)

Transitions:
     To
From  Mic D(oth) D(CVD)  Records:  Events: Risk time:  Persons:
  Mic  67     55     38       160       93    2421.31       160
```

CODE EXPLAINED: The dates `doBase` and `doEnd` both refer to calendar time, here called `per`.

We code the exit status 0 for those alive, 1 for those who have a date of death, and 2 for those who also have an indicator of death from CVD. All included persons start in the `Mic` state, hence the naming of the *first* level of the exit status (where `(!is.na(doDth)) + deathCVD` is 0).

Note that we need to write `(!is.na(doDth)) + deathCVD` because "`+`" has higher precedence then "`!`"; that is, `+` is calculated *before* "`!`". Omitting the outer brackets would give the same result as `!(is.na(doDth) + deathCVD)` which would be a logical, `FALSE` if the expression in brackets is 0, otherwise `TRUE`—not what we want. You may want to consult the operator precedence order by `?Syntax`.

4.8.2 Definition of intermediate states

We can now cut the follow-up at the occasions where albumin has been measured, that is, dates at which we consider a transition can take place. Depending on measurements, persons are classified as being in one of the states normo-albuminuria, micro-albuminuria or macro-albuminuria, in order of severity: `Norm` < `Mic` < `Mac`. Recall that one of the entry criteria were that the persons should be in the state `Mic`.

This will not necessarily produce a transition at every date where current state is recorded; persons may have the same status at a given visit as they had on the previous. But there is no harm done in having a data representation of a transition from a given state to itself:

```
> data(st2alb)
> str(st2alb)
'data.frame':        563 obs. of  3 variables:
 $ id   : num  1 1 1 1 1 2 2 2 2 2 ...
 $ doTr : Date, format: "1993-06-12" "1995-05-13" ...
 $ state: Factor w/ 3 levels "Norm","Mic","Mac": 2 1 2 1 2 1 2 3 2 2 ...

> cut2 <- with(st2alb,
+              data.frame(lex.id = id,
+                         cut = cal.yr(doTr),
+                   new.state = state))
> 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

> L3 <- rcutLexis(L2, cut2, timescale = 'per')
> levels(L3)

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

> L3 <- Relevel(L3, 2:1)
> summary(L3)

Transitions:
     To
From   Norm Mic Mac D(oth) D(CVD)  Records:  Events: Risk time:  Persons:
  Norm   90  31   5     14      7       147       57     608.84        69
  Mic    72 299  65     27     13       476      177    1383.65       160
  Mac     3  20  44     14     18        99       55     428.81        64
  Sum   165 350 114     55     38       722      289    2421.31       160
```

CODE EXPLAINED: In order to cut the follow-up at the dates in `doTr` we must transform these dates to `cal.yr` format to align them with the dates in `L2`. And we must devise a data frame with the variable names `lex.id`, `cut` and `new.state`.

This is generated as `cut2`, to be used as input to `rcutLexis`, which cuts the follow up at the dates of transitions in `cut2` and only keeps track of the most recent state occupied.

We finally reorder the levels of `L3` with `Relevel`; note that we only need to care about the first two levels, the remaining levels will remain in the original order.

From the summary we see there are a few recorded transitions `Norm`→`Mac` and vice versa, but on physiological grounds we will not allow these; we put an artificial visit to the state `Mic` in between `Norm` and `Mac`:

```
> set.seed(1952)
> dd <- subset(L3, (lex.Cst == "Mac" & lex.Xst == "Norm") |
+                  (lex.Xst == "Mac" & lex.Cst == "Norm"))
> cut3 <- data.frame(lex.id = dd$lex.id,
+                        cut = dd$per +
+                              dd$lex.dur * runif(nrow(dd), 0.1, 0.9),
+                  new.state = "Mic")
> cut3
  lex.id      cut new.state
1     70 2001.789       Mic
2     86 2012.232       Mic
3    130 2001.488       Mic
4    131 2001.032       Mic
5    136 1997.610       Mic
6    136 2000.780       Mic
7    171 1997.057       Mic
8    175 2013.472       Mic
> L4 <- rcutLexis(L3, cut3)
> summary(L4)
Transitions:
     To
From   Norm Mic Mac D(oth) D(CVD)  Records:  Events: Risk time:  Persons:
  Norm   90  35   0     13      6       144       54     581.91        66
  Mic    72 312  65     30     14       493      181    1437.48       160
  Mac     0  22  41     12     18        93       52     401.91        60
  Sum   162 369 106     55     38       730      287    2421.31       160
```

CODE EXPLAINED: We find those records in `L3` that represent direct transition between `Norm` and `Mac`, and put them in the data frame `dd`. We then choose a random time point in the middle 80% of the time represented in each of these (`runif(nrow(dd), 0.1, 0.9)` returns `nrow(dd)` random numbers between 0.1 and 0.9) and use these as the transition times to `Mic`, and finally these are used as cut points supplied to `rcutLexis`.

Note that there are now no direct transitions between `Mac` and `Norm` in `L4`. We show the course for two select persons that both have 4 transitions (the maximum in the dataset):

```
> addmargins(table(tt <- with(subset(L4, lex.Cst != lex.Xst),
+                             table(lex.id))))
  1   2   3   4 Sum
 51  60  24  11 146
> who <- names(tt[tt==4])[10:11]
> subset(L4, lex.id %in% who)[,1:6]
 lex.id     per   age lex.dur lex.Cst lex.Xst
    143 1993.95 57.11    0.05     Mic     Mac
    143 1994.00 57.17    2.35     Mac     Mic
    143 1996.35 59.52    3.07     Mic    Norm
    143 1999.41 62.58    3.07    Norm  D(oth)
    159 1994.02 67.50    0.13     Mic     Mic
    159 1994.16 67.63    2.66     Mic    Norm
    159 1996.82 70.29    2.37    Norm     Mic
    159 1999.20 72.67    7.32     Mic     Mac
    159 2006.52 79.99    3.99     Mac  D(CVD)
```

CODE EXPLAINED: First we select the records that represent a transition (i.e. where `lex.Cst` is different from `lex.Xst`), and make a table, `tt`, of the number of this type of records for each person. A table of the entries in `tt` shows that there are 11 persons with 4 transitions, and among these we select the two last, and print the 6 first variables for these two persons for illustration.

Once we have a Lexis object with all desired transitions we can get a graphical overview of the transitions between states:

```
> clr <- c("forestgreen", "orange", "red", "blue", gray(0.4))
> boxes(L4, boxpos = list(x = c(10,10,10,90,90),
+                         y = c( 5,50,95,25,75)),
+       show.BE = "nz",
+       scale.R = 1000,
+       pos.arr = c(0.25,0.5)[c(2,2,1,1,2,1,1,2,1,1)],
+        col.bg = clr,
+       col.txt = c("white", "black")[c(1,2,1,1,1)],
+    col.border = c(clr[1:3], rep("black", 2)),
+           lwd = 3)
```

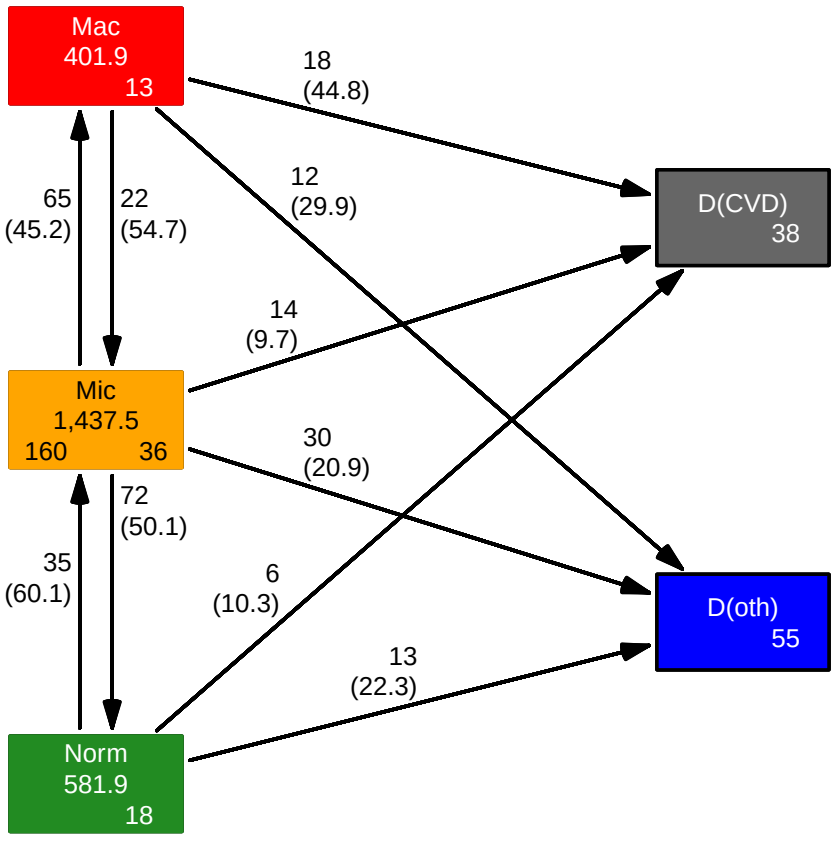


Figure 4.10: *Transitions between states of albuminuria and death in the Steno2 study. The numbers in the center of the boxes are the total risk time (person-years), the numbers in the bottom of the boxes are the number of persons starting (left), resp. ending (right) their follow-up in each state; note that all persons start in the* `Mic` *state. The numbers on the arrows are the number of transitions and (rates per 1000 PY).* `../graph/st2-albbox`

CODE EXPLAINED: When we set `show.BE` to `"nz"` we do not have 0s printed in the boxes where the number of persons entering and exiting are printed.

The `pos.arr` argument determines where on the transition arrows the annotation is located—numbers between 0 and 1, 0 corresponding to the root of the arrow, and 1 to the tip of the arrow. It is a vector of length equal to the number of transitions; the ordering of the transitions (arrows) is row-wise among the non-0, non-diagonal entries in the transition matrix as shown by `summary`.

The `col.` arguments set the background, text and border color of the boxes.

4.8.2.1 Defining transition times: interval censoring

We used the recorded test-times as transition times to new levels of micro- or macro-albuminuria. Instead we might have used the midpoints or some randomly generated time between the two measurements. Strictly speaking we have interval-censored data; the transition occurred sometime between the two measurements. We might even have generated several copies of the dataset where we randomly generated transition times; these could could then be analyzed in parallel and estimates combined in the usual way for multiple imputation. This type of analysis was done by Bjerg *et al.* [2].

4.8.3 Adding clinical measurements

When modeling the transition rates between states we may be interested in the effect of measured variables obtained at clinical visits, visits that are unrelated to state transition times. The follow-up of persons should then be divided at the time of the clinical visit, and the follow-up time *after* the visit have the values of measurements as covariates.

In the Steno2 study, we have also clinical measurements at different visits in the dataset `st2clin`:

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

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

We can add this clinical information to the follow-up by cutting the follow-up at these times using the `addCov.Lexis` function. It takes as arguments a `Lexis` object and a data frame of clinical measurements classified by person and date. It is required that the data frame with the clinical data set has a `lex.id` variable and a variable with the same name as one of the time scales in the `Lexis object`. The remaining variables are considered to be clinical measurement values.

```
> clin <- mutate(st2clin,
+                  lex.id = id,
+                    per = as.numeric(cal.yr(doV))) %>%
+         select(-id, -doV)
> head(clin[order(clin$lex.id, clin$per),], 8)
     a1c chol crea lex.id      per
1   87.3  3.9   83      1 1993.346
161 68.9  4.2   73      1 1995.478
317 63.7  4.4   63      1 1997.477
466 50.6  3.0  105      1 2001.083
594 74.9  3.2  107      1 2006.715
687 32.8  4.2  171      1 2014.761
2   66.5  6.6   83      2 1993.340
162 54.8  5.7  105      2 1995.487
> str(clin)
```

steno2

```
'data.frame':        750 obs. of  5 variables:
 $ 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 ...
 $ lex.id: num  1 2 3 4 5 6 7 8 9 10 ...
 $ per   : num  1993 1993 1993 1993 1993 ...
```

```
> L5 <- addCov.Lexis(L4,
+                    clin,
+              timescale = 'per',
+                    tfc = 'tsClin')
> summary(L4)
```

```
Transitions:
     To
From   Norm Mic Mac D(oth) D(CVD)  Records:  Events: Risk time:  Persons:
  Norm   90  35   0     13      6       144       54     581.91        66
  Mic    72 312  65     30     14       493      181    1437.48       160
  Mac     0  22  41     12     18        93       52     401.91        60
  Sum   162 369 106     55     38       730      287    2421.31       160
```

```
> summary(L5, timeScales = TRUE)
```

```
Transitions:
     To
From   Norm Mic Mac D(oth) D(CVD)  Records:  Events: Risk time:  Persons:
  Norm  231  35   0     13      6       285       54     581.91        66
  Mic    72 708  65     30     14       889      181    1437.48       160
  Mac     0  22 136     12     18       188       52     401.91        60
  Sum   303 765 201     55     38      1362      287    2421.31       160

Timescales:
   per    age tsClin
    ""     ""    "X"
```

CODE EXPLAINED: We use `mutate` to devise the variables `lex.id` and `per` (the timescale for the measurement times), and `select` to remove variables that we do not want carried over to the `Lexis` object. All other variables than `lex.id` and the time scale variable will namely be carried over to the extended `Lexis` object as clinical variables.

The resulting object, `clin`, is then used to extend the `L4` object using `addCov.Lexis`.

Note that when the time scales are listed, `per` and `age` are listed as `""`, which means that neither of them are defined as time since entry into a state. `tsClin` is marked as `"X"`, which means that it is a deficient time scale: it is not defined for some follow-up (namely that preceding the the first clinical visit), and that it is reset at each new clinical visit. But it must be defined a time-scale because it must be updated at subsequent splitting and cutting.

We see that transitions and risk time are the same in `L4` and `L5`, but that `L5` has more records, due to the cut of follow-up at the clinical visit times.

```
> (wh.var <- names(L5)[c(4,1:2,25,3,19:23)])
 [1] "lex.id"  "per"     "age"     "tsClin"  "lex.dur" "lex.Cst" "lex.Xst" "exnam"
 [9] "a1c"     "chol"
```

```
> subset(L5, lex.id %in% who)[,wh.var]
```

```
 lex.id     per   age tsClin lex.dur lex.Cst lex.Xst exnam  a1c chol
    143 1993.95 57.11   0.01    0.05     Mic     Mac   ex1 69.3  4.6
    143 1994.00 57.17   0.06    1.50     Mac     Mac   ex1 69.3  4.6
    143 1995.50 58.67   0.00    0.85     Mac     Mic   ex2 55.2  4.4
    143 1996.35 59.52   0.85    1.16     Mic     Mic   ex2 55.2  4.4
    143 1997.51 60.68   0.00    1.91     Mic    Norm   ex3 73.6  5.4
    143 1999.41 62.58   1.91    1.87    Norm    Norm   ex3 73.6  5.4
    143 2001.28 64.45   0.00    1.20    Norm  D(oth)   ex4 75.5  3.5
    159 1994.02 67.50     NA    0.03     Mic     Mic  <NA>   NA   NA
    159 1994.06 67.53   0.00    0.10     Mic     Mic   ex1 64.0  4.4
    159 1994.16 67.63   0.10    1.35     Mic     Mic   ex1 64.0  4.4
    159 1995.51 68.98   0.00    1.32     Mic    Norm   ex2 50.5  5.0
```

```
    159 1996.82 70.29   1.32    0.66    Norm    Norm   ex2 50.5  5.0
    159 1997.48 70.95   0.00    1.71    Norm     Mic   ex3 43.9  3.9
    159 1999.20 72.67   1.71    2.40     Mic     Mic   ex3 43.9  3.9
    159 2001.60 75.07   0.00    4.92     Mic     Mac   ex4 49.8  3.8
    159 2006.52 79.99   4.92    0.18     Mac     Mac   ex4 49.8  3.8
    159 2006.69 80.16   0.00    3.82     Mac  D(CVD)   ex5 59.7  3.1
```

CODE EXPLAINED: We select names of relevant variables in the data frame we want to see, and invoke the `print` method for `Lexis` objects to show the records from two of the persons.

For person 143 the first value of `tsClin` is not 0, because the earliest clinical measurement is before entry to the study (albeit only by 19 days ($0.052 \times 365.25 = 18.993$)).

Also, the time scale `tsClin` has missing values; namely for those pieces of follow-up that are *before* the first clinical measurement as is seen for person 159. This is something to be wary of when modeling rates—if ignored you may accidentally miss some observation time in your analysis.

Finally, for analysis of the transition rates we can split time in smaller intervals; the point of having `tsClin` as covariate is that we may use it to model how the effect of the clinical variables changes with the time since measurement. Therefore it is defined as a time scale by `addCov.Lexis`:

```
> options("popEpi.datatable" = FALSE)
> L6 <- splitMulti(L5, per = seq(1980, 2020, 0.5))
> summary(L5)
Transitions:
     To
From   Norm Mic Mac D(oth) D(CVD)  Records:  Events: Risk time:  Persons:
  Norm  231  35   0     13      6       285       54     581.91        66
  Mic    72 708  65     30     14       889      181    1437.48       160
  Mac     0  22 136     12     18       188       52     401.91        60
  Sum   303 765 201     55     38      1362      287    2421.31       160
> summary(L6)
Transitions:
     To
From   Norm  Mic  Mac D(oth) D(CVD)  Records:  Events: Risk time:  Persons:
  Norm 1396   35    0     13      6      1450       54     581.91        66
  Mic    72 3565   65     30     14      3746      181    1437.48       160
  Mac     0   22  945     12     18       997       52     401.91        60
  Sum  1468 3622 1010     55     38      6193      287    2421.31       160
```

CODE EXPLAINED: We use the `splitMulti` function from the `popEpi` package, it returns a `data.table` instead of a `data.frame`, unless we set the options `popEpi.datatable` to `FALSE` as shown. This will make the function a bit slower, though.

We see that `tsClin` behaves like a time scale and is properly expanded by the time-splitting. But it should not be used as a time scale to split along, because it is not guaranteed that the result is sensible.

. . . now input from `datarep`

4.9 Summary of multistate data representation

As we have seen, the raw observations in multistate models are just recordings of dates (times, in general) where transitions from one state to another occur along with the information of the states from and to which the transition occur. We assume that the precise time of each transition is known. These are the basic data items required to construct the relevant dataset(s) for estimation of transition rates.

In the `Epi` package is implemented the `Lexis` machinery for representation and manipulation of multistate date; a complete overview of all relevant functions is given in chapter **??**.

We use the term "cut" for the operation of subdividing follow-up at *one* time point, typically some sort of event representing a transition to a new state. We are using the term "split" for the operation of subdividing follow-up into many small intervals along some time scale.

`datarep`

4.9.1 Prerequisites and construction of a Lexis object

1. Make sure that all times of interest are recorded as absolute dates; date of birth, date of entry, date of exit from the study, date of death and date of any intermediate events of interest. This allows calculation of age, disease duration etc. at any time required.

2. First determine the total follow-up period for each person. As a start set up the total follow-up (basically starting with entry in some "alive" state and either ending in some absorbing "dead" state or being censored alive).

   The function `Lexis` [9] will create a data frame with additional structure, called a `Lexis` object which allows representation of multistate follow-up on multiple time scales, and provides tools for manipulation and display of multistate data.

3. Then subdivide the total follow-up according to the state currently occupied. Keep track of how much time is spent in each state, and which transitions occur.

   Tools for subdivision of follow-up in a `Lexis` object at transition times are `cutLexis`, `mcutLexis` and `rcutLexis`. Note that there is no dot ("`.`") before the "`Lexis`" in these function names.

   (a) `cutLexis` will cut records representing follow-up at times where an intermediate event occurs; only one event per person is allowed. For some persons one record will be replaced by two, representing follow-up pre- and post-event, with different values of the state variables.

   There is no requirement that the intermediate event time should be within the observation window for a person; if the event occurs before the beginning of follow-up for a person, all the follow-up of the person will be classified in the new state. If the event occurs after the end of follow-up, no change to the representation of the person's follow-up is made.

   (b) `mcutLexis` will do the same but for several types of intermediate events, that occur at most once each for a person, and where we want to keep track of which events has occurred in which order for each person (the history).

   (c) `rcutLexis` will also cut at multiple events for a person, but it only keeps track of the most recent event type, and moreover multiple occurrences of event types (transitions) are allowed. This will typically be in situations where we are modeling different degrees of illness, and where improvement may occur too.

4. Defining time-dependent variables in a `Lexis` object is done by `addCov.Lexis` and `addDrug.Lexis`. These functions are S3 methods for `Lexis` objects, so when calling them you can leave out the "`.Lexis`".

   (d) `addCov.Lexis` will add clinical covariates to follow-up time in Lexis objects, and keep track of the time since recording of them. This is a bit like `cutLexis` for the event "most recent clinical visit", without recording this as a state, but keeping the time since last visit as a time scale. The latter will not be a proper timescale, because it is reset to 0 at every clinical visit, so it is only useful as a covariate, but it will be respected by the time-splitting functions, so properly updated by the `split` and `cut` functions.

   (e) `addDrug.Lexis` will add drug exposure variables to the `Lexis` object. The object will be cut at all dates of drug purchase plus dates of computed drug expiration. Records representing follow-up between these will receive exposure variables.

   The added covariates are *computed* from the exposure records, they refer to the level of exposure at the beginning of each interval, according to the chosen exposure algorithm.

   This has the consequence that time-splitting or -cutting is not meaningful *after* using `addDrug.Lexis`. Splitting and cutting will carry all covariate values forward in time, and this is not meaningful for the covariates generated by `addDrug.Lexis`.

   This is why the `addDrug.Lexis` should be changed so that no new cuts are added—data will normally already be split. A possible exception might be the date of first purchases of each drug, allowing a more precise assessment of drug allocation effects.

The resulting `Lexis` object will have (at least) one record for each state occupied by a person, with the time at entry to the state defined on all time scales (one variable for each time scale), as well as a variable with the sojourn time in the state and an indication of the state *to* which transition occur at the end of the sojourn.

The `Lexis` object represents the follow-up through the states, and will be the basis for analysis of transition rates between states.

5. Since we only ever consider models for a finite number of states, the states are represented by levels of a *factor* — the designated way in R to represent a variable that only assumes values in a pre-specified finite set. The `Lexis` machinery does allow representation of states as numbers, but it is discouraged since it makes your code more error-prone and results less readable.
6. The follow-up in `Lexis` objects can be summarized in tabular form by `summary` (that is `summary.Lexis`), and as a diagram with states as boxes by `boxes` (`boxes.Lexis`). Finally, follow-up on two select time scales can be illustrated in a Lexis diagram by the `plot` (`plot.Lexis`) function for `Lexis` objects.
7. If we want to model transition rates using smooth functions for the effect of the time scales [6], we must split the follow-up in intervals so small that an assumption of constant rate within each interval is a reasonable assumption; this can be done by `splitLexis`. For most chronic diseases a reasonable interval length would be 0.5 years.

   The functionality of `splitLexis` is duplicated and expanded (in a faster version, with more intuitive syntax) by `splitMulti` from the `popEpi` package. `splitMulti` derives its name from the ability to split follow-up time along multiple time scales simultaneously.
8. The time splitting is not necessary if you model the *cumulative* rates by smooth functions [10], but that will effectively restrict you to models with only one time scale in the description of the rates.

4.10 Summary of analysis of transition rates

A `Lexis` object is a data frame representation of the follow-up of persons through states. This can be exploited to simplify model specification for transition rates because the likelihood for a specific transition rate is a sum of terms that correspond to records in the `Lexis` object.

Joint models for several rates can be specified as well, provided that no two transition rates originate in the same state. This will be the case for most models of practical interest.

Rate models based on `Lexis` objects are specified by one of three functions:

- `glm.Lexis` fits a Poisson model using `glm` with the `poisreg` family.
- `gam.Lexis` fits a Poisson model using `gam` with the `poisreg` family; this is a machinery that allows use of penalized splines for the effect of quantitative variables through the function `s()`.
- `coxph-Lexis` fits a Cox model.

The Cox modeling requires that a baseline time scale be specified as a response, i.e. as the l.h.s. of a formula. For the other two functions only the r.h.s. of a formula can be specified because the baseline time scale(s) are part of the explanatory variables.

For a complete specification of a multistate model it is necessary to have a model for *all* of the observed transitions in the model. The models need not all be different, some groups of transitions may be described by the same model. Normally the `lex.Cst` will then be among the covariates, allowing the estimated rates to depend on the originating state for the model.

At this point we are not going beyond the models for the transition rates; this will be detailed in the chapters on competing risks (6, p. 92) and and estimation from multistate models (7, p. 107).

. . . now input from `recur`

`recur`

Chapter 5

Recurrent events

When an event can occur multiple times for an individual, such as bone fracture or more realistically, hypoglycemia [1] in diabetes patients, the event occurrence is said to be *recurrent*. If the number of times the event occurs is not vast (no more than 15 per person), the normal way of modeling the events is in a multistate model with states 0, 1, 2, ... indicating the attained number of occurrences of the event. The rates of the transitions $0 \to 1$, $1 \to 2$, $2 \to 3$ etc.will typically be modeled with the same effects, but with an additional parameter for each extra occurrence of a transition to accommodate the effect of the attained number of recurrences (that is, the current state).

In practical applications, the estimated effect of the currently attained state on the rate of transition to the next is likely to be increasing by state number because of selection: persons more likely to have many events will be increasingly over-represented by increasing state (that is, increasing no. of events), and by that token the rate of the next event will tend to increase by the number of events so far.

This phenomenon can also be regarded as a reflection of an empirical fact, usable for prediction of rates conditional on the number of attained events. By considering it a true effect we would be assuming that the sheer *number* of attained events were a determinant of the rate of the next event.

When a person can have more events during follow-up it is (at least in theory) possible to estimate a person-specific parameter which is common across the rates in a given model. This can be interpreted as a person-specific propensity for the event of interest, normally termed "frailty", and usually taken to be a multiplicative term on the rate scale.

It will only be possible to estimate a separate (fixed) frailty parameter for each person if *all* persons have at least one transition in the model. Which is very rarely the case. Therefore, a frailty *distribution* is postulated—note that the particular distribution chosen essentially is a statement about the specific study population at hand.

By that token, it would be desirable to estimate the shape of the frailty distribution—in many practical applications it can be suspected that there is a certain fraction of the population that is effectively immune to the recurrent event, corresponding to a frailty distribution with an atom, for example taken to be at 1. This would be a mixture model, that could either be specified flexibly using JAGS or some other Bayesian MCMC machinery or using mixture models like the ones available in the `gnlm` package.

If we consider multiplicative models for the transition rates we may use a normal distribution for the (log-)frailty, by including the frailty as an additive term on the log-rates scale. In the literature it has become the convention to refer to the frailty as the multiplicative term, so the model with a normally distributed term for the log-rates will be referred to as a model with log-normal frailty distribution.

Thus, if we have possible repeat events for each person we may consider using a random effects model. And then report some measure of the random effects distribution, for example the median absolute difference (or ratio) of frailties between two randomly chosen persons, see [8].

It is not possible to separate the frailty effect from the actual effect of number of events conceptually, but computationally we will normally be able to separate the effects. However, the separation (i.e.the estimated effects of state) will depend on a particular *distributional* assumption for the random effects—the systematic effects of state (attained no. of events) is the "residual" not picked up by the random effects.

[1] Hypoglycemia is "too low blood sugar", and can for example occur for diabetes patients if the dosing of insulin is not properly regulated. It is a potentially lethal condition, hence feared by diabetes patients.

5.1 Example: Hospitalized hypoglycemia (HH)

First we load the packages needed:

```
        [,1]
        "R version 4.2.0 (2022-04-22 ucrt)"
LibPath "C:/util/R/R-4.2.0/library"
Package "Epi"
Version "2.47"
```

5.1.1 Data set of recurrences

First we load the two datasets, one with one record per person (pp) and one with one record per hypoglycaemaia event (hh). In the person data set we define sex as a proper factor (as it should always be—like any other categorical variable):

```
> # load(file = "~/teach/SPE/MSbook/data/HH.Rda", v = T)
> load(file = "../data/HH.Rda", v = T)
Loading objects:
  pp
  hh
> pp$sex <- factor(pp$sex, levels = 1:2, labels = c("M","W"))
> str(pp)
'data.frame':        18314 obs. of  14 variables:
 $ dob : num  1954 1972 1968 1935 1951 ...
 $ id  : int  1 3 4 7 8 9 10 11 12 13 ...
 $ bmi : num  26 26.1 25.6 23.5 24.8 ...
 $ dodm: num  2003 1980 2002 1969 1996 ...
 $ doe : num  2007 2006 2006 2009 2008 ...
 $ dox : num  2012 2013 2012 2012 2010 ...
 $ dod : num  NA NA NA NA NA NA NA NA NA NA ...
 $ sex : Factor w/ 2 levels "M","W": 1 2 2 1 1 2 1 1 2 2 ...
 $ hba : num  39.9 76 61.7 45.4 42.1 ...
 $ alb : num  NA NA NA 0.1 NA ...
 $ sbp : int  134 160 144 135 135 145 142 115 116 120 ...
 $ aht : int  1 1 1 2 2 1 1 2 1 1 ...
 $ llt : int  1 1 1 2 2 1 1 1 1 1 ...
 $ smk : Factor w/ 5 levels "Daily smoker",..: 3 1 1 2 1 1 1 4 3 3 ...
> str(hh)
'data.frame':        16194 obs. of  3 variables:
 $ id : int  11 11 11 11 11 11 11 16 16 16 ...
 $ doh: num  2002 2003 2004 2005 2011 ...
 $ nh : int  1 2 3 4 5 6 7 1 2 3 ...
```

The data frame pp only contains follow-up w.r.t. death for the period May 2005 through January 2013:

```
> summary(pp$dod, na.rm=TRUE)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   2007    2010    2010    2010    2011    2012   17641
```

The dates in the datasets are in cal.yr format, but we can convert them to human-readable form by as.Date.cal.yr:

```
> as.Date.cal.yr(range(pp$doe))
[1] "2005-05-02" "2012-09-05"
> as.Date.cal.yr(range(pp$dox))
[1] "2006-07-12" "2013-01-13"
> as.Date.cal.yr(range(pp$dod, na.rm = TRUE))
[1] "2007-01-06" "2012-01-10"
```

recur

The recorded dates of HH span a somewhat longer range back in time, enabling us to assign an absolute ("lifetime") number of each recorded hypoglycemia episode:

```
> as.Date.cal.yr(range(hh$doh))
[1] "1985-02-21" "2013-11-20"
```

We also explore how the entry and exit dates relate in the `pp` data frame:

```
> with(pp, plot(doe, dox, pch = 16, cex = 0.3,
+               xlim = c(2005,2013), ylim = c(2005,2013)))
> for (i in 0:6) abline(i, 1, col = "red", lty = "24")
> abline(v = 2005 + 0:8 + (31 + 15)/365.25,
+        h = 2005 + 0:8 + (31 + 15)/365.25,
+      col = "forestgreen", lty = "24")
> for(si in 1:2) axis(side = si, at = 2005:2013, labels = NA)
```

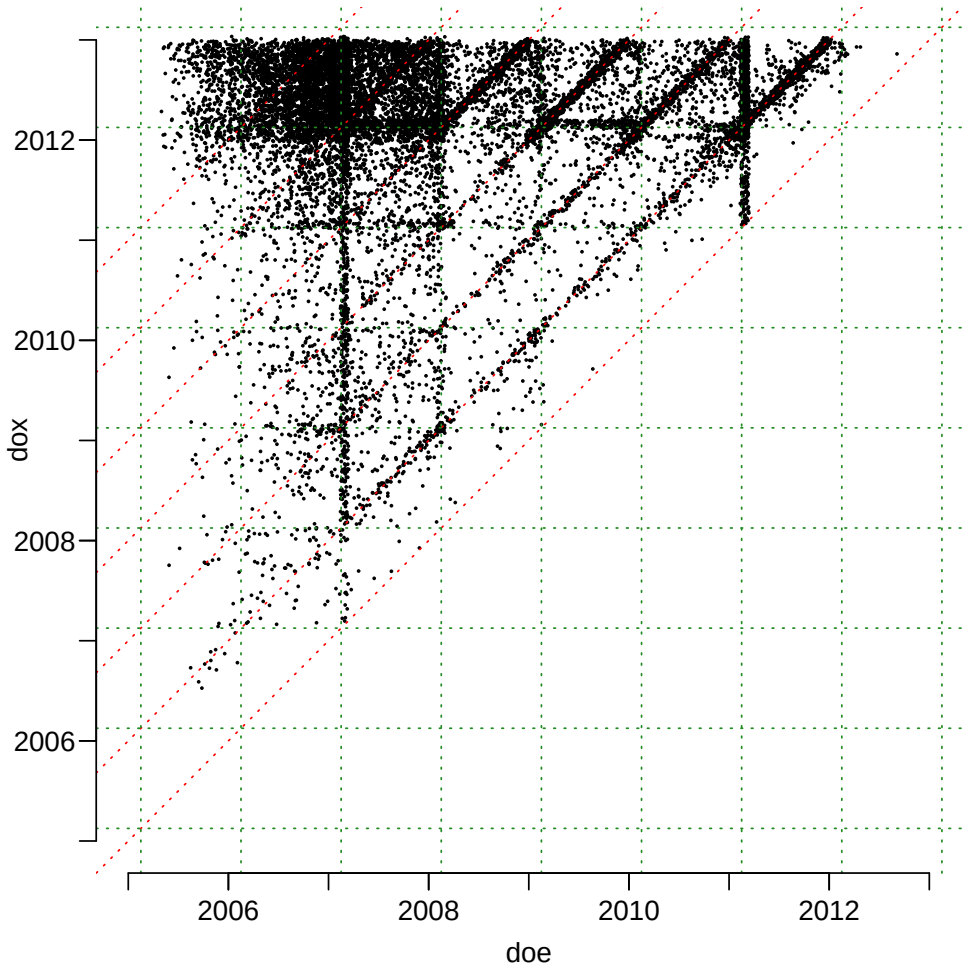


Figure 5.1: *Entry and exit dates in the hypoglycemia study. Note that the green broken lines are located at 15th February each year, both for date of entry (*`doe`*) and date of exit (*`dox`*), while the broken red lines represent integer differences between date of exit and entry.* `../graph/recur-doenex`

From figure **??** it is pretty clear that there is a preference of having the exit date an integer number of years after the entry date (the red dotted lines). Also, there seems to be preferential values both for entry and exit dates at approximately 15th February each year (indicated by green dotted lines). This illustrates how important it is to explore the relationships between dates in the original data. For this data set we do not have any explanations of the pattern, but it is typical for the effect of some kind of date-rounding.

5.1.1.1 Removing presumed duplicated HH events

First we provide an overview of how many persons have how many HH events

```
> nrow(hh) ; tt <- table(hh$id) ; addmargins(table(tt))
[1] 16194
tt
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18
2775 1248  664  335  244  146  108   86   46   58   26   23   14   13   10   11    8    9
  19   20   21   22   23   24   25   26   27   28   29   30   32   33   36   39   44   46
   6    5    4    3    3    3    2    5    1    1    1    1    2    2    1    1    1    2
  49   53   56   63   83  Sum
   1    1    1    2    1 5874
```

CODE EXPLAINED: First we determine the number of rows in `hh` and then make a table of `id`s, (`tt`), with the number of records for each person. A tabulation of `tt` then shows how many persons have 1, 2, ... recorded episodes.

HH events occurring too close together can be suspected to be repeat recordings of the same event. Hence, we want to remove HH episodes too close to the previous one. To this end we first make a histogram of the time between successive HH episodes for those with at least two recorded:

```
> hh$gap <- c(NA,diff(hh$doh))
> dgap <- hh$gap[duplicated(hh$id)] * 365.25
> dgap <- dgap[dgap < 100]
> hist(dgap, col = 1, breaks = 0:100,
+      main = "", xlab = "Gap lengths (days)")
```

CODE EXPLAINED: First we compute the successive differences between HH dates using `diff`. The first value of this within each person is not relevant—it is the difference to a date from the previous person. So we will exclude first record for each person. This is achieved by using `duplicated` that returns the indicator of records with an earlier value of the variable, in this case `id`. Finally we restrict to gaps that are smaller than 100 days, and make a histogram of these.

We see from figure 5.2 that the frequencies of gaps is sinking from 0 through 4 weeks, so there is no clear data feature to reveal a threshold below which we consider two recordings to be one event. Therefore we more or less arbitrarily choose 14 days as an arbitrary cut-off for considering two recordings as being one HH episode:

```
> wh.keep <- which((hh$gap > 13.5/365.25 & duplicated(hh$id)) |
+                                          !duplicated(hh$id))
> length(wh.keep)
[1] 14270
```

Having done this, we must re-number the HH episodes within each individual, i.e. update the `nh` variable:

```
> hh <- data.frame( hh[wh.keep,]
+                %>% group_by(id)
+                %>% mutate(nh = row_number())
+                %>% select(c(id, doh, nh))
+                  )
```

So `hh` is now thinned to have only consecutive episodes of HH that are at least 14 days apart.

5.1.2 Constructing a `Lexis` object

5.1.2.1 Initial `Lexis` data frame

First we construct a `Lexis` data frame with follow-up from entry (`doe`) to exit (`dox`) and with exit status dead (`Dead`) or alive (coded `0.HH` as the default is 0 HHs until information on HH is added):

`recur`

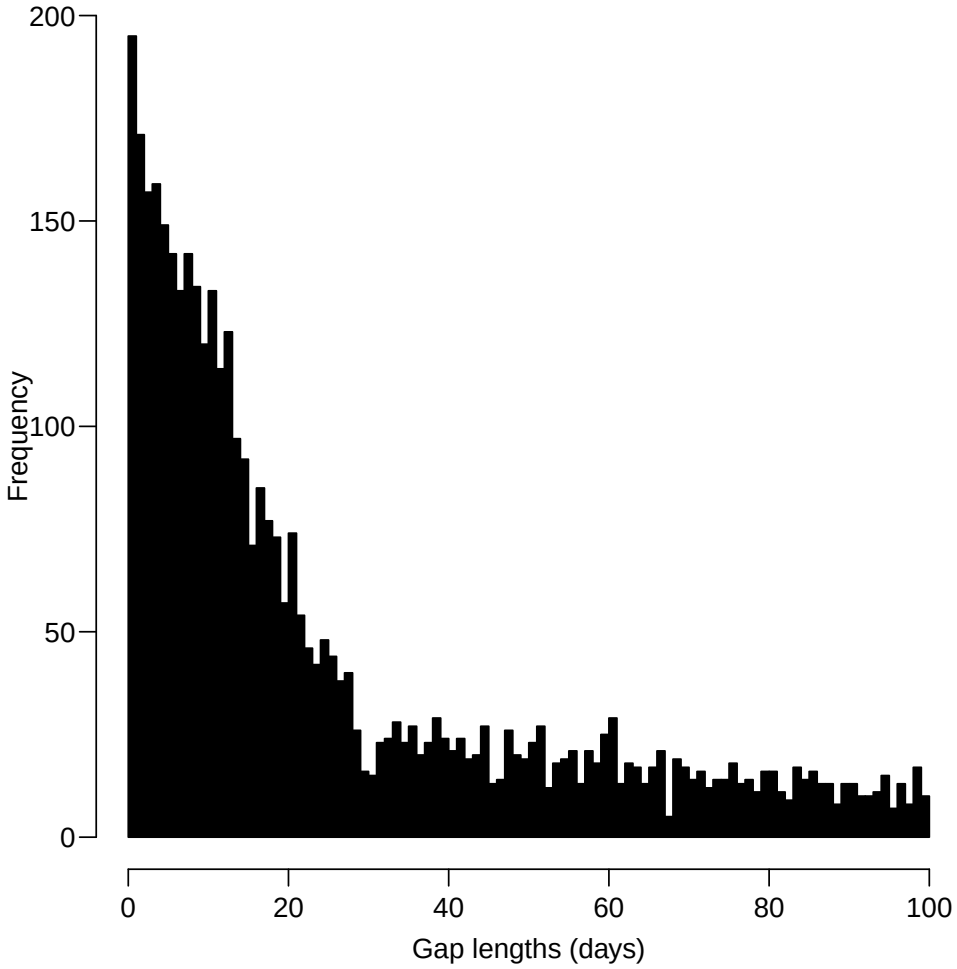


Figure 5.2: *Distribution of within-person gaps between successive HH events; shown in one-day bins. A person may contribute more than one gap in the histogram, but only persons with at least two recorded HH dates contribute.*

../graph/recur-gaphist

```
> Lx <- Lexis(data = pp,
+             entry = list(per = doe,
+                          age = doe - dob,
+                          dur = doe - dodm),
+              exit = list(per = dox),
+                id = id,
+       exit.status = factor(!is.na(dod),
+                            labels = c("0.HH","Dead")))
NOTE: entry.status has been set to "0.HH" for all.
NOTE: Dropping  1  rows with duration of follow up < tol
> summary(Lx, t = T)
Transitions:
     To
From    0.HH Dead  Records:  Events: Risk time:  Persons:
  0.HH 17640  673     18313      673   73223.02     18313

Timescales:
per age dur
 ""  ""  ""
```

Once this has been established, we will use `rcutLexis` to repeatedly cut the follow-up of each person, however only the first 15 HH occurrences:

```
> cutHH <- mutate(subset(hh, nh <= 10),
+                     lex.id = id,
+                        cut = doh,
+                  new.state = ifelse(nh < 10,
+                                     paste0(nh, ".HH"),
+                                     "10+HH")) %>%
+           select(lex.id, cut, new.state)
> head(cutHH, 15)
   lex.id      cut new.state
1      11 2002.174      1.HH
2      11 2003.375      2.HH
3      11 2004.269      3.HH
4      11 2004.565      4.HH
5      11 2011.312      5.HH
6      11 2013.051      6.HH
7      11 2013.522      7.HH
8      16 2003.414      1.HH
9      16 2003.818      2.HH
10     16 2009.943      3.HH
11     17 1999.275      1.HH
12     19 2013.610      1.HH
13     23 2002.245      1.HH
14     23 2002.319      2.HH
15     23 2008.304      3.HH
> table(cutHH$new.state)

 1.HH 10+HH  2.HH  3.HH  4.HH  5.HH  6.HH  7.HH  8.HH  9.HH
 5874   160  2826  1605   997   680   481   351   257   200
> system.time(
+ Rx <- rcutLexis(Lx,
+                 cut = cutHH,
+           timescale = "per"))

   user  system elapsed
   5.96    0.14    6.09
```

CODE EXPLAINED: In order to cut the follow-up at multiple dates (but not keeping track of the history of state visits), we need a data frame with the dates of events (in a variable `cut`, here from `hh$doh`) and the names of the states that events induces to (in the variable `new.state`, here it takes values `x.HH`, where `x` is the number of the hypoglycemic event)—and of course the person id, `lex.id`.

Using this data frame as argument to `rcutLexis` we also need to tell what timescale in `Lx` the values in variable `cut` refer to.

Even if several of the transitions occur before follow-up, the number of HHs before start of follow-up will be correctly counted by `rcutLexis`.

In the output from `summary(Rx)` we see the characteristic transition matrix for recurrent events; the only possible transitions from, say, state `3.HH` are `3.HH` (censoring in state `3.HH`), `4.HH` (the next HH event) or `Dead`:

```
> summary(Rx)
Transitions:
     To
From     0.HH 1.HH 2.HH 3.HH 4.HH 5.HH 6.HH 7.HH 8.HH 9.HH 10+HH Dead  Records:  Events:
  0.HH  12434  887    0    0    0    0    0    0    0    0      0  335     13656     1222
  1.HH      0 2787  568    0    0    0    0    0    0    0      0  141      3496      709
  2.HH      0    0 1112  373    0    0    0    0    0    0      0   62      1547      435
  3.HH      0    0    0  498  260    0    0    0    0    0      0   47       805      307
  4.HH      0    0    0    0  275  193    0    0    0    0      0   30       498      223
  5.HH      0    0    0    0    0  168  144    0    0    0      0   11       323      155
  6.HH      0    0    0    0    0    0  101  105    0    0      0    9       215      114
  7.HH      0    0    0    0    0    0    0   68   70    0      0   13       151       83
```

recur

```
  8.HH      0    0    0    0    0    0    0    0   47   55     0    4       106        59
  9.HH      0    0    0    0    0    0    0    0    0   23    49    5        77        54
  10+HH     0    0    0    0    0    0    0    0    0    0   127   16       143        16
  Sum   12434 3674 1680  871  535  361  245  173  117   78   176  673     21017      3377

Transitions:
     To
From    Risk time:  Persons:
  0.HH    51936.45     13656
  1.HH    11750.50      3496
  2.HH     4425.94      1547
  3.HH     2010.91       805
  4.HH     1110.91       498
  5.HH      628.43       323
  6.HH      383.83       215
  7.HH      236.76       151
  8.HH      168.51       106
  9.HH      116.50        77
  10+HH     454.30       143
  Sum     73223.02     18313
```

We can illustrate the pattern of the recurrent events with a plot of the states:

```
> angles <- seq(pi, -pi * 0.8, length.out = 11)
> boxes(Rx, boxpos = list(x = c(cos(angles) * 42 + 50, 50),
+                         y = c(sin(angles) * 42 + 50, 50)),
+          show.BE = "nz", cex = 0.9,
+          scale.R = 100,
+          pos.arr = 0.4)
```

CODE EXPLAINED: The function `boxes` (really `boxes.Lexis`) takes a `Lexis` object as the first argument and as the second (`boxpos`) a list with x and y coordinates of the centers of the boxes. The `Lexis` object `Rx` has 12 states, so we must provide 12 coordinate-pairs. In order to place the 11 first states on an (almost) full circle we compute 11 equidistant angles from π to $-\pi \times 7/9$; `pi` is a built-in constant in R(guess its value). The x and y coordinates of the boxes are then made by cos, resp. sin of these angles, multiplied by 42 to make a circle of diameter 84, and offset by 50 in both directions in order to center it in the display of (0,100) in both directions in which `boxes` operate. Finally, 50 is appended as both x and y coordinate of the last state (`Dead`) in the center of the display.

The `show.BE` puts two numbers at the bottom om each box, `show`ing how many `B`egin, respectively `E`nd their follow-up in each state. The `scale.R` `scale`s the `R`ates on the arrows by a factor 1000, and the `pos.arr` `pos`itions the number of events and rates on the `arr`ows at 55% from the beginning of the arrow.

From figure 5.3 we see that there are quite a number of persons who enter the study with several previous instances of HH. We also see that both rates of next hypoglycemia and mortality rates increase by the number of hypoglycemic events.

5.1.2.2 Time since last HH as time scale

In the more detailed analysis of rates we want to keep track of the time since the most recent hypoglycemic event, so we define a variable to do this. Some persons enter the study after HH, so we must retrieve the dates of all HH events, including those that occur before study entry.

```
> Rx <- left_join(Rx, rename(cutHH, doHH = cut),
+                 by = c("lex.id"  = "lex.id",
+                        "lex.Cst" = "new.state"))
> Rx <- factorize(Rx)
> Rx <- Relevel(Rx, c(paste0(0:9,".HH"),"10+HH","Dead"))
> Rx$tfHH <- with(Rx, ifelse(lex.Cst == "0.HH", NA, per - doHH))
> attr(Rx, "time.scales") <- c(attr(Rx, "time.scales"), "tfHH")
> attr(Rx, "time.since" ) <- c(attr(Rx, "time.since" ), "")
> attr(Rx, "breaks") <- c(attr(Rx, "breaks"), list(tfHH = NULL))
> str(Rx)
```

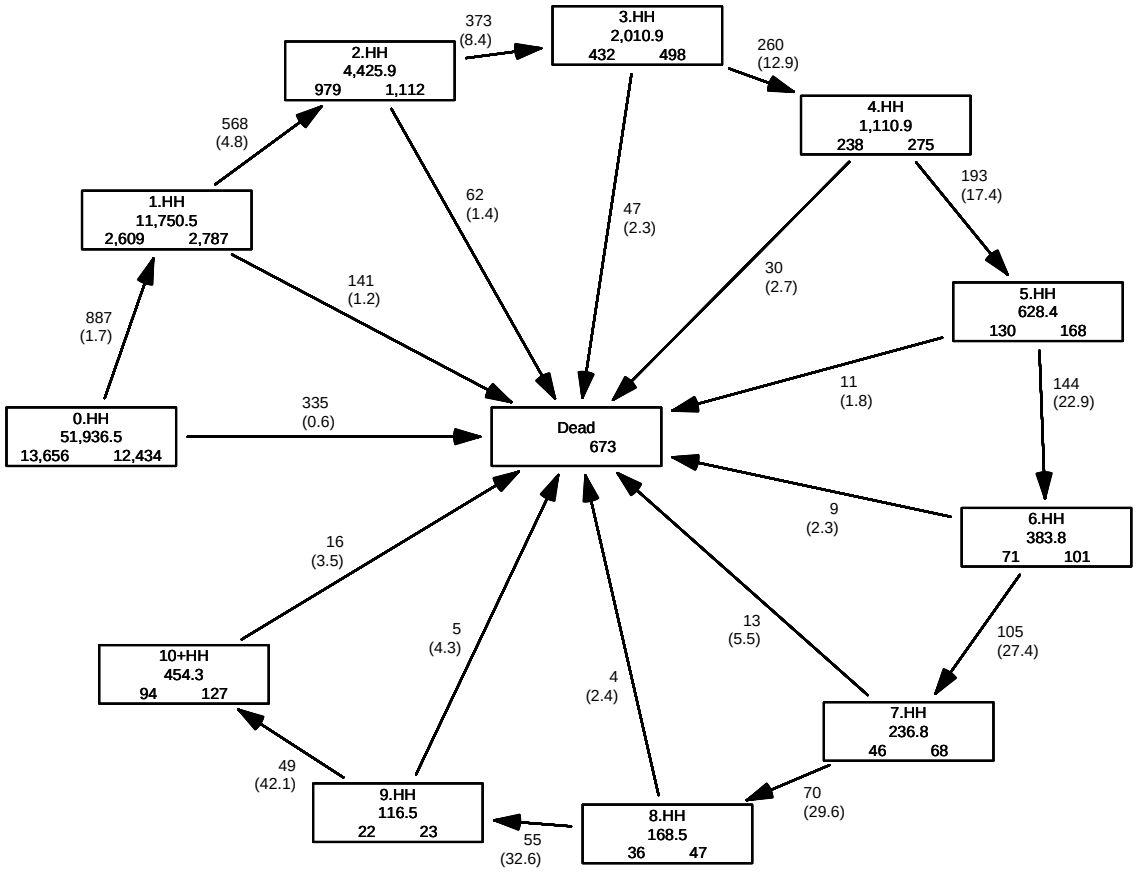


Figure 5.3: *Transitions by recurrences of hypoglycemia. The numbers in the middle of the boxes are the risk time (total sojourn time), at the bottom left the number of persons starting follow-up in the state, at the bottom right the number of persons ending follow-up in the state. The numbers on the arrows are the number of transitions and (rates per 100 person-years).*
`../graph/recur-HH-rec-box`

```
Classes 'Lexis' and 'data.frame':       21017 obs. of  23 variables:
 $ per    : num  2007 2006 2006 2009 2008 ...
 $ age    : num  52.7 34.5 37.9 74 57 ...
 $ dur    : num  4.13 26.31 4.21 40.05 11.64 ...
 $ lex.dur: num  5.28 6.58 6.1 2.95 2.06 ...
 $ lex.Cst: Factor w/ 12 levels "0.HH","1.HH",..: 1 1 1 1 1 1 1 5 6 1 ...
 $ lex.Xst: Factor w/ 12 levels "0.HH","1.HH",..: 1 1 1 1 1 1 1 6 6 1 ...
 $ lex.id : int  1 3 4 7 8 9 10 11 11 12 ...
 $ dob    : num  1954 1972 1968 1935 1951 ...
 $ id     : int  1 3 4 7 8 9 10 11 11 12 ...
 $ bmi    : num  26 26.1 25.6 23.5 24.8 ...
 $ dodm   : num  2003 1980 2002 1969 1996 ...
 $ doe    : num  2007 2006 2006 2009 2008 ...
 $ dox    : num  2012 2013 2012 2012 2010 ...
 $ dod    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ sex    : Factor w/ 2 levels "M","W": 1 2 2 1 1 2 1 1 1 2 ...
 $ hba    : num  39.9 76 61.7 45.4 42.1 ...
 $ alb    : num  NA NA NA 0.1 NA ...
```

`recur`

```
$ sbp    : int  134 160 144 135 135 145 142 115 115 116 ...
$ aht    : int  1 1 1 2 2 1 1 2 2 1 ...
$ llt    : int  1 1 1 2 2 1 1 1 1 1 ...
$ smk    : Factor w/ 5 levels "Daily smoker",..: 3 1 1 2 1 1 1 4 4 3 ...
$ doHH   : num  NA NA NA NA NA ...
$ tfHH   : num  NA NA NA NA NA ...
- attr(*, "time.scales")= chr [1:4] "per" "age" "dur" "tfHH"
- attr(*, "time.since")= chr [1:4] "" "" "" ""
- attr(*, "breaks")=List of 4
 ..$ per : NULL
 ..$ age : NULL
 ..$ dur : NULL
 ..$ tfHH: NULL
```

CODE EXPLAINED: In the data frame `cutHH` we used for cutting follow-up at the dates of HH, we have the dates of HH in the variable `cut`, and the state in the variable `new.state`. We then do a `left_join` of this to the `Lexis` object `Rx`, matching on the variables `lex.Cst` / `new.state`. This way we get the date of the HH event that is the basis for `lex.Cst`, since `lex.Cst` is the state in which follow-up starting at the date `per` takes place.

The `factorize` is necessary in order to make `lex.Cst` a factor, a propterty that was lost by using `left_join`, and the `Relevel` is then needed because `factorize` puts levels in standard alphabetical order.

In the data frame created by `rcutLexis` we have precisely one record for each state defined by no. of hypoglycemic events, where the number of events seen so far is indicated in `lex.Cst`. For records representing follow-up with at least one HH we set the new variable `tfHH` to time since this event (`per-doHH`), and for the state without any HH (`0.HH`) we set `tfHH` to `NA`.

Most importantly however, we declare `tfHH` as a timescale: a `Lexis` object has a number of attributes that are used for manipulation of the object. Among these is `time.scales`—a character vector with the names of the variables that represent time scales, `time.since`—the names of the states that have a time scale representing the time since entry to a particular state and `breaks`—a vector of points where the corresponding time scale is split. These attributes are expanded to include `tfHH` as a time scale, so that the variable `tfHH` will be properly updated when follow-up is split in smaller intervals.

For illustrational purposes we fish out persons that have spent time both in `0.HH` and `4.HH`, and take the first of these:

```
> (wh0 <- intersect(subset(Rx, lex.Cst == "4.HH")[,"lex.id"],
+                   setdiff(subset(Rx, lex.Cst == "3.HH")[,"lex.id"],
+                           subset(Rx, lex.Cst == "0.HH")[,"lex.id"]))[1])
[1] 27

> (wh1 <- intersect(subset(Rx, lex.Cst == "3.HH")[,"lex.id"],
+                   subset(Rx, lex.Cst == "0.HH")[,"lex.id"])[1])

[1] 495

> whc <- grep("HH", names(Rx))
> subset(Rx, lex.id %in% c(wh0, wh1))[,c(1:7,whc)]

 lex.id     per   age   dur tfHH lex.dur lex.Cst lex.Xst     doHH
     27 2006.97 19.62  8.32 6.08    1.69    3.HH    4.HH 2000.887
     27 2008.65 21.31 10.01 0.00    2.56    4.HH    5.HH 2008.655
     27 2011.21 23.87 12.57 0.00    0.68    5.HH    5.HH 2011.210
    495 2009.38 60.27  2.41   NA    0.21    0.HH    1.HH       NA
    495 2009.59 60.47  2.62 0.00    0.40    1.HH    2.HH 2009.592
    495 2009.99 60.87  3.01 0.00    0.53    2.HH    3.HH 2009.988
    495 2010.52 61.40  3.55 0.00    1.63    3.HH    3.HH 2010.518
```

CODE EXPLAINED: To illustrate how `tfHH` is coded, we select ids of a person that has both a 3rd and 4th HH but no recorded follow up in `0.HH`, and of a person that has follow-up both in `0.HH` and `3.HH`. We see that person 27 begins follow-up in state `3.HH` some 6 years after the 3rd HH, and that when the 4th occur, the time scale `tfHH` is reset to 0. Also we see for person 495, the `tfHH` is texttttNA while follow up is in state `0.HH`.

> Note that even if `whc`—which includes the column number of the `tfHH` variable—is mentioned last, it is anyway printed beside the other three timescale variables. This is because the `Lexis` object is printed using the `print.Lexis` method which puts the `lex.id` first, then the timescales and finally the other `lex.` variables. This is however only in the print, the internal ordering of the variables in the `Lexis` object remains the same. If you want variables printed in the order they occur in the data frame, you must use `print.data.frame`.

It is seen that person 27 begins follow-up in state `3.HH` some 6 years after the 3rd HH. Also we see that person 495 begins follow-up in state `0.HH`, that is without HH, and hence with a `NA` value for `tfHH`, and that `tfHH` is reset to 0 at every HH occurrence.

5.1.3 Simple modeling of crude rates

We can summarize the rates of next HH and rates of death by fitting simple models for the rates (corresponding to the overall rates shown in figure 5.3); for the mortality we also fit a model where 5th HH is collapsed with the subsequent ones, that is where mortality rates from states `5.HH`, `6.HH` etc. are assumed identical.

But all models assume all transitions to be constant; not varying by any of the defined time scales. Hence we need not split follow-up time in smaller intervals.

We use the facility in `glm.Lexis` that if `to` but not `from` is given, then all transitions to any state in `to` are modeled with the same model. And also the facility that if neither of the arguments `from` or `to` are given, then all transitions into any absorbing state (in this case only `Dead`) are modeled with the same model.

This does not necessarily mean that we do not distinguish different transitions, we will normally use `lex.Cst` as a covariate to provide models where some covariate effects vary between transitions.

We fit the models twice, using two different parametrizations, one without an intercept, which gives the overall rates as also shown in figure 5.3 (the `0` models), and the other with an intercept, which gives rate-ratios relative to rates originating from state `0.HH`:

```
> levels(Rx)
 [1] "0.HH"  "1.HH"  "2.HH"  "3.HH"  "4.HH"  "5.HH"  "6.HH"  "7.HH"  "8.HH"  "9.HH"
[11] "10+HH" "Dead"
> mh0<- glm.Lexis(Rx, ~ lex.Cst + 0, to = levels(Rx)[2:11])
stats::glm Poisson analysis of Lexis object Rx with log link:
Rates for transitions:
0.HH->1.HH
1.HH->2.HH
2.HH->3.HH
3.HH->4.HH
4.HH->5.HH
5.HH->6.HH
6.HH->7.HH
7.HH->8.HH
8.HH->9.HH
9.HH->10+HH
> mh <- glm.Lexis(Rx, ~ lex.Cst    , to = levels(Rx)[2:11])
stats::glm Poisson analysis of Lexis object Rx with log link:
Rates for transitions:
0.HH->1.HH
1.HH->2.HH
2.HH->3.HH
3.HH->4.HH
4.HH->5.HH
5.HH->6.HH
6.HH->7.HH
7.HH->8.HH
8.HH->9.HH
9.HH->10+HH
> md0<- glm.Lexis(Rx, ~ lex.Cst + 0)

recur
```

```
stats::glm Poisson analysis of Lexis object Rx with log link:
Rates for transitions:
0.HH->Dead
1.HH->Dead
2.HH->Dead
3.HH->Dead
4.HH->Dead
5.HH->Dead
6.HH->Dead
7.HH->Dead
8.HH->Dead
9.HH->Dead
10+HH->Dead
```

```
> md <- glm.Lexis(Rx, ~ lex.Cst)
```

```
stats::glm Poisson analysis of Lexis object Rx with log link:
Rates for transitions:
0.HH->Dead
1.HH->Dead
2.HH->Dead
3.HH->Dead
4.HH->Dead
5.HH->Dead
6.HH->Dead
7.HH->Dead
8.HH->Dead
9.HH->Dead
10+HH->Dead
```

```
> mdg<- glm.Lexis(Rx, ~ Relevel(lex.Cst,
+                               first = FALSE,
+                               list("5+" = 6:11)))
```

```
stats::glm Poisson analysis of Lexis object Rx with log link:
Rates for transitions:
0.HH->Dead
1.HH->Dead
2.HH->Dead
3.HH->Dead
4.HH->Dead
5.HH->Dead
6.HH->Dead
7.HH->Dead
8.HH->Dead
9.HH->Dead
10+HH->Dead
```

With the models fitted we can test whether it is a reasonable reduction of the mortality models to lump 5^{th} and subsequent HHs together as far as mortality rates are concerned:

```
> anova(md, mdg, test = "Chisq")
Analysis of Deviance Table

Model 1: cbind(trt(Lx$lex.Cst, Lx$lex.Xst) %in% trnam, Lx$lex.dur) ~ lex.Cst
Model 2: cbind(trt(Lx$lex.Cst, Lx$lex.Xst) %in% trnam, Lx$lex.dur) ~ Relevel(lex.Cst,
    first = FALSE, list(`5+` = 6:11))
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1     21006     5560.2
2     21011     5569.7 -5  -9.5219  0.08997 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the `anova` we see that there is no compelling evidence against pooling states to `5+` for mortality rates.

We can show the overall rates (per 100 PY) corresponding to the rates shown in figure 5.3

```
> round(ci.exp(mh0) * 100, 1) # HH incidence rates
```

```
            exp(Est.) 2.5% 97.5%
lex.Cst0.HH       1.7  1.6   1.8
lex.Cst1.HH       4.8  4.5   5.2
lex.Cst2.HH       8.4  7.6   9.3
lex.Cst3.HH      12.9 11.4  14.6
lex.Cst4.HH      17.4 15.1  20.0
lex.Cst5.HH      22.9 19.5  27.0
lex.Cst6.HH      27.4 22.6  33.1
lex.Cst7.HH      29.6 23.4  37.4
lex.Cst8.HH      32.6 25.1  42.5
lex.Cst9.HH      42.1 31.8  55.7
> round(ci.exp(md0) * 100, 1) # mortality rates
             exp(Est.) 2.5% 97.5%
lex.Cst0.HH        0.6  0.6   0.7
lex.Cst1.HH        1.2  1.0   1.4
lex.Cst2.HH        1.4  1.1   1.8
lex.Cst3.HH        2.3  1.8   3.1
lex.Cst4.HH        2.7  1.9   3.9
lex.Cst5.HH        1.8  1.0   3.2
lex.Cst6.HH        2.3  1.2   4.5
lex.Cst7.HH        5.5  3.2   9.5
lex.Cst8.HH        2.4  0.9   6.3
lex.Cst9.HH        4.3  1.8  10.3
lex.Cst10+HH       3.5  2.2   5.7
```

We recognize the rates as those from figure **??**, but after modeling we also have confidence limits for the rates.

In order to show how the rate ratios relative to `0.HH` depends on the HH state we make a forest plot of the RRs:

```
> # only the RRs
> rrH <- ci.exp(mh)[-1,]
> rrD <- ci.exp(md)[-1,]
> rrG <- ci.exp(mdg)[c(2:5,rep(6,6)),]
> # groom the parameter labels
> rownames(rrD) <- gsub("lex.Cst", "", rownames(rrD))
> plotEst(rrD, y = 9:0 - 0.05, col = gray(0.5),
+          xlog = TRUE, lwd = 3, xlim = c(1,30),
+  xlab = "RR of death / next HH, relative to rates from state 0.HH (= no HH)")
> linesEst(rrH, y = 9:1 + 0.1, col = "blue", lwd = 3)
> linesEst(rrG, y = 9:0 - 0.2, col = gray(0.0), lwd = 3)
> axis(side = 1, at = c(1,15,30))
> axis(side = 1, at = c(2:10,15,20,25), labels = NA)
```

From figure 5.4 we see that the rates of HH increase by increasing number of HH episodes, whereas mortality only do so till about `5.HH`, and less so than rates of HH.

This analysis of rates is however rudimentary; the only covariate we use is the state, `lex.Cst`, rates are assumed to be constant over time; so we have basically just reproduced the raw rates shown in figure 5.3, albeit with confidence intervals.

5.1.4 Proper analysis of HH rates

A proper analysis of rates of HH and mortality will include both effects of age, diabetes duration as well as time since the most recent HH event (`tfHH` as we defined above). Therefore, proper analysis of rates requires that follow-up be split in intervals so that we can use the value of the time scales at the beginning of each interval. Of course we will make the intervals sufficiently small to make the assumption of constant rates in each interval reasonable.

5.1.4.1 Splitting the follow-up

This can be achieved by `splitMulti`; in this instance we split follow-up in intervals of 3 months length along the age-scale. It does not matter much which time scale we use; all time scales are updated by the splitting machinery:

```
recur
```

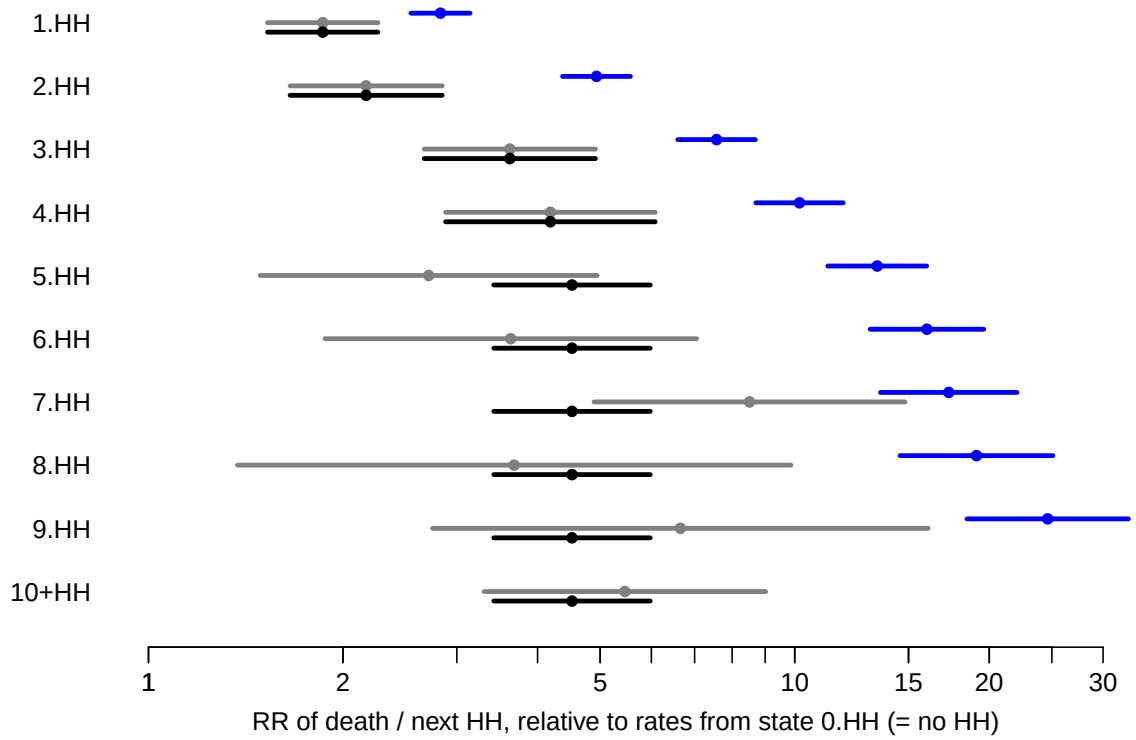


Figure 5.4: *Rate-ratios of death and next HH relative to the* `0.HH` *state. The blue bars are the RRs of next HH event, the black and gray bars are from the two different models for the mortality rates; the black bars are from the model with identical effect for no. of HHs equal to 5 or more.* `../graph/recur-RRforest`

```
> Sx <- splitMulti(Rx, age = seq(0, 90, 1/4))
> rbind(nrow(Rx), nrow(Sx))
      [,1]
[1,]  21017
[2,] 313771
> whx <- subset(Rx, lex.Cst == "0.HH" & lex.dur > 0.4 & lex.dur < 0.9)$lex.id
> why <- subset(Rx, lex.Cst == "1.HH" & lex.dur > 0.4 & lex.dur < 0.9)$lex.id
> (wh2 <- intersect(whx, why)[1])
[1] 378
> subset(Rx, lex.id %in% wh2)[,c(1:8,23)]
 lex.id     per   age   dur tfHH lex.dur lex.Cst lex.Xst      dob
    378 2006.80 53.42 11.32   NA    0.60    0.HH    1.HH 1953.383
    378 2007.40 54.01 11.92    0    0.66    1.HH    2.HH 1953.383
    378 2008.06 54.68 12.58    0    0.80    2.HH    Dead 1953.383
> subset(Sx, lex.id %in% wh2)[,1:9]
 lex.id     per   age   dur tfHH lex.dur lex.Cst lex.Xst      dob
    378 2006.80 53.42 11.32   NA    0.08    0.HH    0.HH 1953.383
    378 2006.88 53.50 11.41   NA    0.25    0.HH    0.HH 1953.383
    378 2007.13 53.75 11.66   NA    0.25    0.HH    0.HH 1953.383
    378 2007.38 54.00 11.91   NA    0.01    0.HH    1.HH 1953.383
    378 2007.40 54.01 11.92 0.00    0.24    1.HH    1.HH 1953.383
    378 2007.63 54.25 12.16 0.24    0.25    1.HH    1.HH 1953.383
    378 2007.88 54.50 12.41 0.49    0.18    1.HH    2.HH 1953.383
    378 2008.06 54.68 12.58 0.00    0.07    2.HH    2.HH 1953.383
    378 2008.13 54.75 12.66 0.07    0.25    2.HH    2.HH 1953.383
    378 2008.38 55.00 12.91 0.32    0.25    2.HH    2.HH 1953.383
    378 2008.63 55.25 13.16 0.57    0.23    2.HH    Dead 1953.383
```

CODE EXPLAINED: `splitMulti` subdivides the follow-up time across several records, each representing a small interval of follow-up, in this case intervals of 6 months age.

For illustration of the data for a single person, we select persons with follow-up in both `0.HH` and `1.HH` between 0.4 (to be sure that the split creates more than one interval) and 0.8 (to be sure that the split does not create too many intervals) — and then just select the first of the intersection of the two sets of persons.

From the last part of the output we see that the timescale `tfHH` is reset to 0 at every new HH occurrence, but also that `tfHH` is correctly updated at the following points of time-splitting, and is correctly propagated as `NA` for the intervals with no preceding HH.

```
> Sx <- tsNA20(Sx, all = TRUE)
> subset(Sx, lex.id %in% wh2)[,1:9]
 lex.id     per   age   dur tfHH lex.dur lex.Cst lex.Xst      dob
    378 2006.80 53.42 11.32 0.00    0.08    0.HH    0.HH 1953.383
    378 2006.88 53.50 11.41 0.00    0.25    0.HH    0.HH 1953.383
    378 2007.13 53.75 11.66 0.00    0.25    0.HH    0.HH 1953.383
    378 2007.38 54.00 11.91 0.00    0.01    0.HH    1.HH 1953.383
    378 2007.40 54.01 11.92 0.00    0.24    1.HH    1.HH 1953.383
    378 2007.63 54.25 12.16 0.24    0.25    1.HH    1.HH 1953.383
    378 2007.88 54.50 12.41 0.49    0.18    1.HH    2.HH 1953.383
    378 2008.06 54.68 12.58 0.00    0.07    2.HH    2.HH 1953.383
    378 2008.13 54.75 12.66 0.07    0.25    2.HH    2.HH 1953.383
    378 2008.38 55.00 12.91 0.32    0.25    2.HH    2.HH 1953.383
    378 2008.63 55.25 13.16 0.57    0.23    2.HH    Dead 1953.383
```

CODE EXPLAINED: The function `tsNA20` (time scales `NA 2 0`) changes `NA`s in time scale variables to 0, to make modeling with `tfHH` viable.

In the modeling we need to change the `NA`s in `tfHH` to 0 in order to be able to use `tfHH` as covariate in a model—records with `tfHH` equal to `NA` would be excluded from computations. We distinguish records that refer to follow up before any HH by the variable `lex.Cst`, which will be `0.HH` before any HH, and `1.HH`, `2.HH` etc. as more instances of HH occur.

We have to keep this coding in mind when manipulating `Sx`—for example it would be meaningless to further split along a(ny) time scale, because the time in `0.HH` would not preserve `tfHH` as 0 as it should.

5.1.5 More realistic models for rates

The relevant variables in models for HH rates will be current age, current duration of diabetes and time since last HH event. Note that the latter is only defined for persons with at least one HH, therefore the time scale representing time since most recent HH (`tfHH`) was initially coded as `NA`, but for analysis converted to 0 in order to be included in the modeling.

We define a model where we have a common effect of duration of diabetes (`dur`) regardless of HH status.

On top of this there will be a separate effect of time since last HH (`tfHH`) for each level of no. of previous HHs. This is where the recoding of the `NA` is important; 0 will both refer to the `0.HH` state (at any time of follow-up), but also to the first intervals in each of the states `x.HH` for any `x`. Therefore, a model must necessarily include state as a factor, otherwise we will impose the restriction on the initial rates in each `x.HH` state that they are equal to the rates in the `0.HH` state—a completely silly assumption.

Thus, the HH effect for persons in state `1.HH` will be the ratio of the rate of the 2nd HH relative to the rate of the 1st HH, as a function of the time since 1st HH. The effect for persons in state `2.HH` will be the rate-ratio of the 3rd HH relative to the rate of the 1st HH, as a function of the time since 2nd HH.

Alternatively we could show the rate-ratio of the 3rd HH relative to the rate of the 2nd HH, as a function of the time since 2nd HH—the most recent HH.

Note that in both cases, we will be comparing rates at a given time since 2nd HH with rates at the *same* time since 1st.

Ultimately, we will of course be showing both sets of parameter estimates from the model; the first one will have the `0.HH` state as reference, the second will be comparisons of rates of HH between successive levels of no. of HH events.

`recur`

In addition to this we will have a separate effect of age for each sex, and an effect of diabetes duration (`dur`), so the model we will be using can be coded as:

```
> system.time(
+ mAC <- glm.Lexis(Sx,
+                   ~ lex.Cst * Ns(tfHH, knots = c(0,1,3,6)) +
+                               Ns(dur, knots = c(1:5 * 10)) +
+                         sex * Ns(age, knots = c(4:8 * 10)),
+                   to = levels(Sx)[2:11],
+                scale = 100))
stats::glm Poisson analysis of Lexis object Sx with log link:
Rates for transitions:
0.HH->1.HH
1.HH->2.HH
2.HH->3.HH
3.HH->4.HH
4.HH->5.HH
5.HH->6.HH
6.HH->7.HH
7.HH->8.HH
8.HH->9.HH
9.HH->10+HH
, lex.dur (person-time) scaled by 100
   user  system elapsed
  22.12    0.79   22.92

> round(ci.exp(mAC), 2)
                                                exp(Est.)  2.5% 97.5%
(Intercept)                                          1.74  1.58  1.91
lex.Cst1.HH                                         15.07 12.00 18.92
lex.Cst2.HH                                         20.07 15.77 25.54
lex.Cst3.HH                                         20.51 15.43 27.27
lex.Cst4.HH                                         32.18 24.13 42.93
lex.Cst5.HH                                         32.16 23.11 44.75
lex.Cst6.HH                                         35.39 24.40 51.33
lex.Cst7.HH                                         49.79 33.75 73.47
lex.Cst8.HH                                         49.90 31.45 79.18
lex.Cst9.HH                                         38.60 22.31 66.79
Ns(tfHH, knots = c(0, 1, 3, 6))1                     1.23  0.40  3.76
Ns(tfHH, knots = c(0, 1, 3, 6))2                     0.23  0.06  0.94
Ns(tfHH, knots = c(0, 1, 3, 6))3                     1.09  0.45  2.63
Ns(dur, knots = c(1:5 * 10))1                        0.94  0.79  1.13
Ns(dur, knots = c(1:5 * 10))2                        1.23  1.05  1.43
Ns(dur, knots = c(1:5 * 10))3                        1.20  1.00  1.44
Ns(dur, knots = c(1:5 * 10))4                        1.16  1.02  1.33
sexW                                                 0.84  0.74  0.95
Ns(age, knots = c(4:8 * 10))1                        1.14  0.89  1.45
Ns(age, knots = c(4:8 * 10))2                        0.95  0.75  1.20
Ns(age, knots = c(4:8 * 10))3                        1.02  0.85  1.22
Ns(age, knots = c(4:8 * 10))4                        1.31  1.04  1.65
lex.Cst1.HH:Ns(tfHH, knots = c(0, 1, 3, 6))1         0.30  0.10  0.95
lex.Cst2.HH:Ns(tfHH, knots = c(0, 1, 3, 6))1         0.19  0.06  0.63
lex.Cst3.HH:Ns(tfHH, knots = c(0, 1, 3, 6))1         0.35  0.11  1.17
lex.Cst4.HH:Ns(tfHH, knots = c(0, 1, 3, 6))1         0.19  0.06  0.65
lex.Cst5.HH:Ns(tfHH, knots = c(0, 1, 3, 6))1         0.54  0.15  1.98
lex.Cst6.HH:Ns(tfHH, knots = c(0, 1, 3, 6))1         0.43  0.11  1.67
lex.Cst7.HH:Ns(tfHH, knots = c(0, 1, 3, 6))1         0.23  0.04  1.22
lex.Cst8.HH:Ns(tfHH, knots = c(0, 1, 3, 6))1         0.42  0.08  2.21
lex.Cst9.HH:Ns(tfHH, knots = c(0, 1, 3, 6))1         1.00  1.00  1.00
lex.Cst1.HH:Ns(tfHH, knots = c(0, 1, 3, 6))2         0.15  0.03  0.69
lex.Cst2.HH:Ns(tfHH, knots = c(0, 1, 3, 6))2         0.22  0.05  1.02
lex.Cst3.HH:Ns(tfHH, knots = c(0, 1, 3, 6))2         0.51  0.11  2.44
lex.Cst4.HH:Ns(tfHH, knots = c(0, 1, 3, 6))2         0.25  0.05  1.23
lex.Cst5.HH:Ns(tfHH, knots = c(0, 1, 3, 6))2         0.41  0.08  2.14
lex.Cst6.HH:Ns(tfHH, knots = c(0, 1, 3, 6))2         0.42  0.08  2.32
lex.Cst7.HH:Ns(tfHH, knots = c(0, 1, 3, 6))2         0.13  0.02  0.85
```

```
lex.Cst8.HH:Ns(tfHH, knots = c(0, 1, 3, 6))2      0.20  0.03  1.40
lex.Cst9.HH:Ns(tfHH, knots = c(0, 1, 3, 6))2      1.00  1.00  1.00
lex.Cst1.HH:Ns(tfHH, knots = c(0, 1, 3, 6))3      0.42  0.17  1.04
lex.Cst2.HH:Ns(tfHH, knots = c(0, 1, 3, 6))3      0.27  0.11  0.67
lex.Cst3.HH:Ns(tfHH, knots = c(0, 1, 3, 6))3      0.29  0.12  0.75
lex.Cst4.HH:Ns(tfHH, knots = c(0, 1, 3, 6))3      0.24  0.09  0.64
lex.Cst5.HH:Ns(tfHH, knots = c(0, 1, 3, 6))3      0.30  0.11  0.82
lex.Cst6.HH:Ns(tfHH, knots = c(0, 1, 3, 6))3      0.39  0.13  1.14
lex.Cst7.HH:Ns(tfHH, knots = c(0, 1, 3, 6))3      0.16  0.04  0.60
lex.Cst8.HH:Ns(tfHH, knots = c(0, 1, 3, 6))3      0.29  0.08  1.09
lex.Cst9.HH:Ns(tfHH, knots = c(0, 1, 3, 6))3      1.00  1.00  1.00
sexW:Ns(age, knots = c(4:8 * 10))1                0.67  0.45  1.01
sexW:Ns(age, knots = c(4:8 * 10))2                1.43  1.00  2.03
sexW:Ns(age, knots = c(4:8 * 10))3                1.43  1.12  1.81
sexW:Ns(age, knots = c(4:8 * 10))4                1.42  1.04  1.93
```

Note that we set the `scale` argument to 100, so that the parameters and predictions we get automatically are rescaled to rates per 100 years (or more precisely, per 100 units of `lex.dur`).

The parameter estimates from the model are not informative *per se*, we can only report the model estimates as curves of predicted rates and rate ratios for select values of the covariates.

We show rates of first HH in ages 30 through 80 for men and women in state `0.HH` with diabetes duration 20 years (approximately the median duration at follow-up) and time since HH equal to 0 (this is the only value that `tfHH` assumes in state `0.HH`, (`lex.Cst` = `0.HH` ⇒ `tfHH` = 0).

Note that since we fix duration of diabetes at 20 years, we are showing *cross-sectional* rates, not rates as they evolve for a cohort of persons:

```
> pra <- data.frame(age = 30:80,
+                   dur = 20,
+                  tfHH = 0,
+                lex.Cst = factor("0.HH", levels = levels(Sx)),
+                   sex = factor( "M", levels = levels(Sx$sex)))
> matshade(pra$age, cbind(aa <- ci.pred(mAC, pra),
+                                ci.pred(mAC, transform(pra, sex = "W"))),
+          plot = TRUE, col = c("blue", "red"),
+          log = "y", lwd = 3, ylim = c(1,100),
+          xlab = "Attained age (years)",
+          ylab = "Rate of first HH (men) per 100 PY")
> abline(h = aa[21,1], lty = "42", lwd = 2)
> points(50, aa[21,1], pch = 16, cex = 1.4, col = "white")
> points(50, aa[21,1], pch =  1, cex = 1.4, col = "blue", lwd = 3)
```

We also want to show how the effect of `x.HH`, so we show the rates of next HH for a 50 year old man for different levels of HH (`lex.Cst`):

```
> tHH <- 0:50 / 10
> prx <- function(x, sx){
+ data.frame(age = 50,
+             dur = 20,
+            tfHH = tHH,
+         lex.Cst = factor(paste0(x,".HH"), levels = levels(Sx)),
+             sex = factor(sx, levels = levels(Sx$sex)))
+                     }
> for(x in 1:9)
+    {
+ matshade(prx(x, "M")$tfHH,
+          # the pmin/pmax is just a dirty trick to allow the c.i.
+          # shades to be shown - it is a windows graphics driver
+          # problem.
+          # pmax(pmin(
+          zz <- ci.pred(mAC, prx(x, "M")),
+          #          100), 1),
+          plot = (x == 1),
+          log = "y", lwd = 3, col = gray(x/12),
+          ylim = c(1,100),
+          xlab = "Time since latest HH (years)",
```

recur

```
+          ylab = "Rate of n'th HH (men) per 100 PY")
+ text(0, zz[1,1], paste(x), adj = 1.6)
+    }
> text(0, 100, "HH number", adj = 0.5)
> abline(h = aa[21,1], lty = "42", lwd = 2)
```

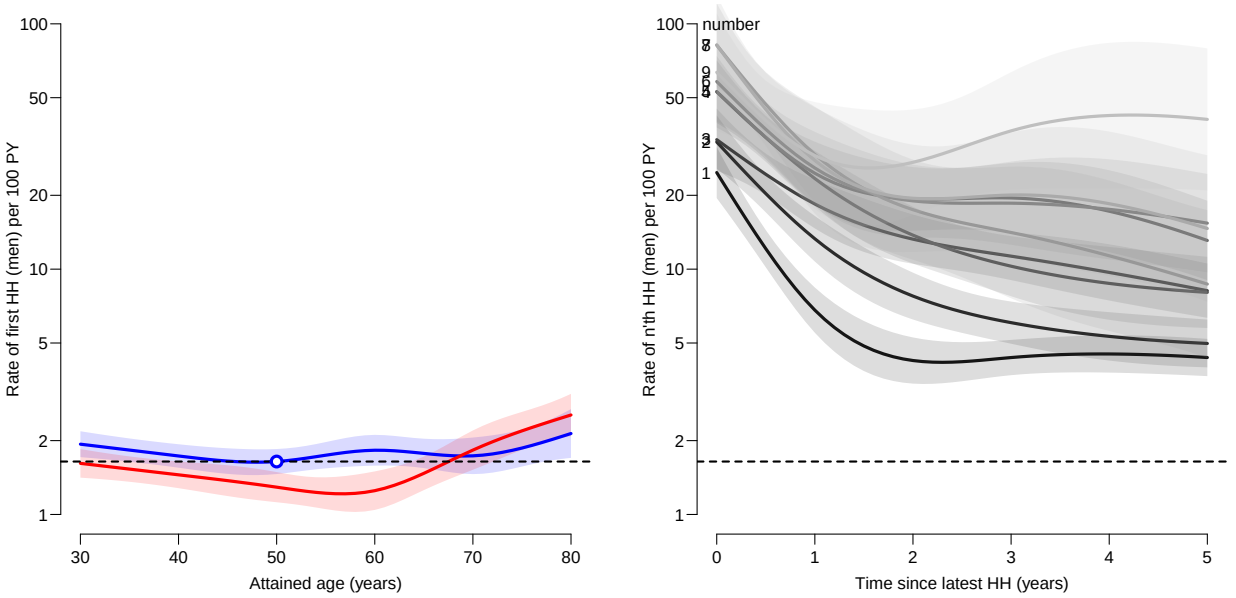


Figure 5.5: *Effects of age (left) and time since last HH (right) at diabetes duration 20 years. The horizontal dotted lines correspond to the rates for men aged 50 in the left panel, as illustrated by the blob.*

The rightmost panel in figure 5.5 is a graphical rendering of the interaction between HH level and time since last HH. The curves have largely the same shape just different levels, so it actually makes sense to assess whether the interaction really is there by fitting the model without interaction between `lex.Cst` and `tfHH`:

```
> system.time(
+ mAc <- glm.Lexis(Sx,
+                   ~ lex.Cst + Ns(tfHH, knots = c(0,1,3,6)) +
+                               Ns(dur, knots = c(1:5 * 10)) +
+                         sex * Ns(age, knots = c(4:8 * 10)),
+                   to = levels(Sx)[2:11],
+                scale = 100))
stats::glm Poisson analysis of Lexis object Sx with log link:
Rates for transitions:
0.HH->1.HH
1.HH->2.HH
2.HH->3.HH
3.HH->4.HH
4.HH->5.HH
5.HH->6.HH
6.HH->7.HH
7.HH->8.HH
8.HH->9.HH
9.HH->10+HH
, lex.dur (person-time) scaled by 100
   user  system elapsed
   6.84    0.51    7.36
> anova(mAC, mAc, test = "Chisq")
```

```
Analysis of Deviance Table

Model 1: cbind(trt(Lx$lex.Cst, Lx$lex.Xst) %in% trnam, Lx$lex.dur) ~ lex.Cst *
    Ns(tfHH, knots = c(0, 1, 3, 6)) + Ns(dur, knots = c(1:5 *
    10)) + sex * Ns(age, knots = c(4:8 * 10))
Model 2: cbind(trt(Lx$lex.Cst, Lx$lex.Xst) %in% trnam, Lx$lex.dur) ~ lex.Cst +
    Ns(tfHH, knots = c(0, 1, 3, 6)) + Ns(dur, knots = c(1:5 *
    10)) + sex * Ns(age, knots = c(4:8 * 10))
  Resid. Df Resid. Dev  Df Deviance Pr(>Chi)
1    311758      27848
2    311782      27896 -24  -48.271 0.002336 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The formal likelihood ratio tests rejects the addititivity hypothesis, confirming an interaction between HH state and time since most recent HH, on strict testing grounds. But we also see that the deviance to d.f. ratio is only about 2. We will use this observation as a the basis for reporting from the model with no interaction between state and time.

The code for predicting from the model is precisely the same as for the interaction model, here in a 2 by 1 layout:

```
> # function to plot age effect
> plage <- function() {
+ matshade(pra$age, aa <<- ci.pred(mAc, pra),
+           plot = TRUE, col = "white",
+           log = "y", lwd = 3, ylim = c(1,100),
+           xlab = "Attained age (years)",
+           ylab = "Rate of first HH per 100 PY")
+ abline(h = aa[21,1], lty = "42", lwd = 2, col = "gray")
+ matshade(pra$age, aa, lwd = 3, col = "blue")
+ matshade(pra$age, ci.pred(mAc, mutate(pra, sex = "W")),
+                       lwd = 3, col = "red")
+ points(50, aa[21,1], pch = 16, cex = 1.4, col = "white")
+ points(50, aa[21,1], pch =  1, cex = 1.4, lwd = 2, col = "blue")
+                      }
> # function to plot duration effects
> pldur <- function() {
+ for(x in 9:1)
+    {
+ matshade(prx(x, "M")$tfHH,
+           dd <<- ci.pred(mAc, prx(x, "M")),
+           plot = (x == 9),
+           log = "y", lwd = 3, ylim = c(1,100), col = gray(x/12),
+           xlab = "Time since last HH (years)",
+           ylab = "Rate of next HH (men, aged 50) per 100 PY")
+ text(0, dd[1,1], paste(x), adj = 1.6)
+    }
+ text(0, 100, "HH number", adj = 0.2)
+ abline(h = aa[21,1], lty = "42", lwd = 2, col = "gray")
+                      }
> par(mfrow = c(1, 2), oma = c(0,0,0,2.5),
+     mgp = c(3,1,0) / 1.6, las = 1, bty = "n")
> plage()
> pldur()
```

This will show the rates of first HH for men and women in different ages as well as the absolute rates of next HH in 50 year old men as a function of time since most recent HH for different numbers of HHs seen so far. The corresponding curves for women or men in other ages would look precisely the same (by model assumption) just at a different absolute level, because there is no interaction between `tfHH` and other variables.

It is seen (figure 5.6) that the rates of the next HH are very high shortly after a HH episode; there is a drop by a factor 2.5 over the first two years. Also, the rates of next HH are increasing by number of previous HH episodes, most pronounced in the beginning where rates for persons with 1, 2 and 3 previous HH episodes are some 5, 10 and 15 times higher than those of persons with no previous HH episodes.

`recur`

5.1.5.1 Choice of reference

Since the time scale `tfHH` is not defined for persons without HH (and hence coded 0 for these) the rate-ratio of rate of next HH between person with 1, 2, 3... HH episodes and persons with 0 episodes will look precisely the same as the right panel in figure 5.6, except for the y-axis, that would have the dotted lines at a y-value of 1. There is no interaction between `tfHH` and age and sex, so it is immaterial what reference for age and sex that was used.

However the *confidence limits* of the rate-ratios will not be the same, because we will be making a comparison between rates in state `1.HH`, say, and state `0.HH`. Here we compute and graph the estimated RRs relative to persons with 0 prior HH:

```
> # function to plot RR relative to 0.HH
> plRR <- function(yl) {
+ pr0 <- data.frame(age = 50,
+                   dur = 20,
+                  tfHH = 0,
+                lex.Cst = factor("0.HH", levels = levels(Sx)),
+                   sex = factor("M", levels = levels(Sx$sex)))
+ for(x in 9:1)
+    {
+ matshade(prx(x, "M")$tfHH,
+          rr <<- ci.exp(mAc,
+                        list(prx(x, "M"),
+                             pr0[,colnames(prx(x, "M"))])),
+          plot = (x == 9),
+          log = "y", lwd = 3, ylim = yl, col = gray(x/12),
+          xlab = "Time since last HH (years)",
+          ylab = "RR of next HH for men, aged 50, relative to 0 HH")
+ text(0, rr[1,1], paste(x), adj = 1.6)
+    }
+ text(0, 100, "HH number", adj = 0.2)
+ abline(h = 1, lty = "42", lwd = 2, col = "gray")
+                       }
```

With the RR function we can now plot the three sets of estimates next to each other:

```
> par(mfrow=c(1,3), mar = c(3,3,0,0), oma = c(0,0,0,0),
+     mgp = c(3,1,0) / 1.6, las = 1, bty = "n")
> plage()
> pldur()
> # yl <- (10^par("usr")[3:4]) / aa[21,1]
> yl <- c(1,100) / aa[21,1]
> plRR(yl)
```

The shape of the curves corresponds to a decline in rates of about 3.8 fold during the first two years:

```
> round(rbind(
+ c(dd[1,1], dd[21,1], dd[1,1] / dd[21,1]),
+ c(rr[1,1], rr[21,1], rr[1,1] / rr[21,1])), 2)
      [,1] [,2] [,3]
[1,] 20.98 5.48 3.83
[2,] 12.75 3.33 3.83
```

Moreover, there is a clear increase of rates of next HH by the number of HHs already seen, most pronounced in the beginning.

We have collected the absolute rates at time 1 in the matrix `dd` and the rate-ratios at time in `rr`, and we see that the rate ratio is equal to the rates divided by the reference rate, but also that the the confidence intervals for the `rr` are narrower:

```
> R <- cbind(rates = dd[1,],
+       "rates/ref" = dd[1,] / aa[21,1],
+         "real RR" = rr[1,])
> rbind(R, "rel. width" = R[3,] / R[2,])
```

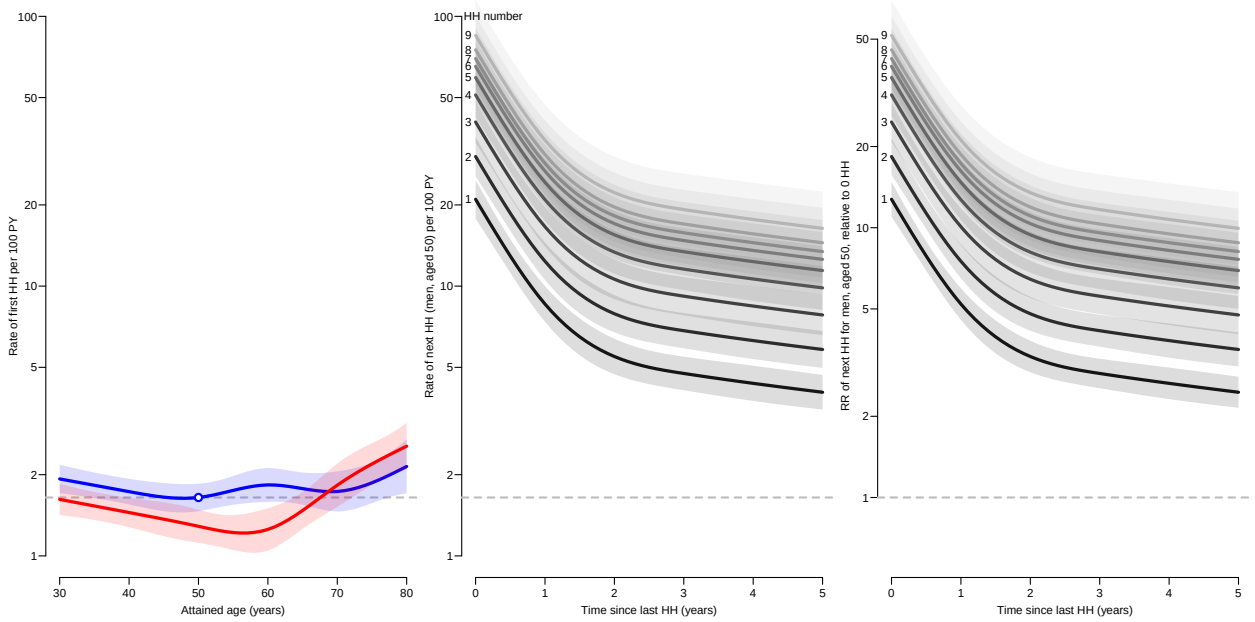


Figure 5.6: *Effects of age, sex and time since last HH from a model with no interaction between HH number of HH and time since latest HH. The left panel shows the rate of 1st HH as a function of age for men and women, while the middle panel shows the rate of next HH for 50 year old men with 1, 2,...previous HH episodes as function of the time since the most recent HH episode, and the rightmost panel shows the RR of next HH episode between persons in states* `x.HH` *(*`x` *= 1,2,...) and persons in state 0.HH.*
The horizontal dotted lines correspond to the blob at age 50 for men in the left panel, that is rates of first HH in 50 year old men. `../graph/recur-R3add`

```
               rates rates/ref   real RR
Estimate   20.983117 12.746024 12.746024
2.5%       17.795702 10.809855 10.985318
97.5%      24.741434 15.028984 14.788933
rel. width  1.390304  1.390304  1.346245
```

Heuristically speaking, the confidence intervals for the rate ratios are narrower (lower limit larger, upper linit smaller) because they do not involve the estimates of the absolute rates, but only the ratios.

5.2 Random effects: frailty

We have seen that the rate of HH occurrence increases by the number of HH episodes seen, and that the rate decreases by time since the most recent one. These two features are typical of selection by frailty—the individual propensity to event occurrence: Early after a HH the most frail experience another event, so as time goes by, the average frailty among those left in a given state drops, causing the rate of events among those (alive and) still without event to decrease.

A similar effect is seen as the number of HH events increase; among those who have another HH event, the average frailty is higher than in those who do not, so the more HH events a person has had the more likely it is that the person has a high frailty, and thus yet another HH event. That is why the rate of the next HH event increases by the number of HH events already seen.

It would be useful if we could estimate a separate frailty for each person, but this is not possible for persons without any HH events, and very shaky for persons with only very few events.

So the way to go about this is using a *random effects* model, where we instead of estimating the frailty explicitly for each person estimate characteristics of the frailty distribution. With that done, we can then derive the posterior me(di)an of each person's frailty.

`recur`

It is common to use a normal distribution for the frailty on the linear predictor scale; corresponding to a log-normal distribution on the rate scale. However, with 18,000 persons of which slightly less than 6,000 see an HH event, the normal distribution is not a credible distribution—those without events would be expected to have a very small frailty.

The logical thing would be to allow for an atom close to 0 in the distribution of the frailty on the rate scale, very large and negative on the on the log-rate scale. Almost all persons with no HH events would be put in that class, we could call it the "immunity" class with a very small rate of first HH. It may very well be so that not all the persons without observed HH event are effectively immune to HH.

If we knew the underlying rates of first HH (as a function of time, age etc.), we could for each person without a HH event compute the survival probability for the person's observed sojourn time in the `0.HH` state, given a frailty of 1. The larger this survival probability is, the more likely it is that the true frailty of the person is very small (that is numerically large but negative).

The proper formulation of this would be that there is some probability that a person belongs to the immune class where the frailty is 0, and with the complementary probability belongs to the class of susceptible where the distribution of frailties is log-normal, say.

Having fitted that sort of model we can for each person compute the posterior probability that the person belongs to one of the two classes, as well as the posterior mean conditional on being in the susceptible class. We could illustrate this by showing the posterior probabilities of immunity, as well as the distribution of frailties among persons with posterior probability of non-immunity larger than 0, 0.1, 0.2 etc.

The main purpose of fitting such a model would be to see how much the frailty between persons could explain of the effect of number of HHs. But we do not currently have any software that would fit these models. Except possibly for JAGS that would estimate parameters by MCMC, but with the analysis dataset `Sx` with 300,000+ records this is likely to be very time consuming.

5.2.0.1 Cox-models

We can fit two different Cox models that mimic the Poisson model, by naming one of the time-scales to be the baseline scale; recall that the Poisson model was specified as:

```
> mha <- glm.Lexis(Sx,
+                  ~ lex.Cst + Ns(tfHH, knots = c(0,1,3,6)) +
+                              Ns(dur, knots = c(1:5 * 10)) +
+                         sex * Ns(age, knots = c(4:8 * 10)),
+                  to = levels(Sx)[2:11])
stats::glm Poisson analysis of Lexis object Sx with log link:
Rates for transitions:
0.HH->1.HH
1.HH->2.HH
2.HH->3.HH
3.HH->4.HH
4.HH->5.HH
5.HH->6.HH
6.HH->7.HH
7.HH->8.HH
8.HH->9.HH
9.HH->10+HH
```

When using `dur` as underlying time scales we will do:

```
> system.time(
+ cmd <- coxph.Lexis(Sx,
+                   dur ~ factor(lex.Cst) +
+                               Ns(tfHH, knots = c(0,1,3,6)) +
+                          sex * Ns(age, knots = c(4:8 * 10)),
+                    to = levels(Sx)[2:11]))
survival::coxph analysis of Lexis object Sx:
Rates for transitions:
0.HH->1.HH
1.HH->2.HH
2.HH->3.HH
```

```
3.HH->4.HH
4.HH->5.HH
5.HH->6.HH
6.HH->7.HH
7.HH->8.HH
8.HH->9.HH
9.HH->10+HH
Baseline timescale: dur
   user  system elapsed
  20.87    0.48   21.36
```

CODE EXPLAINED: Note that we use `factor(lex.Cst)` in the model specification; this is to avoid extra aliased parameters in the coefficients—it returns a factor with only the levels actually assumed. `coxph` does not remove non-existing factor levels from the coefficients vector.

But if we want to use `age` as underlying time scales we must accommodate the fact that we have a model with separate age-effects for the two sexes. That is normally called a stratified model in Cox-model lingo, and fitted by adding a `strata()` function in the model formula:

```
> system.time(
+ cma <- coxph.Lexis(Sx,
+                     age ~ factor(lex.Cst) +
+                           Ns(tfHH, knots = c(0,1,3,6)) +
+                           Ns(dur, knots = c(1:5 * 10)) +
+                           strata(sex),
+                     to = levels(Sx)[2:11]))
survival::coxph analysis of Lexis object Sx:
Rates for transitions:
0.HH->1.HH
1.HH->2.HH
2.HH->3.HH
3.HH->4.HH
4.HH->5.HH
5.HH->6.HH
6.HH->7.HH
7.HH->8.HH
8.HH->9.HH
9.HH->10+HH
Baseline timescale: age
   user  system elapsed
  10.61    0.40   11.03
```

We can compare the estimated regression parameters from the two Cox models with those from the Poisson model where all effects are modeled parametrically. We an extract the regression parameters from each of the Cox models and compare with the corresponding parameters from the Poisson-model.

Note the two extra rows of `NA`s to make up for the fact that `coxph` also produces estimates for factor levels not in the model:

```
> # dirty trick to shrink 1st column label so we get nicer prints
> ci.exp <- function(...){ z <- Epi::ci.exp(...)
+                  colnames(z)[1] <- "   Est"; z }
> round(cbind(
+       ci.exp(mha, subset = "lex.Cst"),
+       ci.exp(cmd, subset = "lex.Cst"),
+       ci.exp(cma, subset = "lex.Cst")), 2)
               Est  2.5% 97.5%    Est  2.5% 97.5%    Est  2.5% 97.5%
lex.Cst1.HH  12.75 10.99 14.79  12.69 10.82 14.88  12.94 11.02 15.20
lex.Cst2.HH  18.36 15.71 21.45  18.28 15.47 21.60  18.69 15.81 22.08
lex.Cst3.HH  24.65 20.87 29.13  24.51 20.68 29.05  24.92 21.03 29.53
lex.Cst4.HH  31.06 25.94 37.20  30.89 25.61 37.26  31.16 25.80 37.64
lex.Cst5.HH  36.02 29.60 43.84  36.17 29.62 44.18  36.95 30.20 45.22
lex.Cst6.HH  39.63 31.79 49.39  39.68 31.52 49.95  39.74 31.53 50.09
lex.Cst7.HH  42.34 32.76 54.74  43.79 33.76 56.79  43.13 32.87 56.59
lex.Cst8.HH  45.65 34.32 60.73  45.52 33.97 60.99  45.57 33.96 61.14
lex.Cst9.HH  51.63 38.25 69.68  52.97 39.17 71.62  52.61 38.87 71.20
```

recur

```
> round(cbind(
+       ci.exp(mha, subset = "tfHH"),
+       ci.exp(cmd, subset = "tfHH"),
+       ci.exp(cma, subset = "tfHH")), 2)

                                    Est 2.5% 97.5%    Est 2.5% 97.5%    Est 2.5% 97.5%
Ns(tfHH, knots = c(0, 1, 3, 6))1   0.33 0.28  0.38   0.33 0.28  0.38   0.32 0.28  0.38
Ns(tfHH, knots = c(0, 1, 3, 6))2   0.06 0.05  0.08   0.06 0.05  0.09   0.06 0.05  0.08
Ns(tfHH, knots = c(0, 1, 3, 6))3   0.36 0.31  0.41   0.36 0.31  0.41   0.36 0.31  0.41

> round(cbind(
+       ci.exp(mha, subset = "dur"),
+       ci.exp(cma, subset = "dur")), 2)

                                  Est 2.5% 97.5%    Est 2.5% 97.5%
Ns(dur, knots = c(1:5 * 10))1   0.96 0.80  1.15   1.00 0.82  1.22
Ns(dur, knots = c(1:5 * 10))2   1.23 1.06  1.43   1.20 1.02  1.40
Ns(dur, knots = c(1:5 * 10))3   1.21 1.01  1.45   1.21 1.00  1.45
Ns(dur, knots = c(1:5 * 10))4   1.17 1.03  1.33   1.16 1.02  1.33

> round(cbind(
+       ci.exp(mha, subset = c("age","sex")),
+       ci.exp(cmd, subset = c("age","sex"))), 2)

                                     Est 2.5% 97.5%    Est 2.5% 97.5%
Ns(age, knots = c(4:8 * 10))1       1.14 0.90  1.46   1.14 0.88  1.49
Ns(age, knots = c(4:8 * 10))2       0.95 0.75  1.20   0.94 0.73  1.21
Ns(age, knots = c(4:8 * 10))3       1.02 0.86  1.22   1.02 0.85  1.22
Ns(age, knots = c(4:8 * 10))4       1.32 1.05  1.66   1.31 1.03  1.66
sexW:Ns(age, knots = c(4:8 * 10))1  0.68 0.45  1.01   0.68 0.45  1.03
sexW:Ns(age, knots = c(4:8 * 10))2  1.43 1.01  2.04   1.40 0.96  2.05
sexW:Ns(age, knots = c(4:8 * 10))3  1.42 1.12  1.80   1.43 1.12  1.84
sexW:Ns(age, knots = c(4:8 * 10))4  1.42 1.05  1.94   1.45 1.07  1.97
sexW                                0.84 0.74  0.94   0.83 0.74  0.95
sexW:Ns(age, knots = c(4:8 * 10))1  0.68 0.45  1.01   0.68 0.45  1.03
sexW:Ns(age, knots = c(4:8 * 10))2  1.43 1.01  2.04   1.40 0.96  2.05
sexW:Ns(age, knots = c(4:8 * 10))3  1.42 1.12  1.80   1.43 1.12  1.84
sexW:Ns(age, knots = c(4:8 * 10))4  1.42 1.05  1.94   1.45 1.07  1.97

> # important to clean up so we call Epi::ci.exp by default
> rm(ci.exp)
```

CODE EXPLAINED: To get a nicer print we redefine `ci.exp` as a local copy that just has a slightly different name for the first column. At the end we remove the local copy, so that future calls of `ci.exp` will (again) be calls of `Epi::ci.exp`.

We see that we get almost identical parameter estimates, whether we use age (`age`) or duration (`dur`) as baseline time scale. The Cox-model does however not provide estimates of the effect of the baseline timescale, as we get from the Poisson model, so the last two comparisons are only with one of the Cox models.

5.2.0.2 Cox or Poisson model(s)?

So it seems that we could get the estimates of all time scales if we just fit two Cox models with different underlying time scales. But the prerequisite for this is that we from regression parameters in the Cox model can estimate the effect of the timescale(s) that is not the baseline—that is from the linear predictor. This will only be possible if we have fitted the Cox models to a split dataset (`Sx` in this case). But if we already have a finely split dataset we might as well fit the Poisson model in the first place and get estimates of the effects of both time scales in one go.

The real reason to set up the Cox models here is that there is a simple machinery to incorporate random effects in Cox models via the `coxme` package.

5.2.0.3 `coxme`: Cox model with mixed effects

The rate models we operated can be expanded with an extra random effect term in the linear predictor:

$$\log(\lambda_{pt}) = \eta_{pt} + A_p, \quad A_p \sim \mathcal{N}(0, \sigma^2)$$

where we assume the A_ps to be independent, so the σ is the between-persons standard deviation of the log-rates. Formally this is a 2-level model; first level is the specification of the random effects (the A_ps — marginal) and the second level is the specification of the rates *conditional* on the random effects.

We want to see how the effect of no. of HHs will change by introduction of a random person-effect, and currently the simplest way to do this is to use the `coxme` function from the package of the same name.

We can now fit the mixed effects Cox-models corresponding to the two Cox-models and inspect the random effects and the parameter estimates.

```
> rdur <- coxme(Surv(dur,
+                    dur + lex.dur,
+                    lex.Xst %in% levels(Sx)[2:11] &
+                    lex.Xst != lex.Cst)
+                ~ factor(lex.Cst) +
+                  Ns(tfHH, knots = c(0,1,3,6)) +
+                  sex * Ns(age, knots = c(4:8*10)) +
+                  (1 | lex.id),
+                data = subset(Sx, lex.Cst %in% levels(Sx)[1:10]))


> summary(rdur)
Cox mixed-effects model fit by maximum likelihood
  Data: subset(Sx, lex.Cst %in% levels(Sx)[1:10])
  events, n = 2700, 311808
  Iterations= 9 76
                    NULL Integrated   Fitted
Log-likelihood -19278.55  -17683.02 -17186.6

                   Chisq     df p     AIC     BIC
Integrated loglik 3191.06  22.00 0 3147.06 3017.24
 Penalized loglik 4183.90 506.27 0 3171.35  183.83

Model:  Surv(dur, dur + lex.dur, lex.Xst %in% levels(Sx)[2:11] & lex.Xst !=      lex.Cst) ~ factor(l
Fixed coefficients
                                          coef   exp(coef)   se(coef)      z      p
factor(lex.Cst)1.HH                 2.28887083  9.86379351 0.07815933  29.28 0.0000
factor(lex.Cst)2.HH                 2.64561203 14.09206722 0.08183166  32.33 0.0000
factor(lex.Cst)3.HH                 2.93592249 18.83887371 0.08799412  33.36 0.0000
factor(lex.Cst)4.HH                 3.15814402 23.52689003 0.09575695  32.98 0.0000
factor(lex.Cst)5.HH                 3.31363874 27.48495428 0.10499416  31.56 0.0000
factor(lex.Cst)6.HH                 3.39389894 29.78184382 0.11843776  28.66 0.0000
factor(lex.Cst)7.HH                 3.48552078 32.63942058 0.13774713  25.30 0.0000
factor(lex.Cst)8.HH                 3.48705669 32.68959046 0.15429226  22.60 0.0000
factor(lex.Cst)9.HH                 3.65616606 38.71263615 0.16379351  22.32 0.0000
Ns(tfHH, knots = c(0, 1, 3, 6))1   -0.88565274  0.41244487 0.07907967 -11.20 0.0000
Ns(tfHH, knots = c(0, 1, 3, 6))2   -2.33707000  0.09661029 0.13685038 -17.08 0.0000
Ns(tfHH, knots = c(0, 1, 3, 6))3   -0.83513836  0.43381445 0.06845402 -12.20 0.0000
sexW                               -0.18975302  0.82716340 0.06377963  -2.98 0.0029
Ns(age, knots = c(4:8 * 10))1       0.12858617  1.13721942 0.12858876   1.00 0.3200
Ns(age, knots = c(4:8 * 10))2      -0.05022957  0.95101108 0.12717992  -0.39 0.6900
Ns(age, knots = c(4:8 * 10))3       0.01660617  1.01674482 0.09677114   0.17 0.8600
Ns(age, knots = c(4:8 * 10))4       0.28447054  1.32905816 0.12725513   2.24 0.0250
sexW:Ns(age, knots = c(4:8 * 10))1 -0.38016556  0.68374820 0.21225148  -1.79 0.0730
sexW:Ns(age, knots = c(4:8 * 10))2  0.32702490  1.38683600 0.19005297   1.72 0.0850
sexW:Ns(age, knots = c(4:8 * 10))3  0.39799401  1.48883511 0.12962200   3.07 0.0021
sexW:Ns(age, knots = c(4:8 * 10))4  0.40297340  1.49626708 0.16912908   2.38 0.0170

Random effects
 Group  Variable  Std Dev   Variance
 lex.id Intercept 0.4454154 0.1983949


recur
```

```
> sqrt(VarCorr(rdur)$lex.id)
Intercept
0.4454154
> sd(ranef(rdur)$lex.id)
[1] 0.07265774
```

CODE EXPLAINED: The syntax for `coxme` is almost the same as for `coxph`, except that `coxme` allows random effects terms in the model formula, such as `(1 | lex.id)`. This means a random intercept for each person (`lex.id`).

We do not have a `coxme.Lexis` that does all the hard work of getting data from a `Lexis` object; so we must do the subsetting and event-definition by hand:

We are modeling the occurrence rates of next HH, so we restict data to be only those at risk of HH, that is subset to records with `lex.Cst` equal to levels `0.HH` through `9.HH` (levels 1 though 10).

And we only want events to be where `lex.Xst` is among the levels `1.HH` through `10+HH` (2 through 11), but since the dataset is split, we should only count records where we have a transition, that is where `lex.Xst` is different from `lex.Cst`.

We see that the estimated standard deviation of the random effects is 0.445— this is the standard deviation of the random effects that are assumed to be normal variates on the log-rate scale. We can interpret this as the between-person variation of frailties—a population characteristic specific for this population.

This implies that the difference between the random effect terms for two randomly chosen persons is normally distributed with standard deviation of $\sqrt{2} \times 0.445$; so the *absolute* difference follows a half-normal distribution. The median in a half-normal is the 75% quantile of the normal distribution, hence the median absolute difference between two randomly chosen persons is ([8]:

```
> qnorm(0.75) * sqrt(2) * sqrt(VarCorr(rdur)$lex.id)
Intercept
0.4248696
```

This means that the median rate-ratio (the one > 1) of HH-occurrence between two randomly chosen persons is:

```
> exp(qnorm(0.75) * sqrt(2) * sqrt(VarCorr(rdur)$lex.id))
Intercept
 1.529391
```

We might instead argue that the relevant quantity to base our reporting on was the `empirical` standard deviation of the posterior random effects:

```
> exp(qnorm(0.75) * sqrt(2) * sd(ranef(rdur)$lex.id))
[1] 1.071764
```

This would imply that the median absolute rate ratio (that is the one > 1) between two randomly chosen persons were only 1.07.

Either way, the between-person sd of the frailties is a population characteristic, so refers to the specific study population at hand; it is not a residual error we may expect to find in other study populations. Moreover, the calculations assume that the (prior) distribution of the log-frailties is normal which is certainly not the case for the posterior as we shall see.

5.2.0.4 The other Cox model

We can also fit the other Cox-model, with age as time scale

```
> rage <- coxme(Surv(age,
+                    age + lex.dur,
+                    lex.Xst %in% levels(Sx)[2:11] &
+                    lex.Xst != lex.Cst)
+               ~ factor(lex.Cst) +
+                 Ns(tfHH, knots = c(0,1,3,6)) +
+                 Ns(dur, knots = c(1:5 * 10)) +
+                 strata(sex) +
+                 (1 | lex.id),
+               data = subset(Sx, lex.Cst %in% levels(Sx)[1:10]))
```

```
> summary(rage)
Cox mixed-effects model fit by maximum likelihood
  Data: subset(Sx, lex.Cst %in% levels(Sx)[1:10])
  events, n = 2700, 311808
  Iterations= 9 76
                    NULL Integrated    Fitted
Log-likelihood -17108.89  -15551.62 -15073.11

                    Chisq    df p     AIC     BIC
Integrated loglik 3114.53  17.0 0 3080.53 2980.21
 Penalized loglik 4071.54 484.5 0 3102.55  243.54

Model:  Surv(age, age + lex.dur, lex.Xst %in% levels(Sx)[2:11] & lex.Xst !=      lex.Cst) ~ factor(l
Fixed coefficients
                                        coef   exp(coef)   se(coef)      z     p
factor(lex.Cst)1.HH                2.31661310 10.14126865 0.07836545  29.56 0.000
factor(lex.Cst)2.HH                2.67640375 14.53273589 0.08176804  32.73 0.000
factor(lex.Cst)3.HH                2.95847831 19.26862859 0.08815713  33.56 0.000
factor(lex.Cst)4.HH                3.17150031 23.84322978 0.09614066  32.99 0.000
factor(lex.Cst)5.HH                3.33747931 28.14808470 0.10561616  31.60 0.000
factor(lex.Cst)6.HH                3.40173383 30.01609783 0.11898842  28.59 0.000
factor(lex.Cst)7.HH                3.48643072 32.66913391 0.13954876  24.98 0.000
factor(lex.Cst)8.HH                3.50219111 33.18809120 0.15510610  22.58 0.000
factor(lex.Cst)9.HH                3.66393387 39.01451936 0.16494355  22.21 0.000
Ns(tfHH, knots = c(0, 1, 3, 6))1 -0.90041888  0.40639939 0.07903074 -11.39 0.000
Ns(tfHH, knots = c(0, 1, 3, 6))2 -2.39192310  0.09145364 0.13772059 -17.37 0.000
Ns(tfHH, knots = c(0, 1, 3, 6))3 -0.84148005  0.43107204 0.06848138 -12.29 0.000
Ns(dur, knots = c(1:5 * 10))1     0.01377948  1.01387486 0.09809459   0.14 0.890
Ns(dur, knots = c(1:5 * 10))2     0.19432355  1.21448917 0.08295019   2.34 0.019
Ns(dur, knots = c(1:5 * 10))3     0.20475620  1.22722583 0.09499085   2.16 0.031
Ns(dur, knots = c(1:5 * 10))4     0.16836010  1.18336266 0.06971433   2.41 0.016

Random effects
 Group  Variable  Std Dev   Variance
 lex.id Intercept 0.4376666 0.1915521
> sqrt(c(VarCorr(rage)$lex.id, VarCorr(rdur)$lex.id))
Intercept Intercept
0.4376666 0.4454154
> c(sd(ranef(rage)$lex.id), sd(ranef(rdur)$lex.id))
[1] 0.07012529 0.07265774
```

So the random variation between persons is very similar in the two models, as is the discrepancy between the estimated sd and the empirical sd of the individual posterior random effects. The discrepancy between the estimated sd and and the empirical sd of the posterior random effects is partly attributable to the fact that the sd is a really bad summary of a distribution that is so far from symmetry as this one is.

5.2.0.5 Individual random effects

We also inspect if the estimated random effects ("posterior" person effects) are similar between the two models

```
> table(names(rage$frail$lex.id) == names(rdur$frail$lex.id))
 TRUE
18215
> range(c(rdur$frail$lex.id, rage$frail$lex.id))
[1] -0.2336867  1.1589616
> ## plot(rage$frail$lex.id,
> ##      rdur$frail$lex.id,
> ##      pch = 16, cex = 0.4,
> ##      xlim = c(-0.25, 1.2), ylim = c(-0.25, 1.2),
> ##      xlab = "log-frailty using age as timescale",
```

recur

```
> ##      ylab = "log-frailty using duration as timescale")
> ## abline(0, 1, lty = 3)
> ## abline(h = 0, v = 0, lty = 3)
>
> plot(exp(rage$frail$lex.id),
+      exp(rdur$frail$lex.id),
+      pch = 16, cex = 0.5, log = "xy",
+      xlim = exp(c(-0.25, 1.2)), ylim = exp(c(-0.25, 1.2)),
+      xlab = "Frailty using age as timescale",
+      ylab = "Frailty using duration as timescale")
> abline(0, 1, lty = 3)
> abline(h = 1, v = 1, lty = 3)
```

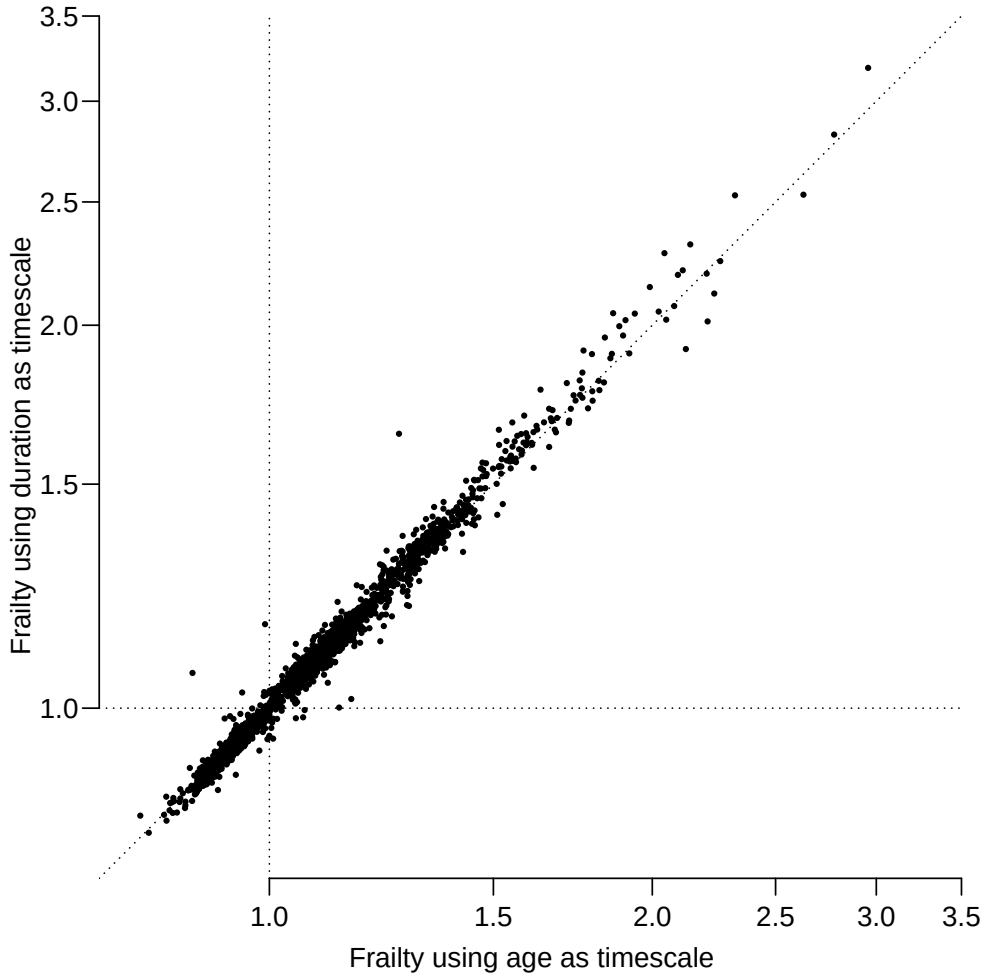


Figure 5.7: *Estimated frailties (on the log-rate-scale) for the Cox-models with duration respectively age as baseline timescale.*

../graph/recur-frail

From figure 5.7 we see that quite a number of the random effects are below 0, so we would like to see how the posterior frailties relate to the number of events for each person:

```
> tt <- with(subset(Sx, lex.Cst %in% levels(Sx)[1:10]),
+            table(lex.id, lex.Cst != lex.Xst & lex.Xst != "Dead"))[,2]
> table(tt)
tt
    0     1     2     3     4     5     6     7     8     9
16401  1289   342    92    47    21     9    10     2     2
```

```
> table(names(tt) == names(rdur$frail$lex.id))
 TRUE
18215
> exp(c(-0.25,1.2))
[1] 0.7788008 3.3201169
> axpt <- c(0.8,seq(1, 3, 0.5))
> frails <-
+ function(frails, mod)
+ {
+ plot(jitter(tt, a = 0.35),
+      frails,
+      pch = 16, cex = 0.5,
+      xlab = "Total no. HH events", xaxt ="n",
+      ylab = "Frailty (log-rate scale)", ylim = c(-0.25, 1.2))
+ abline(h = 0, lty = 3)
+ axis(side = 1, at = 0:9, labels = 0:9,
+      col = "white", col.lab = "black")
+ axis(side = 2, at = -2:12/10, labels = NA, tcl = -0.3)
+ axis(side=4, at = log(axpt), labels = axpt)
+ text(0, 1.2, mod, adj = c(0, 1))
+ }
> par(mfrow=c(1,2), mar=c(3,3,1,3), mgp=c(3,1,0)/1.6, las = 1, bty = "n")
> frails(rdur$frail$lex.id, "Duration baseline model")
> frails(rage$frail$lex.id, "Age baseline model")
```

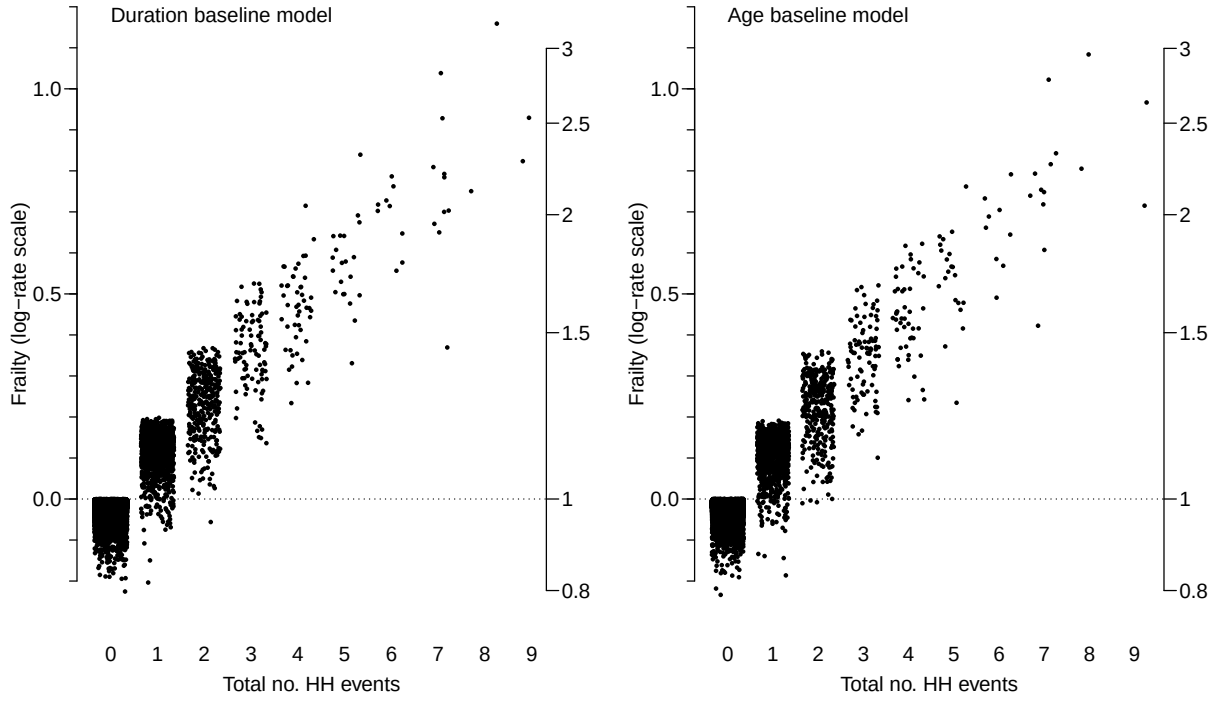


Figure 5.8: *Posterior frailties (log-rate scale) for the two Cox-models, plotted against the number of HH events for each person. Baseline timescales are* dur *(left) resp.* age *(right).* `../graph/recur-frail-ev`

We see that the two mixed effects Cox-models give essentially the same fit; they are both cousins of the Poisson-model, with the same time-scales as covariates, the difference is only which one we have profiled out when forming the Cox-model. So these are not models using different timescales; all three time scales (age, diabetes duration and time since last HH) are in all three models. The Cox models just hide one of the time scales.

`recur`

5.2.1 RR by no. of HH

From figure 5.8 we see that the average frailty increases by the total number of HH events seen, so by the frailty arguments above we will expect that the fixed effect estimates of the no. of HH events to be somewhat diluted by inclusion of person-specific random effects.

Since the two Cox-models with different baseline time scale give substantially the same fit, we only report these measures from the Cox-model model with diabetes duration as baseline time scale.

When we want the RRs relative to state `0.HH` the RR depends on the time since latest HH, here we choose 2 years as reference point. The facility to estimate the effect of the difference between two prediction frames in `ci.exp` is not implemented for `coxme` objects, so we will do the contrast matrix coding by hand; fortunately it only involves the `lex.Cst` and `tfHH` variables:

```
> t2 <- Ns(rep(2,9), knots = c(0, 1, 3, 6))
> t0 <- Ns(rep(0,9), knots = c(0, 1, 3, 6))
> Cd <- diag(9)
> C0 <- Cd * 0
> (CM <- cbind(Cd,t2) - cbind(C0,t0))
                                    1         2          3
 [1,] 1 0 0 0 0 0 0 0 0 0.1190646 0.5676795 -0.3422997
 [2,] 0 1 0 0 0 0 0 0 0 0.1190646 0.5676795 -0.3422997
 [3,] 0 0 1 0 0 0 0 0 0 0.1190646 0.5676795 -0.3422997
 [4,] 0 0 0 1 0 0 0 0 0 0.1190646 0.5676795 -0.3422997
 [5,] 0 0 0 0 1 0 0 0 0 0.1190646 0.5676795 -0.3422997
 [6,] 0 0 0 0 0 1 0 0 0 0.1190646 0.5676795 -0.3422997
 [7,] 0 0 0 0 0 0 1 0 0 0.1190646 0.5676795 -0.3422997
 [8,] 0 0 0 0 0 0 0 1 0 0.1190646 0.5676795 -0.3422997
 [9,] 0 0 0 0 0 0 0 0 1 0.1190646 0.5676795 -0.3422997
```

We can then compute and plot the RRs relative to `0.HH`

```
>  round(ci.exp(rdur, subset = c("Cst","tfHH")), 2)
                                  exp(Est.)  2.5% 97.5%
factor(lex.Cst)1.HH                    9.86  8.46 11.50
factor(lex.Cst)2.HH                   14.09 12.00 16.54
factor(lex.Cst)3.HH                   18.84 15.85 22.38
factor(lex.Cst)4.HH                   23.53 19.50 28.38
factor(lex.Cst)5.HH                   27.48 22.37 33.76
factor(lex.Cst)6.HH                   29.78 23.61 37.56
factor(lex.Cst)7.HH                   32.64 24.92 42.76
factor(lex.Cst)8.HH                   32.69 24.16 44.23
factor(lex.Cst)9.HH                   38.71 28.08 53.37
Ns(tfHH, knots = c(0, 1, 3, 6))1       0.41  0.35  0.48
Ns(tfHH, knots = c(0, 1, 3, 6))2       0.10  0.07  0.13
Ns(tfHH, knots = c(0, 1, 3, 6))3       0.43  0.38  0.50

> rrr <- ci.exp(rdur, subset = c("Cst","tfHH"), ctr.mat = CM)
>  round(ci.exp(mAc , subset = c("Cst","tfHH")), 2)

                                  exp(Est.)  2.5% 97.5%
lex.Cst1.HH                           12.75 10.99 14.79
lex.Cst2.HH                           18.36 15.71 21.45
lex.Cst3.HH                           24.65 20.87 29.13
lex.Cst4.HH                           31.06 25.94 37.20
lex.Cst5.HH                           36.02 29.60 43.84
lex.Cst6.HH                           39.63 31.79 49.39
lex.Cst7.HH                           42.34 32.76 54.74
lex.Cst8.HH                           45.65 34.32 60.73
lex.Cst9.HH                           51.63 38.24 69.70
Ns(tfHH, knots = c(0, 1, 3, 6))1       0.33  0.28  0.38
Ns(tfHH, knots = c(0, 1, 3, 6))2       0.06  0.05  0.08
Ns(tfHH, knots = c(0, 1, 3, 6))3       0.36  0.31  0.41

> ccc <- ci.exp(mAc , subset = c("Cst","tfHH"), ctr.mat = CM)
> rownames(rrr) <- paste0(1:9, ".HH")
> round(cbind(rrr, ccc), 3)
```

```
     exp(Est.)  2.5%  97.5% exp(Est.)  2.5%  97.5%
1.HH     3.135 2.735  3.593     3.331 2.913  3.809
2.HH     4.479 3.864  5.192     4.799 4.152  5.546
3.HH     5.987 5.085  7.050     6.444 5.497  7.554
4.HH     7.477 6.241  8.959     8.119 6.820  9.665
5.HH     8.735 7.128 10.704     9.414 7.749 11.437
6.HH     9.465 7.519 11.915    10.357 8.324 12.887
7.HH    10.373 7.889 13.640    11.067 8.538 14.346
8.HH    10.389 7.670 14.073    11.932 8.964 15.883
9.HH    12.303 8.906 16.997    13.493 9.982 18.240
```

```
> plotEst(rrr,
+         xlog = TRUE, lwd = 3, xlim = c(1,20),
+         xlab = "Hazard ratio of next HH (at 2 years) vs. no HH")
> axis(side = 1, at = 1:2)
> linesEst(ccc, lwd = 3, y = 9:1 - 0.2, col = "gray")
> abline(v=1)
```

5.2.1.1 RRs between successive states

It would also be of interest to see the ratio of hazards of next HH versus that in the previous state. This would be rates from state `n.HH` versus rates from the state `n-1.HH`, for `n` from 1 to 9. This amounts to a contrast matrix relating to the `lex.Cst` parameters

```
> M <- diag(9)[-9,]
> (RM <- (col(M) - row(M) == 1) - M)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]   -1    1    0    0    0    0    0    0    0
[2,]    0   -1    1    0    0    0    0    0    0
[3,]    0    0   -1    1    0    0    0    0    0
[4,]    0    0    0   -1    1    0    0    0    0
[5,]    0    0    0    0   -1    1    0    0    0
[6,]    0    0    0    0    0   -1    1    0    0
[7,]    0    0    0    0    0    0   -1    1    0
[8,]    0    0    0    0    0    0    0   -1    1
```

While this is constant over time for `n` > 1 by the very structure of the model (no interaction between `lex.Cst` and other variables), the RR between between `1.HH` and `0.HH` depends on time since 1st HH, because `tfHH` is 0 for any record with `lex.Cst` equal to `0.HH`, so we omit this:

```
> ci.exp(rdur, subset = "Cst")
                     exp(Est.)      2.5%    97.5%
factor(lex.Cst)1.HH  9.863794  8.462811 11.49670
factor(lex.Cst)2.HH 14.092067 12.003820 16.54360
factor(lex.Cst)3.HH 18.838874 15.854561 22.38493
factor(lex.Cst)4.HH 23.526890 19.500963 28.38396
factor(lex.Cst)5.HH 27.484954 22.372980 33.76496
factor(lex.Cst)6.HH 29.781844 23.612240 37.56349
factor(lex.Cst)7.HH 32.639421 24.916778 42.75560
factor(lex.Cst)8.HH 32.689590 24.158817 44.23268
factor(lex.Cst)9.HH 38.712636 28.082221 53.36715
```

```
> ci.exp(cmd , subset = "Cst")
                     exp(Est.)     2.5%    97.5%
factor(lex.Cst)1.HH  12.68617 10.81829 14.87656
factor(lex.Cst)2.HH  18.27919 15.46970 21.59893
factor(lex.Cst)3.HH  24.51320 20.68414 29.05109
factor(lex.Cst)4.HH  30.89188 25.61215 37.25998
factor(lex.Cst)5.HH  36.17100 29.61616 44.17662
factor(lex.Cst)6.HH  39.67913 31.52301 49.94554
factor(lex.Cst)7.HH  43.78903 33.76462 56.78961
factor(lex.Cst)8.HH  45.52099 33.97474 60.99122
factor(lex.Cst)9.HH  52.96816 39.17351 71.62050
```

```
> ci.exp(rdur, subset = "Cst", ctr.mat = RM)
```

recur

```
     exp(Est.)      2.5%    97.5%
[1,]  1.428666 1.2471014 1.636665
[2,]  1.336842 1.1343411 1.575494
[3,]  1.248848 1.0291594 1.515432
[4,]  1.168236 0.9323518 1.463798
[5,]  1.083569 0.8325148 1.410331
[6,]  1.095950 0.7981266 1.504908
[7,]  1.001537 0.6922460 1.449018
[8,]  1.184250 0.7896019 1.776145
```

```
> CCC <- ci.exp(cmd , subset = "Cst", ctr.mat = RM)
> RRR <- ci.exp(rdur, subset = "Cst", ctr.mat = RM)
> rownames(RRR) <- paste0(2:9," vs. ", 1:8)
> round(cbind(RRR, CCC), 3)

        exp(Est.)  2.5% 97.5% exp(Est.)  2.5% 97.5%
2 vs. 1     1.429 1.247 1.637     1.441 1.260 1.647
3 vs. 2     1.337 1.134 1.575     1.341 1.143 1.574
4 vs. 3     1.249 1.029 1.515     1.260 1.048 1.515
5 vs. 4     1.168 0.932 1.464     1.171 0.946 1.449
6 vs. 5     1.084 0.833 1.410     1.097 0.855 1.407
7 vs. 6     1.096 0.798 1.505     1.104 0.826 1.475
8 vs. 7     1.002 0.692 1.449     1.040 0.737 1.467
9 vs. 8     1.184 0.790 1.776     1.164 0.778 1.739
```

For comparison we re-plot the RRs versus `0.HH` to get a complete overview:

```
> par(mfrow=c(1,2))
> #
> par(xaxt = "n")
> plotEst(rrr,
+         xlog = TRUE, lwd = 3, xlim = c(0.5,20),
+         xlab = paste("Hazard ratio of next HH (at 2 years):",
+                      "current vs. no HH (0.HH)"))
> par(xaxt = "s")
> axis(side = 1, at = c(0.5,1:10,15), labels = NA)
> axis(side = 1, at = c(0.5,1,2,5,10,20),
+            labels = c("0.5",paste(c(1,2,5,10,20))))
> linesEst(ccc, lwd = 3, y = 9:1 - 0.2, col = "gray")
> abline(v = 1, lty = 3)
> #
> par(xaxt = "n")
> plotEst(rbind(NA,RRR),
+         xlog = TRUE, lwd = 3, xlim = c(0.5,20),
+         xlab = "Hazard ratio of next HH: current vs. previous HH")
> par(xaxt = "s")
> axis(side = 1, at = c(0.5,1:10,15), labels = NA)
> axis(side = 1, at = c(0.5,1,2,5,10,20),
+            labels = c("0.5",paste(c(1,2,5,10,20))))
> linesEst(rbind(NA,CCC), lwd = 3, y = 9:1 - 0.2, col = "gray")
> abline(v = 1, lty = 3)
```

From figure **??** we see there is only a very slight dilution of the effect of no. HHs by introducing a frailty term, but also that regardless of model, there is an increase in rates of next HH up to about 5 HHs, after which the rate of next HH does not increase by further HH occurrences.

So even if there is some variation between individuals in their frailties it does not explain much of the variation between HH states. The `coxme` model uses an assumption of normally distributed frailties (on the log-rate scale) which is not an obvious choice when we have a large fraction of the population who see no HH, and therefore might be considered immune (and so should have a frailty of 1).

before

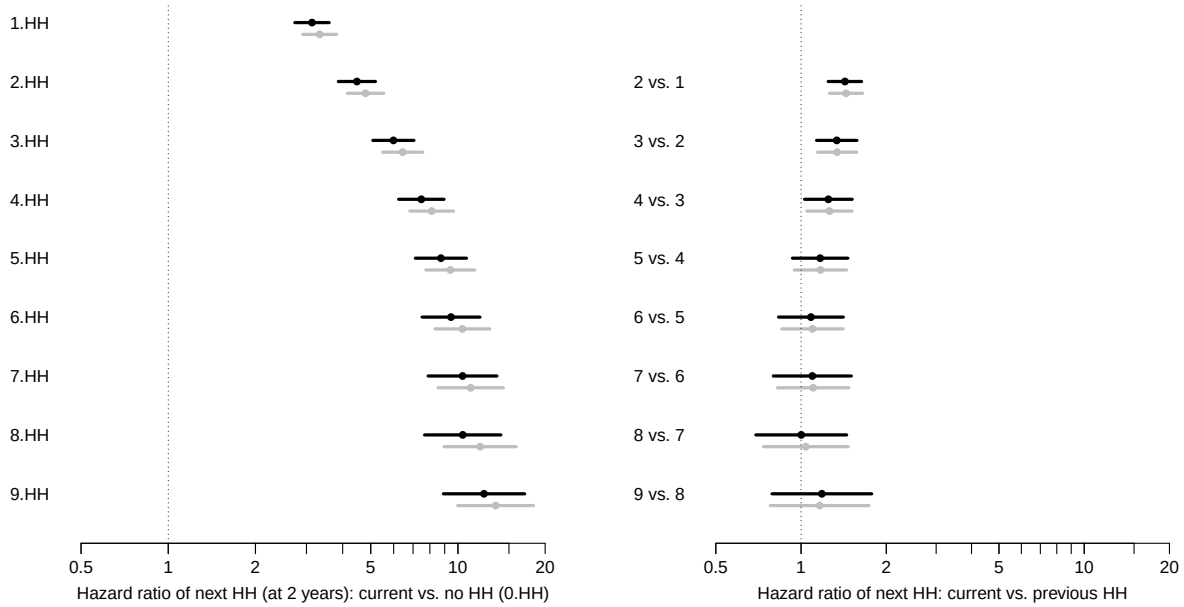


Figure 5.9: *Hazard ratios of next HH event relative to the rate of the first event, respectively previous event. Left panels: The hazard ratio of next HH event relative to the rate of the first event, computed at 2 years after last HH event (*`tfHH` *= 2). Right panel: Ratio of rate of next HH versus rate of previous rate.*
The black estimates are from the mixed-effects Cox model with random person-effect, while the gray estimates are from the usual Cox model; both with diabetes duration as baseline timescale, and separate age-effects for the two sexes. `../graph/recur-cmp-dur`

5.3 Unknown history of events before entry

In some instances the number of events seen *before* entry to the study is not known, so a proper analysis is not possible. In most practical circumstances, all event dates will not be known; only a time-limited set of events will be known.

It is presumably quite common that only events `after` entry instead, so there will be a larger variation in rates `between` persons, but the frailty/selection effects *within* persons will have similar characteristics. In particular, the state without any *recorded* events will be contaminated with follow-up time after event(s).

As an example we look at the example pretending that HH events before entry were not known:

```
> summary(Lx, t = T)
Transitions:
     To
From    0.HH Dead  Records:  Events: Risk time:  Persons:
  0.HH 17640  673     18313      673   73223.02     18313

Timescales:
per age dur
 ""  ""  ""
```

```
> summary(hh)
       id             doh             nh
 Min.   :   11   Min.   :1985   Min.   : 1.000
 1st Qu.: 5580   1st Qu.:2000   1st Qu.: 1.000
 Median :11548   Median :2004   Median : 2.000
 Mean   :11492   Mean   :2004   Mean   : 3.535
 3rd Qu.:17192   3rd Qu.:2009   3rd Qu.: 4.000
 Max.   :23268   Max.   :2014   Max.   :54.000
```

In order to mimic the absence of events prior to study entry we extract the entry times of persons from `Lx`

```
> hhx <- left_join(rename(hh, lex.id = id),
+                  subset(Lx, select = c(lex.id, per)))
> hhx <- data.frame( subset(hhx, per < doh)
+                 %>% group_by(lex.id)
+                 %>% mutate(nh = row_number())
+                 %>% select(c(lex.id, doh, nh))
+                   )
> str(hh)
'data.frame':        14270 obs. of  3 variables:
 $ id : int  11 11 11 11 11 11 11 16 16 16 ...
 $ doh: num  2002 2003 2004 2005 2011 ...
 $ nh : int  1 2 3 4 5 6 7 1 2 3 ...
> str(hhx)
'data.frame':        4142 obs. of  3 variables:
 $ lex.id: int  11 11 11 19 27 27 27 27 27 27 ...
 $ doh   : num  2011 2013 2014 2014 2009 ...
 $ nh    : int  1 2 3 1 1 2 3 4 5 6 ...
```

So we see there is a substantial number of event occurring before entry; the number of events we count is not quite the same because the limit of 10 now refers to the number of events *after* entry and not the total number of events recorded.

```
> cutHHx <- mutate(subset(hhx, nh <= 10),
+                          cut = doh,
+                    new.state = ifelse(nh < 10,
+                                       paste0(nh, ".HH"),
+                                       "10+HH")) %>%
+           select(lex.id, cut, new.state)
> system.time(
+ Rxx <- rcutLexis(Lx,
+                  cut = cutHHx,
+           timescale = "per"))
   user  system elapsed
   6.05    0.17    6.22
> summary(Rx)
Transitions:
     To
From     0.HH 1.HH 2.HH 3.HH 4.HH 5.HH 6.HH 7.HH 8.HH 9.HH 10+HH Dead  Records:  Events:
  0.HH  12434  887    0    0    0    0    0    0    0    0     0  335     13656     1222
  1.HH      0 2787  568    0    0    0    0    0    0    0     0  141      3496      709
  2.HH      0    0 1112  373    0    0    0    0    0    0     0   62      1547      435
  3.HH      0    0    0  498  260    0    0    0    0    0     0   47       805      307
  4.HH      0    0    0    0  275  193    0    0    0    0     0   30       498      223
  5.HH      0    0    0    0    0  168  144    0    0    0     0   11       323      155
  6.HH      0    0    0    0    0    0  101  105    0    0     0    9       215      114
  7.HH      0    0    0    0    0    0    0   68   70    0     0   13       151       83
  8.HH      0    0    0    0    0    0    0    0   47   55     0    4       106       59
  9.HH      0    0    0    0    0    0    0    0    0   23    49    5        77       54
  10+HH     0    0    0    0    0    0    0    0    0    0   127   16       143       16
  Sum   12434 3674 1680  871  535  361  245  173  117   78   176  673     21017     3377

Transitions:
     To
From    Risk time:  Persons:
  0.HH    51936.45     13656
  1.HH    11750.50      3496
  2.HH     4425.94      1547
  3.HH     2010.91       805
  4.HH     1110.91       498
  5.HH      628.43       323
  6.HH      383.83       215
  7.HH      236.76       151
  8.HH      168.51       106
  9.HH      116.50        77
  10+HH     454.30       143
  Sum     73223.02     18313
```

```
> summary(Rxx)
Transitions:
     To
From     0.HH 1.HH 2.HH 3.HH 4.HH 5.HH 6.HH 7.HH 8.HH 9.HH 10+HH Dead  Records:  Events:
  0.HH  15894 1869    0    0    0    0    0    0    0    0      0  550     18313     2419
  1.HH      0 1219  574    0    0    0    0    0    0    0      0   76      1869      650
  2.HH      0    0  328  218    0    0    0    0    0    0      0   28       574      246
  3.HH      0    0    0   94  117    0    0    0    0    0      0    7       218      124
  4.HH      0    0    0    0   48   62    0    0    0    0      0    7       117       69
  5.HH      0    0    0    0    0   23   37    0    0    0      0    2        62       39
  6.HH      0    0    0    0    0    0   14   22    0    0      0    1        37       23
  7.HH      0    0    0    0    0    0    0    9   11    0      0    2        22       13
  8.HH      0    0    0    0    0    0    0    0    1   10      0    0        11       10
  9.HH      0    0    0    0    0    0    0    0    0    4      6    0        10        6
  10+HH     0    0    0    0    0    0    0    0    0    0      6    0         6        0
  Sum   15894 3088  902  312  165   85   51   31   12   14     12  673     21239     3599

Transitions:
     To
From    Risk time:  Persons:
  0.HH    68335.16     18313
  1.HH     3628.12      1869
  2.HH      813.85       574
  3.HH      238.43       218
  4.HH       96.20       117
  5.HH       47.39        62
  6.HH       31.04        37
  7.HH       15.23        22
  8.HH        4.68        11
  9.HH        3.40        10
  10+HH       9.51         6
  Sum     73223.02     18313
```

We see that the amount of risk time and number of deaths is the same; we have more events in total, but the events are more concentrated in the early states.

5.3.1 Comparison with the full data

We can compare the results of analyzing this dataset withe those of the "real" one above, we need to define `tfHH` as a time scale, where we only are concerned about time since events that occur after entry. It is basically the same code as above, but with the new definition of the recurrence numbers and -dates:

```
> # join with the cut dates
> Rxx <- left_join(Rxx, rename(cutHHx, doHH = cut),
+                  by = c("lex.id"  = "lex.id",
+                         "lex.Cst" = "new.state"))
> # make lex.Cst a factor, and reorder levels
> Rxx <- factorize(Rxx)
> Rxx <- Relevel(Rxx, c(paste0(0:9,".HH"),"10+HH","Dead"))
> # define new timescale
> Rxx$tfHH <- with(Rxx, ifelse(lex.Cst == "0.HH", NA, per - doHH))
> attr(Rxx, "time.scales") <- c(attr(Rxx, "time.scales"), "tfHH")
> attr(Rxx, "time.since" ) <- c(attr(Rxx, "time.since" ), "")
> attr(Rxx, "breaks") <- c(attr(Rxx, "breaks"), list(tfHH = NULL))
> # split follow-up and convert NAs to 0
> Sxx <- splitMulti(Rxx, age = seq(0, 90, 1/4))
> Sxx <- tsNA20(Sxx, all = TRUE)
> # then fit the model
> system.time(
+ mAcx <- glm.Lexis(Sxx,
+                   ~ lex.Cst + Ns(tfHH, knots = c(0,1,3,6)) +
+                               Ns(dur, knots = c(1:5 * 10)) +
+                         sex * Ns(age, knots = c(4:8 * 10)),
+                   to = levels(Sxx)[2:11],
+                scale = 100))
```

recur

```
stats::glm Poisson analysis of Lexis object Sxx with log link:
Rates for transitions:
0.HH->1.HH
1.HH->2.HH
2.HH->3.HH
3.HH->4.HH
4.HH->5.HH
5.HH->6.HH
6.HH->7.HH
7.HH->8.HH
8.HH->9.HH
9.HH->10+HH
, lex.dur (person-time) scaled by 100
   user  system elapsed
   6.72    0.80    7.54
```

We now have a fitted model of exactly the same structure and naming of covariates as the previous one, but fitted to a `Lexis` object with a different definition of the recurrent event. The events will be the same but with a differing numbering, all with an identical or smaller number. Since we in both cases use 10 as the cut-off number, the data frame only counting after study entry will have some more events in total.

Therefore we can plot the parameter estimates using the same function as before:

```
> # function to plot age effect
> plage <- function(mAc) {
+ matshade(pra$age, aa <<- ci.pred(mAc, pra),
+           plot = TRUE, col = "white",
+           log = "y", lwd = 3, ylim = c(1,350),
+           xlab = "Attained age (years)",
+           ylab = "Rate of first HH per 100 PY")
+ abline(h = aa[21,1], lty = "42", lwd = 2, col = "gray")
+ matshade(pra$age, aa, lwd = 3, col = "blue")
+ matshade(pra$age, ci.pred(mAc, mutate(pra, sex = "W")),
+                       lwd = 3, col = "red")
+ points(50, aa[21,1], pch = 16, cex = 1.4, col = "white")
+ points(50, aa[21,1], pch =  1, cex = 1.4, lwd = 2, col = "blue")
+                      }
> # function to plot duration effects
> pldur <- function(mAc) {
+ for(x in 9:1)
+    {
+ matshade(prx(x, "M")$tfHH,
+           dd <<- ci.pred(mAc, prx(x, "M")),
+           plot = (x == 9),
+           log = "y", lwd = 3, ylim = c(1,350), col = gray(x/12),
+           xlab = "Time since last HH (years)",
+           ylab = "Rate of next HH (men, aged 50) per 100 PY")
+ text(0, dd[1,1], paste(x), adj = 1.6)
+    }
+ text(0, 350, "HH number", adj = 0.2)
+ abline(h = aa[21,1], lty = "42", lwd = 2, col = "gray")
+                      }
```

We can then use the functions to plot the two different sets of estimates together.

```
> par(mfcol = c(2, 2), oma = c(0,0,0,2.5),
+     mgp = c(3,1,0) / 1.6, las = 1, bty = "n")
> plage(mAc)
> pldur(mAc)
> plage(mAcx)
> pldur(mAcx)
```

From figure 5.10 we see that the age-effects are aprroximately the same, but the effects of HH number are somewhat larger in the follow-up only data set. This is because the events in this are assigned a smaller number, and since the number of event increases the rate of next event, the absoute effect of event number must be larger.

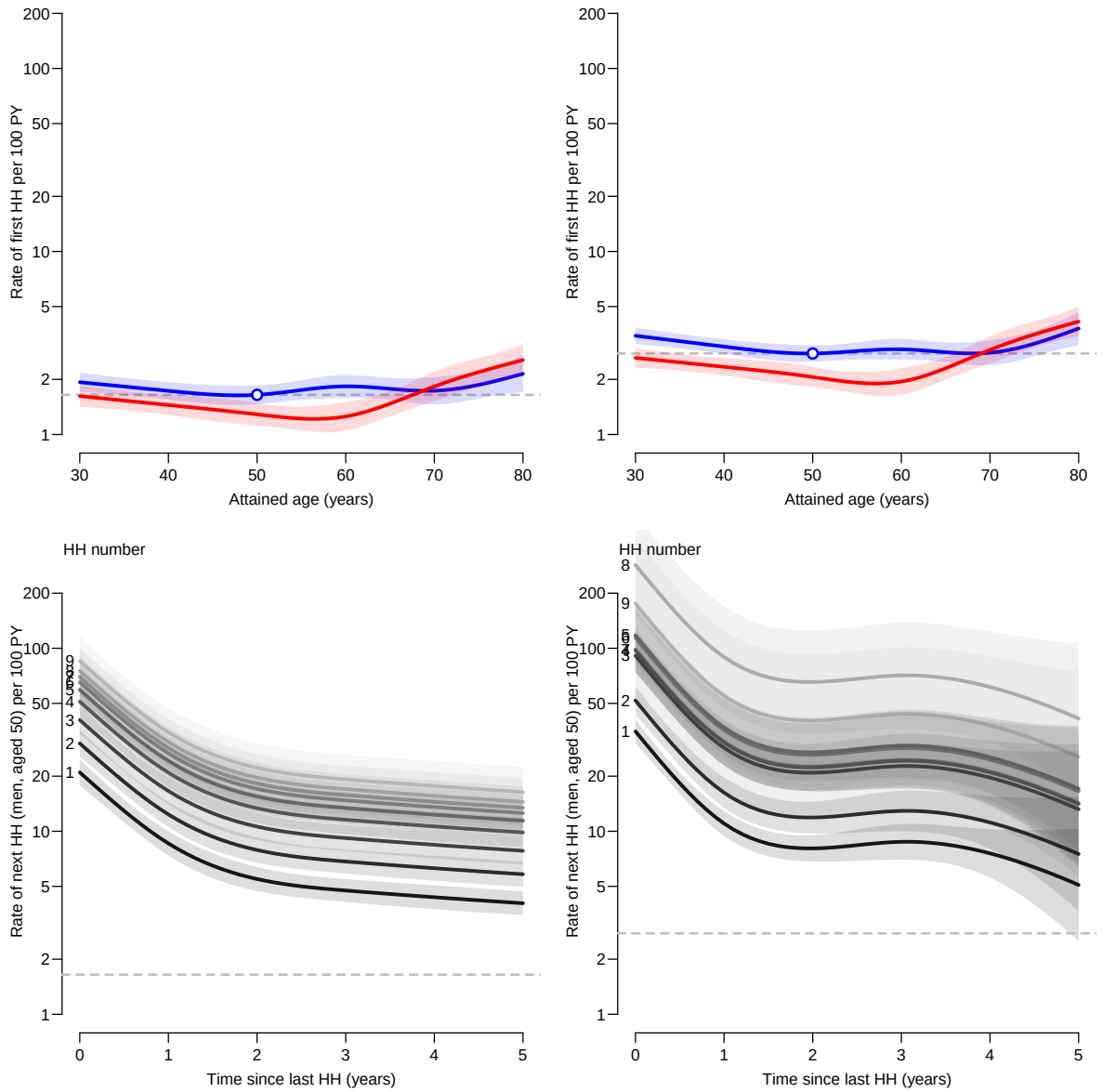


Figure 5.10: *Estimates from two different definitions of recurrence number. The left panels are using all available data, including information of HH events prior to start of follow-up, the right panels are counting only HH events ocurring after strat of follow-up.* `../graph/recur-ageHHaddx`

This is a demonstration that the way event counts are defined from original data may have a profound influence on the estiamted recurrence rates. Unless there is certainty that no events have occurred before start of follow-up, results may be strongly dependent on how recurrent events are numbered.

5.4 Summary of recurrent events

When the same event can occur multiple times for the same person, this should be addressed by defining different states corresponding to the number of recurrent events encountered. The current number of events should be included as a covariate in order to accomodate the effects of increasing number of events.

`recur`

Since the same person may see several events, it is possible to introduce a dependency structure in the models by adding random effects. Here we have shown how this is accomodated in a Cox model using `coxme`, and how the random effects model dilutes the effects of the event number.

In some instances of recurrent events there may be a (large) fraction of the population that is effectively immune to having an event. This implies that if a random effects is introduced, then it shoudl have an allowance for an atom in 0 on the rate scale ($-\infty$ on the log-rate scale), ideally some compound model with a binomial model for the probability of being in the immune class.

In probability terms a model is formulated in terms of susceptibility:

$$\text{P}\{\text{susceptible}\} \times \text{P}\{\text{next event} \mid \text{susceptible}\}$$

where the first terms is a binomial model and the latter a rate model possibly with some random effect allowance.

. . . now input from `comprisk`

Chapter 6

Competing risks in practice

Suppose that persons in a given state, 'alive', say, are subject to a number of different causes of deaths, 'cause1', 'cause2' etc. Causes of death are required to be exhaustive and mutually exclusive. Exhaustive means that any person will eventually died from one if the causes; exclusive means that you can only die from from one cause. In situations where the causes are not causes of death but other events, it is implicit that we only consider the first occurrence of an event from the state 'alive', and ignore what occurs after.

So a competing risk set-up is a special multistate model; one with a singe initial state, and a finite number of absorbing states that can be reached directly from the initial state. By that token the three levels of quantities of interest will also be of relevance here (depending, of course, on the subject matter): 1) cause-specific rates 2) cumulative risks for each cause and 3) expected life time and life time lost to each cause.

The likelihood for observations from a competing risk scenario is a function of the cause-specific transition rates, and is *product* of the likelihoods that would emerge if we considered each cause the only one. Thus analysis is in principle straight forward; just estimate a model for each of the cause-specific rates. These will together form a complete model for the competing risks problem. Note that it is not because the types of event (causes of death) are independent, but because the log-likelihood is a sum with each term depending only on one of the cause-specific rates, see (3.4), p. 21.

If the cause-specific rates are all we want to assess then we will be done.

But often we would like also to have estimates of the cumulative risks, that is the probability of dying from a specific cause before a given time as function of time. Each of these cumulative risks are functions of *all* cause-specific rates. Specifically, if the cause-specific rates are $\lambda_j(t), j = 1, 2, \ldots$, then the cumulative risk of death from cause c before time y, $R_c(t)$ is (where $S(u)$ is the survival function):

$$\begin{aligned} R_c(t) &= \int_0^t \lambda_c(u) S(u)\,\mathrm{d}u \\ &= \int_0^t \lambda_c(u) \exp\Big(-\int_0^u \sum_j \lambda_j(s)\,\mathrm{d}s\Big)\,\mathrm{d}u \\ &= \int_0^t \lambda_c(u) \exp\Big(-\sum_j \int_0^u \lambda_j(s)\,\mathrm{d}s\Big)\,\mathrm{d}u \\ &= \int_0^t \lambda_c(u) \exp\Big(-\sum_j \Lambda_j(s)\Big)\,\mathrm{d}u \end{aligned}$$

$\Lambda_j(s)$ is the cumulative rate (integrated intensity) for the j^{th} cause of death.

Even if we from the modeling of the λs have standard errors of $\log(\lambda_c)$ the standard errors of R_cs will be analytically intractable from these.

The only viable way to get confidence intervals for the cumulative risks, R_c, is by simulation. This is implemented through calculation of simulated rates $\lambda(t)$ by sampling from the posterior distribution of the parameters in the models for $\log(\lambda(s))$, and computing the integrals numerically for each simulated sample.

The simulation approach also allows calculation of confidence intervals for sums of the cumulative risks, $R_1(t) + R_2(t)$ for example, which will be needed if we want to show stacked cumulative risks.

Finally, simulation will also allow calculation of standard errors of sojourn times in each of the states

'alive' and 'cause1', 'cause2', etc. While the latter two may not be of direct interest, then *differences* between such sojourn times between different groups can be interpreted as years of life lost to each cause between groups.

6.1 Example data

As an illustrative data example we use the (fake) diabetes register data; we set up the Lexis object, cut the follow-up time at dates of OAD, resp Ins. There is a considerable fraction of persons who are put on medication immediately at diagnosis a who claim their medication during the first one or two months, so we disregard the first two months of follow up to get a more realistic picture of beginning of medication among persons initially without medication. This is done by moving the date of DM diagnosis 2 months, which also has the effect of increasing the number of persons beginning their follow-up on OAD or insulin.

```
> library(Epi)
> library(popEpi)
> data(DMlate)
> DMlate$dodm <- DMlate$dodm + 2/12
> 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  284  rows with duration of follow up < tol
```

The 284 rows dropped are persons in the original data that end follow-up less than 2 months after date of diabetes (`dodm`).

```
> summary(Ldm)
Transitions:
     To
From   DM Dead  Records:  Events: Risk time:  Persons:
  DM 7383 2333      9716     2333   52633.05      9716
```

We can cut the follow-up at the dates of beginning OAD (`dooad` and insulin (`doins`:

```
> set.seed(1952)
> Mdm <- mcutLexis(Ldm,
+                  wh = c('dooad', 'doins'),
+           new.states = c('OAD', 'Ins'),
+            precursor = 'DM',
+           seq.states = FALSE,
+                 ties = TRUE)
NOTE: 14 records with tied events times resolved (adding 0.01 random uniform),
      so results are only reproducible if the random number seed was set.
> summary(Mdm)
Transitions:
     To
From        DM Dead  OAD Ins Ins+OAD  Records:  Events: Risk time:  Persons:
  DM      2775  925 2044 234       0      5978     3203   21817.05      5978
  OAD        0  968 3271   0     952      5191     1920   22498.35      5191
  Ins        0  141    0 461     126       728      267    3821.08       728
  Ins+OAD    0  299    0   0     876      1175      299    4496.57      1175
  Sum     2775 2333 5315 695    1954     13072     5689   52633.05      9716
```

CODE EXPLAINED: `mcutLexis` cuts the follow-up at multiple event times keeping track of the persons' history, but requires that each type of event occurs at most once, see 4.4.2 on pages 36 ff.

Also requires that no two events occur at the same time; if that is so arbitrary (small) random quantities are added to the times in order to break the tie.

In order to model the cause-specific rates by duration we initially split the the follow-up by duration of diabetes in intervals of 1/48 year (about 1 week) for the two first months and after that in intervals of 1 month (= 1/12 year), creating a `Lexis` object for a competing risks situation with three possible event types:

```
> Sdm <- splitMulti(factorize(subset(Mdm, lex.Cst == "DM")),
+                   tfd = c(seq(   0, 2/12, 1/48),
+                           seq(3/12,  100, 1/12)))
> summary(Sdm)
Transitions:
     To
From     DM Dead  OAD Ins  Records:  Events: Risk time:  Persons:
  DM 295344  925 2044 234    298547     3203   21817.05      5978
```

CODE EXPLAINED: We subset the `Mdm` to the follow-up in state `DM` in order to have a proper competing risks set-up. We need to use the `factorize` in order to restrict the levels of `lex.Cst` and `lex.Xst` to those actually present in data set. The `summary(Sdm)` shows that this cuts the number of persons and risk time substantially—compare to `summary(Mdm)`, or see figure **??**,

In order to be able to model the effect of duration at early durations we split the early follow-up finely along the diabetes duration (`tfd`, time from diagnosis).

We can illustrate the follow-up in the full data set and in the data set restricted to those at risk in the `DM` state:

```
> par(mfrow = c(1,2))
> boxes(Mdm,
+       boxpos = list(x = c(18, 50, 18, 82, 82),
+                     y = c(85, 50, 15, 85, 15)),
+       scale.R = 100,
+       show.BE = TRUE)
> boxes(Sdm,
+       boxpos = list(x = c(18, 50, 18, 82),
+                     y = c(85, 50, 15, 85)),
+       scale.R = 100,
+       show.BE = 'nz' )
```

The following is concerned with analysis of the follow-up shown in the right panel of figure 6.1.

6.2 Models for rates

Now that we have set up a dataset with three competing events, we can model the cause-specific rates separately by time from diagnosis as the only underlying time scale. Note that we only need to specify the `to=` argument because there is only one possible `from` for each `to` (incidentally the same for all `to` states), namely `DM`.

We can use either `gam` models with penalized smoothing, where we only set an initial upper bound for the degrees of effective degrees of freedom used:

```
> mD <- gam.Lexis(Sdm, ~ s(tfd, k = 6), to = 'Dead')
> mO <- gam.Lexis(Sdm, ~ s(tfd, k = 6), to = 'OAD' )
> mI <- gam.Lexis(Sdm, ~ s(tfd, k = 6), to = 'Ins' )
```

It turns out that this automated approach also requires some grooming in order to render credible estimates, so in this context we refrain from that. The code is given above, but not run; it is left as an exercise to the reader to try out this set of model, varying the `k` to see what happens.

A bit more brutal approach is to use natural spline (restricted cubic splines) with a predefined set of knots — here we explicitly use 7 knots corresponding to 7 degrees for freedom for the `tfd` term—that is each model uses a total of 7 parameters

```
> kn <- c(0,0.2,0.5,1,3,7,10)
> mD <- glm.Lexis(Sdm, ~ Ns(tfd, knots = kn), to = 'Dead')
```

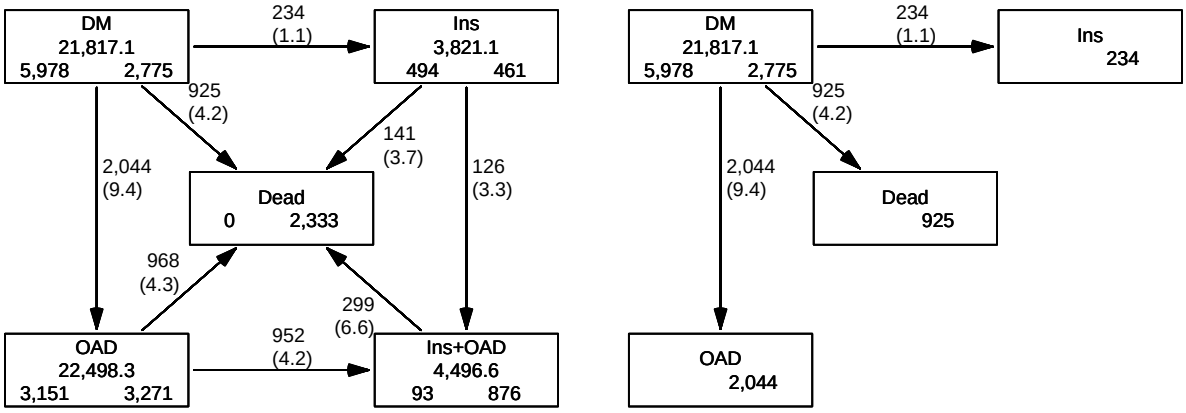


Figure 6.1: *The transitions in the multistate model with and without follow-up extended after beginning of first drug exposure. Follow-up begins two months after the recorded date of diagnosis of diabetes. Rates in brackets are per 100 PY. In the rightmost panel only persons in the* `DM` *state contribute person-years—all follow-up after exit from the* `DM` *state is discarded; a characteristic of a competing risks situation.* `../graph/crisk-boxes54`

```
stats::glm Poisson analysis of Lexis object Sdm with log link:
Rates for the transition:
DM->Dead
> mD <- glm.Lexis(Sdm, ~ Ns(tfd, knots = kn), to = 'OAD' )
stats::glm Poisson analysis of Lexis object Sdm with log link:
Rates for the transition:
DM->OAD
> mI <- glm.Lexis(Sdm, ~ Ns(tfd, knots = kn), to = 'Ins' )
stats::glm Poisson analysis of Lexis object Sdm with log link:
Rates for the transition:
DM->Ins
```

With these models fitted we can compute the rates, cumulative rates and risks and sojourn times in different states using the usual formulae—basically integrating the rates. First we compute the rates in intervals of length 1/100 years. Note that these models only have time since diagnosis as covariates, so in that sense they are similar to Nelson-Aalen estimates, albeit in a biologically more meaningful guise, and they provide estimates on the rate scale, not on the scale of the cumulative rates.

The points where we compute the predicted rates are midpoints of intervals of length 1/100 year. These points are unrelated to the follow-up intervals in which we split the data for analysis—those were 1 month intervals (1/12 year), here we use 1/100 year (about 3.7 days) throughout:

```
> int <- 1/100
> nd <- data.frame(tfd = seq(0, 14, int))
> rownames(nd) <- nd$tfd
> str(nd)
'data.frame':        1401 obs. of  1 variable:
 $ tfd: num  0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 ...
```

With this we can show the rates as a function of the time since diagnosis:

```
> par(mfrow = c(1,2))
> matshade(nd$tfd, cbind(ci.pred(mD, nd),
+                        ci.pred(mI, nd),
+                        ci.pred(mO, nd))*1000,
```

```
+          ylim = c(1,300), yaxt = "n", xlim = c(0,10),
+          ylab = "Rates per 1000 PY",
+          xlab = "Time since DM diagnosis (years)",
+          col = c("black","red","blue"), log = "y", lwd = 3,
+          plot = TRUE)
> axis(side = 2, at = ll<-outer(c(1,2,5),-2:3,function(x,y) x*10^y),
+               labels = formatC(ll,digits = 4), las = 1)
> axis(side = 2, at = ll<-outer(c(1.5,2:9),-2:3,function(x,y) x*10^y),
+               labels = NA, tcl = -0.3)
> text(0, 2.5*0.7^c(1,2,0),
+      c("Dead","Ins","OAD"),
+      col = c("black","red","blue"), adj = 0)
> matshade(nd$tfd, cbind(ci.pred(mD, nd),
+                        ci.pred(mI, nd),
+                        ci.pred(m0, nd))*1000,
+          ylim = c(0,300), yaxs = "i",  xlim = c(0,10),
+          ylab = "Rates per 1000 PY",
+          xlab = "Time since DM diagnosis (years)",
+          col = c("black","red","blue"), lwd = 3, plot = TRUE)
> axis(side = 2, at = 1:9 * 50,
+               labels = NA, tcl = -0.3)
```

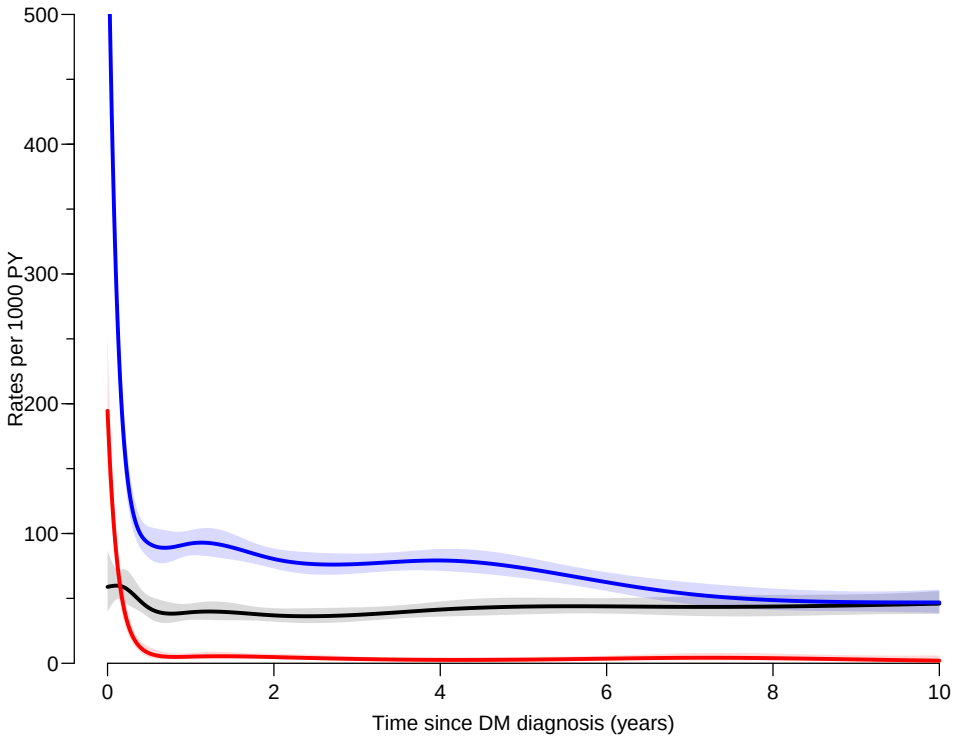


Figure 6.2: *Estimated rates of OAD, Ins and Dead from the* `DM` *state. Estimates are from* `glm` *models fitted to data split in 1 month intervals (1/12 year, that is), using natural splines. The y-axis is linear in the rates and starts exactly at 0, so allowing visualization of the integrals as areas under the curves.* `../graph/crisk-rates`

Note that the graphs in figures **??** (logarithmic y-scale) and 6.2 are not normally shown in analyses of competing risks; the competing cause-specific rates are hardly ever shown. I suspect that this is largely because they are often modeled by a Cox model and so are buried in the model.

`comprisk`

6.3 Cumulative rates and risks

For the calculation of the cumulative rates and state probabilities, we need just the rates without CIs:

```
> # utility to compute midpoints in a vector x
> mp <- function(x) x[-1] - diff(x) / 2
> # rates at midpoints
> lD <- mp(ci.pred(mD, nd)[,1])
> lI <- mp(ci.pred(mI, nd)[,1])
> l0 <- mp(ci.pred(m0, nd)[,1])
> # cumulative rates at right border of the intervals
> LD <- cumsum(lD) * int
> LI <- cumsum(lI) * int
> L0 <- cumsum(l0) * int
> # but when integrating to get the cumulative risks we use the average
> # of the survival function at the two endpoints (adding 1 as the first)
> Sv <- c(1, exp(- LD - LI - L0))
> rD <- c(0, cumsum(lD * mp(Sv)) * int)
> rI <- c(0, cumsum(lI * mp(Sv)) * int)
> r0 <- c(0, cumsum(l0 * mp(Sv)) * int)
```

Now we have the cumulative risks for the three causes and the survival, computed at the end of each of the intervals, at any time point the sum of the 3 cumulative risks and the survival should be 1:

```
> summary(Sv + rD + rI +r0)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1       1       1       1       1       1
> oo <- options(digits = 20)
> cbind(summary(Sv + rD + rI + r0))
                          [,1]
Min.    1.0000000000000000000
1st Qu. 1.0000003131293013769
Median  1.0000003337919416424
Mean    1.0000003247574191789
3rd Qu. 1.0000003432689923422
Max.    1.0000003488362141368
> options(oo)
```

We can then plot the 3 cumulative risk functions together:

```
> zz <- mat2pol(cbind(rD, rI, r0, Sv), x = nd$tfd,
+               xlim = c(0,10), xaxs = "i", yaxs = "i", las = 1,
+               xlab = "Time since DM diagnosis (years)",
+               ylab = "Probability",
+                col =  c(gray(0.3),"red","blue","forestgreen"))
> mm <- t(apply(zz, 1, mp))
> text(9, mm[900,], c("Dead","Ins","OAD","DM"), col = "white")
> box(col = "white",lwd = 3)
```

6.4 Confidence intervals for cumulative risks

We want confidence intervals for each of the 4 cumulative risks, but we may also be interested in confidence intervals for *sums* of any subset of the cumulative risks, corresponding to the borders between the colors in figure 6.3. If we only had two competing risks (and hence three states) the latter would not be an issue, because the sum of any two cumulative risks will be 1 minus the cumulative risk of the remainder, so we could get away with the confidence intervals for the single cumulative risks. This is the reason we have chosen an example with 3 competing risks and not just 2; we then have 4 probabilities to sum in different order.

A short look at the formulae for cumulative risks will reveal that analytic approximation to the standard error of these probabilities (or some transform of them) is not really a viable way to go. Particularly if we also want confidence intervals of sums of the state probabilities as those shown in stacked plots.

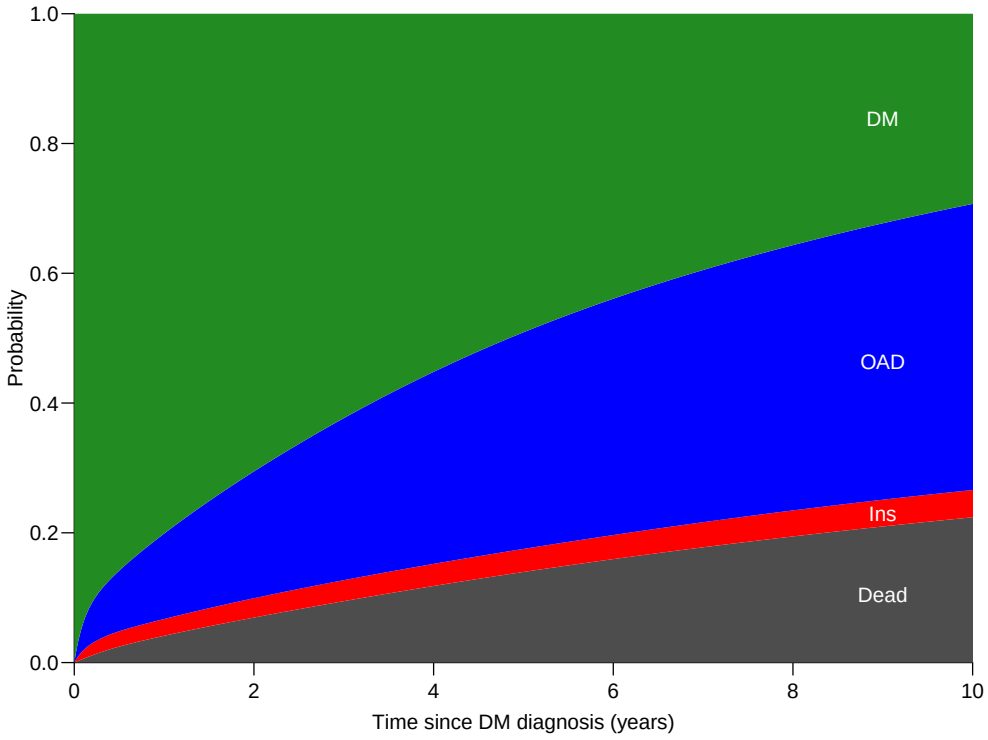


Figure 6.3: *Probabilities of being in the 4 different states as a function of time since diagnosis. Note that* `OAD` *means that OAD was initiated first, and similarly for* `Ins`*. We are not concerned about what occur after these events.* `Dead` *means dead without being on any drug.*
`../graph/crisk-stack`

So in practice, if we want confidence intervals not only for the state probabilities, but also for any sum of subsets of them we would want a large number of simulated copies of the cumulative risks, each copy of the same structure as the one we just extracted from the model.

Moreover, we might also want confidence intervals for sojourn times (i.e. time spent) in each state up to a given time, which would come almost for free from the simulation approach.

This means that we must devise a method to make predictions from the estimated model, but where we instead of the estimated model parameters use a sample from the posterior distribution of the estimated parameters. The posterior distribution of the parameters is taken to be multivariate normal with mean equal to the vector of parameter estimates and variance-covariance matrix equal to the estimated variance-covariance matrix of the parameters (the Hessian).

The simulation is taking place at the parameter level and the transformation to survival and cumulative risks is simply a function applied to every simulated set of parameters.

This simulation scheme is sometimes called the parametric bootstrap.

6.5 Joint models for several transitions

We have implicitly been assuming that the transition rates are modeled separately. If some transitions are modeled jointly—for example assuming that the rates of `OAD` and `Ins` are proportional as functions of time since entry, using one model—we are in trouble, because we then need one sample from the posterior generating two predictions, one for each of the transitions modeled together. Moreover the model will have to be a model fitted to a `stack.Lexis` object (see p. 22), so a little more complicated to work with.

A simple way to program this would be to reset the seed to the same value before simulating with different values of `nd`, this is what is intended to be implemented, but is not yet. This is mainly the complication of having different prediction frames for different risks in this case.

However, this is not a very urgent need, since the situation where you want common parameters for

different rates out of a common state is quite rare. Try to come up with an example like: We assumed that for persons in state `A` the effect of covariate Z was the same for the rates of `B` and `C`.

6.6 Simulation based confidence intervals

. . . has been implemented in the function `ci.Crisk` (confidence `i`ntervals for `C`umulative `risk`s) in the `Epi` package. The confidence intervals for the cause-specific rates are easily derived from the models using `ci.pred`, but confidence intervals for the cumulative risks and the sojourn times are from a practical point of view analytically intractable, hence a function to do this by simulation (parametric bootstrap).

We can now run the function using the model objects for the three competing events, using a common prediction data frame, `nd` for the rates:

```
> system.time(
+ res <- ci.Crisk(list(OAD = mO,
+                       Ins = mI,
+                      Dead = mD),
+                  nd = data.frame(tfd = seq(0, 10, 0.01)),
+                  nB = 500,
+                perm = 4:1))
NOTE: Times are assumed to be in the column tfd at equal distances of 0.01
   user  system elapsed
  11.39    0.35   11.75
> str(res)
List of 4
 $ Crisk: num [1:1001, 1:4, 1:3] 1 0.992 0.984 0.977 0.97 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ tfd  : chr [1:1001] "0" "0.01" "0.02" "0.03" ...
  .. ..$ cause: chr [1:4] "Surv" "OAD" "Ins" "Dead"
  .. ..$      : chr [1:3] "50%" "2.5%" "97.5%"
 $ Srisk: num [1:1001, 1:3, 1:3] 0 0.000609 0.001211 0.001803 0.002389 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ tfd  : chr [1:1001] "0" "0.01" "0.02" "0.03" ...
  .. ..$ cause: chr [1:3] "Dead" "Dead+Ins" "Dead+Ins+OAD"
  .. ..$      : chr [1:3] "50%" "2.5%" "97.5%"
 $ Stime: num [1:1001, 1:4, 1:3] 0 0.00996 0.01983 0.02964 0.03937 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ tfd  : chr [1:1001] "0" "0.01" "0.02" "0.03" ...
  .. ..$ cause: chr [1:4] "Surv" "OAD" "Ins" "Dead"
  .. ..$      : chr [1:3] "50%" "2.5%" "97.5%"
 $ time : num [1:1001] 0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 ...
 - attr(*, "int")= num 0.01
```

CODE EXPLAINED: `ci.Crisk` requires as input a list of fitted models for the cause-specific rates (first argument), and a data frame of prediction points which will be used for all models in the list. `nB` gives the number of simulated replicates of the rates.

The result is a list with 4 elements, three 3-way arrays and a vector of times associate with each row of the matrices.

One of slots in the returned object will refer to sums of probabilities of causes (stacked probabilities), the `perm` argument governs in what order the sum is formed. `perm` must have one more component than the list in the first argument; the first will refer to the survival. Thus in this case we will have the components `Dead`, `Dead`+`Ins` and `Dead`+`Ins`+`OAD`.

As we see, the returned object (`res`) is a list of length 4, the first three elements are 3-way arrays, and the last a vector of times. The three first components of `res` are:

- `Crisk` Cumulative `risk`s for each state
- `Srisk` Stacked cumulative `risk`s across states
- `Stime` Sojourn `time` for each state, truncated at each point of the time dimension, hence the first slice in the time-dimension is uniformly 0.

The first dimension of each is time starting with 0, and named corresponding to endpoints of intervals of length `int`. The second dimension is states (or combinations thereof). The last dimension of the arrays is the type of statistic; `50%` the median of the samples, and the confidence intervals as defined by the the `alpha` argument to `ci.Crisk`.

The argument `perm` governs in which order the state probabilities are stacked in the `Srisk` element of the returned list, the default is the `Surv` (survival, probability of being in initial state) followed by states in the order given in the list of models in the first argument to `ci.Crisk`.

6.6.1 Keeping the bootstrap samples

If we want the bootstrap samples for other calculations we can ask the function to return the bootstrap samples of the rates by using the argument `sim.res='rates'` (defaults to `'none'`):

```
> # prediction data frame
> prfr <- data.frame(tfd = seq(0.01, 10, 0.01) - 0.01/2)
> system.time(
+ rsm <- ci.Crisk(list(OAD = mO,
+                      Ins = mI,
+                     Dead = mD),
+                 nd = prfr,
+                 nB = 2000,
+             sim.res = 'rates'))
NOTE: Times are assumed to be in the column tfd at equal distances of 0.01
   user  system elapsed
   0.31    0.00    0.31

> str(rsm)

 num [1:1000, 1:3, 1:2000] 0.603 0.563 0.525 0.49 0.458 ...
 - attr(*, "dimnames")=List of 3
  ..$ time: chr [1:1000] "0.005" "0.015" "0.025" "0.035" ...
  ..$ mod : chr [1:3] "OAD" "Ins" "Dead"
  ..$ sim : chr [1:2000] "1" "2" "3" "4" ...
 - attr(*, "int")= num 0.01
 - attr(*, "time")= num [1:1000] 0.005 0.015 0.025 0.035 0.045 0.055 0.065 0.075 0.085 0.095 ...
```

These are 2000 (parametric) bootstrap samples of the rates evaluated at the 1000 midpoints of intervals of length 1/100 year between 0 and 10 years of diabetes duration.

Alternatively we can get the bootstrap samples of the cumulative risks by setting `sim.res='crisk'`:

```
> system.time(
+ csm <- ci.Crisk(list(OAD = mO,
+                       Ins = mI,
+                      Dead = mD),
+                  nd = prfr,
+                  nB = 2000,
+              sim.res = 'crisk'))
NOTE: Times are assumed to be in the column tfd at equal distances of 0.01
   user  system elapsed
  21.47    0.28   21.76

> str(csm)

 num [1:1000, 1:4, 1:2000] 1 0.991 0.983 0.976 0.969 ...
 - attr(*, "dimnames")=List of 3
  ..$ tfd  : chr [1:1000] "0.005" "0.015" "0.025" "0.035" ...
  ..$ cause: chr [1:4] "Surv" "OAD" "Ins" "Dead"
  ..$ sim  : chr [1:2000] "1" "2" "3" "4" ...
 - attr(*, "int")= num 0.01
```

This is the bootstrap samples of cumulative risks evaluated at the 1001 endpoints of the 1000 intervals, and also includes the survival probability in the first slot of the 1st dimension of `rsm`.

`comprisk`

6.6.2 Rates

In figure 6.2 we showed the rates with confidence intervals from the model. But in `rsm` we have 2000 (parametric) bootstrap samples of the occurrence rates, so we can derive the bootstrap medians and the bootstrap c.i. We use the function `Epi:::mnqt` to compute the model estimate and the mean, median and quantiles of the simulated values.

```
> Epi:::mnqt
function (x, alpha = 0.05)
{
    c(quantile(x, probs = c(0.5, alpha/2, 1 - alpha/2), na.rm = TRUE))
}
<bytecode: 0x000001fe50c0bef0>
<environment: namespace:Epi>
> Brates <- aperm(apply(rsm, 1:2, Epi:::mnqt), c(2,3,1))
> str(Brates)
 num [1:1000, 1:3, 1:3] 0.596 0.553 0.513 0.476 0.442 ...
 - attr(*, "dimnames")=List of 3
  ..$ time: chr [1:1000] "0.005" "0.015" "0.025" "0.035" ...
  ..$ mod : chr [1:3] "OAD" "Ins" "Dead"
  ..$     : chr [1:3] "50%" "2.5%" "97.5%"
```

Then we can plot the bootstrap estimates on top of the estimates based on the normal approximation to the distribution of the parameters. They are (not surprisingly) in close agreement since they are both based on an assumption of normality of the parameters on the log-rate scale:

```
> matshade(nd$tfd, cbind(ci.pred(mD, nd),
+                        ci.pred(mI, nd),
+                        ci.pred(mO, nd))*1000,
+          ylim = c(1,500), yaxt = "n",
+          ylab = "Rates per 1000 PY",
+          xlab = "Time since DM diagnosis (years)",
+          col = c("black","red","blue"), log = "y", lwd = 3, plot = TRUE)
> matlines(as.numeric(dimnames(Brates)[[1]]),
+          cbind(Brates[,"Dead",],
+                Brates[,"Ins" ,],
+                Brates[,"OAD" ,])*1000,
+          col = c("white","black","black"), lty = 3, lwd=c(3,1,1))
> axis(side = 2, at = ll <- outer(c(1,2,5), -2:3, function(x,y) x * 10^y),
+               labels = formatC(ll,digits = 4), las = 1)
> axis(side = 2, at = ll <- outer(c(1.5,2:9),-2:3, function(x,y) x * 10^y),
+               labels = NA, tcl = -0.3)
> text(0, 2.5*0.7^c(1,2,0),
+      c("Dead","Ins","OAD"),
+      col = c("black","red","blue"), adj = 0)
```

6.6.3 Cumulative risks

In the `Crisk` component of `res` we have the cumulative risks as functions of of time, with bootstrap confidence intervals, so we can immediately plot the three cumulative risks:

```
> matshade(res$time,
+          cbind(res$Crisk[,"Dead",],
+                res$Crisk[,"OAD" ,],
+                res$Crisk[,"Ins" ,]), plot = TRUE,
+          xlim = c(0,10), xaxs = "i", yaxs = "i", las = 1,
+          xlab = "Time since DM diagnosis (years)",
+          ylab = "Cumulative probability",
+           col = c("black","blue","red"))
> text(8, 0.3 + c(1,2,0)/25,
+      c("Dead","OAD","Ins"),
+      col = c("black","blue","red"), adj = 0)
```

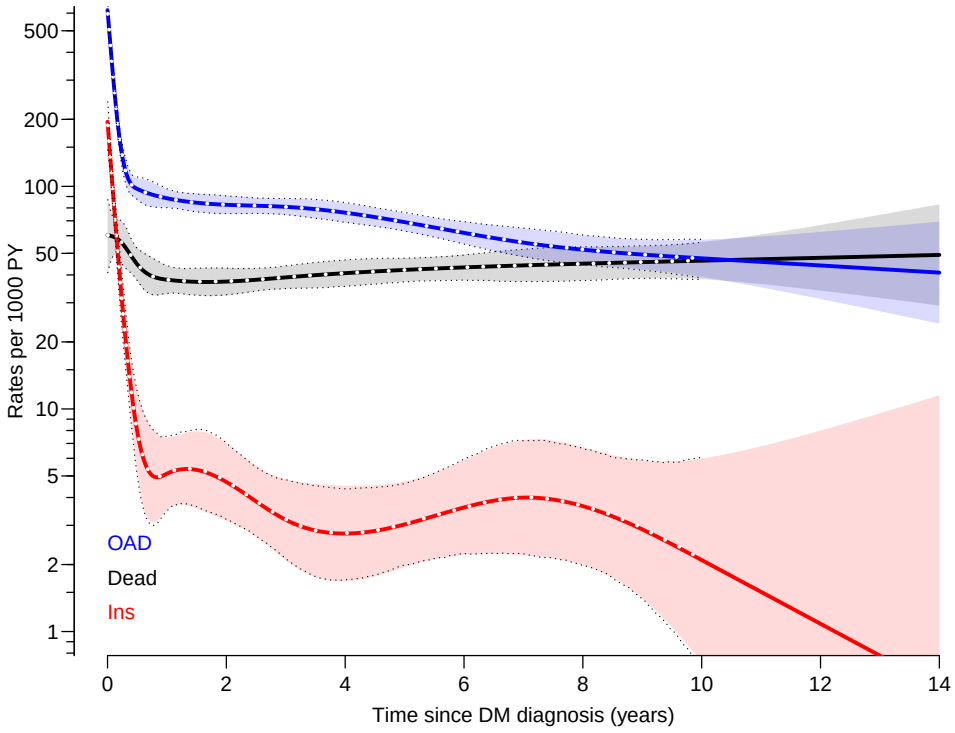


Figure 6.4: *Estimated rates (log-scale) from the* DM *state. Estimates are from* glm *models with natural splines fitted to data split in 1 month intervals (1/12 year, that is). The white dotted curves are the bootstrap medians, black dotted curves are the bootstrap 95% c.i.s.* ../graph/crisk-rates-ci

6.6.4 Stacked cumulative risks

With the single cumulative risks we have a confidence interval for each cumulative risk, but if we want to show the stacked probabilities we must deliver the confidence intervals for the relevant sum of cumulative risks (state probabilities). These are in the Srisk component of res. We can use these to show the stacked cumulative probabilities with confidence intervals:

```
> zz <- mat2pol(res$Crisk[,c("Dead","Ins","OAD","Surv"),"50%"],
+               x = res$time,
+            xlim = c(0,10), xaxs = "i", yaxs = "i", las = 1,
+            xlab = "Time since DM diagnosis (years)",
+            ylab = "Probability",
+             col =  c("black","red","blue","forestgreen") )
> mm <- t(apply(zz, 1, mp))
> text(9, mm["9",], c("Dead","Ins","OAD","DM"), col = "white" )
> matshade(res$time,
+          cbind(res$Srisk[,"Dead",],
+                res$Srisk[,"Dead+Ins",],
+                res$Srisk[,"Dead+Ins+OAD",]),
+          col = 'transparent', col.shade = "white", alpha = 0.4)
```

CODE EXPLAINED: The stacked probabilities are plotted as stacked polygons using mat2pol (matrix to polygon), where the input required is the single components from the Crisk component. Then we can add the confidence intervals by using matshade with the components of the Srisk component to add white semi-transparent confidence intervals, while asking for a fully transparent curve.

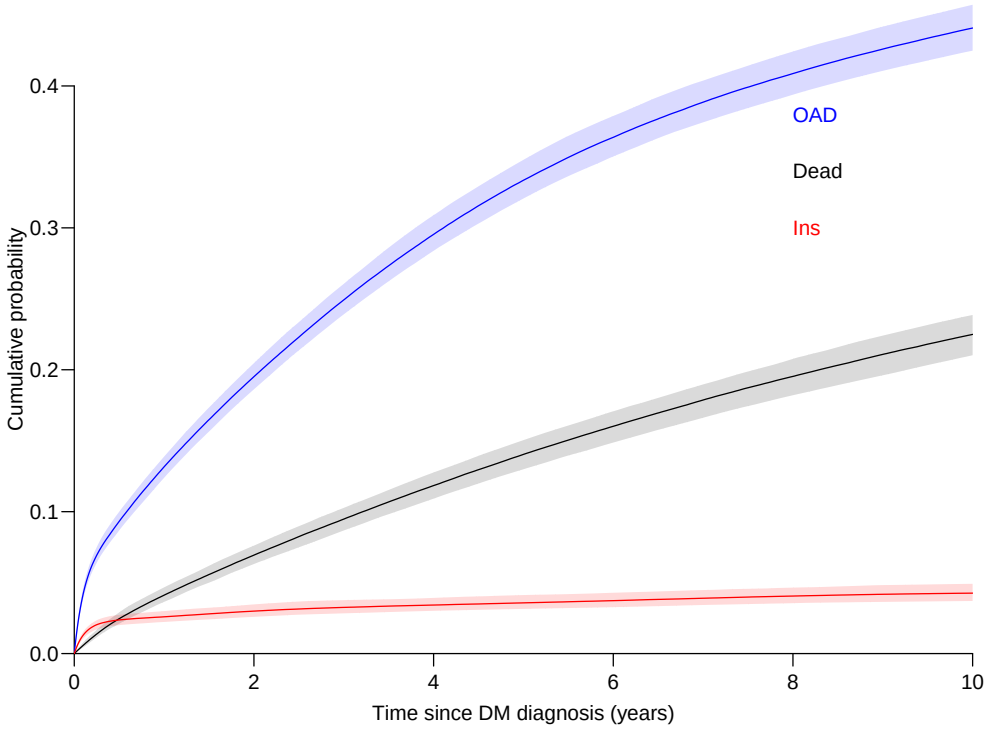


Figure 6.5: *Cumulative risks for the three types of events, with 95% bootstrap-based confidence intervals as shades.* `../graph/crisk-crates`

6.6.5 Sojourn times

From the `Stime` component of the `res` we can derive the estimated time spent in each state during the first, say, 1, 2, 5 or 10 years:

```
> ftable(round(res$Stime[paste(c(1,3,5,10)),,], 2), row.vars = 2)
      tfd    1                3                5               10
           50% 2.5% 97.5%  50% 2.5% 97.5%  50% 2.5% 97.5%  50% 2.5% 97.5%
cause
Surv      0.87 0.86  0.87 2.28 2.25  2.31 3.39 3.33  3.44 5.29 5.17  5.40
OAD       0.09 0.08  0.09 0.48 0.45  0.50 1.06 1.02  1.11 3.04 2.93  3.16
Ins       0.02 0.02  0.02 0.08 0.07  0.09 0.15 0.13  0.17 0.35 0.30  0.40
Dead      0.02 0.02  0.03 0.16 0.15  0.18 0.40 0.37  0.43 1.33 1.24  1.41
```

CODE EXPLAINED: The first dimension of the component `Stime` is the time, and the *names* of this are the endpoints of the intervals in the time scale (time since diagnosis). Thus if want the ones referring to specific times, we must supply these as a character vector, if we index by a numerical vector we would be referring to the sequence numbers. Hence the `paste()`.

So we see that the expected life lived without pharmaceutical treatment during the first 10 years after DM diagnosis is 5.29 years with a 95% CI of (5.17; 5.4) and during the first 5 years 3.39 (3.33; 3.44) The quantity `OAD` is the years spent after OAD medication, regardless of subsequent insulin medication or death, and similarly for `Ins` and `Dead`.

6.7 Aalen-Johansen approach

Another estimator of the cumulative risk is the Aalen-Johansen estimator, which is a so-called non-parametric estimator of the cumulative risks or more precisely the state probabilities.

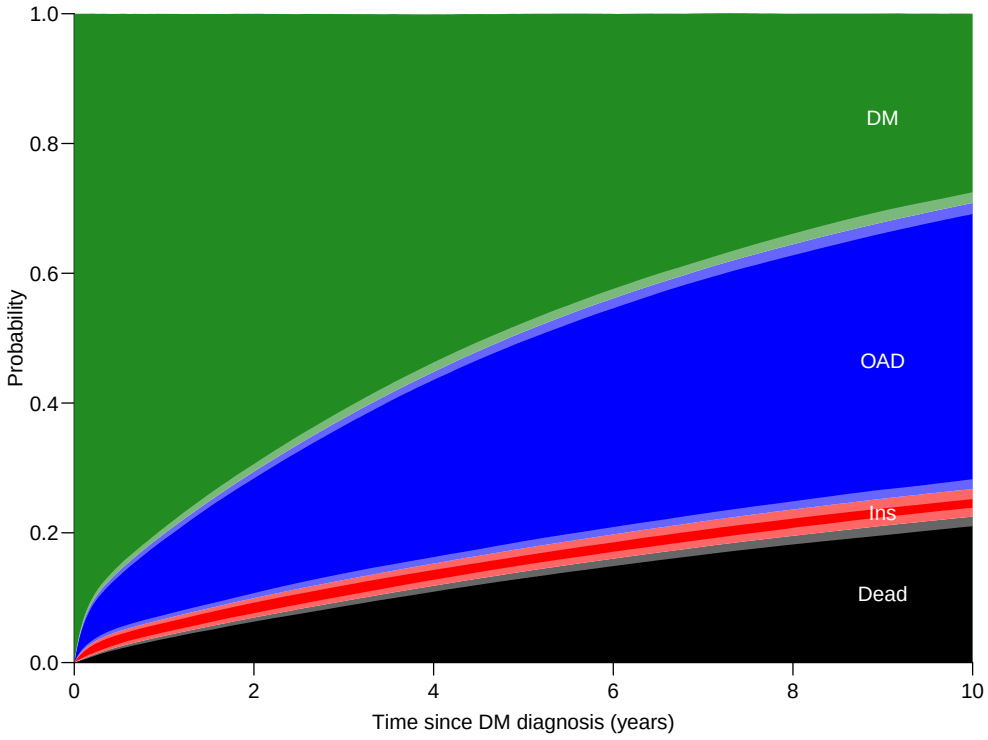


Figure 6.6: *Probabilities of being in the 4 different states as a function of time since diagnosis. Note that* **`OAD`** *means that OAD was initiated first, and similarly for* **`Ins`**. *We are not concerned about what occurs* after *these events.* **`Dead`** *means dead without being on any drug. The white shadings around the borders between colored areas represent the 95% confidence intervals for the (sum of) probabilities.* `../graph/crisk-stack-ci`

The following is based on the dataset which has one record per person, where the outcome is defined by a factor; precisely what is present in the `Lexis` object `Rdm`, but we must first subset it to those at risk in the state `DM`:

```
> library(survival)
> Rdm <- factorize(subset(Mdm, lex.Cst == "DM"))
```

In order to use the multistate machinery from the `survival` package, the initial state must be the *first* of the states in the factor `lex.Xst`—this can be checked by `levels`:

```
> levels(Rdm)
[1] "DM"   "Dead" "OAD"  "Ins"
```

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

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

In the `Epi` package is a wrapper for survfit that uses the features of a `Lexis` object, so that you only have to write:

```
> aaj <- AaJ.Lexis(Rdm, ~ 1, "tfd")
NOTE: Timescale is tfd
```

comprisk

There are 4 columns in each of slots in `pstate`, `lower` and `upper`, representing the states in the order of the factor `lex.Xst`, also available in the slot `states`. Here we only want the 3 last from each, namely the ones referring to the cumulative risks of `Dead`, `OAD` and `Ins`. We want estimate, lower, upper for each in that order:

```
> aaj$states
[1] "DM"   "Dead" "Ins"  "OAD"
> (wh <- c(2 + 0:2 * 4, 3 + 0:2 * 4, 4 + 0:2 * 4))
[1]  2  6 10  3  7 11  4  8 12
> aajCr <- with(aaj, cbind(pstate, lower, upper)[,wh])
> colnames(aajCr) <- rep(c("Est","lo","up"), 3)
> colnames(aajCr)[c(1,4,7)] <- aaj$states[-1]
> round(head(aajCr), 4)
      Dead     lo     up    Ins     lo     up    OAD     lo     up
[1,] 0e+00     NA     NA 0.0003 0.0001 0.0013 0.0018 0.0010 0.0033
[2,] 0e+00     NA     NA 0.0007 0.0003 0.0018 0.0030 0.0019 0.0048
[3,] 3e-04 1e-04 0.0013 0.0010 0.0005 0.0022 0.0059 0.0042 0.0081
[4,] 3e-04 1e-04 0.0013 0.0015 0.0008 0.0029 0.0074 0.0055 0.0099
[5,] 3e-04 1e-04 0.0013 0.0023 0.0014 0.0040 0.0082 0.0062 0.0108
[6,] 3e-04 1e-04 0.0013 0.0030 0.0019 0.0048 0.0090 0.0069 0.0118
```

CODE EXPLAINED: Each of the elements `pstate`, `lower` and `upper` have 4 columns corresponding the the elements in `states`, but we only want columns 2, 3 and 4 from each. `wh` gives the columns in the object we get from `cbind`ing the elements, in the desired order.

Finally we provide column names for better readability.

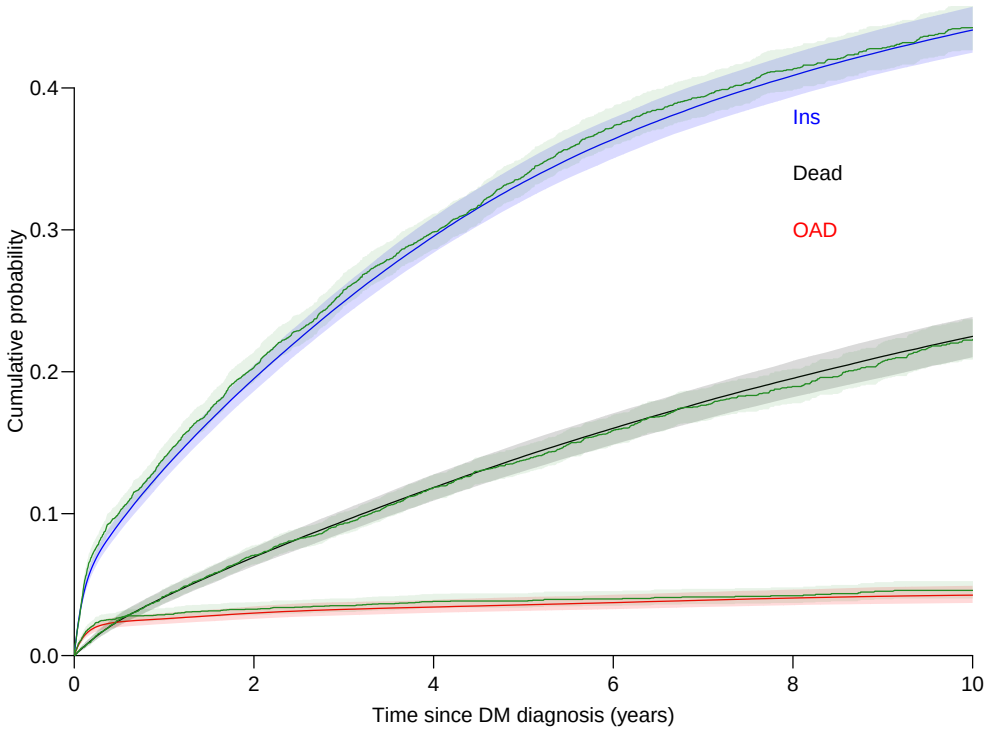


Figure 6.7: *Cumulative risks for the three types of events based on parametric models, compared to the Nelson-Aalen estimator 95% bootstrap-based confidence intervals shown as shades, Aalen-Johansen estimator in green; we see that the two approaches give very similar results.* `../graph/crisk-cratesx`

```
> matshade(res$time,
+          cbind(res$Crisk[,"Dead",],
+                res$Crisk[,"OAD" ,],
+                res$Crisk[,"Ins" ,]), plot = TRUE,
+          xlim = c(0,10), xaxs = "i", yaxs = "i", las = 1,
+          xlab = "Time since DM diagnosis (years)",
+          ylab = "Cumulative probability",
+           col = c("black","blue","red"))
> matshade(aaj$time, aajCr, col = "forestgreen", lty = 1, alpha = 0.1)
> text(8, 0.3 + c(1,2,0)/25,
+      c("Dead","Ins","OAD"),
+      col = c("black","blue","red"), adj = 0)
```

From figure 6.7 we see that the general pattern of cumulative risks is the same as from the parametric model, but with a tendency of estimating the cumulative risks of `Ins` and `OAD` a bit higher than the parametric model. This is a consequence of early events that are captured differently in the parametric model, even if we excluded the first two months of observation.

6.8 Interpretation

In the example here we used only time since diagnosis as explanatory variable for the rates. But there is presumably a substantial effect of age and calendar time as well. So the results shown here are for a diabetes population with the age-distribution of the Danish, observed over a calendar time as the Danish. So not necessarily of major interest outside of Denmark.

But we could get results that were more generalizable if we included effects of, say, sex, age and calendar time in models for the rates too. This could be done with the parametric models as well as with Cox-models for the cause-specific rates.

That would provide estimates of the multiplicative effects of the covariates on the rates, but only the parametric approach would make it possible to show the cumulative risks and sojourn times for select values of the covariates—the only thing required would be a suitably constructed prediction data frame as `nd` argument to `ci.Crisk`.

. . . now input from `simLex`

Chapter 7

Estimation from multistate models

If we set up a multistate model for the follow-up of a cohort of persons, and estimate rate-models for all transitions, we will in principle have a complete probabilistic description of the model. This means that for any person (i.e. any set of initial covariate values) in any state at any time we can compute the person's probability of being in any state at any future time.

Because it is implicit that any covariate value should be known at any time in the future, this excludes models where transition rates depend on clinical measurements obtained during follow-up. In practice this means that only initial clinical measurements can be accommodated. This does not apply to future state occupancy, because the state occupancy is a part of the probability model.

Unlike the situation for simple survival and competing risks, there is no practically tractable expression for the state probabilities in a general multistate model for a given person at a given time in a given state. Let alone for sojourn times and other measures derived from the state probabilities.

State probabilities can instead be estimated as empirical fractions of persons in each state in a simulated data set based on the model. The drawback is that if we estimate the state probabilities from a simulated dataset there is no simple way of getting a handle on the uncertainty of the estimates as for example confidence intervals.

Generating simulated datasets of persons from a multistate model is often termed micro-simulation; this is because we are simulating the history of individuals and not of populations.

7.1 Simulated multistate objects

When a multistate model has been set up (states and transitions) and models for all transition rates have been estimated, we have a complete statistical model for the multistate data.

This means that we can generate data in the form of (events, time) from the multistate model, that is state-histories for a given set of persons beginning follow-up in the defined states.

The requirements for this endeavor are:

- A data set with initial values of state, time scales and other covariates for a set of persons (a cohort).
- A statistical model for each of the possible transitions between states. This is basically a means to produce transition rates at any future time for all possible transitions out of each state, based on the status at entry to the state.
- Technical parameters: the position of time points for evaluation of transition rates and the length of simulation time before censoring.

The (life-)course of each of the initial persons through the states can then be simulated. The simulated data will be in the same form as the original multistate data; the `simLexis` function will produce a `Lexis` object—a simulated follow-up of the initial cohort.

From the simulated data we can then derive the desired quantities such as state probabilities, expected sojourn times in states etc.

7.1.1 Simulation machinery

Inside the `simLexis` the simulation works as follows. Suppose we have a person in some current state (either the initial state or newly arrived from a previous one) and that we want to simulate a transition time and transition type out of the current state.

If there are a number of possible transitions out of the current state, with transition rates $\lambda_i(t), i = 1, 2, \ldots$ we simulate a transition time from each, pretending that the specific rate is the rate of the only possible exit from the current state as follows.

The survival function in this case is $S_i(t) = \mathrm{e}^{-\Lambda_i(t)}$, where $\Lambda_i(t) = \int_0^t \lambda(s)\,\mathrm{d}s$. Some call the function $S_i(t)$ the *net* survival, but it really does not refer to any probability of this world.

We then choose independent random uniform variates $u_i \in [0, 1]$, and solve numerically for t_i:

$$S_i(t_i) = u \quad \Leftrightarrow \quad -\log(u_i) = \Lambda(t_i)$$

In practice we compute $\Lambda_i(t)$ at a number of times by numerical integration of the predicted rates, $\hat{\lambda}_i$ and then find t_i by linear interpolation, by finding two successive t-values that have Λ-values on either side of $-\log(u)$.

This way we get a transition time t_i for each of the states reachable from the current state. We then pick the smallest of these times and the corresponding state as the time and state for transition out of the current state. If there is only one possible transition out of the current state we only have one time and one state to choose between, so much simpler.

This is then repeated in the next state till the person reaches an absorbing state or exceeds the stipulated censoring time. The result is a dataset that resembles follow-up of the persons through the possible states.

7.1.1.1 Alternative simulation scheme

A time and type of transition can also be simulated by first simulating the transition time, t, using the sum of the intensities, $\sum_i \lambda_i(t)$, and hence survival function $S_i(t) = \mathrm{e}^{-\sum_i \Lambda_i(t)}$. Subsequently we then simulate the type of event at t as a random draw from the multinomial distribution with probabilities $p_i = \lambda_i(t) / \sum_j \lambda_j(t)$.

It can be shown that this produces the same joint distribution of events and event times as the one used in `simLexis` described above.

7.2 Using simulated multistate data

Once a complete model (states, transitions and models for the transitions) has been set up and a suitably large cohort of persons have been simulated, we can use the simulated data to derive measures of interest:

- State probabilities can be derived as the fraction of the initial cohort in each state at the time points of interest
- Sojourn times (in a given set of states between specified time points) can be derived by aggregating these among the simulated persons
- Probability of ever visiting a given state can be derived by simply enumerating the fraction ever visiting the state.

Trivial as they may sound, these operations on the simulated cohort are not simple, so special functions are needed to perform them.

But note that the results depend *both* on the multistate model *and* the chosen initial population; the derived statements will refer to a population of a particular composition (for example a population of identical individuals).

7.3 A worked example of using `simLexis`

7.3.1 `Lexis` object for the `steno2` data frame

The `steno2` dataset is a simulated dataset with the same structure and characteristics as the original data from the 20 year follow-up; it is described a bit more thorough in chapter 4, **??**, p. 46.

```
simLex
```

First we 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. Also, we make Conv (conventional treatment) the first level of the allocation factor; this will give estimates of intervention effects directly in the standard parametrization:

```
> data(steno2)
> steno2 <- cal.yr(steno2) %>%
+           mutate(doEnd = ifelse(!is.na(doDth),
+                                 doDth,
+                                 doEnd),
+                  allo = relevel(allo, 2))
> 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 "Conv","Int": 2 2 1 1 1 1 1 2 2 2 ...
 $ 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   : num  2015 2015 2002 2003 1998 ...
 $ doDth   : 'cal.yr' num  NA NA 2002 2003 1998 ...
```

We then define a Lexis data frame for the entire follow-up time for each person; from entry (doBase, date of baseline) to exit, doEnd. Note that we name 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.
```

CODE EXPLAINED: We set up a Lexis object with 3 time scales: calendar time, per; current age age and time from inclusion, tfi.

The coding of exit.status is as follows: deathCVD is 1 if the person died from cardiovascular disease, otherwise 0. Therefore, the expression deathCVD + !is.na(doDth) is $0+0=0$ if no death occurs, $0+1=1$ if a non-CVD death occur and $1+1=2$ if a CVD death occur at the end of the person's follow-up. [1]

```
> summary(L2)
Transitions:
     To
From  Mic D(oth) D(CVD)  Records:  Events: Risk time:  Persons:
  Mic  67     55     38       160       93    2420.91       160
```

In this set-up we can study the CVD and the non-CVD mortality rates, a classical competing risks problem, but we also want to see how the mortality rates depend on albuminuria status (lex.Cst) and treatment allocation (allo).

In order to allocate follow-up (person-time and events) to *current* albuminuria status we need to know when 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 we end use the function rcutLexis (recurrent cuts); this function does not keep track of the state-history, but only records the current state. It requires a data frame of transitions with columns lex.id, cut (time of transition) and new.state (the state that the transition is to)—see ?rcutLexis.

We change the scale of the date of transition in st2alb to year by cal.yr (to align with the per variable in L2), and in order to comply with the requirements of rcutLexis we 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 <- rename(cal.yr(st2alb),
+                lex.id = id,
+                   cut = doTr,
+             new.state = 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 ...
```

An alternative way of achieving the same is to start with st2alb and then pipe it through cal.yr and rename. This can be done in different styles according to your preferences:

```
>   ( st2alb
+ %>% cal.yr
+ %>% rename(lex.id = id,
+                cut = doTr,
+          new.state = state)
+   ) -> cut2
> #
> # or:
> #
> cut2 <- st2alb %>%
+         cal.yr %>%
+         rename(lex.id = id,
+                   cut = doTr,
+             new.state = 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 ...
```

> CODE EXPLAINED: The initial data frame, st2alb is first processed by cal.yr which turns all date variables to class cal.yr, that is measured in years (chunks of 365.25 days that is) since 1970-1-1. have Then variables are renamed (newname = oldname) as needed for rcutLexis, and finally assign the result to a data frame cut2. The layout of the code is the point here; the pipe operators (%>%) are placed at the beginning of the lines, so that they are aligned and clearly indicates where each step starts. Finally, *after* the data operations, the result (a data frame) is assigned to cut2. It is the initial opening bracket that makes it possible to place the pipes at the beginning of each line; it ensures that the expression is not complete till the last closing bracket. If you place the pipe operators at the end of the line you do not have that worry, but they are less easily noted when reading the code.

We see that 156 persons have recorded transitions, mostly 3, 4 or 5 of them:

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

We then cut at intermediate transition times (note that rcutLexis by default assumes that values in the cut column refer to the first timescale, and the first of the timescales in L2 is per, so we need not use the timescale argument:

```
> 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    1383.56       160
  Norm  31   90   5     14      7       147       57     608.75        69
  Mac   20    3  44     14     18        99       55     428.60        64
  Sum  350  165 114     55     38       722      289    2420.91       160
```

We note that there are transitions both ways between all three transient states `Norm`, `Mic` and `Mac`, which is a bit illogical, since we have a natural ordering of states: `Norm` < `Mic` < `Mac`, so transitions from `Norm` to `Mac` (and vice versa) should therefore go through `Mic`. In order to remedy this anomaly we find all transitions `Norm` → `Mac` and provide a transition `Norm` → `Mic` in between. And of course similarly for transitions `Mac` → `Norm`.

The relevant "jump" transitions are easily found (we keep all timescales to be able to use the print method for `Lexis` objects):

```
> (jump <-
+ subset(L3, (lex.Cst == "Norm" & lex.Xst == "Mac") |
+             (lex.Xst == "Norm" & lex.Cst == "Mac"))[,
+             c("lex.id",timeScales(L3),"lex.dur","lex.Cst","lex.Xst")])
 lex.id     per   age   tfi lex.dur lex.Cst lex.Xst
     70 1999.49 64.68  6.42    2.67     Mac    Norm
     86 2001.76 68.88  8.34   12.82    Norm     Mac
    130 2000.91 61.42  7.12    1.88     Mac    Norm
    131 1997.76 65.99  3.85    4.24    Norm     Mac
    136 1997.21 47.41  3.26    0.47     Mac    Norm
    136 1997.69 47.88  3.73    4.24    Norm     Mac
    171 1996.39 56.89  2.08    5.34    Norm     Mac
    175 2004.58 67.60 10.24    9.88    Norm     Mac
```

CODE EXPLAINED: The jumps we are interested in are records representing time in `Norm` ending in a jump to `Mac`, that is where `lex.Cst` is `Norm` and `lex.Xst` is `Mac`. Or vice versa.

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 `Norm` or `Mac`, i.e. somewhere between `per` and `per` + `lex.dur` in these records; we choose a random time in the middle 80% between the dates. What we are doing here is really *single* imputation; we know there is a missing transition time to `Mic` and we just invent one and pretend it is a real data point.

```
> set.seed(1952)
> xcut <- select(transform(as.data.frame(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
291     70 2001.789       Mic
353     86 2012.232       Mic
506    130 2001.488       Mic
511    131 2001.032       Mic
525    136 1997.610       Mic
526    136 2000.780       Mic
654    171 1997.057       Mic
676    175 2013.472       Mic
```

Then we make extra cuts (transitions to `Mic`) at these dates using `rcutLexis` with `xcut` on the `L3` object:

```
> L4 <- rcutLexis(L3, xcut)
> summary(L4)
```

```
Transitions:
     To
From    Mic Norm Mac D(oth) D(CVD)  Records:  Events: Risk time:  Persons:
  Mic   312   72  65     30     14       493      181    1437.39       160
  Norm   35   90   0     13      6       144       54     581.83        66
  Mac    22    0  41     12     18        93       52     401.70        60
  Sum   369  162 106     55     38       730      287    2420.91       160
```

We see that there are now no transitions directly between `Norm` and `Mac` in `L4`, so we can make a plot of the transitions to consider:

```
> clr <- c("forestgreen", "orange", "red", "blue", gray(0.4))
> boxes(L4, boxpos = list(x = c(20,20,20,80,80),
+                         y = c(50,90,10,75,25)),
+          scale.R = 100,
+          show.BE = "nz",
+          pos.arr = c(2,2,1,1,2,1,1,2,1,1) * 0.25,
+             font = 2,
+           col.bg = clr,
+          col.txt = c("white", "black")[c(1,2,1,1,1)],
+       col.border = c(clr[1:3], rep("black", 2)),
+              lwd = 2, cex = 1.1)
```

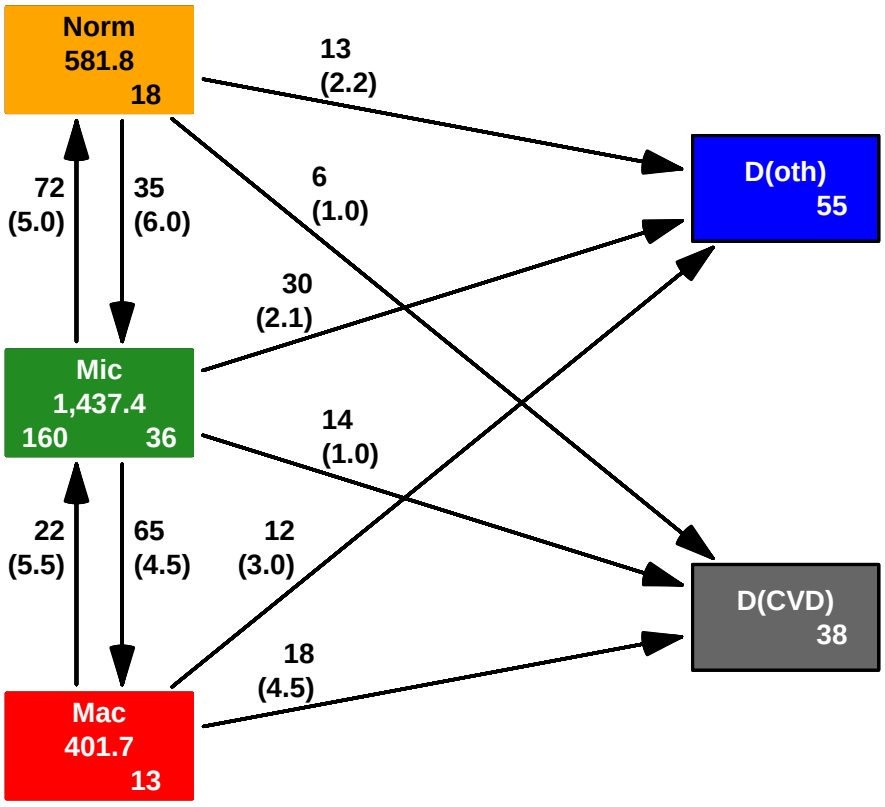


Figure 7.1: *Transitions between albuminuria and death states in the Steno2 study.* `../graph/simLex-boxL4`

CODE EXPLAINED: The function boxes.Lexis has a zillion arguments to regulate colors and other appearance parameters of boxes and arrows.

The boxpos argument is a list with x and y coordinates (between 0 and 100) of the midpoints of the state-boxes. scale.R scales the rates printed along the arrows, show.BE shows the number of persons beginning, respectively ending their follow-up in each state. pos.arr sets the position of the arrow annotation (as a fraction, measured from the base of the arrow); it is a vector referring to the internal numbering of the arrows (see below), here the annotation between the transient states is at the middle, and the annotation of transitions to the absorbing states at 0.25 from the origin.

font=2 sets the display font to 2 which is bold face.

The col. arguments set the background, text and border color of the boxes. Note the we initially put the colors in a vector for later use. Finally, cex sets the font size relative to the font size of the current driver—the default value for the argument cex is 1.4.

Here are two functions that show the sequence of the arrows used in the arguments to boxes that modifies arrow appearances:

```
> arr.no <-
+ function(Lx) # arrow numbers
+     {
+ ttm <- t(tmat(Lx))
+ ano <- cumsum(!is.na(ttm)) * as.vector(!is.na(ttm))
+ ttm[,] <- ifelse(ano == 0, NA, ano)
+ t(ttm)
+     }
> arr.ft <-
+ function(Lx) # arrows From To
+     {
+ tt <- arr.no(Lx)
+ ft <- data.frame(expand.grid(from = rownames(tt),
+                              to = colnames(tt)),
+                  arrow = as.vector(tt))
+ ft <- subset(ft, !is.na(arrow))
+ ft[order(ft$arrow),]
+     }
> arr.no(L4)
         Mic Norm Mac D(oth) D(CVD)
  Mic     NA    1   2      3      4
  Norm     5   NA  NA      6      7
  Mac      8   NA  NA      9     10
  D(oth)  NA   NA  NA     NA     NA
  D(CVD)  NA   NA  NA     NA     NA
> arr.ft(L4)
   from     to arrow
6   Mic   Norm     1
11  Mic    Mac     2
16  Mic D(oth)     3
21  Mic D(CVD)     4
2  Norm    Mic     5
17 Norm D(oth)     6
22 Norm D(CVD)     7
3   Mac    Mic     8
18  Mac D(oth)     9
23  Mac D(CVD)    10
```

Within this multistate model we fit models for the mortality rates to see how they depend on the current albuminuria state and treatment allocation, as well as models for the transition rates between the different albuminuria states and assess how these depend on various other covariates.

First we look at how the mortality rates depends on albuminuria status. We will model the mortality rates with parametric functions, so we must split the dataset along some time scale; we will use 3 month intervals (they should be sufficiently small to accommodate an assumption of constant rates in each interval):

```
> S4 <- splitMulti(L4, tfi = seq(0, 25, 1/4))
> summary(S4)
Transitions:
     To
From    Mic Norm  Mac D(oth) D(CVD)  Records:  Events: Risk time:  Persons:
  Mic  5986   72   65     30     14      6167      181    1437.39       160
  Norm   35 2418    0     13      6      2472       54     581.83        66
  Mac    22    0 1644     12     18      1696       52     401.70        60
  Sum  6043 2490 1709     55     38     10335      287    2420.91       160
> summary(Relevel(S4, 2:1))
Transitions:
     To
From   Norm  Mic  Mac D(oth) D(CVD)  Records:  Events: Risk time:  Persons:
  Norm 2418   35    0     13      6      2472       54     581.83        66
  Mic    72 5986   65     30     14      6167      181    1437.39       160
  Mac     0   22 1644     12     18      1696       52     401.70        60
  Sum  2490 6043 1709     55     38     10335      287    2420.91       160
```

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

We fit separate models for the two causes of death, using the wrapper glm.Lexis, allowing different allocation effects for different levels of albuminuria:

```
> mo <- 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:
Mic->D(oth)
Norm->D(oth)
Mac->D(oth)
> round(ci.exp(mo), 3)
                                  exp(Est.)  2.5%        97.5%
(Intercept)                           0.000 0.000 9.000000e-03
Ns(tfi, knots = seq(0, 20, 5))1     170.742 3.261 8.940449e+03
Ns(tfi, knots = seq(0, 20, 5))2      41.348 1.667 1.025863e+03
Ns(tfi, knots = seq(0, 20, 5))3   50328.536 6.222 4.070989e+08
Ns(tfi, knots = seq(0, 20, 5))4       2.460 0.412 1.468400e+01
Ns(age, knots = seq(50, 80, 10))1     2.680 0.887 8.100000e+00
Ns(age, knots = seq(50, 80, 10))2     1.547 0.148 1.615100e+01
Ns(age, knots = seq(50, 80, 10))3    11.777 4.223 3.284100e+01
lex.CstNorm                           0.965 0.396 2.353000e+00
lex.CstMac                            0.569 0.208 1.556000e+00
lex.CstMic:alloInt                    0.519 0.249 1.080000e+00
lex.CstNorm:alloInt                   0.558 0.183 1.695000e+00
lex.CstMac:alloInt                    1.538 0.484 4.890000e+00
> mC <- 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:
Mic->D(CVD)
Norm->D(CVD)
Mac->D(CVD)
> round(ci.exp(mC), 3)
                                  exp(Est.)  2.5%      97.5%
(Intercept)                           0.001 0.000      0.015
Ns(tfi, knots = seq(0, 20, 5))1       1.042 0.150      7.244
Ns(tfi, knots = seq(0, 20, 5))2       2.230 0.350     14.206
Ns(tfi, knots = seq(0, 20, 5))3       1.242 0.016     96.367
```

```
Ns(tfi, knots = seq(0, 20, 5))4       0.125 0.015      1.022
Ns(age, knots = seq(50, 80, 10))1     7.340 1.120     48.117
Ns(age, knots = seq(50, 80, 10))2   702.868 1.940 254610.105
Ns(age, knots = seq(50, 80, 10))3    21.053 4.747     93.374
lex.CstNorm                           1.026 0.270      3.902
lex.CstMac                            4.438 1.842     10.690
lex.CstMic:alloInt                    0.594 0.204      1.727
lex.CstNorm:alloInt                   0.717 0.142      3.613
lex.CstMac:alloInt                    0.206 0.058      0.733
```

For a complete description of all transitions in the model we also need models for the transitions between albuminuria states; we will use different models for deterioration and improvement of albuminuria status:

```
> 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
> round(ci.exp(det), 3)
                                  exp(Est.)  2.5%  97.5%
(Intercept)                           0.055 0.028  0.109
Ns(tfi, knots = seq(0, 20, 5))1       0.635 0.243  1.659
Ns(tfi, knots = seq(0, 20, 5))2       0.270 0.077  0.947
Ns(tfi, knots = seq(0, 20, 5))3       0.278 0.045  1.739
Ns(tfi, knots = seq(0, 20, 5))4       0.220 0.064  0.757
Ns(age, knots = seq(50, 80, 10))1     1.988 0.841  4.702
Ns(age, knots = seq(50, 80, 10))2     3.548 0.939 13.405
Ns(age, knots = seq(50, 80, 10))3     2.692 0.773  9.382
lex.CstNorm                           0.634 0.296  1.358
lex.CstMic:alloInt                    0.510 0.305  0.850
lex.CstNorm:alloInt                   2.044 0.925  4.517
> 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(imp), 3)
                                  exp(Est.)  2.5% 97.5%
(Intercept)                           0.108 0.062 0.187
Ns(tfi, knots = seq(0, 20, 5))1       0.239 0.074 0.767
Ns(tfi, knots = seq(0, 20, 5))2       0.067 0.011 0.420
Ns(tfi, knots = seq(0, 20, 5))3       0.046 0.008 0.273
Ns(tfi, knots = seq(0, 20, 5))4       0.176 0.034 0.913
Ns(age, knots = seq(50, 80, 10))1     0.841 0.289 2.453
Ns(age, knots = seq(50, 80, 10))2     0.340 0.068 1.712
Ns(age, knots = seq(50, 80, 10))3     0.571 0.069 4.715
lex.CstMac                            2.689 1.383 5.230
lex.CstMic:alloInt                    1.902 1.170 3.093
lex.CstMac:alloInt                    0.748 0.304 1.841
```

For the two deterioration rates (`Norm`→`Mic` and `Mic`→`Mac`) the effects of the intervention is in different directions. This is also the case for the improvement rates (`Mic`→`Norm` and `Mac`→`Mic`):

```
> Wald(det, subset="allo")
```

```
     Chisq       d.f.          P
9.76430733 2.00000000 0.00758067
> Wald(imp, subset="allo")
     Chisq       d.f.          P
7.12765757 2.00000000 0.02833015
> Wald(det, subset="allo", ctr.mat = rbind(c(1,-1)))
      Chisq        d.f.           P
8.299873983 1.000000000 0.003964783
> Wald(imp, subset="allo", ctr.mat = rbind(c(1,-1)))
    Chisq      d.f.         P
3.1991556 1.0000000 0.0736763
```

We see that the Wald-tests based on the parameters are all (almost) significant, so not only are there significant allocation effects, but they are also different within each direction.

However, we are mostly interested in how the rates impact the probabilities of being in different states as a function of time, not in the individual rates between albuminuria states.

7.3.2 Simulation of state probabilities

We have fitted statistical models for all transitions, a model for each of the two sets of cause specific mortality rates, and models for transitions in either direction between albuminuria states. All models include smooth effects of time since study entry (`tfi`) and current age (`age`), as well as state-specific effects of the treatment allocation.

We can therefore assess the probability of being in each of the states at a given time after entry to the study, separately for the the two intervention groups. These probabilities depend on the age at entry to the study because current age (`age`) and time since entry, (`tfi`) are both in the model. They also depend on the treatment allocation.

The state probabilities can be approached in (at least) two different ways, defined by how the initial population from which we simulate is defined:

- Use a population with the same covariate distribution as the entire study population, and compute the state probabilities for this. This would be an approximation to the results from the Aalen-Johansen estimator of state probabilities.
- Evaluate the probabilities for persons with specific covariate values at entry. This is an approach we could call the *conditional* (on the specific initial values).

In either case, the state probabilities are not trivial to compute, from a practical point of view they can only be computed by simulation.[2]

For simulation of state probabilities we need a data frame of persons, each record indicating a person's initial status. `simLexis` will then simulate individual trajectories through states (what transitions take place when) and produce a simulated cohort of persons in the form of a `Lexis` object. The initial data frame should be a `Lexis` object, but the values of `lex.Xst` and `lex.dur` need not be given, since these will be simulated. The initial object must have class `Lexis`, in order to identify the variables that are time scales, and which of these that are defined as time since entry into a given state. The reason that it should be a `Lexis` object is that the time scale variables must be updated during the simulation process.

Of course the object should contain all variables in any of the models fro the transition rates. Hence, we first construct a cohort with the same covariates as the entire study for each of the allocation groups. The intention is to simulate what would have happened if the entire Steno2 population were assigned to either `Int` or `Conv`:

```
> ini <- L2[,c("per", "age", "tfi")]
> ini <- rbind(transform(ini, allo = "Int"),
+              transform(ini, allo = "Conv")) %>%
+        mutate(lex.Cst = factor("Mic", levels = levels(L4)),
+                  allo = factor(allo))
> str(ini)
```

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

```
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 ...
 $ allo   : Factor w/ 2 levels "Conv","Int": 2 2 2 2 2 2 2 2 2 2 ...
 $ lex.Cst: Factor w/ 5 levels "Mic","Norm","Mac",..: 1 1 1 1 1 1 1 1 1 1 ...
 - 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: NULL
```

CODE EXPLAINED: The `ini` data frame should contain the variables that are in any of the models for the transitions; in this case `tfi`, `age` and `lex.Cst`. But also all of the timescale variables from the `Lexis` object used for fitting the models. Finally, it should also contain `lex.Cst` even if this variable were not in any of the models.

We take the `ini` as the time scale columns of `L2`, the (`Lexis`) data frame with one record per person. This is then duplicated with each half assigned one of the two levels of `allo`, `Int`, respectively `Conv`, and finally `lex.Cst` is set to `Mic`, but defined as a factor with all possible states as levels.

These will be the initial values in the cohort we follow through states. `allo` will remain the same during follow-up, but `lex.Cst`, `age` and `tfi` will change during follow-up—the two latter will change deterministically; they are time scales.

Note that we could not have used `mutate` here; `mutate` does not have `Lexis` method, here we are using `transform.Lexis` and `rbind.Lexis`, because `ini` is a `Lexis` object.

We also need a specification of what transitions the rate models refer to, since the simulated transitions will be using predictions from these models. This is specified in a list of lists:

```
> Tr <- list(Norm = list(Mic = det,
+                        "D(oth)" = mo,
+                        "D(CVD)" = mC),
+            Mic = list(Mac = det,
+                       Norm = imp,
+                        "D(oth)" = mo,
+                        "D(CVD)" = mC),
+            Mac = list(Mic = imp,
+                        "D(oth)" = mo,
+                        "D(CVD)" = mC))
```

For example, the object `Tr$Norm$Mic` is a 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 **??**. Note that some of the entries in `Tr` point to the same model; all the models fitted were actually joint models for more than one transition. The models all include an effect of `lex.Cst`, the indicator of the current state; the state *from* which the transition goes.

7.3.3 Using the Steno2 population

First we simulate transitions from a large cohort that looks like the study population, say 20 copies of each person in the original data set (well, 40 actually, `ini` contains two copies of each person; one assumed allocated to `Conv`entional treatment and one to `Int`ensive treatment)—see `?simLexis`. So we will simulate histories of $160 \times 2 \times 50 = 16000$ persons:

```
> set.seed(181783)
> system.time(
+ Sorg <- simLexis(Tr = Tr,  # models for each transition
+                 init = ini, # cohort of starters
+                    N = 50,  # how many copies of each person in ini
+              t.range = 21,  # when do we censor follow-up
+                n.int = 85)  # how many intervals for evaluating rates
+              )
```

```
   user  system elapsed
  59.98    7.11   67.17
```

```
> summary(Sorg)
```

```
Transitions:
     To
From    Mic Norm  Mac D(oth) D(CVD)  Records:  Events: Risk time:  Persons:
  Mic  3267 7222 6651   2944   1296     21380    18113  141932.98     16000
  Norm 3442 1878    0   1301    601      7222     5344   60628.02      6643
  Mac  1938    0 1544   1236   1933      6651     5107   40701.84      6087
  Sum  8647 9100 8195   5481   3830     35253    28564  243262.85     16000
```

```
> save(Sorg, file = "Sorg.rda")
```

```
> load(file = "Sorg.rda", v = TRUE)
```

```
Loading objects:
  Sorg
```

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

```
> levels(Sorg)
[1] "Mic"    "Norm"   "Mac"    "D(oth)" "D(CVD)"
```

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

```
Transitions:
     To
From   Norm  Mic  Mac D(CVD) D(oth)  Records:  Events: Risk time:  Persons:
  Norm 1878 3442    0   601   1301      7222     5344   60628.02      6643
  Mic  7222 3267 6651  1296   2944     21380    18113  141932.98     16000
  Mac     0 1938 1544  1933   1236      6651     5107   40701.84      6087
  Sum  9100 8647 8195  3830   5481     35253    28564  243262.85     16000
```

For illustration we show the simulated records for 5 select persons:

```
> 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   12.29     Mic  D(oth)  Int 2014.326
     29 1993.33 61.05  0.00    6.38     Mic     Mac  Int 2014.326
     29 1999.70 67.43  6.38   12.26     Mac  D(oth)  Int 2014.326
     30 1993.33 61.05  0.00   18.74     Mic  D(oth)  Int 2014.326
     31 1993.33 61.05  0.00    6.48     Mic    Norm  Int 2014.326
     31 1999.81 67.53  6.48    5.20    Norm     Mic  Int 2014.326
     31 2005.01 72.73 11.68    9.32     Mic     Mic  Int 2014.326
     32 1993.33 61.05  0.00    0.86     Mic     Mac  Int 2014.326
     32 1994.19 61.91  0.86    4.37     Mac     Mic  Int 2014.326
     32 1998.56 66.28  5.23   15.77     Mic     Mic  Int 2014.326
```

We see that there is a considerable difference between persons in the number of records simulated:

```
> addmargins(table(table(Sorg$lex.id)))
    1     2     3     4     5     6     7     8   Sum
 4252  6845  2909  1533   334   111    12     4 16000
```

We count how many of the original 3200 persons are in each of the states at each of a prespecified set of times; this is done by the function nState:

```
> system.time(
+ Nst <- nState(Sorg,
+                at = seq(0, 20, 0.1),
+               from = 0,
+         time.scale = "tfi"))
   user  system elapsed
  16.42    0.42   16.84
```

simLex

```
> str(Nst)
```

```
 'table' int [1:201, 1:5] 0 209 403 611 799 992 1175 1344 1480 1630 ...
 - attr(*, "dimnames")=List of 2
  ..$ when : chr [1:201] "0" "0.1" "0.2" "0.3" ...
  ..$ State: chr [1:5] "Norm" "Mic" "Mac" "D(CVD)" ...
```

```
> head(Nst)
```

```
     State
when   Norm   Mic   Mac D(CVD) D(oth)
  0       0 16000     0      0      0
  0.1   209 15710    75      6      0
  0.2   403 15433   154     10      0
  0.3   611 15148   227     14      0
  0.4   799 14882   299     19      1
  0.5   992 14606   378     23      1
```

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

```
> Nconv<- nState(subset(Sorg, allo == "Conv"),
+                at = seq(0, 20, 0.1),
+               from = 0,
+         time.scale = "tfi")
> head(Nconv)
```

```
     State
when  Norm  Mic  Mac D(CVD) D(oth)
  0      0 8000    0      0      0
  0.1   79 7869   48      4      0
  0.2  146 7748   99      7      0
  0.3  209 7638  144      9      0
  0.4  280 7518  188     13      1
  0.5  346 7400  237     16      1
```

```
> Nint <- nState(subset(Sorg, allo == "Int"),
+                at = seq(0, 20, 0.1),
+               from = 0,
+         time.scale = "tfi")
> head(Nint)
```

```
     State
when  Norm  Mic  Mac D(CVD) D(oth)
  0      0 8000    0      0      0
  0.1  130 7841   27      2      0
  0.2  257 7685   55      3      0
  0.3  402 7510   83      5      0
  0.4  519 7364  111      6      0
  0.5  646 7206  141      7      0
```

If we divide each of these by 3200, we get the probabilities of being in each if the states at the different times since entry.

If we want the cumulated state probabilities over states we can derive these by `pState`, that yields a matrix with the cumulative state probabilities. This requires an ordering of that states; this is why we ordered the states in the desired order:

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

```
 'pState' num [1:201, 1:5] 0 0.0163 0.0321 0.0503 0.0649 ...
 - attr(*, "dimnames")=List of 2
  ..$ when : chr [1:201] "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   0.000000 1.000000 1.000000      1      1
  0.1 0.016250 0.996375 0.999750      1      1
  0.2 0.032125 0.992750 0.999625      1      1
  0.3 0.050250 0.989000 0.999375      1      1
  0.4 0.064875 0.985375 0.999250      1      1
  0.5 0.080750 0.981500 0.999125      1      1
```

CODE EXPLAINED: For each time point (row of nState result) pState generates the cumulative sum of the state probabilities; this is just a convenience for plotting the stacked state probabilities.

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 the states are already in the order we want, so the default of 1:n is used):

```
> par(mfrow = c(1,2), mar=c(3,3,2,2))
> # we previously assigned this vector of colors - this is just a reminder
> # clr <- c("forestgreen", "orange", "red", "blue", gray(0.4))
> #
> plot(Pconv, col = clr, xlim = c(0, 19.5))
> # the survival curve
> lines(as.numeric(rownames(Pconv)), Pconv[,"Mac"], lwd = 2, col = "white")
> # state names
> text(rownames(Pconv)[150],
+      Pconv[150,] - diff(c(0, Pconv[150,]))/2,
+      colnames(Pconv),
+      col = "white")
> #
> plot(Pint, col = clr, xlim = c(19.5, 0))
> # the survival curve
> lines(as.numeric(rownames(Pint)), Pint[,"Mac"], lwd = 2, col = "white")
> # state names
> text(rownames(Pint)[150],
+      Pint[150,] - diff(c(0, Pint[150,]))/2,
+      colnames(Pint),
+      col = "white")
> #
> mtext(c("Conv","Int"), side = 3, at = c(1,3)/4, outer = TRUE, line = -2)
```

CODE EXPLAINED: Remember that we put the colors in a vector clr so that they can be used a several different places, it is then easier to change colors should we want so. The plot is really plot.pState since Pconv is of class pState; it plots the state probabilities cumulated over states (which is what is in Pconv). The column "Mac" is the sum of the state probabilities for Norm, Mic and Mac, that is the probability of being alive. The rownames of Pconv is the time, and for the 150^{th} time point we have the cumulated state probabilities in Pconv[150,]; these are used to compute the midpoints of each of the areas at this time.

The same is repeated for the state probabilities for the intervention group, Pint (with reversed x-axis), and finally mtext annotates the two panels,

The plot 7.2 is however 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 into the study. The plot is only partially relevant for showing the intervention effect, the absolute sizes of the state probabilities are irrelevant, since they refer to the particular distribution of age (and other possible determinants) in the Steno2 patient population.

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 deceptive terms such as "population averaged".

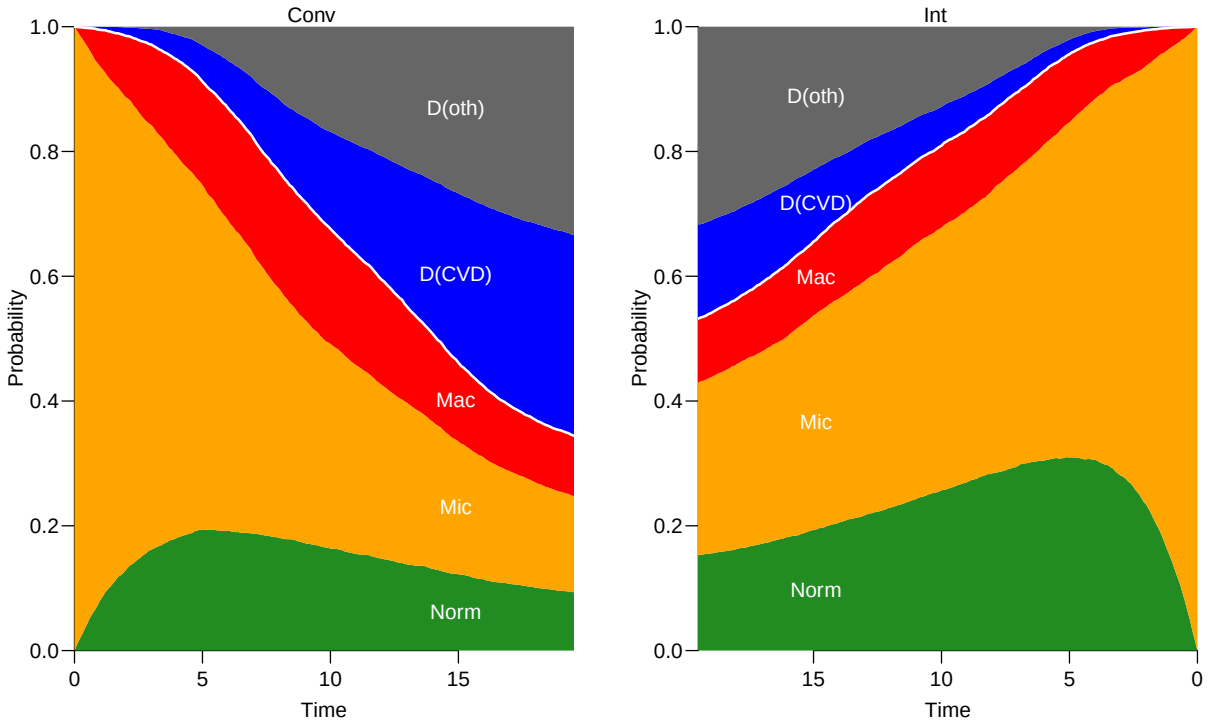


Figure 7.2: *State probabilities for the two intervention groups, for populations of the same structure w.r.t. age as the original* total *Steno2 population. Clearly, the main difference is that the intervention group has a much smaller CVD-mortality compared to the conventional group.* `../graph/simLex-pStates`

7.3.4 Age-stratified initial data

Therefore, it would be more relevant to show the results for a homogeneous population of persons that enter at a few select ages.

This would just require a different `init` data frame:

```
> ini <- S4[1:6,c(timeScales(S4), "lex.dur", "lex.Cst", "allo")]
> ini[,"per"]     <- 1993 # not used but it is a time scale in S4
> ini[,"ain"]     <- # entry age, not changed in the simulation
+ ini[,"age"]     <- rep(seq(45,65,10), 2) # current age
> 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 = 3))
> ini
  per age tfi lex.dur lex.Cst allo ain
 1993  45   0    0.12     Mic  Int  45
 1993  55   0    0.13     Mic  Int  55
 1993  65   0    0.25     Mic  Int  65
 1993  45   0    0.25     Mic Conv  45
 1993  55   0    0.25     Mic Conv  55
 1993  65   0    0.25     Mic Conv  65
```

CODE EXPLAINED: We make a copy of the first 6 rows of `S4`, to make sure that the relevant `Lexis` attributes necessary are present in the `ini` object. The rest is just replacing the values with the desired ones for the simulation.

We must enter the variable `lex.Cst` as a factor with the same levels as states in the `Lexis` object `S4`, in the order we want the states when reporting results.

The predictor `allo` must also be entered as a factor, otherwise it is not possible to compute predictions from the models where `allo` were included as a factor.

For each of these combinations of age (at entry) and treatment allocation we will simulate 5000 persons from each age / sex combination (note that we are using the same transition rates as before, from the models in Tr):

```
> set.seed(181783)
> system.time(
+ Sdef <- simLexis(Tr = Tr,    # models for each transition
+                 init = ini,  # cohort of starters
+                    N = 5000, # how many copies of each person in ini
+              t.range = 21,    # how long should we simulate before censoring
+                n.int = 85))  # no points to evaluate rates
   user  system elapsed
 113.42   13.03  126.58
> save(Sdef, file = "Sdef.rda")
```

Once simulated, we can take a look at the simulated cohort:

```
> summary(Sdef)
Transitions:
     To
From    Norm   Mic    Mac D(CVD) D(oth)  Records:  Events: Risk time:  Persons:
  Norm  3639  6593      0   1065   2545     13842    10203  115595.65     12658
  Mic  13842  6129  12134   2527   5600     40232    34103  262698.91     30000
  Mac      0  3639   2706   3343   2446     12134     9428   71434.07     11110
  Sum  17481 16361  14840   6935  10591     66208    53734  449728.62     30000
> 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    7.23     Mic    Norm  Int  45 2014
      1 2000.23 52.23 7.23   13.77    Norm    Norm  Int  45 2014
      2 1993.00 45.00 0.00    2.67     Mic    Norm  Int  45 2014
      2 1995.67 47.67 2.67   18.33    Norm    Norm  Int  45 2014
      3 1993.00 45.00 0.00    7.92     Mic  D(oth)  Int  45 2014
      4 1993.00 45.00 0.00    1.66     Mic    Norm  Int  45 2014
      4 1994.66 46.66 1.66   19.34    Norm    Norm  Int  45 2014
```

We now repeat the graph above by generating a pState for each of the 6 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 5000    0      0      0
  0.1  124 4862   14      0      0
  0.2  261 4717   22      0      0
  0.3  369 4597   34      0      0
  0.4  479 4474   47      0      0
  0.5  585 4359   56      0      0
> head(pState(P45i))
     State
when    Norm    Mic Mac D(CVD) D(oth)
  0   0.0000 1.0000   1      1      1
  0.1 0.0248 0.9972   1      1      1
  0.2 0.0522 0.9956   1      1      1
  0.3 0.0738 0.9932   1      1      1
  0.4 0.0958 0.9906   1      1      1
  0.5 0.1170 0.9888   1      1      1
```

This should then be repeated for all three ages at enrollment and the two allocations, but now we only store the (cumulated) state probabilities:

```
> P45c <- pState(nState(subset(Sdef, ain == 45 & allo == "Conv"),
+                at = seq(0, 20, 0.1),
+              from = 0,
+        time.scale = "tfi"))
> P45i <- pState(nState(subset(Sdef, ain == 45 & allo == "Int"),
+                at = seq(0, 20, 0.1),
+              from = 0,
+        time.scale = "tfi"))
> P55c <- pState(nState(subset(Sdef, ain == 55 & allo == "Conv"),
+                at = seq(0, 20, 0.1),
+              from = 0,
+        time.scale = "tfi"))
> P55i <- pState(nState(subset(Sdef, ain == 55 & allo == "Int"),
+                at = seq(0, 20, 0.1),
+              from = 0,
+        time.scale = "tfi"))
> P65c <- pState(nState(subset(Sdef, ain == 65 & allo == "Conv"),
+                at = seq(0, 20, 0.1),
+              from = 0,
+        time.scale = "tfi"))
> P65i <- pState(nState(subset(Sdef, ain == 65 & allo == "Int"),
+                at = seq(0, 20, 0.1),
+              from = 0,
+        time.scale = "tfi"))
```

We can then plot these in a multiple lay-out:

```
> par(mfrow = c(3,2),
+       mar = c(1,2,0,0),
+       oma = c(3,3,1,0),
+       mgp = c(3,1,0)/1.6)
> plot(P45c, col = clr, xlim = c(0,20))
> plot(P45i, col = clr, xlim = c(20,0))
> plot(P55c, col = clr, xlim = c(0,20))
> plot(P55i, col = clr, xlim = c(20,0))
> plot(P65c, col = clr, xlim = c(0,20))
> plot(P65i, col = clr, xlim = c(20,0))
> mtext(c("Conv","Int"), side = 3, at = c(1,3)/4, outer = TRUE, line = 0)
> mtext(paste(seq(45,65,10)), side = 2, at = (3:1*2-1)/6,
+       outer = TRUE, line = 1, las = 1)
```

This approach entails a lot of hard-coding; we would like to be able to easily get a plot with any set of ages at entry. To this end it is more convenient to collect the state probabilities in an array:

```
> (ain <- seq(45, 65, 10))
[1] 45 55 65
> (agr <- levels(S4$allo))
[1] "Conv" "Int"
> pdef <- NArray(c(list(ain = ain,
+                       allo = agr),
+                  dimnames(P45i)))
> str(pdef)
 logi [1:3, 1:2, 1:201, 1:5] NA NA NA NA NA NA ...
 - attr(*, "dimnames")=List of 4
  ..$ ain  : chr [1:3] "45" "55" "65"
  ..$ allo : chr [1:2] "Conv" "Int"
  ..$ when : chr [1:201] "0" "0.1" "0.2" "0.3" ...
  ..$ State: chr [1:5] "Norm" "Mic" "Mac" "D(CVD)" ...
```

CODE EXPLAINED: `NArray` defines an empty array (full of `NA`s), classified by the vectors given in a `list` argument. Note that the way lists are concatenated is by "`c`".

We lose the `pState` class of the results, so we resort to the `mat2pol` function that stacks probabilities (well any set of columns of a matrix) and plots them, so we simply take the result from `nState` and divide by the number in the initial state (`Mic`) using `sweep`:

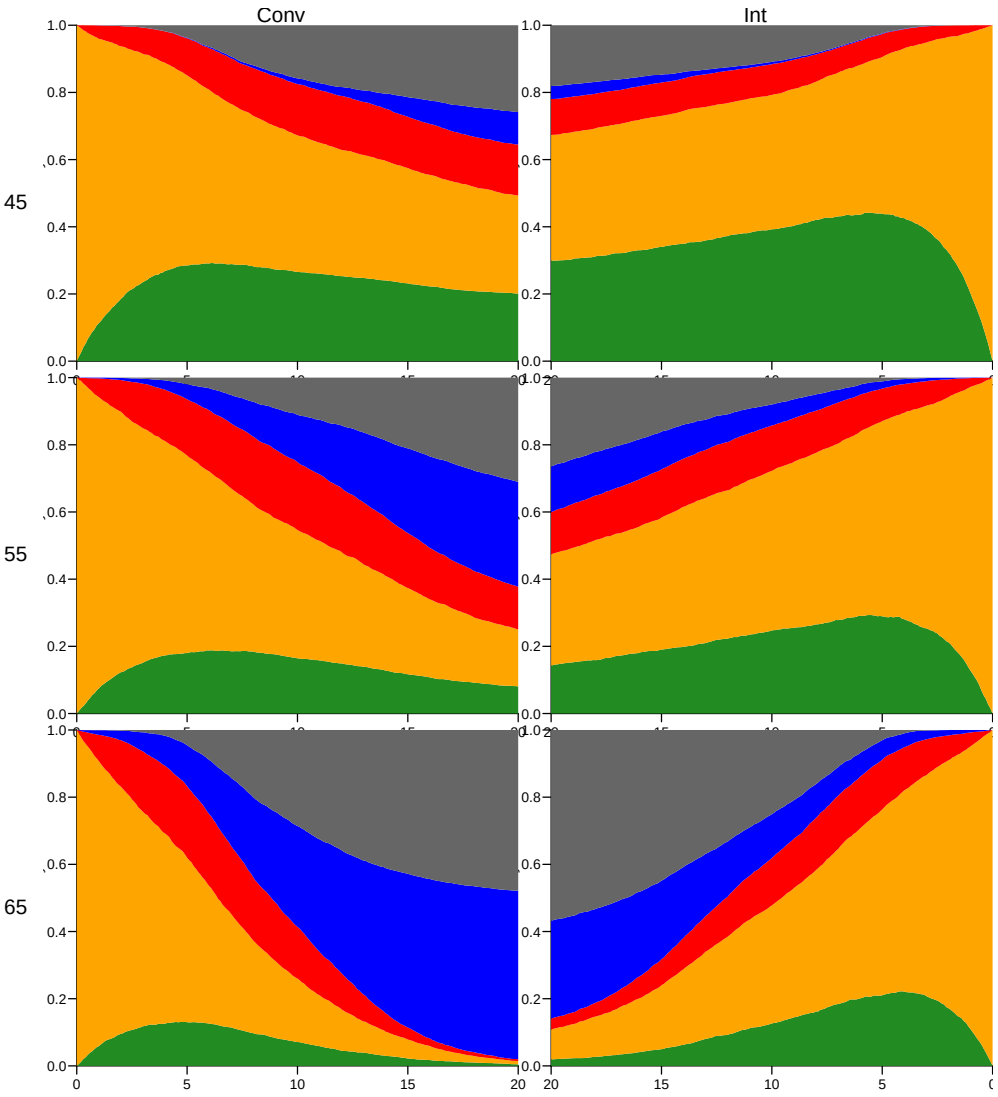


Figure 7.3: *Predicted probabilities of being in each of the states for persons aged 45, 55 and 65 at entry , separately for the two intervention groups. Clearly the graph needs some doctoring.* ../graph/simLex-panel5

```
> for(aa in ain)
+ for(gg in agr)
+    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 [1:3, 1:2, 1:201, 1:5] 0 0 0 0 0 0 0.014 0.0082 0.0062 0.0248 ...
 - attr(*, "dimnames")=List of 4
  ..$ ain  : chr [1:3] "45" "55" "65"
  ..$ allo : chr [1:2] "Conv" "Int"
  ..$ when : chr [1:201] "0" "0.1" "0.2" "0.3" ...
  ..$ State: chr [1:5] "Norm" "Mic" "Mac" "D(CVD)" ...
```

CODE EXPLAINED: Each iteration of the loop over `ain` (age at entry) and `all` (`allocation`) computes the number of person in each of the 5 states at a set of times (3rd dimension of `pdef`).

```
> # panel3, fig = TRUE, width = 8, height = 9
> ain <- seq(45, 65, 10)
> par(mfrow = c(length(ain), 2),
+       mar = c(1,0,1,0)/20,
+       oma = c(3,5,2,3),
+       mgp = c(3,1,0) / 1.6,
+       xaxt = "n", yaxt = "n",
+       xaxs = "i", yaxs = "i", las = 1)
> for(aa in ain)
+    {
+ mat2pol(pdef[paste(aa),"Conv",,], col = clr, xlim = c(0,20))
+ axis(side = 2, yaxt = "s", at = 0:10/10, labels = NA )
+ axis(side = 2, yaxt = "s", at = (1:5*2-1)/10)
+ axis(side = 4, yaxt = "s", at = 0:10/10,
+      labels = NA, tcl = 0.4, col="white")
+ axis(side = 4, yaxt = "s", at = 0:20/20,
+      labels = NA, tcl = 0.3, col="white")
+ axis(side = 4, yaxt = "s", at = 0:100/100,
+      labels = NA, tcl = 0.2, col="white")
+ if(aa == ain[length(ain)]) axis(side = 1, xaxt = "s")
+
+ mat2pol(pdef[paste(aa),"Int" ,,], col = clr, xlim = c(20,0), yaxt = "n")
+ axis(side = 2, yaxt = "s", at = 0:10/10,
+      labels = NA, tcl = 0.4, col="white")
+ axis(side = 2, yaxt = "s", at = 0:20/20,
+      labels = NA, tcl = 0.3, col="white")
+ axis(side = 2, yaxt = "s", at = 0:100/100,
+      labels = NA, tcl = 0.2, col="white")
+ axis(side = 4, yaxt = "s", at = 0:10/10, labels = NA )
+ axis(side = 4, yaxt = "s", at = (1:5*2-1)/10)
+ if(aa == ain[length(ain)]) axis(side = 1, xaxt = "s")
+    }
> mtext(c("Conv","Int"), side = 3, at = c(1,3)/4, outer = TRUE, line = 0,
+       cex = 0.67)
> mtext(paste("Age\n",ain),
+       side = 2,
+       at = (length(ain):1 * 2 - 1) / (length(ain) * 2) + 0.1,
+       outer = TRUE, line = 2.5, cex = 0.67)
> mtext("Time since entry (years)", side = 1, line = 3/1.6,
+       outer = TRUE, cex=0.67)
```

7.3.5 Time spent in albuminuria states

We have observation time till almost 22 years, but we have only made predictions of state probabilities in `pdef` till a bit longer than 20 years; `pdef` contains the state probabilities by time since entry for all combinations of age at entry and treatment allocation:

```
> str(pdef)
 num [1:3, 1:2, 1:201, 1:5] 0 0 0 0 0 0 0.014 0.0082 0.0062 0.0248 ...
 - attr(*, "dimnames")=List of 4
  ..$ ain  : chr [1:3] "45" "55" "65"
  ..$ allo : chr [1:2] "Conv" "Int"
  ..$ when : chr [1:201] "0" "0.1" "0.2" "0.3" ...
  ..$ State: chr [1:5] "Norm" "Mic" "Mac" "D(CVD)" ...

> ftable(pdef[,,1:5,], row.vars = c(1,3))
```

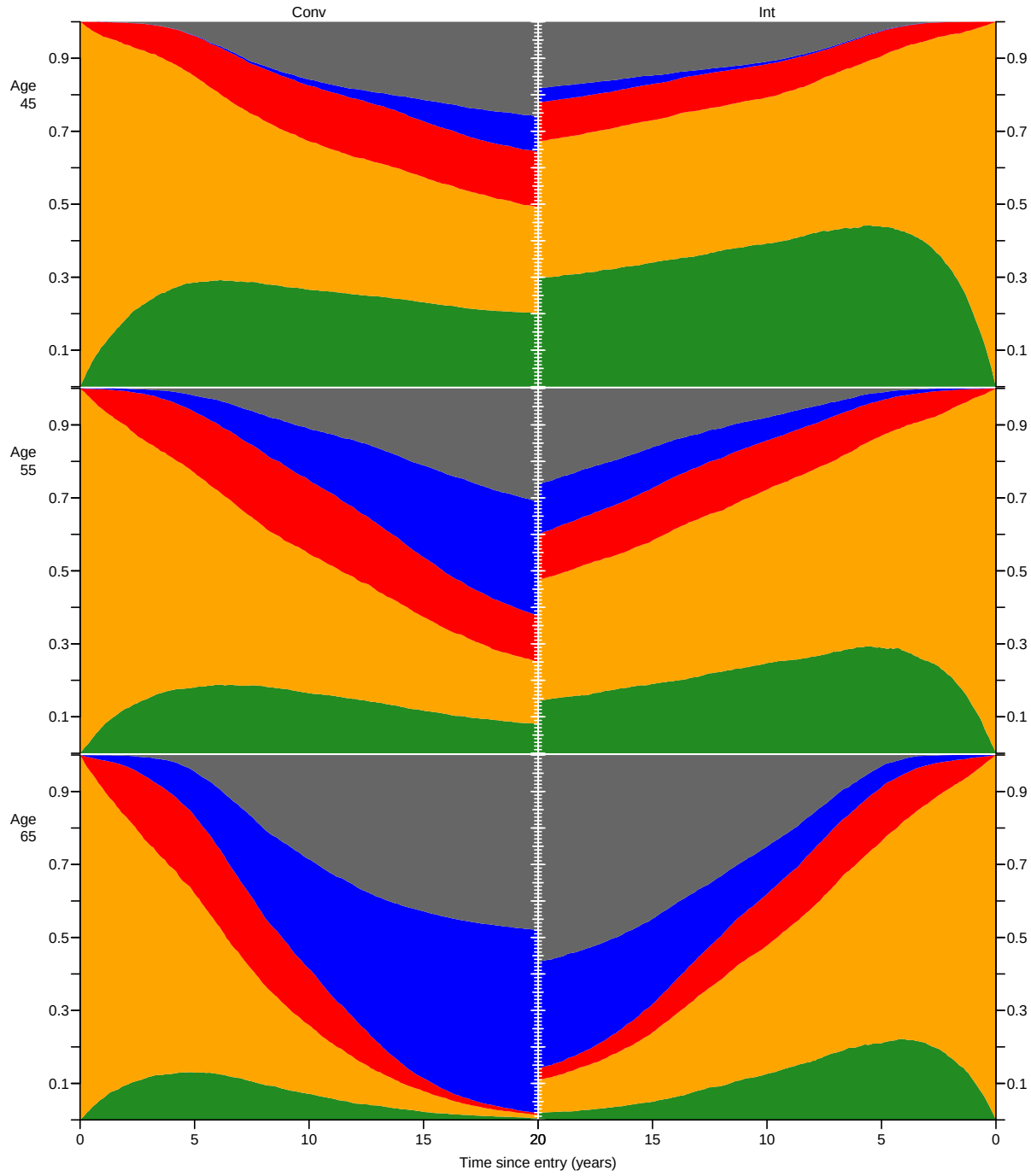


Figure 7.4: *Predicted probabilities of being in each of the states for persons aged 45, 55 and 65 at entry, separately for the two intervention groups. Estimates are based on proportional hazards models with spline effects of time since entry and current age, and hazard ratios depending on treatment allocation (*`Conv`*/*`Int`*), and current microalbuminuria status (current state) (*`Norm`*/*`Mic`*/*`Mac`*).* `../graph/simLex-panel3`

```
         allo    Conv                                  Int
         State   Norm    Mic    Mac D(CVD) D(oth)   Norm    Mic    Mac D(CVD) D(oth)
ain when
45  0          0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000
    0.1        0.0140 0.9816 0.0044 0.0000 0.0000 0.0248 0.9724 0.0028 0.0000 0.0000
    0.2        0.0264 0.9642 0.0094 0.0000 0.0000 0.0522 0.9434 0.0044 0.0000 0.0000
    0.3        0.0418 0.9432 0.0150 0.0000 0.0000 0.0738 0.9194 0.0068 0.0000 0.0000
    0.4        0.0518 0.9302 0.0178 0.0000 0.0002 0.0958 0.8948 0.0094 0.0000 0.0000
55  0          0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000
    0.1        0.0082 0.9858 0.0056 0.0004 0.0000 0.0154 0.9808 0.0034 0.0004 0.0000
    0.2        0.0162 0.9734 0.0096 0.0008 0.0000 0.0270 0.9666 0.0060 0.0004 0.0000
    0.3        0.0222 0.9600 0.0166 0.0012 0.0000 0.0430 0.9480 0.0084 0.0006 0.0000
    0.4        0.0296 0.9472 0.0220 0.0012 0.0000 0.0538 0.9334 0.0120 0.0008 0.0000
65  0          0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000
    0.1        0.0062 0.9838 0.0086 0.0012 0.0002 0.0138 0.9814 0.0042 0.0006 0.0000
    0.2        0.0126 0.9650 0.0190 0.0032 0.0002 0.0266 0.9634 0.0084 0.0016 0.0000
    0.3        0.0206 0.9466 0.0276 0.0048 0.0004 0.0394 0.9462 0.0118 0.0026 0.0000
    0.4        0.0256 0.9334 0.0350 0.0056 0.0004 0.0510 0.9300 0.0154 0.0036 0.0000
```

CODE EXPLAINED: The `str` shows that `pdef` is a 4-way array; the `ftable` prints the first 5 levels of the 3rd dimension, with the 1st and 3rd as the row classification in a flat `table` layout.

7.3.5.1 Using estimated probabilities

We may want to compare groups by the expected time spent in the normoalbuminuric state during the first, say, 20 years. The expected time in a state is simply the time-integral of the probabilities, so we can easily compute it from `pdef`; each probability in `pdef` represents is computed at a specific time, the times being 0.1 year apart, so we just take the midpoint of the probabilities at the ends of each interval and multiply by the interval width:

```
> mid <- function(x) x[-1] - diff(x) / 2
> pmid <- apply(pdef, c(1,2,4), mid)
> str(pmid)
 num [1:200, 1:3, 1:2, 1:5] 0.007 0.0202 0.0341 0.0468 0.059 ...
 - attr(*, "dimnames")=List of 4
  ..$      : chr [1:200] "0.1" "0.2" "0.3" "0.4" ...
  ..$ ain  : chr [1:3] "45" "55" "65"
  ..$ allo : chr [1:2] "Conv" "Int"
  ..$ State: chr [1:5] "Norm" "Mic" "Mac" "D(CVD)" ...

> pyr <- apply(pmid[1:200,,,], 2:4, sum) * 0.1
> str(pyr)

 num [1:3, 1:2, 1:5] 4.69 2.72 1.27 7.11 4.31 ...
 - attr(*, "dimnames")=List of 3
  ..$ ain  : chr [1:3] "45" "55" "65"
  ..$ allo : chr [1:2] "Conv" "Int"
  ..$ State: chr [1:5] "Norm" "Mic" "Mac" "D(CVD)" ...

> round(ftable(pyr, col.vars = 3:2), 1)

    State Norm       Mic       Mac      D(CVD)      D(oth)
    allo  Conv  Int Conv  Int Conv  Int   Conv  Int   Conv  Int
ain
45         4.7  7.1  9.5  9.2  2.5  1.6    0.6  0.3    2.7  1.8
55         2.7  4.3  8.7 10.2  3.1  2.3    3.0  1.3    2.4  1.9
65         1.3  2.3  5.8  7.9  2.2  2.0    5.7  2.8    5.0  5.0
```

CODE EXPLAINED: The `mid` function will return the midpoints between successive values in a vector, a result that is one shorter than the argument. The `apply` returns an array classified by dimensions 1, 2 and 4 of `pdef` *and* (as the first dimension) the result of the `mid` function applied to the remaining dimensions of `pdef` (in this case only the 3rd).

The second apply adds the first 200 entries of the first dimension of pmid, which, when multiplied by the interval length give the integral of the state probabilities over the first 20 years (200 intervals) of follow-up.

Note that the apply function automatically keeps track of the three dimensions by which we want the result classified.

These numbers are the expected time (in years) spent in each state during the first 20 years after enrollment; we see that the intervention group spend far more time in Norm than do the conventional group, regardless of the age at entry.

The time spent in the two dead states are not really interpretable, it would be something like the number of years (during the first 20 years after enrollment) lost to each of the causes. We see that the most dramatic differences are for the CVD deaths.

We can also take a look at the differences in sojourn times between the two intervention groups; this is the advantage of having results stored in an array—it is quite easy to do simple manipulations across multiple dimensions:

```
> round(pyr[,"Int",] - pyr[,"Conv",], 1)
    State
ain  Norm  Mic  Mac D(CVD) D(oth)
  45  2.4 -0.3 -1.0   -0.4   -0.8
  55  1.6  1.5 -0.9   -1.7   -0.6
  65  1.0  2.0 -0.2   -2.9    0.0
```

7.3.5.2 Using simulated sojourn times

We also have the possibility to compute the sojourn times directly from the simulated object; this however will not include time spent in the absorbing states, as this is not part of the simulated follow-up:

```
> tpyr <- with(Sdef, tapply(lex.dur,
+                           list(ain, allo, lex.Cst),
+                           sum)) / 2000
> round(ftable(tpyr, col.vars = 3:2), 1)
    Norm       Mic       Mac      D(CVD)      D(oth)
    Conv  Int Conv  Int Conv  Int   Conv  Int   Conv  Int

45  12.2 18.5 24.4 24.0  6.7  4.2     NA   NA     NA   NA
55   7.0 11.1 22.3 26.4  8.1  6.0     NA   NA     NA   NA
65   3.2  5.8 14.5 19.8  5.6  5.1     NA   NA     NA   NA
> round(ftable( pyr, col.vars = 3:2), 1)
    State Norm       Mic       Mac      D(CVD)      D(oth)
    allo  Conv  Int Conv  Int Conv  Int   Conv  Int   Conv  Int
ain
45         4.7  7.1  9.5  9.2  2.5  1.6    0.6  0.3    2.7  1.8
55         2.7  4.3  8.7 10.2  3.1  2.3    3.0  1.3    2.4  1.9
65         1.3  2.3  5.8  7.9  2.2  2.0    5.7  2.8    5.0  5.0
> round(ftable(tpyr / pyr, col.vars = 3:2), 2)
    Norm       Mic       Mac      D(CVD)      D(oth)
    Conv  Int Conv  Int Conv  Int   Conv  Int   Conv  Int

45  2.61 2.60 2.58 2.60 2.65 2.67     NA   NA     NA   NA
55  2.57 2.58 2.55 2.58 2.60 2.64     NA   NA     NA   NA
65  2.51 2.52 2.50 2.53 2.51 2.54     NA   NA     NA   NA
> round(ftable(tpyr - pyr, col.vars = 3:2), 2)
     Norm         Mic         Mac       D(CVD)       D(oth)
     Conv   Int  Conv   Int  Conv   Int   Conv   Int   Conv   Int

45   7.53 11.41 14.92 14.75  4.19  2.63     NA    NA     NA    NA
55   4.27  6.81 13.53 16.14  5.01  3.73     NA    NA     NA    NA
65   1.92  3.47  8.73 11.99  3.34  3.07     NA    NA     NA    NA
```

Discrepancy to numerical integrals We see that the simulated sojourn times are longer than those computed from the integration of the simulated state probabilities. Suppose namely that we censored all simulated sojourn times immediately after the times where we calculated the state probabilities. Then the simulated probabilities would still be the same, but the simulated sojourn times would be smaller. So we see that the numerical integration gives downward biased sojourn times; the actually simulated sojourn times gives the most correct result.

Time spent in absorbing states The time spent in the dead states during the first 20 years of follow-up is however available from `Sdef` (only the results from the absorbing states are meaningful):

```
> apyr <- with(subset(Sdef, lex.Xst %in% absorbing(Sdef)),
+              tapply(20 - (tfi + lex.dur),
+                     list(ain, allo, lex.Xst),
+                     sum)) / 2000
> round(ftable(pmin(apyr, tpyr, na.rm = TRUE), col.vars = 3:2), 1)
    Norm       Mic       Mac      D(CVD)      D(oth)
    Conv  Int Conv  Int Conv  Int   Conv  Int   Conv  Int

45  12.2 18.5 24.4 24.0  6.7  4.2    1.5  0.6    6.7  4.6
55   7.0 11.1 22.3 26.4  8.1  6.0    7.4  3.3    6.1  4.7
65   3.2  5.8 14.5 19.8  5.6  5.1   14.2  7.1   12.6 12.6
> round(ftable( pyr, col.vars = 3:2), 1)
    State Norm       Mic       Mac      D(CVD)      D(oth)
    allo  Conv  Int Conv  Int Conv  Int   Conv  Int   Conv  Int
ain
45         4.7  7.1  9.5  9.2  2.5  1.6    0.6  0.3    2.7  1.8
55         2.7  4.3  8.7 10.2  3.1  2.3    3.0  1.3    2.4  1.9
65         1.3  2.3  5.8  7.9  2.2  2.0    5.7  2.8    5.0  5.0
```

7.4 State probabilities from the Aalen-Johansen estimator in `survival`

The `survival` package allows estimation of state probabilities by the Aalen-Johansen estimator similar to what we did in competing risks. However, the Aalen-Johansen machinery in the `survival` package is not aligned with the way `Lexis` objects are defined, so a naive application will give wrong results or crash the `survfit` function.

Therefore, a wrapper for the multistate version of `survfit` has been devised, called `AaJ.Lexis`, so we can just do:

```
> AaJ4 <- AaJ.Lexis(L4, time = "tfi")
NOTE: Timescale is tfi
```

The result is an object which is a list that inherits from `survfitms` and `survfit`. Each element of the list are referred to as a slot. The predicted state probabilities from `survfit` are in the slot called `pstate`, and the confidence intervals in the corresponding slots `lower` and `upper`.

```
> class(AaJ4)
[1] "survfitms" "survfit"
> names(AaJ4)
 [1] "n"           "time"        "n.risk"      "n.event"     "n.censor"    "pstate"
 [7] "p0"          "cumhaz"      "std.err"     "sp0"         "logse"       "transitions"
[13] "lower"       "upper"       "conf.type"   "conf.int"    "states"      "type"
[19] "call"
> AaJ4$states
[1] "Mic"    "Norm"   "Mac"    "D(CVD)" "D(oth)"
```

```
> colnames(AaJ4$pstate) <- AaJ4$states
> head(cbind(time = AaJ4$time,
+                    AaJ4$pstate))
           time      Mic     Norm      Mac D(CVD) D(oth)
[1,] 0.05201916 0.99375 0.00000 0.00625      0      0
[2,] 0.05749487 0.98750 0.00625 0.00625      0      0
[3,] 0.06570842 0.98750 0.00625 0.00625      0      0
[4,] 0.09034908 0.98125 0.01250 0.00625      0      0
[5,] 0.09308693 0.98125 0.01250 0.00625      0      0
[6,] 0.10130048 0.98125 0.01250 0.00625      0      0
```

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

```
> mat2pol(AaJ4$pstate,
+         perm = c(2,1,3,5,4),
+            x = AaJ4$time,
+          col = clr)
> lines(AaJ4$time,
+       apply(AaJ4$pstate[,1:3], 1, sum),
+       lwd = 1,
+       col = "White")
```

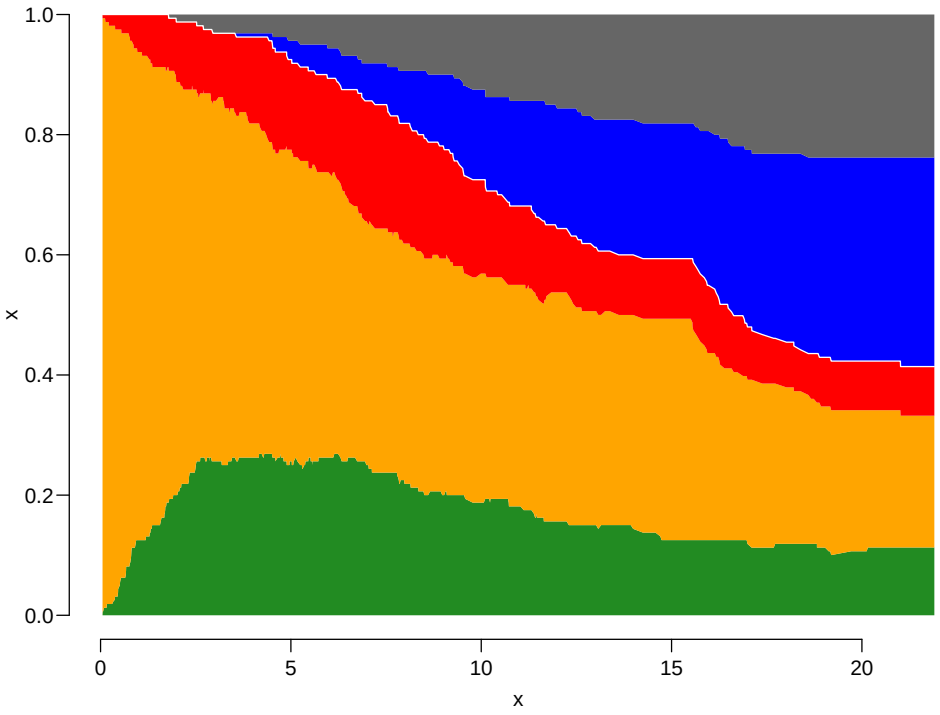


Figure 7.5: *Aalen-Johansen estimator of state probabilities for the total Steno2 study population, disregarding intervention allocation.* ../graph/simLex-AaJ0

CODE EXPLAINED: `mat2pol` plots the columns of the first argument (a matrix) as coloured areas stacked on top of each other, `perm` gives the order in which they are stacked. The `x` argument is the common x-coordinate against which the columns of the first argument is plotted. `col` is a vector of colours used for the areas.

Formally, the Aalen-Johansen estimator is a step function, so instead of just connecting the points we should have made a proper step-function, but this is a minor point with a dataset of this size.

However, we are interested in seeing the results from each of the allocation groups, so we use the `formula` argument to `AaJ.Lexis` (which is just passed on to `Survfit`):

```
> AaJg <- AaJ.Lexis(L4, ~ allo, time = "tfi")
NOTE: Timescale is tfi
```

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

```
> AaJg$strata
allo=Conv  allo=Int
      337       375
```

```
> wh <- rep(substr(names(AaJg$strata), 6, 9), AaJg$strata)
> table(wh)
wh
Conv  Int
 337  375
```

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

```
> par(mfrow = c(1,2), mar=c(3,3,0,0), oma=c(0,0,1,1), mgp = c(3,1,0)/1.6)
> mat2pol(AaJg$pstate[wh=="Conv",],
+         perm = c(2,1,3:5),
+         x = AaJg$time[wh=="Conv"],
+         col = clr, xlim = c(0,21), xaxs = "i", yaxs = "i")
> lines(AaJg$time[wh=="Conv"],
+       apply(AaJg$pstate[,1:3], 1, sum)[wh=="Conv"],
+       lwd = 2, col = "white")
> mat2pol(AaJg$pstate[wh=="Int",],
+         perm = c(2,1,3:5),
+         x = AaJg$time[wh=="Int"],
+         col = clr, xlim = c(21,0), xaxs = "i", yaxs = "i")
> lines(AaJg$time[wh=="Int"],
+       apply(AaJg$pstate[,1:3], 1, sum)[wh=="Int"],
+       lwd = 2, col = "white")
> mtext(c("Conv","Int"), side = 3, at = c(1,3)/4, outer = TRUE, line = 0)
```

7.4.1 Relation to the simulation approach by `simLexis`

The Aalen-Johansen estimators for the two allocation groups can be considered the empirical counterpart of the simulation based estimates in figure 7.2; the state probabilities for a population like the one in the study. But not quite so; the models underlying figure 7.2 are proportional hazards models in the sense that the effects of current age and time since enrollment are assumed proportional between allocation groups with proportionality depending on current state. Moreover, the effect of age is assumed to be the same at all times since enrollment and vice versa. Basically "proportional hazards" just means models without interaction with any of the time scales.

The plots in figure 7.6 are based on separate models for the effect of time from entry (`tfi`) for each transition and allocation, and with no effect of current age or age at entry. So an unspecified interaction between `tfi` and allocation, separately for each transition.

In the Aalen-Johansen approach in `survfit` we get confidence intervals for each of the state probabilities in the slots `lower` and `upper`, but not for sums of subsets of these. Since it is the sums of state probabilities we have shown in the graph, we cannot directly use the confidence intervals for the single state probabilities.

We would also want confidence intervals for areas under the curves—the sojourn times. Neither are available from the Aalen-Johansen approach nor from the simulation approach—at least not in its current state of development.

Both the results in figure 7.2 and figure 7.6 refer to a patient population identical to the study population. Thus to the extent that age also has an effect on the transition rates in the model in figure 7.1 the results will depend on the age-distribution in the study population. The two approaches are not identical, but both answer the question of state probabilities as a function of time on study for a population with the same age-distribution as the Steno2 study population.

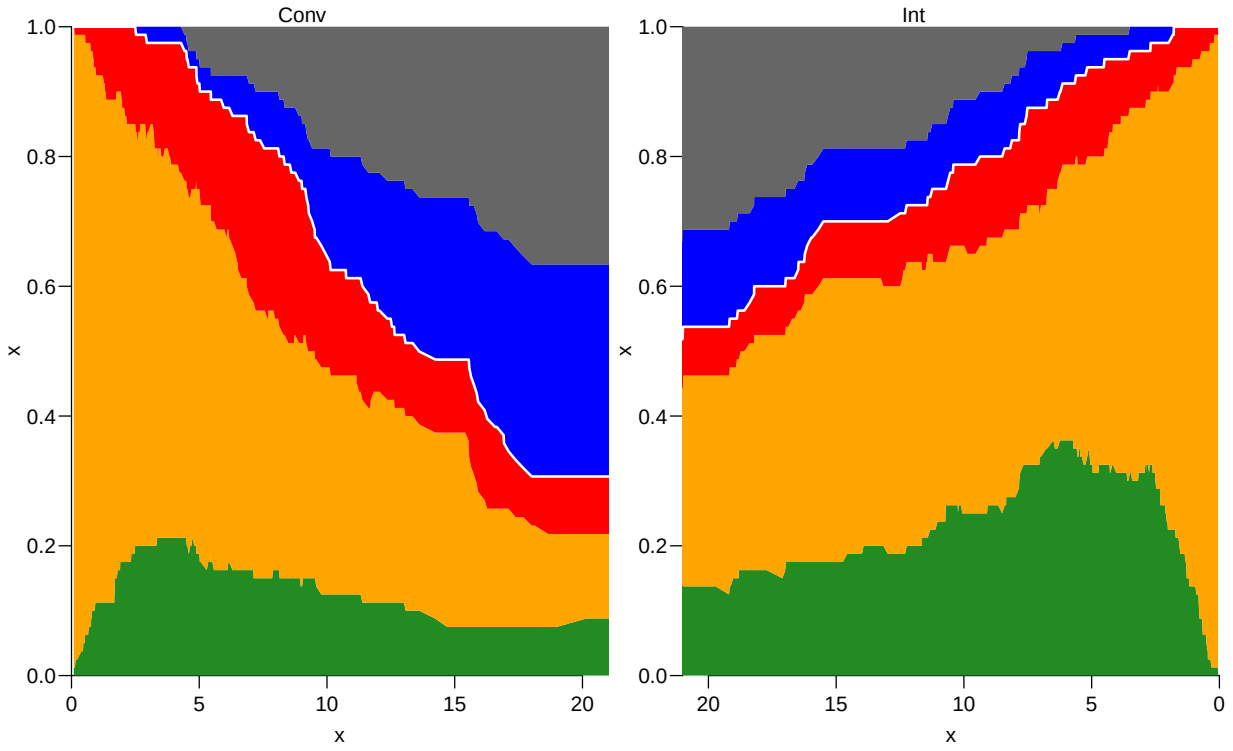


Figure 7.6: *Aalen-Johansen estimator of the state probabilities for the two intervention groups , for the original* total *Steno2 population, subdivided by intervention allocation.* `../graph/simLex-AaJstates`

A parametric counterpart of the Aalen-Johansen estimator would entail fit of separate models for each of the 10 transitions using a spline model for the effect of time since entry and for each transition separate models for each of the two intervention groups `Conv` and `Int` (corresponding to the baseline hazards). The simulation of cohorts would then be done separately for the two groups—they would be quite similar because of the randomization, though. It would still not be *identical* to the Aalen-Johansen analysis, but the basic assumptions about the intensities would be the same:

1. Time since entry is the only covariate for all transitions
2. Separate models for the two randomization groups
3. There is no relationship between models for different transitions

If we instead want to assess how e.g. age influences state probabilities we would fit models with age-effect on the transition rates and use simulation to translate these to the effects on state probabilities such as figure 7.4. This depends both on the chosen models for the transition rates and on the choice of initial cohort to simulate from.

Chapter 8

Multistate packages

. . . now input from `msm`

8.1 The `msm` package

8.1.1 Relationship between `Lexis` and `msm`

The `Lexis` representation of follow-up assumes that times of all transitions between states are known, so a `Lexis` object is a an exact representation of persons entire follow-up through the states. Each record in a `Lexis` object represents a piece of follow-up in a state (`lex.Cst`) of a certain length (`lex.dur`), at the end of which a transition to the next state (`lex.Xst`) occurs.

The `msm` package is unlike the `Lexis` machinery aimed at partially observed multistate models—what is often referred to as "panel data";it is assumed that you only know the state of a person at a finite set of times, but nothing is known about the exact transition times. There is though a provision in the `msm` package to assume that some or all transition times are are known exactly. This will typically be the case with transitions to "Dead" states, but it is required that this is for all transitions to a given state.

8.1.2 Exact transition times?

When states are disease states where transitions times are defined as date of recording of a given diagnosis, it may be argued that some conditions (such as for example bladder cancer) in reality may have occurred some time before the recording date. This is however to some extent a matter of state-definition; using the recorded date of the event as an exact occurrence date means that model is not a model for the occurrence of bladder cancer but for the occurrence of *recorded* bladder cancer. This is essentially just moving the problem to another realm; the relationship between occurrence and recording of bladder cancer is in this way moved outside the realm of the practical modeling, and into the realm of results interpretation.

If we want to use `msm` for data recorded in a `Lexis` object we must restructure it. The `Lexis` representation is as the state in which follow-up time occurs (`lex.Cst`) as well as the next state (`lex.Xst`—which however may be the same as `lex.Cst`).

But if we have a `Lexis` object with, say, n consecutive intervals with the follow-up of a person we must supply to `msm` the states occupied at the beginning of each interval (which will be the "next" state for the previous interval if any), but also the state at the end of the last interval, so a total of $n + 1$ records.

For the first n intervals the time will be the values of the timescale, while the time we need for the last "extra" record is the time of transition, which for time scale `ts` will be `ts + lex.dur`.

However, if we have non-consecutive follow-up intervals for a person, this will have to be done for each set of consecutive intervals within each person.

8.2 Data transformation from `Lexis` to `msm`

The function `Lexis2msm` in the `Epi` package does exactly this job.

```
> library(Epi)
> library(msm)
> library(tidyverse)
```

For illustration we run (a slight extension) of the example code for `mcutlexis` to provide a `Lexis` object with 5 states:

```
> set.seed(1952)
> dd <- data.frame(id = 1:30,
+                  doN = round(runif(30,-30, 0),1), # birth
+                  doE = round(runif(30,  0,20),1), # entry
+                  doX = round(runif(30, 50,60),1), # exit
+                  doD = round(runif(30, 50,60),1), # death
+                  # these are the event times fo A and B
+                  doA = c(NA,21,NA,27,35,NA,52, 5,43,80,
+                          NA,22,56,28,53,NA,51, 5,43,80,
+                          NA,23,NA,33,51,NA,55, 5,43,80),
+                  doB = c(NA,20,NA,53,27,NA, 5,52,34,83,
+                          NA,20,23,37,35,NA,52, 8,33,NA,
+                          25,NA,37,40,NA,NA,15,23,36,61))
> # set up a Lexis object with time from entry to death/exit
> Lx <- Lexis(entry = list(time = doE,
+                          age = doE - doN),
+             exit = list(time=pmin(doX, doD)),
+       exit.status = factor(doD < doX, labels = c("OK", "D")),
+             data = dd)
NOTE: entry.status has been set to "OK" for all.
> summary( Lx )
Transitions:
     To
From OK  D  Records:  Events: Risk time:  Persons:
  OK 14 16        30       16       1288        30
> # cut the follow-up at dates doA and doB
> L3 <- mcutLexis( Lx, "time",
+                  wh = c("doA","doB"),
+                 new.states = c("A","B"),
+            precursor.states = "OK",
+                 seq.states = FALSE,
+                 new.scales = c("tfA","tfB") )
> L3 <- Relevel(L3, c("OK","A","B","A+B","D"))
> summary( L3 )
Transitions:
     To
From  OK A  B A+B  D  Records:  Events: Risk time:  Persons:
  OK   6 6 10   0  3        25       19      743.8        25
  A    0 0  0   6  2         8        8      143.5         8
  B    0 0  3   6  3        12        9      196.9        12
  A+B  0 0  0   5  8        13        8      203.8        13
  Sum  6 6 13  17 16        58       44     1288.0        30
> # Make a graphical display of the data
> boxes(L3, boxpos = list(x = c(12,12,88,88,50),
+                         y = c(88,12,88,12,50)),
+           scale.R = 1000, show.BE = TRUE)
```

With this data set we can illustrate how the `Lexis2msm` works —we just use the three first persons for illustration:

```
> (Lx <- subset(L3, lex.id %in% 1:3))
 lex.id time  age tfA tfB lex.dur lex.Cst lex.Xst id   doN  doE  doX  doD doA doB
      1 16.5 18.0  NA  NA    36.7      OK      OK  1  -1.5 16.5 53.2 55.1  NA  NA
      3 11.9 34.1  NA  NA    40.9      OK       D  3 -22.2 11.9 53.1 52.8  NA  NA
      2  3.0  6.1  NA  NA    17.0      OK       B  2  -3.1  3.0 50.1 55.3  21  20
      2 20.0 23.1  NA   0     1.0       B     A+B  2  -3.1  3.0 50.1 55.3  21  20
      2 21.0 24.1   0   1    29.1     A+B     A+B  2  -3.1  3.0 50.1 55.3  21  20
```

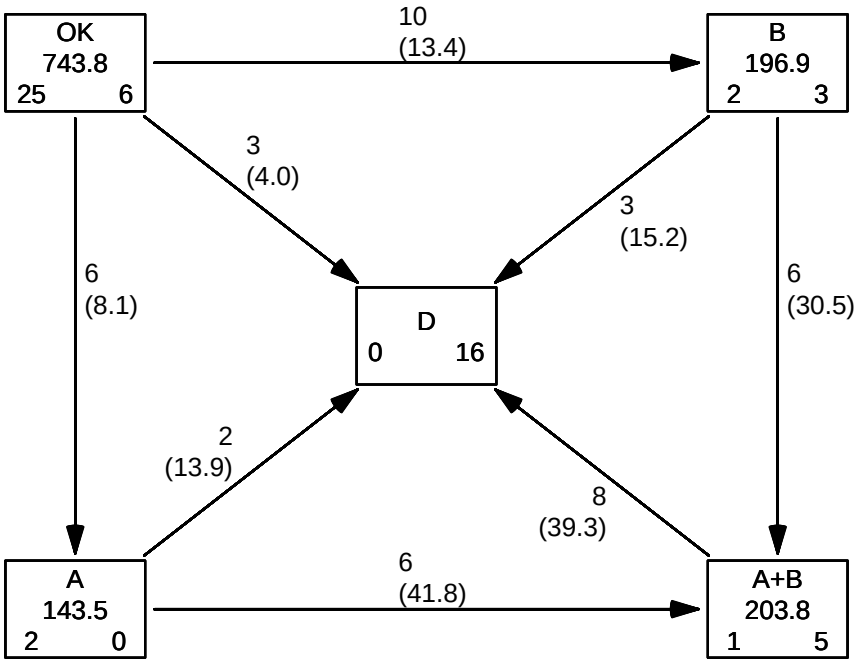


Figure 8.1: *Simple artificial example of a 5-state model. The numbers in the middle of each box is the risk time spent in each state (in years), and teh numbers at the bottom are the number of persons beginning, resp. ending follow-up in the state. The number of transitions and the empirical rates are given (per 1000 PY) on the transition arrows.*

../graph/msm-exbox

```
> Lexis2msm(Lx, v = T)
```

```
Object with state occupied at time points on the time scales:
 time age tfA tfB
  lex.id time  age  tfA  tfB state id   doN  doE  doX  doD doA doB
4      1 16.5 18.0   NA   NA    OK  1  -1.5 16.5 53.2 55.1  NA  NA
1      1 53.2 54.7   NA   NA    OK  1  -1.5 16.5 53.2 55.1  NA  NA
5      2  3.0  6.1   NA   NA    OK  2  -3.1  3.0 50.1 55.3  21  20
6      2 20.0 23.1   NA  0.0     B  2  -3.1  3.0 50.1 55.3  21  20
7      2 21.0 24.1  0.0  1.0   A+B  2  -3.1  3.0 50.1 55.3  21  20
2      2 50.1 53.2 29.1 30.1   A+B  2  -3.1  3.0 50.1 55.3  21  20
8      3 11.9 34.1   NA   NA    OK  3 -22.2 11.9 53.1 52.8  NA  NA
3      3 52.8 75.0   NA   NA     D  3 -22.2 11.9 53.1 52.8  NA  NA
```

Note that if we delete a record of follow-up for person 2 in state `A`, we still get the same results even if we have a "hole" in the follow-up of person 2, because the surrounding two records have the states occupied.

```
> (Lx <- subset(Lx, !(lex.id == 2 & lex.Cst == "A")))
```

```
 lex.id time  age tfA tfB lex.dur lex.Cst lex.Xst id   doN  doE  doX  doD doA doB
      1 16.5 18.0  NA  NA    36.7      OK      OK  1  -1.5 16.5 53.2 55.1  NA  NA
      3 11.9 34.1  NA  NA    40.9      OK       D  3 -22.2 11.9 53.1 52.8  NA  NA
      2  3.0  6.1  NA  NA    17.0      OK       B  2  -3.1  3.0 50.1 55.3  21  20
      2 20.0 23.1  NA   0     1.0       B     A+B  2  -3.1  3.0 50.1 55.3  21  20
      2 21.0 24.1   0   1    29.1     A+B     A+B  2  -3.1  3.0 50.1 55.3  21  20
```

```
> Lexis2msm(Lx, v = T)
```

```
Object with state occupied at time points on the time scales:
 time age tfA tfB
```

```
  lex.id time  age  tfA  tfB state id   doN  doE  doX  doD doA doB
4      1 16.5 18.0   NA   NA    OK  1  -1.5 16.5 53.2 55.1  NA  NA
1      1 53.2 54.7   NA   NA    OK  1  -1.5 16.5 53.2 55.1  NA  NA
5      2  3.0  6.1   NA   NA    OK  2  -3.1  3.0 50.1 55.3  21  20
6      2 20.0 23.1   NA  0.0     B  2  -3.1  3.0 50.1 55.3  21  20
7      2 21.0 24.1  0.0  1.0   A+B  2  -3.1  3.0 50.1 55.3  21  20
2      2 50.1 53.2 29.1 30.1   A+B  2  -3.1  3.0 50.1 55.3  21  20
8      3 11.9 34.1   NA   NA    OK  3 -22.2 11.9 53.1 52.8  NA  NA
3      3 52.8 75.0   NA   NA     D  3 -22.2 11.9 53.1 52.8  NA  NA
```

The more realistic example is:

```
> mLx <- Lexis2msm(L3)
> c(nid(L3), nid(mLx))
[1] 30 30
> c(nrow(L3), nrow(mLx))
[1] 58 88
> str(mLx)
Classes 'msmLexis' and 'data.frame':        88 obs. of  13 variables:
 $ lex.id: int  1 1 2 2 2 2 3 3 4 4 ...
 $ time  : num  16.5 53.2 3 20 21 50.1 11.9 52.8 14.7 27 ...
 $ age   : num  18 54.7 6.1 23.1 24.1 53.2 34.1 75 19.4 31.7 ...
 $ tfA   : num  NA NA NA NA 0 29.1 NA NA NA 0 ...
 $ tfB   : num  NA NA NA 0 1 30.1 NA NA NA NA ...
 $ state : Factor w/ 5 levels "OK","A","B","A+B",..: 1 1 1 3 4 4 1 5 1 2 ...
 $ id    : int  1 1 2 2 2 2 3 3 4 4 ...
 $ doN   : num  -1.5 -1.5 -3.1 -3.1 -3.1 -3.1 -22.2 -22.2 -4.7 -4.7 ...
 $ doE   : num  16.5 16.5 3 3 3 3 11.9 11.9 14.7 14.7 ...
 $ doX   : num  53.2 53.2 50.1 50.1 50.1 50.1 53.1 53.1 60 60 ...
 $ doD   : num  55.1 55.1 55.3 55.3 55.3 55.3 52.8 52.8 55.6 55.6 ...
 $ doA   : num  NA NA 21 21 21 21 NA NA 27 27 ...
 $ doB   : num  NA NA 20 20 20 20 NA NA 53 53 ...
 - attr(*, "time.scales")= chr [1:4] "time" "age" "tfA" "tfB"
 - attr(*, "time.since")= chr [1:4] "" "" "A" "B"
```

We see that the data frame is of class `msmLexis`, and nicely sorted, but also that there are more records in the `msmLexis` dataset, namely the number of records in the original data set *plus* the number of persons in the dataset. The latter is only true because all persons have consecutive follow-up intervals in the `Lexis` data frame.

8.3 Analysis of rates

8.3.1 Follow-up analysis using `glm.Lexis`

We can use the `glm.Lexis` function to model all 8 transitions; 4 mortality transitions in one model and the transitions to other transient states in two other. Note that we are not providing any values for the arguments `from` and `to` for the mortality analyses, because the default behaviour of `glm.Lexis` is to model all transitions *to* all *absorbing* states—in this case `Dead` is the only absorbing state:

```
> library(Epi)
> # mortality rates
> mm <- glm.Lexis(L3, ~ lex.Cst - 1)
stats::glm Poisson analysis of Lexis object L3 with log link:
Rates for transitions:
OK->D
A->D
B->D
A+B->D
> # occurrence of A then B
> mab <- glm.Lexis(L3, ~ lex.Cst -1,
+                       from = c("OK","A"),
+                         to = c("A","A+B"))
```

```
stats::glm Poisson analysis of Lexis object L3 with log link:
Rates for transitions:
OK->A
A->A+B
```

```
> # occurrence of B then A
> mba <- glm.Lexis(L3, ~ lex.Cst -1,
+                       from = c("OK","B"),
+                         to = c("B","A+B"))
```

```
stats::glm Poisson analysis of Lexis object L3 with log link:
Rates for transitions:
OK->B
B->A+B
```

We can arrange the estimated rates and the corresponding confidence intervals from the three models nicely:

```
> rates <- rbind(ci.exp(mab),
+                ci.exp(mba),
+                ci.exp(mm))
> from <- gsub("lex.Cst", "", rownames(rates))
>   to <- c("A","B","A+B","D")[c(1,3,2,3,4,4,4,4)]
> rownames(rates) <- NULL
> colnames(rates) <- c("Rate", "L", "U")
> data.frame(from, to, round(rates * 1000, 1))
  from  to Rate    L    U
1   OK   A  8.1  3.6 18.0
2    A A+B 41.8 18.8 93.1
3   OK   B 13.4  7.2 25.0
4    B A+B 30.5 13.7 67.8
5   OK   D  4.0  1.3 12.5
6    A   D 13.9  3.5 55.7
7    B   D 15.2  4.9 47.2
8  A+B   D 39.3 19.6 78.5
```

We see that we reproduce the empirical rates as shown in the plot with the boxes. This is a model assuming that all rates are constant, a model that can also be fitted using `msm`

8.3.2 Markov analysis by `msm`

We can also fit the model using `msm` by specifying that times given are excact times. However, `msm` requires that that states be numbered (and not, and moreover we want estimated rates in rates per 1000 PY, so we rescale `time`:

```
> Lm <- Lexis2msm(L3)
> Lt <- transform(Lm, state = as.integer(state),
+                     t1000 = time / 1000)
```

We illustrate that we have the same transition matrix

```
> library(msm)
> summary(L3)
Transitions:
     To
From  OK A  B A+B  D  Records:  Events: Risk time:  Persons:
  OK   6 6 10   0  3        25       19      743.8        25
  A    0 0  0   6  2         8        8      143.5         8
  B    0 0  3   6  3        12        9      196.9        12
  A+B  0 0  0   5  8        13        8      203.8        13
  Sum  6 6 13  17 16        58       44     1288.0        30
```

```
> statetable.msm(state, lex.id, data = Lm)
```

```
    to
from OK  A  B A+B  D
   1  6  6 10   0  3
   2  0  0  0   6  2
   3  0  0  3   6  3
   4  0  0  0   5  8
```

We need to specify a transtion matrix:

```
> qm <- tmat(L3) / 10
> qm[is.na(qm)] <- 0
> rownames(qm) <-
+ colnames(qm) <- 1:5
> qm
    1   2 3   4   5
  1 0 0.6 1 0.0 0.3
  2 0 0.0 0 0.6 0.2
  3 0 0.0 0 0.6 0.3
  4 0 0.0 0 0.0 0.8
  5 0 0.0 0 0.0 0.0
```

Then we can specify the Markov model:

```
> msmM <- msm(state ~ t1000,
+             subject = lex.id,
+                data = Lt,
+             qmatrix = qm,
+             obstype = 2)
> msmM
Call:
msm(formula = state ~ t1000, subject = lex.id, data = Lt, qmatrix = qm,     obstype = 2)

Maximum likelihood estimates

Transition intensities
      Baseline
1 - 1 -25.545 ( -40.048,-16.29)
1 - 2   8.067 (   3.624, 17.96)
1 - 3  13.445 (   7.234, 24.99)
1 - 5   4.033 (   1.301, 12.51)
2 - 2 -55.749 (-111.477,-27.88)
2 - 4  41.812 (  18.784, 93.07)
2 - 5  13.938 (   3.486, 55.73)
3 - 3 -45.709 ( -87.848,-23.78)
3 - 4  30.473 (  13.690, 67.83)
3 - 5  15.236 (   4.914, 47.24)
4 - 4 -39.254 ( -78.493,-19.63)
4 - 5  39.254 (  19.631, 78.49)

-2 * log-likelihood:  -168.7931
```

We can extract the estimated transition rates with c.i. using the extractor function `qmatrix.msm`:

```
> str(qmatrix.msm(msmM))
List of 5
 $ estimates: num [1:5, 1:5] -25.5 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:5] "1" "2" "3" "4" ...
  .. ..$ : chr [1:5] "1" "2" "3" "4" ...
 $ SE       : num [1:5, 1:5] 5.86 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:5] "1" "2" "3" "4" ...
  .. ..$ : chr [1:5] "1" "2" "3" "4" ...
 $ L        : num [1:5, 1:5] -40 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
```

msm

```
  .. ..$ : chr [1:5] "1" "2" "3" "4" ...
  .. ..$ : chr [1:5] "1" "2" "3" "4" ...
 $ U        : num [1:5, 1:5] -16.3 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:5] "1" "2" "3" "4" ...
  .. ..$ : chr [1:5] "1" "2" "3" "4" ...
 $ fixed    : logi [1:5, 1:5] FALSE FALSE FALSE FALSE FALSE FALSE ...
 - attr(*, "class")= chr "msm.est"
```

```
> qm <- qmatrix.msm(msmM)$estimates
> ql <- qmatrix.msm(msmM)$L
> qu <- qmatrix.msm(msmM)$U
```

We can then doctor this to a more readable from by applying state names, and then use these to get the estimated rates in the same form as those from the `Lexis` analysis:

```
> rownames(qm) <- rownames(ql) <- rownames(qu) <-
+ colnames(qm) <- colnames(ql) <- colnames(qu) <- levels(L3)
> data.frame(from, to,
+            round(rates * 1000, 1),
+            round(cbind(qm[cbind(from,to)],
+                        ql[cbind(from,to)],
+                        qu[cbind(from,to)]), 1))
  from  to Rate    L    U   X1   X2   X3
1   OK   A  8.1  3.6 18.0  8.1  3.6 18.0
2    A A+B 41.8 18.8 93.1 41.8 18.8 93.1
3   OK   B 13.4  7.2 25.0 13.4  7.2 25.0
4    B A+B 30.5 13.7 67.8 30.5 13.7 67.8
5   OK   D  4.0  1.3 12.5  4.0  1.3 12.5
6    A   D 13.9  3.5 55.7 13.9  3.5 55.7
7    B   D 15.2  4.9 47.2 15.2  4.9 47.2
8  A+B   D 39.3 19.6 78.5 39.3 19.6 78.5
```

. . . and bingo, we see we get exactly the same results.

So we can get the same results from `msm` as from traditional calculation of transition rates. But there are a couple of derivatives that come much easier from `msm`

. . . now input from `dogma`

Chapter 9

Advice and dogma

9.1 Practical advice for multistate analysis

The following is a summary of things to keep in mind when gathering data, doing analysis and reporting multistate models.

1. Record all event times as dates — facilitates calculation of any other time scale.
2. Draw boxes and arrows with person-years and transitions; this gives an overview of the data.
3. Draw Lexis diagrams of follow-up for (all) pairs of time scales, it may reveal (some of the) fishy features of your data.
4. Divide absorbing states by transition type (which state was immediately before the absorbing state, *e.g.* death with or without previous CVD) — then you will be able to address the risk of ever being in a given state.
5. Different occurrence rates (transition intensities) *out* of a state are not likely to be proportional. Specifically, they are not likely to depend on time (and other covariates) in the same way. Hence they will therefore normally be modeled separately. For example, it is unlikely that rates of cancer and CVD among persons not suffering from any of them will depend on age in the same way.
6. Transition intensities from different states *into* the *same* (type of) state are more likely to depend on time (and other covariates) in the same way. For example it is quite likely that cancer rates depend on age in the same way for persons with and without CVD.
7. The stacking of data (also called the *long* format, see e.g. [?], and p. ??) when modeling several transition rates with the common parameters is only needed if several rates *out* of the *same* state are modeled jointly (that is, with common parameters). If the joint modeling of several rates concerns at most one rate out of any state, the stacked data are not needed, simple subsetting of the the follow-up in the `Lexis` representations will suffice.

   By the previous token, this means the stacking of follow-up data is quite rarely needed. But there is a `stack` function that will transform a `Lexis` object to the long format.
8. Adding clinical measurements to a Lexis object naturally induces a time-scale for each (type of) clinical examination, namely the time since the measurement was taken.

   This is because we would expect that the effect of a measurement attenuates with time since the measurement. This can be modeled because we will have a timescale variable indicating the time since last clinical measurement. It simply amounts to including an interaction between the measurement value and the time since measurement.
9. Rates are smooth functions of time scales (and other covariates).
10. The choice of timescales to use in modeling is an empirical question, as is the in- or ex-clusion of any other covariate of potential interest from the model.

11. Differences between time scales may be relevant as covariates, *e.g.* age at diagnosis = current age − disease duration.
12. It is easier to obtain expected sojourn times, cumulative risks and other derived measures when rates are modeled by smooth parametric functions.

9.2 Dogma for multistate analysis

The following is a collection of dogma that I consider essential to obtain a valid and sensible statistical analysis and reporting from observations in a multistate model with multiple time scales.

1. do not condition on the future (see [1])
2. do not label quantities 'probability' or 'survival' if they are not (see [1])
3. do label interactions "*interactions*" if they are — it is confusing to invent new words such as "non-proportional hazards"
4. do not test for interactions—quantify them
5. stick to this world:
   (a) rates are smooth functions of time scale*s*
   (b) rates are likely to depend on more than one time scale
   — empirical examination is required to establish which
   (c) report multiple time scale effects jointly
   — they advance at the same time in real life

. . . now input from `Rtools`

Chapter 10

Useful R functions and packages

10.1 Lexis functions

The Lexis machinery has evolved over time since it was first introduced in a workable version in Epi_1.0.5 in August 2008, as described in [9, 3].

Over the years there have been additions of tools for handling multistate data. Below is a list of the current functions relating to Lexis objects with a very brief description of each; it does not replace the documentation so you have to RTFM. Unless otherwise stated, functions named something.Lexis (with a ".") are S3 methods for Lexis objects, so you can skip the ".Lexis" in daily use.

Define

Lexis defines a Lexis object

Cut and split

cutLexis cut follow-up at one intermediate event
mcutLexis cut follow-up at several intermediate events, keep track of history, no recurrences allowed
rcutLexis cut follow-up at several intermediate events, only most recent event kept, recurrences allowed
countLexis cut follow-up at intermediate event count the no. events so far
addCov.Lexis add clinical measurements at a given date to a Lexis object
splitLexis split follow up along a time scale
popEpi::splitMulti split follow up along a time scale — from the popEpi package, faster and has simpler syntax than splitLexis

Boxes and plots

boxes.Lexis draw a diagram of states and transitions
cohorttools::boxesLx draw a diagram of states and transitions using the graphVz machinery
plot.Lexis draw a standard Lexis diagram
points.Lexis add points to a Lexis diagram
lines.Lexis add lines to a Lexis diagram
PY.ann.Lexis annotate life lines in a Lexis diagram

Summarize and query

summary.Lexis overview of transitions, risk time etc.
levels.Lexis what are the states in the Lexis object
nid.Lexis number of persons in the Lexis object — how many unique values of lex.id are present
entry entry time
exit exit time
status status at entry or exit
timeBand factor of time bands
timeScales what time scales are in the Lexis object

`timeSince` what time scales are defined as time since a given state
`breaks` what breaks are currently defined
`absorbing` what are the absorbing states
`transient` what are the transient states
`preceding`, `before` which states precede this
`succeeding`, `after` which states can follow this
`tmat.Lexis` transition matrix for the `Lexis` object

Manipulate

`subset.Lexis`, `[` subset of a `Lexis` object
`merge.Lexis` merges a `Lexis` objects with a `data.frame`
`cbind.Lexis` bind a `data.frame` to a `Lexis` object
`rbind.Lexis` put two `Lexis` objects head-to-foot
`transform.Lexis` transform and add variables
`order.Lexis`, `orderLexis` return order by (`lex.id`, time scale)
`sortLexis` sort by (`lex.id`, time scale)—note: no S3 method available
`tsNA20` turn `NA`s to 0s for time scales
`factorize.Lexis` make `lex.Cst` and `lex.Xst` factors, removing empty levels
`Relevel.Lexis` reorder and combine states
`rm.tr` remove transitions from a `Lexis` object
`bootLexis` bootstrap sample of *persons* (`lex.id`) from a `Lexis` object

Simulate

`simLexis` simulate a `Lexis` object from specified transition rate models
`nState`, `pState` count state occupancy from a simulated `Lexis` object
`plot.pState`, `lines.pState` plot state occupancy from a `pState` object

Stack

`stack.Lexis` make a stacked object for simultaneous analysis of transitions — returns a `stacked.Lexis` object
`subset.stacked.Lexis` subsets of a `stacked.Lexis` object
`transform.stacked.Lexis` transform a `stacked.Lexis` object

Interfaces to other packages

`msdata.Lexis` interface to `mstate` package
`etm.Lexis` interface to `etm` package
`crr.Lexis` interface to `cmprsk` package

Statistical modeling —these are *not* S3 methods; you cannot omit `.Lexis`

`glm.Lexis` fit a `glm` model using the `poisreg` family to (hopefully) time split data
`gam.Lexis` fit a `gam` model (from the `mgcv` package) using the `poisreg` family to (hopefully) time split data
`coxph.Lexis` fit a Cox model to follow-up in a `Lexis` object

10.2 Modeling and reporting functions

- `coxph.Lexis`
- `glm.Lexis`
- `gam.Lexis`
- `ci.lin`
- `ci.exp`
- `ci.pred`
- `ci.cum`
- `ci.surv`
- `ci.Crisk`

10.3 R Packages

- survival
- cmprsk
- etm
- flexsurv
- mstate
- msm
- pseudo
- timereg
- cohorttools

. . . now input from appendix

appendix

Chapter 11

Appendices

. . . now input from `mrcut`

11.1 Comparing `mcut` and `rcut`

This is a short overview of the functioning of these two functions, the situations where they are relevant, and how the results relate to each other.

```
> library(Epi)
> library(popEpi)
```

The following is a modified version of the help page for `mcutLexis`; where we first set up a data frame of simulated dates, mimicking a multistate model with survival from `A` to `D`, and with possible intermediate states `Z` and `W`.

```
> set.seed(56324)
> N <- 500 # no. persons simulated
> dd <- data.frame(id = 1:N,
+                  doB = runif(N,-30, 0), # date of birth
+                  doE = runif(N,  0,20), # date of entry
+                  doX = runif(N, 50,60), # date of exit
+                  doD = runif(N, 50,60), # date of death
+                  doZ = runif(N, -6,20), # date of Z
+                  doW = runif(N, -2,20)) # date of W
> dd$doD[sample(1:N,N/3)] <- NA
> dd$doZ[sample(1:N,N/2)] <- NA
> dd$doW[sample(1:N,N/4)] <- NA
> dd$doZ <- dd$doZ + dd$doE
> dd$doW <- dd$doW + dd$doE
> dd$doX <- pmin(dd$doX, dd$doD, na.rm = TRUE)
> dd[,-1] <- dd[,-1] + 1940
> round(dd[1:10,], 2)
   id     doB     doE     doX     doD     doZ     doW
1   1 1927.25 1955.39 1995.58      NA 1970.91 1970.46
2   2 1937.28 1957.90 1992.72 1992.72      NA 1969.56
3   3 1916.47 1953.62 1996.79 1999.94 1967.15 1954.14
4   4 1918.35 1950.14 1991.40 1991.40 1956.44 1959.23
5   5 1935.66 1940.56 1998.77 1998.77      NA 1948.83
6   6 1938.15 1954.53 1997.71 1998.52 1965.80 1956.97
7   7 1922.91 1956.24 1992.41      NA      NA 1961.81
8   8 1930.93 1951.31 1993.38 1993.38 1952.24 1957.62
9   9 1913.37 1943.33 1997.02 1997.02      NA 1957.69
10 10 1932.00 1941.75 1991.78 1996.55 1939.55 1960.28
```

We then set up a `Lexis` object with follow-up time from entry to death/exit:

```
> Lx <- Lexis(entry = list(time = doE,
+                          age = doE - doB),
+             exit = list(time = doX),
+       exit.status = factor(!is.na(doD),
+                            labels=c("A", "D")),
+             data = dd)
NOTE: entry.status has been set to "A" for all.

> summary(Lx)

Transitions:
     To
From   A   D  Records:  Events: Risk time:  Persons:
   A 166 334       500      334   22018.58       500
```

11.1.1 mcutLexis: Cutting at dates of transitions, keeping history

11.1.1.1 If state sequence matters

Here we cut the follow-up at dates of Z and W, keeping track of whether Z occurred before W or vice versa:

```
> L2 <- mcutLexis(Lx, "time",
+                 wh = c("doZ", "doW"),
+         new.states = c("Z", "W"),
+         seq.states = TRUE,
+         new.scales = c("tfZ", "tfW"),
+       ties.resolve = TRUE)
NOTE: Precursor states set to A

> levels(L2)

[1] "A"   "W"   "W-Z" "D"   "Z"   "Z-W"
```

CODE EXPLAINED: The second argument is the name of the time scale in Lx object that the variables named in the wh argument refer to (here, the calendar time). The new.states argument names the states that persons transition to at these dates. The new.scales names new variables measuring times since entry to the states.

For display reasons we reorder the sequence of the state names (levels):

```
> L2 <- Relevel(L2, c("A", "Z", "Z-W", "W", "W-Z", "D"))
> summary(L2)
Transitions:
     To
From   A   Z Z-W   W W-Z   D  Records:  Events: Risk time:  Persons:
  A   16 121   0 237   0  45       419      403    5621.22       419
  Z    0  20 109   0   0  44       173      153    3304.06       173
  Z-W  0   0  41   0   0  73       114       73    3582.81       114
  W    0   0   0  65  70 124       259      194    7293.06       259
  W-Z  0   0   0   0  24  48        72       48    2217.43        72
  Sum 16 141 150 302  94 334      1037      871   22018.58       500
```

The reordering makes it a bit easier to get the coordinates for the boxes right:

```
> boxes( L2, boxpos = list(x = c(10, 65, 90, 65, 90, 50),
+                          y = c(50, 95, 70,  5, 30, 50)),
+            show.R = FALSE, show.BE = TRUE, cex = 1 )
```

The resulting graph corresponds to the top panel in figure 11.1.

mrcut

11.1.1.2 If state sequence does not matter

We may consider the relative timing of the events Z and W irrelevant from the beginning and decide to ignore the order in which Z and W occur by specifying seq.states=FALSE (TRUE is the default), in which case we get a state called W+Z. As above we also reorder the states to get an easier readable table:

```
> L3 <- mcutLexis(Lx, "time",
+                 wh = c("doZ", "doW"),
+          new.states = c("Z", "W"),
+          seq.states = FALSE,
+          new.scales = c("tfZ", "tfW"),
+        ties.resolve = TRUE)
NOTE: Precursor states set to A
> L3 <- Relevel(L3, c("A", "Z", "W", "W+Z", "D"))
> summary(L3)
Transitions:
     To
From    A   Z   W W+Z   D  Records:  Events: Risk time:  Persons:
  A    16 121 237   0  45       419      403    5621.22       419
  Z     0  20   0 109  44       173      153    3304.06       173
  W     0   0  65  70 124       259      194    7293.06       259
  W+Z   0   0   0  65 121       186      121    5800.25       186
  Sum  16 141 302 244 334      1037      871   22018.58       500


> boxes( L3, boxpos = list(x = c(10,70,70,90,45),
+                          y = c(50,90,10,50,50)),
+            show.R = FALSE, show.BE = TRUE, cex = 1)
```

The resulting graph corresponds to the middle panel in figure 11.1.

It is of course also possible to use Relevel.Lexis to merge the two states in the object L2 "by hand" to obtain the same:

```
> L3m <- Relevel(L2, list(1, 2, 4, "W+Z" = c("W-Z","Z-W"), 6))
> summary(L3m)
Transitions:
     To
From    A   Z   W W+Z   D  Records:  Events: Risk time:  Persons:
  A    16 121 237   0  45       419      403    5621.22       419
  Z     0  20   0 109  44       173      153    3304.06       173
  W     0   0  65  70 124       259      194    7293.06       259
  W+Z   0   0   0  65 121       186      121    5800.25       186
  Sum  16 141 302 244 334      1037      871   22018.58       500
```

We see we get the same result as L3. In fact it would have sufficient to say Relevel(L2, list("W+Z" = c("W-Z","Z-W"))), but the ordering of the states would have been different.

11.1.2 rcutLexis: Recurrent events — no memory

A somewhat different variant is where we do not keep track of persons' history but only the currently occupied state. This may for example be the case where it is possible to have both Z and W as recurrent states. This is handled by the function rcutLexis ("r" for recurrent), which requires the cut dates in "long" form, that is with one record per transition with variables lex.id, cut—date of transition and new.state—the state *to* which a transition occur:

```
> lcut <- with(Lx, rbind(data.frame(lex.id,
+                                   cut = doZ,
+                                 new.state = "Z"),
+                        data.frame(lex.id,
+                                   cut = doW,
+                                 new.state = "W")))
> lcut <- lcut[!is.na(lcut$cut),]
> lcut <- lcut[order(lcut$lex.id,lcut$cut),]
> print(lcut[1:10,], row.names=FALSE)
```

```
 lex.id      cut new.state
      1 1970.457         W
      1 1970.906         Z
      2 1969.559         W
      3 1954.139         W
      3 1967.152         Z
      4 1956.440         Z
      4 1959.234         W
      5 1948.834         W
      6 1956.971         W
      6 1965.801         Z
```

```
> L4 <- rcutLexis(Lx, cut = lcut)
> L4 <- Relevel(L4, c("A","Z","W","D"))
> summary(L4)
```

```
Transitions:
     To
From   A   Z   W   D  Records:  Events: Risk time:  Persons:
  A   16 121 237  45       419      403    5621.22       419
  Z    0  44 109  92       245      201    5521.49       245
  W    0  70 106 197       373      267   10875.87       373
  Sum 16 235 452 334      1037      871   22018.58       500
```

```
> boxes(L4, boxpos = list(x = c(10,70,70,45),
+                         y = c(50,85,15,50)),
+           show.R = FALSE, show.BE = TRUE, cex = 1)
```

The resulting graph corresponds to the lower panel in figure 11.1.

```
> par(mfcol = c(3, 1))
> boxes(L2, boxpos = list(x = c(10, 70, 95, 70, 95, 45),
+                         y = c(50, 85, 70, 15, 30, 50)),
+           show.R = FALSE, show.BE = TRUE)
> text(5, 90, "Full history\n(mcutLexis)", adj = c(0,1), cex = 1.7)
> boxes(L3, boxpos = list(x = c(10,70,70,95,45),
+                         y = c(50,85,15,50,50)),
+           show.R = FALSE, show.BE = TRUE)
> text(5, 90, "Order ignored\n(mcutLexis)", adj = c(0,1), cex = 1.7)
> boxes(L4, boxpos = list(x = c(10,70,70,45),
+                         y = c(50,85,15,50)),
+           show.R = FALSE, show.BE = TRUE)
> text(5, 90, "History ignored\n(rcutLexis)", adj = c(0,1), cex = 1.7)
```

The rcutLexis gives a result corresponding to merging of levels W and Z-W one one hand and Z and W-Z on the other. But rcutLexis also accommodates the more general case where recurrent transitions between Z and W occur, here is simple illustration:

```
> subset(L4, lex.id == 4)[,1:6]
```

```
 lex.id    time   age lex.dur lex.Cst lex.Xst
      4 1950.14 31.79    6.30       A       Z
      4 1956.44 38.09    2.79       Z       W
      4 1959.23 40.88   32.17       W       D
```

```
> xcut <- data.frame(lex.id = 4,
+                    cut = c(1956 + 0:4, 1975, 1990, 1995) + 0.1,
+                new.state = c("W","Z","W","Z","W","Z","W","Z") )
> xcut
```

```
  lex.id    cut new.state
1      4 1956.1         W
2      4 1957.1         Z
3      4 1958.1         W
4      4 1959.1         Z
5      4 1960.1         W
6      4 1975.1         Z
7      4 1990.1         W
8      4 1995.1         Z
```

mrcut

```
> subset(X4 <- rcutLexis(L4, xcut), lex.id == 4)[,1:6]
 lex.id    time   age lex.dur lex.Cst lex.Xst
      4 1950.14 31.79    5.96       A       W
      4 1956.10 37.75    0.34       W       W
      4 1956.44 38.09    0.66       W       Z
      4 1957.10 38.75    1.00       Z       W
      4 1958.10 39.75    1.00       W       Z
      4 1959.10 40.75    0.13       Z       Z
      4 1959.23 40.88    0.87       Z       W
      4 1960.10 41.75   15.00       W       Z
      4 1975.10 56.75   15.00       Z       W
      4 1990.10 71.75    1.30       W       D
```

The original transitions to `Z` at 1956.440 and to `W` at 1959.234 are preserved, and we also see that the new transition recorded after death (well after end of follow-up) is ignored.

Finally, we demonstrate that it is perfectly admissible with repeat recordings of the same state, it just results in

```
> xcut <- data.frame(lex.id = 4,
+                        cut = c(1956 + 0:4, 1975, 1990, 1995) + 0.1,
+                  new.state = c("W","W","Z","Z","W","Z","W","Z") )
> xcut
  lex.id    cut new.state
1      4 1956.1         W
2      4 1957.1         W
3      4 1958.1         Z
4      4 1959.1         Z
5      4 1960.1         W
6      4 1975.1         Z
7      4 1990.1         W
8      4 1995.1         Z
> subset(X4 <- rcutLexis(L4, xcut), lex.id == 4)[,1:6]
 lex.id    time   age lex.dur lex.Cst lex.Xst
      4 1950.14 31.79    5.96       A       W
      4 1956.10 37.75    0.34       W       W
      4 1956.44 38.09    0.66       W       W
      4 1957.10 38.75    1.00       W       Z
      4 1958.10 39.75    1.00       Z       Z
      4 1959.10 40.75    0.13       Z       Z
      4 1959.23 40.88    0.87       Z       W
      4 1960.10 41.75   15.00       W       Z
      4 1975.10 56.75   15.00       Z       W
      4 1990.10 71.75    1.30       W       D
```

. . . now input from `CovDrug`

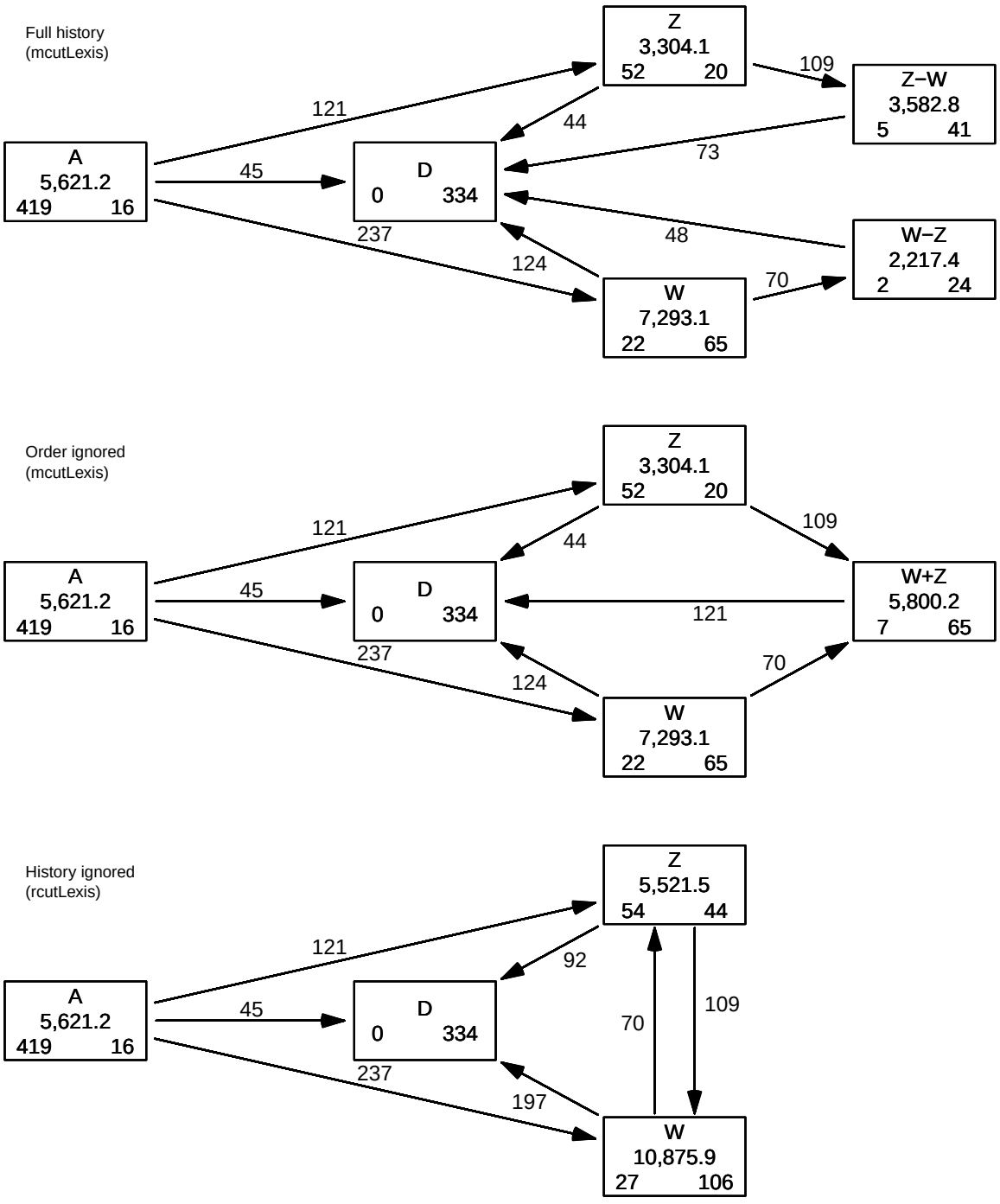


Figure 11.1: *States and transitions in the three different cases.*
Top: keeping track of the sequence of states visited, constructed by mcutLexis *using the argument* seq.states=TRUE *(the default).*
Middle: keeping track of states visited, bug ignoring visit order. constructed by mcutLexis *using the argument* seq.states=FALSE*. Can be obtained from the top panel by merging states* Z-W *and* W-Z*.*
Bottom: only keeping track of the last visited state, ignoring history, constructed by rcutLexis*. Can be obtained from the top panel by merging states* Z-W *and* W *and states* W-Z *and* Z*.* ../graph/cuts-all3

11.2 Adding time-dependent variables

If time-dependent variables are binary, such as for example "occurrence of CVD diagnosis" it may be relevant to define a new state as, say, `CVD`; this is the business of `cut.Lexis` and its cousins. But if we want to append quantitative variables that in principle can take any (positive) value at different times we need special functions for that. The functions `addCov.Lexis` and `addDrug.Lexis` are designed for this in different situations. The functions are `S3` methods for `Lexis` objects, so in code you can omit the "`.Lexis`".

The function `addCov.Lexis` provides the ability to amend a `Lexis` object with clinical measurements taken at different times, and propagate the values as LOCF (Last Observation Carried Forward) to all subsequent records. This means that time-splitting of a `Lexis` object *after* adding clinical measurements will be meaningful, because both `splitLexis` and `splitMulti` will carry variables forward to the split records. The follow-up in the resulting `Lexis` object will be cut at dates of clinical measurement.

As opposed to this, `addDrug.Lexis` will use drug information at each date of recorded drug purchase, and subsequently *compute* cumulative exposure measures at the times in the resulting `Lexis` object. This is essentially done by linear interpolation, so it will not be meaningful to further split an object resulting from `addDrug.Lexis`—LOCF is not meaningful for continuously time-varying covariates such as cumulative exposure.

If persons have very frequent drug purchases, the intervals may become very small and the sheer number of records may present an impediment to practical analysis. Therefore the function `coarse.Lexis` is provided to collapse adjacent follow-up records, to facilitate practical analysis.

In order to illustrate the working of the two functions we load the relevant packages and define a small utility to compactify the printing of data frames with a given number of digits after the decimal point.

```
> library(Epi)
> library(popEpi)
> library(tidyverse)
```

11.2.1 `addCov.Lexis`

11.2.1.1 Rationale

The function has arisen out of a need to attach values measured at clinical visits to a Lexis object representing follow-up for events constituting a multistate model. Hence the data frame with measurements at clinical visits will be called `clin` for mnemonic reasons.

11.2.1.2 The example data

For illustration we devise a small bogus cohort of 3 people, where we convert the character dates into numerical variables (fractional years) using `cal.yr`. Note that we are using a character variable as `id`:

```
> xcoh <- structure(list(id = c("A", "B", "C"),
+                       birth = c("1952-07-14", "1954-04-01", "1987-06-10"),
+                       entry = c("1965-08-04", "1972-09-08", "1991-12-23"),
+                        exit = c("1997-06-27", "1995-05-23", "1998-07-24"),
+                        fail = c(1, 0, 1) ),
+                      .Names = c("id", "birth", "entry", "exit", "fail"),
+                   row.names = c("1", "2", "3"),
+                       class = "data.frame" )
> xcoh$dob <- cal.yr(xcoh$birth)
> xcoh$doe <- cal.yr(xcoh$entry)
> xcoh$dox <- cal.yr(xcoh$exit )
> xcoh
  id      birth      entry       exit fail      dob      doe      dox
1  A 1952-07-14 1965-08-04 1997-06-27    1 1952.533 1965.589 1997.485
2  B 1954-04-01 1972-09-08 1995-05-23    0 1954.246 1972.686 1995.388
3  C 1987-06-10 1991-12-23 1998-07-24    1 1987.437 1991.974 1998.559
```

11.2.1.3 A `Lexis` object

Define this as a `Lexis` object with timescales calendar time (`per`, period) and age (`age`):

```
> Lcoh <- Lexis(entry = list(per = doe),
+                exit = list(per = dox,
+                            age = dox - dob),
+                  id = id,
+         exit.status = factor(fail, 0:1, c("Alive","Dead")),
+                data = xcoh)
NOTE: entry.status has been set to "Alive" for all.
```

```
> str(Lcoh)
```

```
Classes 'Lexis' and 'data.frame':        3 obs. of  14 variables:
 $ per    : 'cal.yr' num  1966 1973 1992
 $ age    : 'cal.yr' num  13.06 18.44 4.54
 $ lex.dur: 'cal.yr' num  31.9 22.7 6.58
 $ lex.Cst: Factor w/ 2 levels "Alive","Dead": 1 1 1
 $ lex.Xst: Factor w/ 2 levels "Alive","Dead": 2 1 2
 $ lex.id : chr  "A" "B" "C"
 $ id     : chr  "A" "B" "C"
 $ birth  : chr  "1952-07-14" "1954-04-01" "1987-06-10"
 $ entry  : chr  "1965-08-04" "1972-09-08" "1991-12-23"
 $ exit   : chr  "1997-06-27" "1995-05-23" "1998-07-24"
 $ fail   : num  1 0 1
 $ dob    : 'cal.yr' num  1953 1954 1987
 $ doe    : 'cal.yr' num  1966 1973 1992
 $ dox    : 'cal.yr' num  1997 1995 1999
 - attr(*, "time.scales")= chr [1:2] "per" "age"
 - attr(*, "time.since")= chr [1:2] "" ""
 - attr(*, "breaks")=List of 2
  ..$ per: NULL
  ..$ age: NULL
```

```
> (Lx <- Lcoh[,1:6])
```

```
 lex.id     per   age lex.dur lex.Cst lex.Xst
      A 1965.59 13.06   31.90   Alive    Dead
      B 1972.69 18.44   22.70   Alive   Alive
      C 1991.97  4.54    6.58   Alive    Dead
```

11.2.1.4 Factor or character `lex.id`?

Note that when the `id` argument to `Lexis` is a character variable then the `lex.id` will be a factor. Which, if each person has a lot of records may save time, but if you subset a `Lexis` object may be a waste of space. Moreover, merging (*i.e.* joining in the language of `tidyverse`) may present problems with different levels of factors. `merge` from the `base` R, will coerce to factor with union of levels as levels (including also levels that are not assumed in data), whereas the `_join` functions from `dplyr` will coerce to character.

Thus the most reasonable strategy thus seems to keep `lex.id` as a character variable.

```
> Lx$lex.id <- as.character(Lx$lex.id)
> str(Lx)
```

```
Classes 'Lexis' and 'data.frame':        3 obs. of  6 variables:
 $ per    : 'cal.yr' num  1966 1973 1992
 $ age    : 'cal.yr' num  13.06 18.44 4.54
 $ lex.dur: 'cal.yr' num  31.9 22.7 6.58
 $ lex.Cst: Factor w/ 2 levels "Alive","Dead": 1 1 1
 $ lex.Xst: Factor w/ 2 levels "Alive","Dead": 2 1 2
 $ lex.id : chr  "A" "B" "C"
 - attr(*, "time.scales")= chr [1:2] "per" "age"
 - attr(*, "time.since")= chr [1:2] "" ""
 - attr(*, "breaks")=List of 2
  ..$ per: NULL
  ..$ age: NULL
```

```
> Lx
```

```
 lex.id     per   age lex.dur lex.Cst lex.Xst
      A 1965.59 13.06   31.90   Alive    Dead
      B 1972.69 18.44   22.70   Alive   Alive
      C 1991.97  4.54    6.58   Alive    Dead
```

11.2.1.5 Clinical measurements

Then we generate data frame with clinical examination data, that is date of examination in `per`, some bogus clinical measurements and also names of the examination rounds:

```
> clin <- data.frame(lex.id = c("A", "A", "A", "C", "B", "C"),
+                        per = c(1983.3, 1975.2, 1971.7, 1996.2, 1990.6, 1989.2),
+                         bp = c(120, 135, 140, 160, 157, 145),
+                       chol = c(6, NA, 5, 8, 9, 6),
+                       xnam = c("X3", "X2", "X1", "X1", "X2", "X0"),
+          stringsAsFactors = FALSE)
> str(clin)
'data.frame':        6 obs. of  5 variables:
 $ lex.id: chr  "A" "A" "A" "C" ...
 $ per   : num  1983 1975 1972 1996 1991 ...
 $ bp    : num  120 135 140 160 157 145
 $ chol  : num  6 NA 5 8 9 6
 $ xnam  : chr  "X3" "X2" "X1" "X1" ...
> clin
  lex.id    per  bp chol xnam
1      A 1983.3 120    6   X3
2      A 1975.2 135   NA   X2
3      A 1971.7 140    5   X1
4      C 1996.2 160    8   X1
5      B 1990.6 157    9   X2
6      C 1989.2 145    6   X0
```

Note that we have chosen a measurement for person `C` from 1989—before the person's entry to the study, and have an `NA` for `chol` for person `A`.

11.2.1.6 Adding clinical data

There is a slightly different behaviour according to whether the variable with the name of the examination is given or not, and whether the name of the (incomplete) time scale is given or not:

```
> (Cx <- addCov.Lexis(Lx, clin))
 lex.id     per   age  tfc lex.dur lex.Cst lex.Xst exnam  bp chol xnam
      A 1965.59 13.06   NA    6.11   Alive   Alive  <NA>  NA   NA <NA>
      A 1971.70 19.17 0.00    3.50   Alive   Alive   ex1 140    5   X1
      A 1975.20 22.67 0.00    8.10   Alive   Alive   ex2 135   NA   X2
      A 1983.30 30.77 0.00   14.19   Alive    Dead   ex3 120    6   X3
      B 1972.69 18.44   NA   17.91   Alive   Alive  <NA>  NA   NA <NA>
      B 1990.60 36.35 0.00    4.79   Alive   Alive   ex1 157    9   X2
      C 1991.97  4.54 2.77    4.23   Alive   Alive   ex1 145    6   X0
      C 1996.20  8.76 0.00    2.36   Alive    Dead   ex2 160    8   X1
```

Note that the clinical measurement preceding the entry of person C is included, and that the `tfc` (time from clinical measurement) is correctly rendered, we a non-zero value at date of entry.

We also see that a variable `exnam` is constructed with consecutive numbering of examinations within each person, while the variable `xnam` is just carried over as any other.

If we explicitly give the name of the variable holding the examination names we do not get a constructed `exnam`. We can also define the name of the (incomplete) timescale to hold the time since measurement, in this case as `tfCl`:

```
> (Dx <- addCov.Lexis(Lx, clin, exnam = "xnam", tfc = "tfCl"))
```

```
 lex.id     per   age tfCl lex.dur lex.Cst lex.Xst xnam  bp chol
      A 1965.59 13.06   NA    6.11   Alive   Alive <NA>  NA   NA
      A 1971.70 19.17 0.00    3.50   Alive   Alive   X1 140    5
      A 1975.20 22.67 0.00    8.10   Alive   Alive   X2 135   NA
      A 1983.30 30.77 0.00   14.19   Alive    Dead   X3 120    6
      B 1972.69 18.44   NA   17.91   Alive   Alive <NA>  NA   NA
      B 1990.60 36.35 0.00    4.79   Alive   Alive   X2 157    9
      C 1991.97  4.54 2.77    4.23   Alive   Alive   X0 145    6
      C 1996.20  8.76 0.00    2.36   Alive    Dead   X1 160    8
```

```
> summary(Dx, t=T)
```

```
Transitions:
     To
From    Alive Dead  Records:  Events: Risk time:  Persons:
  Alive     6    2         8        2      61.18         3

Timescales:
 per  age tfCl
  ""   ""  "X"
```

11.2.1.7 Exchanging split and add

As noted before `addCov.Lexis` uses LOCF, and so it is commutative with `splitLexis`:

```
> # split BEFORE add
> Lb <- addCov.Lexis(splitLexis(Lx,
+                       time.scale = "age",
+                           breaks = seq(0, 80, 5)),
+                    clin,
+                    exnam = "xnam" )
> Lb
```

```
 lex.id     per   age  tfc lex.dur lex.Cst lex.Xst xnam  bp chol
      A 1965.59 13.06   NA    1.94   Alive   Alive <NA>  NA   NA
      A 1967.53 15.00   NA    4.17   Alive   Alive <NA>  NA   NA
      A 1971.70 19.17 0.00    0.83   Alive   Alive   X1 140    5
      A 1972.53 20.00 0.83    2.67   Alive   Alive   X1 140    5
      A 1975.20 22.67 0.00    2.33   Alive   Alive   X2 135   NA
      A 1977.53 25.00 2.33    5.00   Alive   Alive   X2 135   NA
      A 1982.53 30.00 7.33    0.77   Alive   Alive   X2 135   NA
      A 1983.30 30.77 0.00    4.23   Alive   Alive   X3 120    6
      A 1987.53 35.00 4.23    5.00   Alive   Alive   X3 120    6
      A 1992.53 40.00 9.23    4.95   Alive    Dead   X3 120    6
      B 1972.69 18.44   NA    1.56   Alive   Alive <NA>  NA   NA
      B 1974.25 20.00   NA    5.00   Alive   Alive <NA>  NA   NA
      B 1979.25 25.00   NA    5.00   Alive   Alive <NA>  NA   NA
      B 1984.25 30.00   NA    5.00   Alive   Alive <NA>  NA   NA
      B 1989.25 35.00   NA    1.35   Alive   Alive <NA>  NA   NA
      B 1990.60 36.35 0.00    3.65   Alive   Alive   X2 157    9
      B 1994.25 40.00 3.65    1.14   Alive   Alive   X2 157    9
      C 1991.97  4.54 2.77    0.46   Alive   Alive   X0 145    6
      C 1992.44  5.00 3.24    3.76   Alive   Alive   X0 145    6
      C 1996.20  8.76 0.00    1.24   Alive   Alive   X1 160    8
      C 1997.44 10.00 1.24    1.12   Alive    Dead   X1 160    8
```

```
> #
> # split AFTER add
> La <- splitLexis(addCov.Lexis(Lx,
+                               clin,
+                              exnam = "xnam" ),
+                  time.scale = "age",
+                      breaks = seq(0, 80, 5))
> La
```

```
 lex.id     per   age  tfc lex.dur lex.Cst lex.Xst xnam  bp chol
      A 1965.59 13.06   NA    1.94   Alive   Alive <NA>  NA   NA
      A 1967.53 15.00   NA    4.17   Alive   Alive <NA>  NA   NA
```

```
A 1971.70 19.17 0.00    0.83   Alive   Alive   X1 140    5
A 1972.53 20.00 0.83    2.67   Alive   Alive   X1 140    5
A 1975.20 22.67 0.00    2.33   Alive   Alive   X2 135   NA
A 1977.53 25.00 2.33    5.00   Alive   Alive   X2 135   NA
A 1982.53 30.00 7.33    0.77   Alive   Alive   X2 135   NA
A 1983.30 30.77 0.00    4.23   Alive   Alive   X3 120    6
A 1987.53 35.00 4.23    5.00   Alive   Alive   X3 120    6
A 1992.53 40.00 9.23    4.95   Alive    Dead   X3 120    6
B 1972.69 18.44   NA    1.56   Alive   Alive <NA>  NA   NA
B 1974.25 20.00   NA    5.00   Alive   Alive <NA>  NA   NA
B 1979.25 25.00   NA    5.00   Alive   Alive <NA>  NA   NA
B 1984.25 30.00   NA    5.00   Alive   Alive <NA>  NA   NA
B 1989.25 35.00   NA    1.35   Alive   Alive <NA>  NA   NA
B 1990.60 36.35 0.00    3.65   Alive   Alive   X2 157    9
B 1994.25 40.00 3.65    1.14   Alive   Alive   X2 157    9
C 1991.97  4.54 2.77    0.46   Alive   Alive   X0 145    6
C 1992.44  5.00 3.24    3.76   Alive   Alive   X0 145    6
C 1996.20  8.76 0.00    1.24   Alive   Alive   X1 160    8
C 1997.44 10.00 1.24    1.12   Alive    Dead   X1 160    8
```

We see that the results are identical bar the sequence of variables and attributes.

We can more explicitly verify that the resulting data frames are the same:

```
> La$tfc == Lb$tfc
 [1]    NA    NA TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE    NA    NA    NA    NA    NA
[16] TRUE TRUE TRUE TRUE TRUE TRUE

> La$age == Lb$age

 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[16] TRUE TRUE TRUE TRUE TRUE TRUE

> La$per == Lb$per

 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[16] TRUE TRUE TRUE TRUE TRUE TRUE
```

The same goes for `splitMulti`:

```
> # split BEFORE add
> Mb <- addCov.Lexis(splitMulti(Lx, age = seq(0, 80, 5)),
+                    clin,
+                    exnam = "xnam" )
> #
> # split AFTER add
> Ma <- splitMulti(addCov.Lexis(Lx,
+                               clin,
+                              exnam = "xnam" ),
+                  age = seq(0, 80, 5))
> La$tfc == Lb$tfc
 [1]    NA    NA TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE    NA    NA    NA    NA    NA
[16] TRUE TRUE TRUE TRUE TRUE TRUE

> Ma$tfc == Mb$tfc

 [1]    NA    NA TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE    NA    NA    NA    NA    NA
[16] TRUE TRUE TRUE TRUE TRUE TRUE
```

In summary, because both `addCov.Lexis` and `splitLexis`/`splitMulti` use LOCF for covariates the order of splitting and adding does not matter.

This is certainly not the case with addDrug.Lexis as we shall see.

11.2.1.8 Filling the NAs

As mentioned in the beginning, clinical measurements given as NA in the clin data frame are carried forward. If you want to have these replaced by 'older' clinical measurements you can do that explicitly by dplyr::fill:

```
> cov <- c("bp", "chol")
> Lx <- La
> Lx[,cov] <- as.data.frame(group_by(La, lex.id) %>% fill(cov))[,cov]
> La
 lex.id     per   age  tfc lex.dur lex.Cst lex.Xst xnam  bp chol
      A 1965.59 13.06   NA    1.94   Alive   Alive <NA>  NA   NA
      A 1967.53 15.00   NA    4.17   Alive   Alive <NA>  NA   NA
      A 1971.70 19.17 0.00    0.83   Alive   Alive   X1 140    5
      A 1972.53 20.00 0.83    2.67   Alive   Alive   X1 140    5
      A 1975.20 22.67 0.00    2.33   Alive   Alive   X2 135   NA
      A 1977.53 25.00 2.33    5.00   Alive   Alive   X2 135   NA
      A 1982.53 30.00 7.33    0.77   Alive   Alive   X2 135   NA
      A 1983.30 30.77 0.00    4.23   Alive   Alive   X3 120    6
      A 1987.53 35.00 4.23    5.00   Alive   Alive   X3 120    6
      A 1992.53 40.00 9.23    4.95   Alive    Dead   X3 120    6
      B 1972.69 18.44   NA    1.56   Alive   Alive <NA>  NA   NA
      B 1974.25 20.00   NA    5.00   Alive   Alive <NA>  NA   NA
      B 1979.25 25.00   NA    5.00   Alive   Alive <NA>  NA   NA
      B 1984.25 30.00   NA    5.00   Alive   Alive <NA>  NA   NA
      B 1989.25 35.00   NA    1.35   Alive   Alive <NA>  NA   NA
      B 1990.60 36.35 0.00    3.65   Alive   Alive   X2 157    9
      B 1994.25 40.00 3.65    1.14   Alive   Alive   X2 157    9
      C 1991.97  4.54 2.77    0.46   Alive   Alive   X0 145    6
      C 1992.44  5.00 3.24    3.76   Alive   Alive   X0 145    6
      C 1996.20  8.76 0.00    1.24   Alive   Alive   X1 160    8
      C 1997.44 10.00 1.24    1.12   Alive    Dead   X1 160    8
> Lx
 lex.id     per   age  tfc lex.dur lex.Cst lex.Xst xnam  bp chol
      A 1965.59 13.06   NA    1.94   Alive   Alive <NA>  NA   NA
      A 1967.53 15.00   NA    4.17   Alive   Alive <NA>  NA   NA
      A 1971.70 19.17 0.00    0.83   Alive   Alive   X1 140    5
      A 1972.53 20.00 0.83    2.67   Alive   Alive   X1 140    5
      A 1975.20 22.67 0.00    2.33   Alive   Alive   X2 135    5
      A 1977.53 25.00 2.33    5.00   Alive   Alive   X2 135    5
      A 1982.53 30.00 7.33    0.77   Alive   Alive   X2 135    5
      A 1983.30 30.77 0.00    4.23   Alive   Alive   X3 120    6
      A 1987.53 35.00 4.23    5.00   Alive   Alive   X3 120    6
      A 1992.53 40.00 9.23    4.95   Alive    Dead   X3 120    6
      B 1972.69 18.44   NA    1.56   Alive   Alive <NA>  NA   NA
      B 1974.25 20.00   NA    5.00   Alive   Alive <NA>  NA   NA
      B 1979.25 25.00   NA    5.00   Alive   Alive <NA>  NA   NA
      B 1984.25 30.00   NA    5.00   Alive   Alive <NA>  NA   NA
      B 1989.25 35.00   NA    1.35   Alive   Alive <NA>  NA   NA
      B 1990.60 36.35 0.00    3.65   Alive   Alive   X2 157    9
      B 1994.25 40.00 3.65    1.14   Alive   Alive   X2 157    9
      C 1991.97  4.54 2.77    0.46   Alive   Alive   X0 145    6
      C 1992.44  5.00 3.24    3.76   Alive   Alive   X0 145    6
      C 1996.20  8.76 0.00    1.24   Alive   Alive   X1 160    8
      C 1997.44 10.00 1.24    1.12   Alive    Dead   X1 160    8
```

The slightly convoluted code where the covariate columns are explicitly selected, owes to the fact that the dplyr functions will strip the data frames of the Lexis attributes. So you need to use fill to just generate the covariates and not touch the Lexis object itself.

The facility of propagating older clinical measurements should of course be built into addCov.Lexis as a separate argument.

Note that the tfc, time from clinical measurement is now not a valid variable any more; the 5 in chol is measured in 1971.5 but tfc is reset to 0 at 1977.3, despite only bp but not chol is measured at that time. If you want that remedied you will have to use addCov.Lexis twice, one with a clin data frame with only

bp and another with a data frame with only chol, each generating a differently named time from clinical measurement.

This is a problem that comes from the structure of the supplied *data* not from the program features; in the example we had basically measurements of different clinical variables at different times, and so necessarily also a need for different times since last measurement.

11.2.2 addDrug.Lexis

The general purpose of the function is to amend a Lexis object with drug exposure data. The data base with information on a specific drug is assumed to be a data frame with one entry per drug purchase (or prescription), containing the date and the amount purchased and optionally the prescribed dosage (that is how much is supposed to be taken per time). We assume that we have such a data base for each drug of interest, which also includes an id variable, lex.id, that matches the lex.id variable in the Lexis object.

For each type of drug the function derives 4 variables:

ex : logical, is the person currently exposed
tf : numeric, time since first purchase
ct : numeric, cumulative time on the drug
cd : numeric, cumulative dose of the drug

These names are pre- or suf-fixed by the drug name, so that exposures to different drugs can be distinguished; see the examples. The variables all refer to the status of the perons at the beginning of the intervals.

The resulting Lexis object has extra records corresponding to cuts at each drug purchase and at each expiry date of a purchase. For each purchase the coverage period is derived (different methods for this are available), and if the end of this (the expiry date) is earlier than the next purchase of the person, the person is considered off the drug from the expiry date, and a cut in the follow-up is generated with ex set to FALSE.

11.2.2.1 The help example

The following is a slight modification of the code from the example subsection of the help page for addDrug.Lexis

First we generate follow-up of 2 persons, and split the follow-up in intervals of length 0.6 years along the calendar time scale, per:

```
> fu <- data.frame(doe = c(2006, 2008),
+                  dox = c(2015, 2018),
+                  dob = c(1950, 1951),
+                  xst = factor(c("A","D")))
> Lx <- Lexis(entry = list(per = doe,
+                          age = doe- dob),
+             exit = list(per = dox),
+       exit.status = xst,
+             data = fu)
NOTE: entry.status has been set to "A" for all.

> Lx <- subset(Lx, select = -c(doe, dob, dox, xst))
> Sx <- splitLexis(Lx, "per", breaks = seq(1990, 2020, 0.6))
> summary(Sx)

Transitions:
     To
From  A D  Records:  Events: Risk time:  Persons:
   A 32 1        33        1         19         2
```

Then we generate drug purchases for these two persons, one data frame for each of the drugs F and G. Note that we generate lex.id$\in (1,2)$ referring to the values of lex.id in the lexis object Sx.

```
> set.seed(1952)
> rf <- data.frame(per = c(2005 + runif(12, 0, 10)),
+                  amt = sample(2:4, 12, replace = TRUE),
+                lex.id = sample(1:2, 12, replace = TRUE)) %>%
+       arrange(lex.id, per)
> rg <- data.frame(per = c(2009 + runif(10, 0, 10)),
+                  amt = sample(round(2:4/3,1), 10, replace = TRUE),
+                lex.id = sample(1:2, 10, replace = TRUE)) %>%
+       arrange(lex.id, per)
```

We do not need to sort the drug purchase data frames (it is done internally by `addDrug.Lexis`), but it makes it easier to grasp the structure.

The way purchase data is supplied to the function is in a `list` where each element is a data frame of purchase records for one type of drug. The list must be named, the names will be used as prefixes of the generated exposure variables. We can show the resulting data:

```
> pdat <- list(F = rf, G = rg)
> pdat
$F
        per amt lex.id
1  2013.964   4      1
2  2014.251   4      1
3  2014.509   3      1
4  2014.990   2      1
5  2005.311   2      2
6  2007.595   3      2
7  2011.710   4      2
8  2011.812   3      2
9  2012.865   2      2
10 2013.331   2      2
11 2013.417   2      2
12 2013.932   2      2

$G
        per amt lex.id
1  2009.630 1.3      1
2  2012.987 1.3      1
3  2013.018 1.3      1
4  2016.954 0.7      1
5  2017.924 1.0      1
6  2009.089 1.3      2
7  2011.599 0.7      2
8  2014.566 1.0      2
9  2016.912 0.7      2
10 2017.004 1.0      2
> Lx
 lex.id  per age lex.dur lex.Cst lex.Xst
      1 2006  56       9       A       A
      2 2008  57      10       A       D
```

Note that we have generated data so that there are drug purchases of drug `F` that is *before* start of follow-up for person 2.

We can then expand the time-split `Lexis` object, `Sx` with the drug information. `addDrug.Lexis` not only adds 8 *variables* (4 from each drug), it also adds *records* representing cuts at the purchase dates and possible expiry dates.

```
> summary(Sx) ; names(Sx)
Transitions:
     To
From  A D  Records:  Events: Risk time:  Persons:
   A 32 1        33        1         19         2
[1] "lex.id"  "per"     "age"     "lex.dur" "lex.Cst" "lex.Xst"
> ex1 <- addDrug.Lexis(Sx, pdat, method = "ext") # default
```

CovDrug

```
NOTE: timescale taken as 'per'
NOTE: end of exposure based on differences in purchase times (per)
 and amount purchased (amt).
```

```
> summary(ex1) ; names(ex1)
```

```
Transitions:
     To
From  A D  Records:  Events: Risk time:  Persons:
   A 64 1        65        1         19         2
```

```
 [1] "lex.id"  "per"     "age"     "lex.dur" "lex.Cst" "lex.Xst" "F.ex"
 [8] "F.tf"    "F.ct"    "F.cd"    "G.ex"    "G.tf"    "G.ct"    "G.cd"
```

```
> ex1
```

lex.id	per	age	lex.dur	lex.Cst	lex.Xst	F.ex	F.tf	F.ct	F.cd	G.ex
1	2006.00	56.00	0.20	A	A	FALSE	0.000	0.000	0.000	FALSE
1	2006.20	56.20	0.60	A	A	FALSE	0.000	0.000	0.000	FALSE
1	2006.80	56.80	0.60	A	A	FALSE	0.000	0.000	0.000	FALSE
1	2007.40	57.40	0.60	A	A	FALSE	0.000	0.000	0.000	FALSE
1	2008.00	58.00	0.60	A	A	FALSE	0.000	0.000	0.000	FALSE
1	2008.60	58.60	0.60	A	A	FALSE	0.000	0.000	0.000	FALSE
1	2009.20	59.20	0.43	A	A	FALSE	0.000	0.000	0.000	FALSE
1	2009.63	59.63	0.17	A	A	FALSE	0.000	0.000	0.000	TRUE
1	2009.80	59.80	0.60	A	A	FALSE	0.000	0.000	0.000	TRUE
1	2010.40	60.40	0.60	A	A	FALSE	0.000	0.000	0.000	TRUE
1	2011.00	61.00	0.60	A	A	FALSE	0.000	0.000	0.000	TRUE
1	2011.60	61.60	0.60	A	A	FALSE	0.000	0.000	0.000	TRUE
1	2012.20	62.20	0.60	A	A	FALSE	0.000	0.000	0.000	TRUE
1	2012.80	62.80	0.19	A	A	FALSE	0.000	0.000	0.000	TRUE
1	2012.99	62.99	0.03	A	A	FALSE	0.000	0.000	0.000	TRUE
1	2013.02	63.02	0.03	A	A	FALSE	0.000	0.000	0.000	TRUE
1	2013.05	63.05	0.35	A	A	FALSE	0.000	0.000	0.000	FALSE
1	2013.40	63.40	0.56	A	A	FALSE	0.000	0.000	0.000	FALSE
1	2013.96	63.96	0.00	A	A	TRUE	0.000	0.000	0.000	FALSE
1	2013.96	63.96	0.04	A	A	TRUE	0.000	0.000	0.000	FALSE
1	2014.00	64.00	0.25	A	A	TRUE	0.036	0.036	0.497	FALSE
1	2014.25	64.25	0.00	A	A	TRUE	0.286	0.286	4.000	FALSE
1	2014.25	64.25	0.26	A	A	TRUE	0.286	0.286	4.000	FALSE
1	2014.51	64.51	0.00	A	A	TRUE	0.545	0.545	8.000	FALSE
1	2014.51	64.51	0.09	A	A	TRUE	0.545	0.545	8.000	FALSE
1	2014.60	64.60	0.10	A	A	TRUE	0.636	0.636	9.406	FALSE
1	2014.70	64.70	0.00	A	A	FALSE	0.739	0.739	11.000	FALSE
1	2014.70	64.70	0.29	A	A	FALSE	0.739	0.739	11.000	FALSE
1	2014.99	64.99	0.01	A	A	TRUE	1.025	0.739	11.000	FALSE
2	2008.00	57.00	0.60	A	A	TRUE	0.000	0.405	0.354	FALSE
2	2008.60	57.60	0.49	A	A	TRUE	0.600	1.005	0.879	FALSE
2	2009.09	58.09	0.11	A	A	TRUE	1.089	1.494	1.308	TRUE
2	2009.20	58.20	0.60	A	A	TRUE	1.200	1.605	1.405	TRUE
2	2009.80	58.80	0.60	A	A	TRUE	1.800	2.205	1.930	TRUE
2	2010.40	59.40	0.60	A	A	TRUE	2.400	2.805	2.455	TRUE
2	2011.00	60.00	0.02	A	A	TRUE	3.000	3.405	2.981	TRUE
2	2011.02	60.02	0.00	A	A	FALSE	3.022	3.427	3.000	TRUE
2	2011.02	60.02	0.58	A	A	FALSE	3.022	3.427	3.000	TRUE
2	2011.60	60.60	0.00	A	A	FALSE	3.599	3.427	3.000	TRUE
2	2011.60	60.60	0.11	A	A	FALSE	3.600	3.427	3.000	TRUE
2	2011.71	60.71	0.00	A	A	TRUE	3.710	3.427	3.000	TRUE
2	2011.71	60.71	0.10	A	A	TRUE	3.710	3.427	3.000	TRUE
2	2011.81	60.81	0.08	A	A	TRUE	3.812	3.529	7.000	TRUE
2	2011.89	60.89	0.31	A	A	FALSE	3.889	3.605	10.000	TRUE
2	2012.20	61.20	0.60	A	A	FALSE	4.200	3.605	10.000	TRUE
2	2012.80	61.80	0.07	A	A	FALSE	4.800	3.605	10.000	TRUE
2	2012.87	61.87	0.09	A	A	TRUE	4.865	3.605	10.000	TRUE
2	2012.95	61.95	0.38	A	A	TRUE	4.951	3.691	10.368	FALSE
2	2013.33	62.33	0.07	A	A	TRUE	5.331	4.071	12.000	FALSE
2	2013.40	62.40	0.02	A	A	TRUE	5.400	4.140	13.606	FALSE
2	2013.42	62.42	0.09	A	A	TRUE	5.417	4.157	14.000	FALSE
2	2013.50	62.50	0.43	A	A	FALSE	5.503	4.243	16.000	FALSE

```
     2 2013.93 62.93    0.07       A       A  TRUE 5.932 4.243 16.000 FALSE
     2 2014.00 63.00    0.45       A       A  TRUE 6.000 4.311 16.265 FALSE
     2 2014.45 63.45    0.12       A       A FALSE 6.447 4.758 18.000 FALSE
     2 2014.57 63.57    0.03       A       A FALSE 6.566 4.758 18.000  TRUE
     2 2014.60 63.60    0.60       A       A FALSE 6.600 4.758 18.000  TRUE
     2 2015.20 64.20    0.60       A       A FALSE 7.200 4.758 18.000  TRUE
     2 2015.80 64.80    0.60       A       A FALSE 7.800 4.758 18.000  TRUE
     2 2016.40 65.40    0.51       A       A FALSE 8.400 4.758 18.000  TRUE
     2 2016.91 65.91    0.09       A       A FALSE 8.912 4.758 18.000  TRUE
     2 2017.00 66.00    0.00       A       A FALSE 9.000 4.758 18.000  TRUE
     2 2017.00 66.00    0.13       A       A FALSE 9.004 4.758 18.000  TRUE
     2 2017.14 66.14    0.46       A       A FALSE 9.135 4.758 18.000 FALSE
     2 2017.60 66.60    0.40       A       D FALSE 9.600 4.758 18.000 FALSE
 G.tf  G.ct  G.cd
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.170 0.170 0.066
0.770 0.770 0.298
1.370 1.370 0.531
1.970 1.970 0.763
2.570 2.570 0.995
3.170 3.170 1.228
3.357 3.357 1.300
3.388 3.388 2.600
3.420 3.420 3.900
3.770 3.420 3.900
4.334 3.420 3.900
4.334 3.420 3.900
4.370 3.420 3.900
4.621 3.420 3.900
4.621 3.420 3.900
4.879 3.420 3.900
4.879 3.420 3.900
4.970 3.420 3.900
5.073 3.420 3.900
5.073 3.420 3.900
5.360 3.420 3.900
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.111 0.111 0.057
0.711 0.711 0.368
1.311 1.311 0.679
1.911 1.911 0.990
1.933 1.933 1.001
1.933 1.933 1.001
2.510 2.510 1.300
2.511 2.511 1.300
2.621 2.621 1.358
2.621 2.621 1.358
2.723 2.723 1.410
2.800 2.800 1.450
3.111 3.111 1.611
3.711 3.711 1.922
3.776 3.776 1.956
3.862 3.862 2.000
4.242 3.862 2.000
4.311 3.862 2.000
4.328 3.862 2.000
4.414 3.862 2.000
```

```
 4.843 3.862 2.000
 4.911 3.862 2.000
 5.357 3.862 2.000
 5.477 3.862 2.000
 5.511 3.895 2.014
 6.111 4.495 2.270
 6.711 5.095 2.526
 7.311 5.695 2.782
 7.822 6.207 3.000
 7.911 6.295 3.673
 7.914 6.299 3.700
 8.046 6.430 4.700
 8.511 6.430 4.700
> ex2 <- addDrug.Lexis(Sx, pdat, method = "ext", grace = 0.5)
NOTE: timescale taken as 'per'
Values of grace has been recycled across 2 drugs
NOTE: end of exposure based on differences in purchase times (per)
 and amount purchased (amt).
> summary(ex2)
Transitions:
     To
From  A D  Records:  Events: Risk time:  Persons:
   A 62 1        63        1         19         2
> dos <- addDrug.Lexis(Sx, pdat, method = "dos", dpt = 6)
NOTE: timescale taken as 'per'
NOTE: end of exposure based on purchase and dosage (dpt).
> summary(dos)
Transitions:
     To
From  A D  Records:  Events: Risk time:  Persons:
   A 66 1        67        1         19         2
> dos
 lex.id     per   age lex.dur lex.Cst lex.Xst  F.ex  F.tf  F.ct   F.cd  G.ex
      1 2006.00 56.00    0.20       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2006.20 56.20    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2006.80 56.80    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2007.40 57.40    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2008.00 58.00    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2008.60 58.60    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2009.20 59.20    0.43       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2009.63 59.63    0.17       A       A FALSE 0.000 0.000  0.000  TRUE
      1 2009.80 59.80    0.05       A       A FALSE 0.000 0.000  0.000  TRUE
      1 2009.85 59.85    0.55       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2010.40 60.40    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2011.00 61.00    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2011.60 61.60    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2012.20 62.20    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2012.80 62.80    0.19       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2012.99 62.99    0.03       A       A FALSE 0.000 0.000  0.000  TRUE
      1 2013.02 63.02    0.22       A       A FALSE 0.000 0.000  0.000  TRUE
      1 2013.23 63.23    0.17       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2013.40 63.40    0.56       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2013.96 63.96    0.00       A       A  TRUE 0.000 0.000  0.000 FALSE
      1 2013.96 63.96    0.04       A       A  TRUE 0.000 0.000  0.000 FALSE
      1 2014.00 64.00    0.25       A       A  TRUE 0.036 0.036  0.497 FALSE
      1 2014.25 64.25    0.00       A       A  TRUE 0.286 0.286  4.000 FALSE
      1 2014.25 64.25    0.26       A       A  TRUE 0.286 0.286  4.000 FALSE
      1 2014.51 64.51    0.00       A       A  TRUE 0.545 0.545  8.000 FALSE
      1 2014.51 64.51    0.09       A       A  TRUE 0.545 0.545  8.000 FALSE
      1 2014.60 64.60    0.39       A       A  TRUE 0.636 0.636  8.567 FALSE
      1 2014.99 64.99    0.00       A       A  TRUE 1.025 1.025 11.000 FALSE
      1 2014.99 64.99    0.01       A       A  TRUE 1.025 1.025 11.000 FALSE
```

```
     2 2008.00 57.00    0.10       A       A  TRUE 0.000 0.405  2.428 FALSE
     2 2008.10 57.10    0.00       A       A FALSE 0.095 0.500  3.000 FALSE
     2 2008.10 57.10    0.50       A       A FALSE 0.095 0.500  3.000 FALSE
     2 2008.60 57.60    0.49       A       A FALSE 0.600 0.500  3.000 FALSE
     2 2009.09 58.09    0.11       A       A FALSE 1.089 0.500  3.000  TRUE
     2 2009.20 58.20    0.11       A       A FALSE 1.200 0.500  3.000  TRUE
     2 2009.31 58.31    0.49       A       A FALSE 1.306 0.500  3.000 FALSE
     2 2009.80 58.80    0.60       A       A FALSE 1.800 0.500  3.000 FALSE
     2 2010.40 59.40    0.60       A       A FALSE 2.400 0.500  3.000 FALSE
     2 2011.00 60.00    0.60       A       A FALSE 3.000 0.500  3.000 FALSE
     2 2011.60 60.60    0.00       A       A FALSE 3.599 0.500  3.000  TRUE
     2 2011.60 60.60    0.11       A       A FALSE 3.600 0.500  3.000  TRUE
     2 2011.71 60.71    0.01       A       A  TRUE 3.710 0.500  3.000  TRUE
     2 2011.72 60.72    0.10       A       A  TRUE 3.716 0.506  3.216 FALSE
     2 2011.81 60.81    0.39       A       A  TRUE 3.812 0.602  7.000 FALSE
     2 2012.20 61.20    0.11       A       A  TRUE 4.200 0.990  9.326 FALSE
     2 2012.31 61.31    0.49       A       A FALSE 4.312 1.102 10.000 FALSE
     2 2012.80 61.80    0.07       A       A FALSE 4.800 1.102 10.000 FALSE
     2 2012.87 61.87    0.33       A       A  TRUE 4.865 1.102 10.000 FALSE
     2 2013.20 62.20    0.13       A       A FALSE 5.199 1.435 12.000 FALSE
     2 2013.33 62.33    0.07       A       A  TRUE 5.331 1.435 12.000 FALSE
     2 2013.40 62.40    0.02       A       A  TRUE 5.400 1.504 13.606 FALSE
     2 2013.42 62.42    0.33       A       A  TRUE 5.417 1.521 14.000 FALSE
     2 2013.75 62.75    0.18       A       A FALSE 5.750 1.855 16.000 FALSE
     2 2013.93 62.93    0.07       A       A  TRUE 5.932 1.855 16.000 FALSE
     2 2014.00 63.00    0.27       A       A  TRUE 6.000 1.923 16.409 FALSE
     2 2014.27 63.27    0.30       A       A FALSE 6.265 2.188 18.000 FALSE
     2 2014.57 63.57    0.03       A       A FALSE 6.566 2.188 18.000  TRUE
     2 2014.60 63.60    0.13       A       A FALSE 6.600 2.188 18.000  TRUE
     2 2014.73 63.73    0.47       A       A FALSE 6.733 2.188 18.000 FALSE
     2 2015.20 64.20    0.60       A       A FALSE 7.200 2.188 18.000 FALSE
     2 2015.80 64.80    0.60       A       A FALSE 7.800 2.188 18.000 FALSE
     2 2016.40 65.40    0.51       A       A FALSE 8.400 2.188 18.000 FALSE
     2 2016.91 65.91    0.09       A       A FALSE 8.912 2.188 18.000  TRUE
     2 2017.00 66.00    0.00       A       A FALSE 9.000 2.188 18.000  TRUE
     2 2017.00 66.00    0.17       A       A FALSE 9.004 2.188 18.000  TRUE
     2 2017.17 66.17    0.43       A       A FALSE 9.170 2.188 18.000 FALSE
     2 2017.60 66.60    0.40       A       D FALSE 9.600 2.188 18.000 FALSE
 G.tf  G.ct  G.cd
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.170 0.170 1.020
0.217 0.217 1.300
0.770 0.217 1.300
1.370 0.217 1.300
1.970 0.217 1.300
2.570 0.217 1.300
3.170 0.217 1.300
3.357 0.217 1.300
3.388 0.248 2.600
3.605 0.465 3.900
3.770 0.465 3.900
4.334 0.465 3.900
4.334 0.465 3.900
4.370 0.465 3.900
4.621 0.465 3.900
4.621 0.465 3.900
4.879 0.465 3.900
4.879 0.465 3.900
4.970 0.465 3.900
```

CovDrug

```
 5.360 0.465 3.900
 5.360 0.465 3.900
 0.000 0.000 0.000
 0.000 0.000 0.000
 0.000 0.000 0.000
 0.000 0.000 0.000
 0.000 0.000 0.000
 0.111 0.111 0.665
 0.217 0.217 1.300
 0.711 0.217 1.300
 1.311 0.217 1.300
 1.911 0.217 1.300
 2.510 0.217 1.300
 2.511 0.217 1.304
 2.621 0.328 1.967
 2.627 0.333 2.000
 2.723 0.333 2.000
 3.111 0.333 2.000
 3.223 0.333 2.000
 3.711 0.333 2.000
 3.776 0.333 2.000
 4.109 0.333 2.000
 4.242 0.333 2.000
 4.311 0.333 2.000
 4.328 0.333 2.000
 4.661 0.333 2.000
 4.843 0.333 2.000
 4.911 0.333 2.000
 5.176 0.333 2.000
 5.477 0.333 2.000
 5.511 0.367 2.201
 5.644 0.500 3.000
 6.111 0.500 3.000
 6.711 0.500 3.000
 7.311 0.500 3.000
 7.822 0.500 3.000
 7.911 0.588 3.673
 7.914 0.592 3.700
 8.081 0.759 4.700
 8.511 0.759 4.700
```

```
> fix <- addDrug.Lexis(Sx, pdat, method = "fix", maxt = 1)
```

```
NOTE: timescale taken as 'per'
Values of maxt has been recycled across 2 drugs
NOTE: end of exposure based on fixed coverage time of 1 .
```

```
> summary(fix)
```

```
Transitions:
     To
From  A D  Records:  Events: Risk time:  Persons:
   A 63 1        64        1         19         2
```

```
> fix
```

```
 lex.id     per   age lex.dur lex.Cst lex.Xst  F.ex  F.tf  F.ct   F.cd  G.ex
      1 2006.00 56.00    0.20       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2006.20 56.20    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2006.80 56.80    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2007.40 57.40    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2008.00 58.00    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2008.60 58.60    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2009.20 59.20    0.43       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2009.63 59.63    0.17       A       A FALSE 0.000 0.000  0.000  TRUE
      1 2009.80 59.80    0.60       A       A FALSE 0.000 0.000  0.000  TRUE
      1 2010.40 60.40    0.23       A       A FALSE 0.000 0.000  0.000  TRUE
      1 2010.63 60.63    0.37       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2011.00 61.00    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2011.60 61.60    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
```

```
      1 2012.20 62.20    0.60       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2012.80 62.80    0.19       A       A FALSE 0.000 0.000  0.000 FALSE
      1 2012.99 62.99    0.03       A       A FALSE 0.000 0.000  0.000  TRUE
      1 2013.02 63.02    0.38       A       A FALSE 0.000 0.000  0.000  TRUE
      1 2013.40 63.40    0.56       A       A FALSE 0.000 0.000  0.000  TRUE
      1 2013.96 63.96    0.00       A       A  TRUE 0.000 0.000  0.000  TRUE
      1 2013.96 63.96    0.04       A       A  TRUE 0.000 0.000  0.000  TRUE
      1 2014.00 64.00    0.02       A       A  TRUE 0.036 0.036  0.497  TRUE
      1 2014.02 64.02    0.23       A       A  TRUE 0.054 0.054  0.752 FALSE
      1 2014.25 64.25    0.00       A       A  TRUE 0.286 0.286  4.000 FALSE
      1 2014.25 64.25    0.26       A       A  TRUE 0.286 0.286  4.000 FALSE
      1 2014.51 64.51    0.00       A       A  TRUE 0.545 0.545  8.000 FALSE
      1 2014.51 64.51    0.09       A       A  TRUE 0.545 0.545  8.000 FALSE
      1 2014.60 64.60    0.39       A       A  TRUE 0.636 0.636  8.567 FALSE
      1 2014.99 64.99    0.00       A       A  TRUE 1.025 1.025 11.000 FALSE
      1 2014.99 64.99    0.01       A       A  TRUE 1.025 1.025 11.000 FALSE
      2 2008.00 57.00    0.60       A       A  TRUE 0.000 0.405  1.214 FALSE
      2 2008.60 57.60    0.00       A       A FALSE 0.595 1.000  3.000 FALSE
      2 2008.60 57.60    0.00       A       A FALSE 0.595 1.000  3.000 FALSE
      2 2008.60 57.60    0.49       A       A FALSE 0.600 1.000  3.000 FALSE
      2 2009.09 58.09    0.11       A       A FALSE 1.089 1.000  3.000  TRUE
      2 2009.20 58.20    0.60       A       A FALSE 1.200 1.000  3.000  TRUE
      2 2009.80 58.80    0.29       A       A FALSE 1.800 1.000  3.000  TRUE
      2 2010.09 59.09    0.31       A       A FALSE 2.089 1.000  3.000 FALSE
      2 2010.40 59.40    0.60       A       A FALSE 2.400 1.000  3.000 FALSE
      2 2011.00 60.00    0.60       A       A FALSE 3.000 1.000  3.000 FALSE
      2 2011.60 60.60    0.00       A       A FALSE 3.599 1.000  3.000  TRUE
      2 2011.60 60.60    0.11       A       A FALSE 3.600 1.000  3.000  TRUE
      2 2011.71 60.71    0.10       A       A  TRUE 3.710 1.000  3.000  TRUE
      2 2011.81 60.81    0.39       A       A  TRUE 3.812 1.102  7.000  TRUE
      2 2012.20 61.20    0.40       A       A  TRUE 4.200 1.490  8.163  TRUE
      2 2012.60 61.60    0.20       A       A  TRUE 4.599 1.889  9.361 FALSE
      2 2012.80 61.80    0.01       A       A  TRUE 4.800 2.090  9.963 FALSE
      2 2012.81 61.81    0.05       A       A FALSE 4.812 2.102 10.000 FALSE
      2 2012.87 61.87    0.47       A       A  TRUE 4.865 2.102 10.000 FALSE
      2 2013.33 62.33    0.07       A       A  TRUE 5.331 2.568 12.000 FALSE
      2 2013.40 62.40    0.02       A       A  TRUE 5.400 2.637 13.606 FALSE
      2 2013.42 62.42    0.51       A       A  TRUE 5.417 2.654 14.000 FALSE
      2 2013.93 62.93    0.07       A       A  TRUE 5.932 3.168 16.000 FALSE
      2 2014.00 63.00    0.57       A       A  TRUE 6.000 3.237 16.136 FALSE
      2 2014.57 63.57    0.03       A       A  TRUE 6.566 3.803 17.269  TRUE
      2 2014.60 63.60    0.33       A       A  TRUE 6.600 3.837 17.336  TRUE
      2 2014.93 63.93    0.27       A       A FALSE 6.932 4.168 18.000  TRUE
      2 2015.20 64.20    0.37       A       A FALSE 7.200 4.168 18.000  TRUE
      2 2015.57 64.57    0.23       A       A FALSE 7.566 4.168 18.000 FALSE
      2 2015.80 64.80    0.60       A       A FALSE 7.800 4.168 18.000 FALSE
      2 2016.40 65.40    0.51       A       A FALSE 8.400 4.168 18.000 FALSE
      2 2016.91 65.91    0.09       A       A FALSE 8.912 4.168 18.000  TRUE
      2 2017.00 66.00    0.00       A       A FALSE 9.000 4.168 18.000  TRUE
      2 2017.00 66.00    0.60       A       A FALSE 9.004 4.168 18.000  TRUE
      2 2017.60 66.60    0.40       A       D FALSE 9.600 4.168 18.000  TRUE
 G.tf  G.ct  G.cd
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.170 0.170 0.221
0.770 0.770 1.001
1.000 1.000 1.300
1.370 1.000 1.300
1.970 1.000 1.300
2.570 1.000 1.300
```

CovDrug

```
3.170 1.000 1.300
3.357 1.000 1.300
3.388 1.031 2.600
3.770 1.413 3.096
4.334 1.978 3.830
4.334 1.978 3.830
4.370 2.013 3.876
4.388 2.031 3.900
4.621 2.031 3.900
4.621 2.031 3.900
4.879 2.031 3.900
4.879 2.031 3.900
4.970 2.031 3.900
5.360 2.031 3.900
5.360 2.031 3.900
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
0.111 0.111 0.144
0.711 0.711 0.924
1.000 1.000 1.300
1.311 1.000 1.300
1.911 1.000 1.300
2.510 1.000 1.300
2.511 1.001 1.301
2.621 1.111 1.378
2.723 1.213 1.449
3.111 1.601 1.721
3.510 2.000 2.000
3.711 2.000 2.000
3.723 2.000 2.000
3.776 2.000 2.000
4.242 2.000 2.000
4.311 2.000 2.000
4.328 2.000 2.000
4.843 2.000 2.000
4.911 2.000 2.000
5.477 2.000 2.000
5.511 2.034 2.034
5.843 2.365 2.365
6.111 2.634 2.634
6.477 3.000 3.000
6.711 3.000 3.000
7.311 3.000 3.000
7.822 3.000 3.000
7.911 3.088 3.673
7.914 3.092 3.700
8.511 3.688 4.299
```

11.2.2.2 A more realistic example with timing

Follow-up data: `DMlate` As example data we use the `DMlate` example data from the `Epi` package:

```
> data(DMlate) ; 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 ...
```

```
> Lx <- 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(Lx)
Transitions:
     To
From   DM Dead  Records:  Events: Risk time:  Persons:
  DM 7497 2499      9996     2499   54273.27      9996
```

We split the data along the age-scale (omitting the variables we shall not need):

```
> Sx <- splitMulti(Lx[,1:7], age = 0:120)
> summary(Sx)
Transitions:
     To
From    DM Dead  Records:  Events: Risk time:  Persons:
  DM 61627 2499     64126     2499   54273.27      9996
```

Artificial prescription data To explore how `addDrug.Lexis` works, we also need some drug exposure data, but these are unfortunately not available, so we simulate three datasets representing three types of drugs:

```
> set.seed(1952)
> purA <-
+   ( data.frame(lex.id = rep(Lx$lex.id,
+                             round(runif(nrow(Lx), 0, 20))))
+ %>% left_join(Lx[,c("lex.id", "dodm", "dox")])
+ %>% mutate(per = dodm + runif(length(dodm), -0.1, 0.99) * (dox - dodm),
+             amt = sample(4:20*10, length(dodm), replace = TRUE),
+             dpt = amt * round(runif(length(dodm), 3, 7)))
+ %>% select(-dodm, -dox)
+ %>% arrange(lex.id, per)
+   )
> addmargins(table(table(purA$lex.id)))
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
 504  529  474  480  489  445  485  476  494  489  518  487  501  525  522  524
  17   18   19   20  Sum
 505  524  498  254 9723
> str(purA)
'data.frame':        100736 obs. of  4 variables:
 $ lex.id: int  1 1 1 1 1 1 1 1 1 1 ...
 $ per   : num  1999 1999 1999 2000 2000 ...
 $ amt   : num  200 160 190 90 160 90 100 90 90 190 ...
 $ dpt   : num  1000 960 1330 360 640 630 500 450 630 950 ...
> purB <-
+   ( data.frame(lex.id = rep(Lx$lex.id,
+                             round(pmax(runif(nrow(Lx), -10, 15), 0))))
+ %>% left_join(Lx[,c("lex.id", "dodm", "dox")])
+ %>% mutate(per = dodm + runif(length(dodm), -0.1, 0.99) * (dox - dodm),
+             amt = sample(4:20*10, length(dodm), replace = TRUE),
+             dpt = amt * round(runif(length(dodm), 5, 9)))
+ %>% select(-dodm, -dox)
+ %>% arrange(lex.id, per)
+   ) -> purB
> addmargins(table(table(purB$lex.id)))
```

```
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15  Sum
 374  418  414  406  391  383  361  429  375  415  394  401  375  394  223 5753
> str(purB)
'data.frame':        44695 obs. of  4 variables:
 $ lex.id: int  1 1 1 1 1 1 1 1 1 1 ...
 $ per   : num  1998 1998 1999 1999 2000 ...
 $ amt   : num  100 190 110 140 120 90 140 70 150 180 ...
 $ dpt   : num  800 1520 770 980 960 810 1260 560 1050 1440 ...
> purC <-
+   ( data.frame(lex.id = rep(Lx$lex.id,
+                             round(pmax(runif(nrow(Lx), -5, 12), 0))))
+ %>% left_join(Lx[,c("lex.id", "dodm", "dox")])
+ %>% mutate(per = dodm + runif(length(dodm), -0.1, 0.99) * (dox - dodm),
+            amt = sample(4:20*10, length(dodm), replace = TRUE),
+            dpt = amt * round(runif(length(dodm), 5, 7)))
+ %>% select(-dodm, -dox)
+ %>% arrange(lex.id, per)
+   )
> addmargins(table(table(purB$lex.id)))
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15  Sum
 374  418  414  406  391  383  361  429  375  415  394  401  375  394  223 5753
> str(purC)
'data.frame':        42302 obs. of  4 variables:
 $ lex.id: int  1 1 1 1 1 5 5 5 5 5 ...
 $ per   : num  1998 2000 2003 2007 2008 ...
 $ amt   : num  90 40 200 100 100 160 60 90 170 60 ...
 $ dpt   : num  450 240 1200 600 700 1120 360 630 1190 360 ...
> head(purC)
  lex.id      per amt  dpt
1      1 1998.363  90  450
2      1 2000.059  40  240
3      1 2002.642 200 1200
4      1 2006.568 100  600
5      1 2007.684 100  700
6      5 2008.775 160 1120
```

Note that the time scale is in years, so the `dpt` must be in amount per year, so that `dpt/amt` is the approximate number of annual drug purchases.

We now have three artificial drug purchase datasets so we can see how `addDrug.Lexis` performs on larger datasets:

Using `addDrug`

1000 and 500 persons We start out with a small sample and a two month grace period to limit the number of gaps:

```
> Sx1 <- subset(Sx, lex.id < 1000)
> pur <- list(A = subset(purA, lex.id < 1000),
+             B = subset(purB, lex.id < 1000),
+             C = subset(purC, lex.id < 1000))
> system.time(ad1 <- addDrug.Lexis(Sx1, pur, tnam = "per", grace = 1/4))
Values of grace has been recycled across 3 drugs
NOTE: end of exposure based on differences in purchase times (per)
 and amount purchased (amt).
   user  system elapsed
  25.42    0.97   26.42
> summary(Sx1)
Transitions:
     To
From   DM Dead  Records:  Events: Risk time:  Persons:
  DM 6037  241      6278      241    5303.26       999
```

```
> summary(ad1)
Transitions:
     To
From    DM Dead  Records:  Events: Risk time:  Persons:
  DM 28720  241     28961      241    5303.26       999
```

We then cut the number of persons in half:

```
> Sx2 <- subset(Sx, lex.id < 500)
> pur <- list(A = subset(purA, lex.id < 500),
+             B = subset(purB, lex.id < 500),
+             C = subset(purC, lex.id < 500))
> system.time(ad2 <- addDrug.Lexis(Sx2, pur, tnam = "per", grace = 1/6))
Values of grace has been recycled across 3 drugs
NOTE: end of exposure based on differences in purchase times (per)
 and amount purchased (amt).
   user  system elapsed
  12.83    0.15   13.02
> summary(Sx2)
Transitions:
     To
From   DM Dead  Records:  Events: Risk time:  Persons:
  DM 3023  118      3141      118    2653.65       499
> summary(ad2)
Transitions:
     To
From    DM Dead  Records:  Events: Risk time:  Persons:
  DM 15413  118     15531      118    2653.65       499
```

It looks like timing is broadly proportional to the number of persons.

Fewer prescription records We can try to cut the number of purchases in half:

```
> pur <- list(A = subset(purA, lex.id < 500 & runif(nrow(purA)) < 0.5),
+             B = subset(purB, lex.id < 500 & runif(nrow(purB)) < 0.5),
+             C = subset(purC, lex.id < 500 & runif(nrow(purC)) < 0.5))
> sapply(pur, nrow)
   A    B    C
2557 1098 1063
> system.time(ad3 <- addDrug.Lexis(Sx2, pur, tnam = "per", grace = 1/6))
Values of grace has been recycled across 3 drugs
NOTE: end of exposure based on differences in purchase times (per)
 and amount purchased (amt).
   user  system elapsed
   5.84    0.21    6.08
> summary(Sx2)
Transitions:
     To
From   DM Dead  Records:  Events: Risk time:  Persons:
  DM 3023  118      3141      118    2653.65       499
> summary(ad3)
Transitions:
     To
From   DM Dead  Records:  Events: Risk time:  Persons:
  DM 9646  118      9764      118    2653.65       499
```

It appears that the number of purchases per person is also a determinant of the run time too; the timing is largely proportional to the number of drug records.

In any concrete application it is recommended to run the function on a fairly small sample of persons, say 1000 to get a feel for the run time. It may also be a good idea to run the function on chunks of the persons, to make sure that you do not lose all the processed data in a crash.

`CovDrug`

Fewer prescription types Finally we try to cut the number of drugs, to assess how this influences the run time:

```
> pur <- list(B = subset(purB, lex.id < 500),
+             C = subset(purC, lex.id < 500))
> sapply(pur, nrow)
   B    C
2197 2170
> system.time(ad4 <- addDrug.Lexis(Sx2, pur, tnam = "per", grace = 1/6))
Values of grace has been recycled across 2 drugs
NOTE: end of exposure based on differences in purchase times (per)
 and amount purchased (amt).
   user  system elapsed
   4.81    0.08    4.90
> summary(Sx2)
Transitions:
     To
From   DM Dead  Records:  Events: Risk time:  Persons:
  DM 3023  118      3141      118    2653.65       499
> summary(ad4)
Transitions:
     To
From   DM Dead  Records:  Events: Risk time:  Persons:
  DM 8976  118      9094      118    2653.65       499
```

We see that the number of records have been greatly expanded, almost tripled.

Too many records `–coarse.Lexis` If we look at the length of the intervals as given in `lex.dur` we see that some are are quite small:

```
> summary(ad1$lex.dur)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00000 0.02848 0.10277 0.18312 0.25620 1.00000
```

Half are smaller then 0.11 years, 40 days. We could without much loss of precision in the analysis merge adjacent records that have total risk time less than 3 months.

The function `coarse.Lexis` will collapse records with short `lex.dur` with the subsequent record. The collapsing will use the covariates from the first record, and so the entire follow-up from the two records will have the characteristics of the first. Therefore it is wise to choose first records with reasonably short `lex.dur`—the approximation will be better than if the first record was with a larger `lex.dur`. Therefore there are two values supplied to `coarse.Lexis`; the maximal length of the first record's `lex.dur` and the maximal length of the `lex.dur` in the resulting combined record. The larger these parameters are, the more the `Lexis` object is coarsened.

```
> summary(ad1)
Transitions:
     To
From    DM Dead  Records:  Events: Risk time:  Persons:
  DM 28720  241     28961      241    5303.26       999
> summary(adc <- coarse.Lexis(ad1, lim = c(1/6,1/2)))
Transitions:
     To
From    DM Dead  Records:  Events: Risk time:  Persons:
  DM 14670  241     14911      241    5303.26       999
> summary(adc$lex.dur)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.2079  0.3048  0.3557  0.4406  1.0000
> min(adc$lex.dur)
[1] 4.547474e-12
```

This could cut the number of units for analysis substantially, in this case from about 27,000 to some 13,000.

Records to be kept When we are dealing with drug exposure data we will be interested keeping the record that holds the start of a drug exposure. Some may argue that it does not matter much, though.

The records (*i.e.* beginnings of FU intervals) that should be kept must be given in logical vector in the argument `keep`:

```
> summary(Sx2)
Transitions:
     To
From   DM Dead  Records:  Events: Risk time:  Persons:
  DM 3023  118      3141      118    2653.65       499
> system.time(ad4 <- addDrug.Lexis(Sx2,
+                                  pur,
+                                 tnam = "per",
+                                grace = 1/6))
Values of grace has been recycled across 2 drugs
NOTE: end of exposure based on differences in purchase times (per)
 and amount purchased (amt).
   user  system elapsed
   5.34    0.11    5.48
> summary(ad4)
Transitions:
     To
From   DM Dead  Records:  Events: Risk time:  Persons:
  DM 8976  118      9094      118    2653.65       499
> #
> ad5 <- coarse.Lexis(ad4,
+                     lim = c(1/4, 1/2))
> summary(ad5)
Transitions:
     To
From   DM Dead  Records:  Events: Risk time:  Persons:
  DM 5460  118      5578      118    2653.65       499
```

We can identify the first date of exposure to drug `B`, say, by the exposure (`B.ex`) being true and the cumulative time on the drug (`B.ct`) being 0:

```
> ad4$keep <- with(ad4, (B.ex & B.ct == 0) |
+                       (C.ex & C.ct == 0))
> ad6 <- coarse.Lexis(ad4,
+                     lim = c(1/4, 1/2),
+                    keep = ad4$keep)
> summary(ad6)
Transitions:
     To
From   DM Dead  Records:  Events: Risk time:  Persons:
  DM 5578  118      5696      118    2653.65       499
```

We see the expected behaviour when we use `coarse.Lexis`: we get fewer records, but identical follow-up. And the `keep` argument gives the possibility to keep selected records, or more precisely beginnings. `keep` prevents a record to be collapsed with a previous one, but not with a subsequent one.

References

[1] P. K. Andersen and N. Keiding. Interpretability and importance of functionals in competing risks and multistate models. *Stat Med*, 31:1074–1088, 2012.

[2] L. Bjerg, A. Hulman, B. Carstensen, M. Charles, M. E. Jørgensen, and D. R. Witte. Development of Microvascular Complications and Effect of Concurrent Risk Factors in Type 1 Diabetes: A Multistate Model From an Observational Clinical Cohort Study. *Diabetes Care*, 41(11):2297–2305, 11 2018.

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

[4] M. J. Crowther and P. C. Lambert. Parametric multistate survival models: Flexible modelling allowing transition-specific distributions with application to estimating clinically useful measures of effect differences. *Stat Med*, 36(29):4719–4742, Dec 2017.

[5] L. Huo, D. J. Magliano, F. Ranciere, J. L. Harding, N. Nanayakkara, J. E. Shaw, and B. Carstensen. Impact of age at diagnosis and duration of type 2 diabetes on mortality in Australia 1997-2011. *Diabetologia*, Feb 2018.

[6] S. Iacobelli and B. Carstensen. Multiple time scales in multi-state models. *Stat Med*, 32(30):5315–5327, Dec 2013.

[7] P. Kragh Andersen, M. Pohar Perme, H. C. van Houwelingen, R. J. Cook, P. Joly, T. Martinussen, J. M. G. Taylor, M. Abrahamowicz, and T. M. Therneau. Analysis of time-to-event for observational studies: Guidance to the use of intensity models. *Stat Med*, 40(1):185–211, Jan 2021.

[8] K Larsen, JH Petersen, E Budtz-Jørgensen, and L Endahl. Interpreting parameters in the logistic regression model with random effects. *Biometrics*, 56(3):909–914, 2000.

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

[10] Patrick Royston and Paul C. Lambert. *Flexible Parametric Survival Analysis Using Stata: Beyond the Cox Model*. Stata Press, 2011.

Index

+, 47

AaJ.Lexis, 105, 129, 131
Aalen-Johansen estimator, 103
absorbing state, 4, 7
addCov.Lexis, 45
addDrug.Lexis, 45, 46
addmargins, 110, 118
age
    attained, 29
    current, 29
anova, 65
aperm, 101
apply, 9, 12, 97, 101, 117, 127, 130
as.Date.cal.yr, 57
attained age, 29
axis, 13, 29, 96, 101

Bayes' formula, 18
bootstrap, 99, 101
    parametric, 99, 101
box, 97
Boxes, 38
boxes
    arguments, 112
boxes, 35, 49, 61, 94, 112, 146
boxes argument
    boxpos, 112
    cex, 112
    pos.arr, 112
    scale.R, 112
    show.BE, 112
boxes.Lexis, 35, 61, 146
boxesLx, 36

cal.yr, 47, 48, 109
cause-specific rate, 96
cause-specific rates, 92
cbind, 29, 101
censoring date, 3
censoring time, 3
ci.Crisk, 99
ci.exp, 41, 115
ci.pred, 29, 31, 42, 97
clinical variables, 45
clock-back model, 15
cohorttools functions
    boxesLx, 36
colnames, 9
combining states, 38
competing risks, 21, 92
    stacking data, 22
conditional probabilities, 18
conditional survival, 4
confidence interval, 97
covariates
    clinical, 45
    timedependent, 45
Cox model
    mixed effects, 79
coxme, 79
coxme functions
    coxme, 79
cumsum, 97
cumulative intensity, 6
cumulative rate, 6, 31
cumulative risk, 4, 6
    confidence interval, 97
    stacked, 102
current age, 29
cutLexis, 33
cutLexis arguments
    new.state, 33
    precursor.states, 33
    split.states, 33
    timescale, 33
cutting time, 32, 36

data sets
    steno2 (Epi), 108
data.frame, 29
data.frame, 29, 31, 42
data.table, 29
dates, 3
deficient time scale, 51
diag, 9, 12
dplyr functions
    left_join, 63
    select, 111

empirical rate, 17
Epi data sets
    steno2, 108
Epi functions
    AaJ.Lexis, 129, 131
    addCov.Lexis, 45

addDrug.Lexis, 45, 46
as.Date.cal.yr, 57
boxes, 61, 94, 112
boxes.Lexis, 61
cal.yr, 47, 48, 109
ci.Crisk, 99
ci.exp, 41, 115
ci.pred, 42, 97
factorize, 63, 104
gam.Lexis, 27, 40, 95
glm.Lexis, 27, 65, 95, 115
Lexis, 26, 59, 93, 109
Lexis2msm, 137
lines.pState, 120
linesEst, 66
mat2pol, 13, 97, 102, 130
matshade, 29, 31, 42, 96, 101, 106
mcutLexis, 93
NArray, 13, 123
nState, 119
plot.pState, 120
plotEst, 66
pState, 120
rbind.Lexis, 117
rcutLexis, 60, 111
Relevel, 36, 118
Relevel.Lexis, 118
simLexis, 107, 118, 122
splitLexis, 28
subset.Lexis, 111, 118
summary.Lexis, 26, 109, 118
transform.Lexis, 117
Wald, 116
event, 6
simulated, 108
events
recurrent, 55
expand.grid, 42
expected life time, 6
expected lifetime, 4, 17
exponential life time, 20
exponential model, 20
exposure, 6
exposure time, 6

factor, 109
factorize, 63, 104
follow-up, 6
Follow-up likelihood, 19
follow-up time, 6
ftable, 103, 127
functions
+, 47
AaJ.Lexis (Epi), 129, 131
AaJ.Lexis (Lexis), 105
addCov.Lexis (Epi), 45
addDrug.Lexis (Epi), 45, 46
addmargins, 110, 118
anova, 65
aperm, 101
apply, 9, 12, 97, 101, 117, 127, 130
as.Date.cal.yr (Epi), 57
axis, 13, 29, 96, 101
box, 97
Boxes (Lexis), 38
boxes (Epi), 61, 94, 112
boxes (Lexis), 35, 49, 146
boxes.Lexis (Epi), 61
boxes.Lexis (Lexis), 35, 146
boxesLx (cohorttools), 36
boxesLx (gv2image), 36
cal.yr (Epi), 47, 48, 109
cbind, 29, 101
ci.Crisk (Epi), 99
ci.exp (Epi), 41, 115
ci.pred, 29, 31
ci.pred (Epi), 42, 97
colnames, 9
coxme (coxme), 79
cumsum, 97
cutLexis (Lexis), 33
data.frame, 29, 31, 42
diag, 9, 12
expand.grid, 42
factor, 109
factorize (Epi), 63, 104
ftable, 103, 127
gam (mgcv), 40
gam.Lexis (Epi), 27, 40, 95
glm.Lexis (Epi), 27, 65, 95, 115
head, 42
ifelse, 109
left_join (dplyr), 63
Levels (Lexis), 38
Lexis (Epi), 26, 59, 93, 109
Lexis (Lexis), 26, 47, 146
Lexis2msm (Epi), 137
lines, 13, 130
lines.pState (Epi), 120
linesEst (Epi), 66
list, 99, 123
mat2pol (Epi), 13, 97, 102, 130
matlines, 101
matrix, 9
MatrixExp (msm), 12
matshade (Epi), 29, 31, 42, 96, 101, 106
mcutLexis (Epi), 93
mcutLexis (Lexis), 36, 146
msm (msm), 138
mtext, 120, 123
mutate (tidyverse), 42, 109
names, 9
NArray (Epi), 13, 123

nState (Epi), 119
outer, 101
paste, 103
plot.pState (Epi), 120
plotEst (Epi), 66
pState (Epi), 120
qmatrix.msm (msm), 139
rbind.Lexis (Epi), 117
rcutLexis (Epi), 60, 111
rcutLexis (Lexis), 48, 148
Relevel (Epi), 36, 118
Relevel (Lexis), 36, 38
relevel, 109
Relevel.Lexis (Epi), 118
Relevel.Lexis (Lexis), 48
round, 41, 103
rownames, 9, 95
select (dplyr), 111
set.seed, 118
simLexis (Epi), 107, 118, 122
splitLexis (Epi), 28
splitMulti (popEpi), 28, 38, 94, 114
subset, 48, 104, 111
subset (Lexis), 48
subset.Lexis (Epi), 111, 118
summary (Lexis), 34, 146
summary.gam (mgcv), 41
summary.Lexis (Epi), 26, 109, 118
summary.Lexis (Lexis), 26, 48
Surv (survival), 31
survfit (survival), 31, 104
sweep, 125
table, 118
text, 13, 96, 97, 101
transform, 111
transform.Lexis (Epi), 117
transient (Lexis), 34
Wald (Epi), 116

gam, 40
gam.Lexis, 27, 40, 95
glm.Lexis, 27, 65, 95, 115
graphics functions
axis, 29, 96, 101
box, 97
mtext, 120, 123
text, 96, 97, 101
gv2image functions
boxesLx, 36

hazard, 6
hazard rate, 6
hazard ratio, 6, 30
head, 42
history, 6
homogeneous Markov model, 7, 16
hypoglycemia, 56

identity link, 20
ifelse, 109
incidence density, 6
inhomogeneous Markov model, 15, 16
integrated intensity, 6, 12
intensity, 6
intensity matrix, 11
interaction, 6

joint likelihood, 21

landmarking, 4
last observation carried forward, 45
left_join, 63
Levels, 38
Lexis, 26, 47, 59, 93, 109, 146
Lexis functions
AaJ.Lexis, 105
Boxes, 38
boxes, 35, 49, 146
boxes.Lexis, 35, 146
cutLexis, 33
Levels, 38
Lexis, 26, 47, 146
mcutLexis, 36, 146
rcutLexis, 48, 148
Relevel, 36, 38
Relevel.Lexis, 48
subset, 48
summary, 34, 146
summary.Lexis, 26, 48
transient, 34
Lexis functions, 142
Lexis, Wilhelm, 25
Lexis2msm, 137
lifetime lost, 4
likelihood, 17
competing risks, 21
multistate, 17, 23
Poisson, 19
lines, 13, 130
lines.pState, 120
linesEst, 66
link function
identity, 20
list, 99, 123
LOCF, 45

Markov
model semi, 16
Markov model, 7
homogeneous, 7, 16
inhomogeneous, 15, 16
time inhomogeneous, 19
mat2pol, 13, 97, 102, 130
matlines, 101
matrix, 9

matrix exponential, 12
matrix multiplication, 9
MatrixExp, 12
matshade, 29, 31, 42, 96, 101, 106
mcutLexis, 36, 93, 146
mgcv functions
  gam, 40
  summary.gam, 41
micro simulation, 16
micro-simulation, 16, 107
mixed effects Cox model, 79
model
  joint for transitions, 98
  semi-Markov, 15
model for mortality, 27
morbidity rate, 6
mortality, 27
  model, 27
mortality rate, 6, **17**, 17, 96
msm, 138
msm functions
  MatrixExp, 12
  msm, 138
  qmatrix.msm, 139
mtext, 120, 123
multiple time scales, 15, 21
multiple timescales, 16
multistate likelihood, 23
mutate, 42, 109

names, 9
NArray, 13, 123
new.state
  cutLexis argument, 33
non-proportional hazards, 6
nState, 119

observation, 6
operator precedence, 47
origin
  time, 4
outcome, 6
outer, 101

panel data, 133
parameter estimates, 98
  posterior, 98
  simulated, 98
parametric bootstrap, 98, 99, 101
paste, 103
person years, 21
plot.pState, 120
plotEst, 66
Poisson likelihood, 19
Poisson model, 19
popEpi functions
  splitMulti, 28, 38, 94, 114
precedence
  operator, 47
precursor.states
  cutLexis argument, 33
probability, 6
probability model, 17
pState, 120

qmatrix.msm, 139

R packages
  Epi, 133
  msm, 12, 133
rate, **17**, 96
  cause-specific, 96
  cumulative, 97
  empirical, 17
rate ratio, 6, 30
rates, 101
  time varying, 14
rbind.Lexis, 117
rcutLexis, 48, 60, 111, 148
recurrent events, 55
relative risk, 6
Relevel, 36, 38, 118
relevel, 109
Relevel.Lexis, 48, 118
response, 6
risk, 6
  cumulative, 97
  stacked, 102
risk ratio, 6
risk time, 6, 21
round, 41, 103
rownames, 9, 95

select, 111
semi Markov model, 16
semi-Markov model, 15, 19
set.seed, 118
simLexis, 107, 118, 122
simulating transitions, 108
simulation, 16, 98, 99, 101, 107
  micro, 16
sojourn time, 2, 4, 6
sojourn times, 103
split.states
  cutLexis argument, 33
splitLexis, 28
splitMulti, 28, 38, 94, 114
splitting time, 28
stacked data, 22
stacked risks, 102
stacking data, 22
state
  absorbing, 4, 7
  transient, 4, 7

state occupancy probability, 6
state probability, 6, 107
    Aalen-Johansen, 103
state time, 6
states
    coloring, 44
    combining, 38
`steno2`, 108
`subset`, 48, 104, 111
`subset.Lexis`, 111, 118
`summary`, 34, 146
`summary.gam`, 41
`summary.Lexis`, 26, 48, 109, 118
`Surv`, 31
`survfit`, 31, 104
survival, 4
    conditional, 4
survival analysis, 31
survival function, 17, 31
`survival` functions
    `Surv`, 31
    `survfit`, 31, 104
survival model, 17
`sweep`, 125

`table`, 118
`text`, 13, 96, 97, 101
`tidyverse` functions
    `mutate`, 42, 109
time, 6
    covariate, 21
    cutting, 32, 36
    multiple cuts, 36
    origin, 4
    outcome, 21
    response, 21
    since state, 63
    splitting, 28
time inhomogeneous Markov model, 19
time interaction, 6
time scale, 6
    deficient, 51
time scales, 4
    multiple, 15, 21
time varying rates, 14
time-inhomogeneous Markov model, 15
timedependent covariates, 45
`timescale`
    `cutLexis` argument, 33
`transform`, 111
`transform.Lexis`, 117
`transient`, 34
transient state, 4, 7
transition dates, 3
transition intensity, 6
transition probability, 6
transition time, 3
    simulated, 108

`Wald`, 116
Wilhelm Lexis, 25

About this book

This book takes you all the way through multistate models—from data definition and -manipulation through graphical representation of data to modeling of transition rates, and further to calculation of state probabilities and sojourn times.

The book includes the traditional methods based on semiparametric models for rates, but the main emphasis is on more credible parametric models for transition rates, allowing simultaneous dependence on more than one time scale (such as age and disease duration).

You will find a detailed demonstration of parametric modeling of transition rates and the application of these to derive estimates of state probabilities, sojourn times and other relevant measures.

Unlike most other books on this topic, the data representation and manipulation plays a prominent role; using carefully worked examples it covers in detail the `Lexis` machinery from the `Epi` and `popEpi` packages for R as well as the multistate tools from the R-packages `survival`, `mstate` and `msm`.

The style of the book is to illustrate concepts by examples in R, as close as possible to real modeling. That means that not only is the multistate machinery illustrated, also the paraphernalia to graph and report analyses and results. All the code in the book is also available on the book's website

http://bendixcarstensen.com/PMM