

Follow-up data with the Lexis functions in Epi

SDCC

February 2019

<http://bendixcarstensen.com/>

Version 3

Compiled Thursday 23rd May, 2019, 10:21

from: /home/bendix/stat/R/lib.src/Epi/pkg/vignettes/flup.tex

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

Contents

Introduction	1
1 Representation of follow-up data in the Epi package	2
1.1 Time scales	2
1.2 Splitting the follow-up time along a time scale	5
1.3 Cutting follow up time at dates of intermediate events	8
2 Modeling rates from Lexis objects	12
2.1 Covariates	12
2.1.1 Time scales as covariates	12
2.1.2 Differences between time scales	13
2.1.3 Keeping the relation between time scales	13
2.2 Modeling of rates	14
2.2.1 Interval length	15
2.2.2 Practicalities for splines	15
2.2.3 Poisson model	16
2.3 Time dependent covariate	18
2.3.1 Time since insulin start	19
2.4 The Cox model	21
2.5 Marginal effect of time since insulin	22
2.6 Age×duration interaction	25
2.6.1 Age at insulin start	25
2.6.2 General interaction	29
2.6.3 Evaluating interactions	29
2.7 Separate models	29
3 More states	33
3.1 Subdividing states	33
3.2 Multiple intermediate events	33
4 Lexis functions	38

Introduction

This is an introduction to the **Lexis** machinery in the **Epi** package. The machinery is intended for representation and manipulation of follow-up data (event history data) from studies where exact dates of events are known. It accommodates follow-up through multiple states and on multiple time scales.

This vignette uses an example from the **Epi** package to illustrate the set-up of a simple **Lexis** object (a data frame of follow-up intervals), as well as the subdivision of follow-up intervals needed for multistate representation and analysis of transition rates.

The first chapter is exclusively on manipulation of the follow-up representation, but it points to the subsequent chapter where analysis is based on a **Lexis** representation with very small follow-up intervals.

Chapter 2 uses analysis of mortality rates among Danish diabetes patients currently on insulin treatment or not to illustrate the use of the the **Lexis** machinery.

I owe much thanks to my colleague Lars Jorge Diaz for careful reading and many constructive suggestions.

Chapter 1

Representation of follow-up data in the Epi package

In the `Epi`-package, follow-up data is represented by adding some extra variables to a data frame. Such a data frame is called a `Lexis` object. The tools for handling follow-up data then use the structure of this for special plots, tabulations and modeling.

Follow-up data basically consists of a time of entry, a time of exit and an indication of the status at exit (normally either “alive” or “dead”) for each person. Implicitly is also assumed a status *during* the follow-up (usually “alive”).

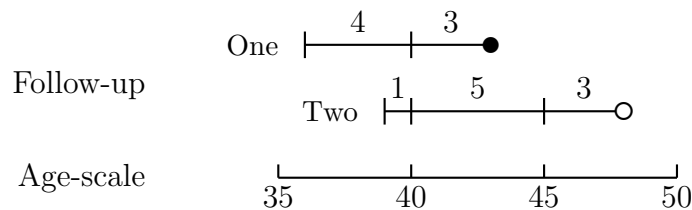


Figure 1.1: *Follow-up of two persons*

1.1 Time scales

A time scale is a variable that varies deterministically *within* each person during follow-up, *e.g.*:

- Age
- Calendar time
- Time since start of treatment
- Time since relapse

All time scales advance at the same pace, so the time followed is the same on all time scales. Therefore, it will suffice to use only the entry point on each of the time scales, for example:

- Age at entry
- Date of entry

- Time at treatment (*at* treatment this is 0)
- Time at relapse (*at* relapse this is 0)

For illustration we need to load the *Epi* package:

```
> library(Epi)
> print( sessionInfo(), l=F )
R version 3.6.0 (2019-04-26)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 14.04.6 LTS

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

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

other attached packages:
[1] Epi_2.37

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.0      lattice_0.20-38 zoo_1.8-4      MASS_7.3-51.1
 [5] grid_3.6.0      plyr_1.8.4      nlme_3.1-139  etm_1.0.4
 [9] data.table_1.12.0 Matrix_1.2-17    splines_3.6.0 tools_3.6.0
[13] cmprsk_2.2-7     numDeriv_2016.8-1 survival_2.44-1.1 parallel_3.6.0
[17] compiler_3.6.0  mgcv_1.8-28
```

In the *Epi* package, follow-up in a cohort is represented in a *Lexis* object. As mentioned, a *Lexis* object is a data frame with some extra structure representing the follow-up. For the *DMlate* data — follow-up of diabetes patients in Denmark recording date of birth, date of diabetes, date of insulin use, date of first oral drug use and date of death — we can construct a *Lexis* object by:

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

> dmL <- 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

```
> timeScales(dmL)
[1] "per" "age" "tfD"
```

(The excluded persons are persons with date of diabetes equal to date of death.)

The **entry** argument is a *named* list with the entry points on each of the time scales we want to use. It defines the names of the time scales and the entry points of the follow-up of each person. The **exit** argument gives the exit time on *one* of the time scales, so the name of the element in this list must match one of the names of the **entry** list. This is sufficient, because the follow-up time on all time scales is the same, in this case **dodm**.

The **exit.status** is a categorical variable (a *factor*) that indicates the exit status — in this case whether the person (still) is in state **DM** or exits to **Dead** at the end of follow-up. In principle we should also indicate the **entry.status**, but the default is to assume that all persons enter in the **first** of the mentioned **exit.states** — in this case **DM**, because **FALSE < TRUE**.

Now take a look at the result:

```
> str( dmL )
Classes 'Lexis' and 'data.frame':      9996 obs. of  14 variables:
 $ per      : num  1999 2003 2005 2009 2009 ...
 $ age      : num  58.7 64.1 86.3 44 75.8 ...
 $ tfD      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ lex.dur: num  11.08 6.689 5.446 0.736 1.344 ...
 $ lex.Cst: Factor w/ 2 levels "DM","Dead": 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Xst: Factor w/ 2 levels "DM","Dead": 1 1 1 1 1 2 1 1 2 1 ...
 $ lex.id  : int   1 2 3 4 5 6 7 8 9 10 ...
 $ 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 ...
 - attr(*, "time.scales")= chr  "per" "age" "tfD"
 - attr(*, "time.since")= chr  "" "" ""
 - attr(*, "breaks")=List of 3
 ..$ per: NULL
 ..$ age: NULL
 ..$ tfD: NULL

> head( dmL )[,1:10]
```

	per	age	tfD	lex.dur	lex.Cst	lex.Xst	lex.id	sex	dobth	dodm
50185	1998.917	58.66119	0	11.0800821	DM	DM	1	F	1940.256	1998.917
307563	2003.309	64.09035	0	6.6885695	DM	DM	2	M	1939.218	2003.309
294104	2004.552	86.25051	0	5.4455852	DM	DM	3	F	1918.301	2004.552
336439	2009.261	44.03559	0	0.7364819	DM	DM	4	F	1965.225	2009.261
245651	2008.653	75.77550	0	1.3442847	DM	DM	5	M	1932.877	2008.653
216824	2007.886	80.01643	0	2.0369610	DM	Dead	6	F	1927.870	2007.886

The **Lexis** object **dmL** has a variable for each time scale which is the entry point on this time scale. The follow-up time is in the variable **lex.dur** (duration). Note that the exit status is in the variable **lex.Xst** (eXit state). The variable **lex.Cst** is the state where the follow-up takes place (Current state), in this case **DM** (alive with diabetes) for all persons. This implies that *censored* observations are characterized by having **lex.Cst = lex.Xst**.

There is a **summary** function for **Lexis** objects that lists the number of transitions and records as well as the total amount of follow-up time; it also (optionally) prints information about the names of the variables that constitute the time scales:

```
> summary.Lexis( dmL, timeScales=TRUE )
Transitions:
      To
From   DM Dead  Records:  Events: Risk time:  Persons:
DM 7497 2499      9996      2499   54273.27      9996

Timescales:
per age tFD
"" "" ""
```

It is possible to get a visualization of the follow-up along the time scales chosen by using the `plot` method for `Lexis` objects. `dmL` is an object of *class* `Lexis`, so using the function `plot()` on it means that R will look for the function `plot.Lexis` and use this function.

```
> plot( dmL )
```

The function allows quite a bit of control over the output, and a `points.Lexis` function allows plotting of the endpoints of follow-up:

```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( dmL, 1:2, lwd=1, col=c("blue","red")[dmL$sex],
+       grid=TRUE, lty.grid=1, col.grid=gray(0.7),
+       xlim=1960+c(0,60), xaxs="i",
+       ylim= 40+c(0,60), yaxs="i", las=1 )
> points( dmL, 1:2, pch=c(NA,3)[dmL$lex.Xst],
+         col="lightgray", lwd=3, cex=0.3 )
> points( dmL, 1:2, pch=c(NA,3)[dmL$lex.Xst],
+         col=c("blue","red")[dmL$sex], lwd=1, cex=0.3 )
> box(bty='o')
```

In the above code you will note that the values of the arguments `col` and `pch` are indexed by factors, using the convention in R that the index is taken as *number of the level* of the supplied factor. Thus `c("blue","red")[dmL$sex]` is "blue" when `sex` is M (the first level).

The results of these two plotting commands are in figure 1.2, p. 6.

1.2 Splitting the follow-up time along a time scale

In next chapter we shall conduct statistical analysis of mortality rates, and a prerequisite for parametric analysis of rates is that follow-up time is subdivided in smaller intervals, where we can reasonably assume that rates are constant.

The follow-up time in a cohort can be subdivided (“split”) along a time scale, for example current age. This is achieved by the `splitLexis` (note that it is *not* called `split.Lexis`). This requires that the time scale and the breakpoints on this time scale are supplied. Try:

```
> dmS1 <- splitLexis( dmL, "age", breaks=seq(0,100,5) )
> summary( dmL )
Transitions:
      To
From   DM Dead  Records:  Events: Risk time:  Persons:
DM 7497 2499      9996      2499   54273.27      9996

> summary( dmS1 )
```

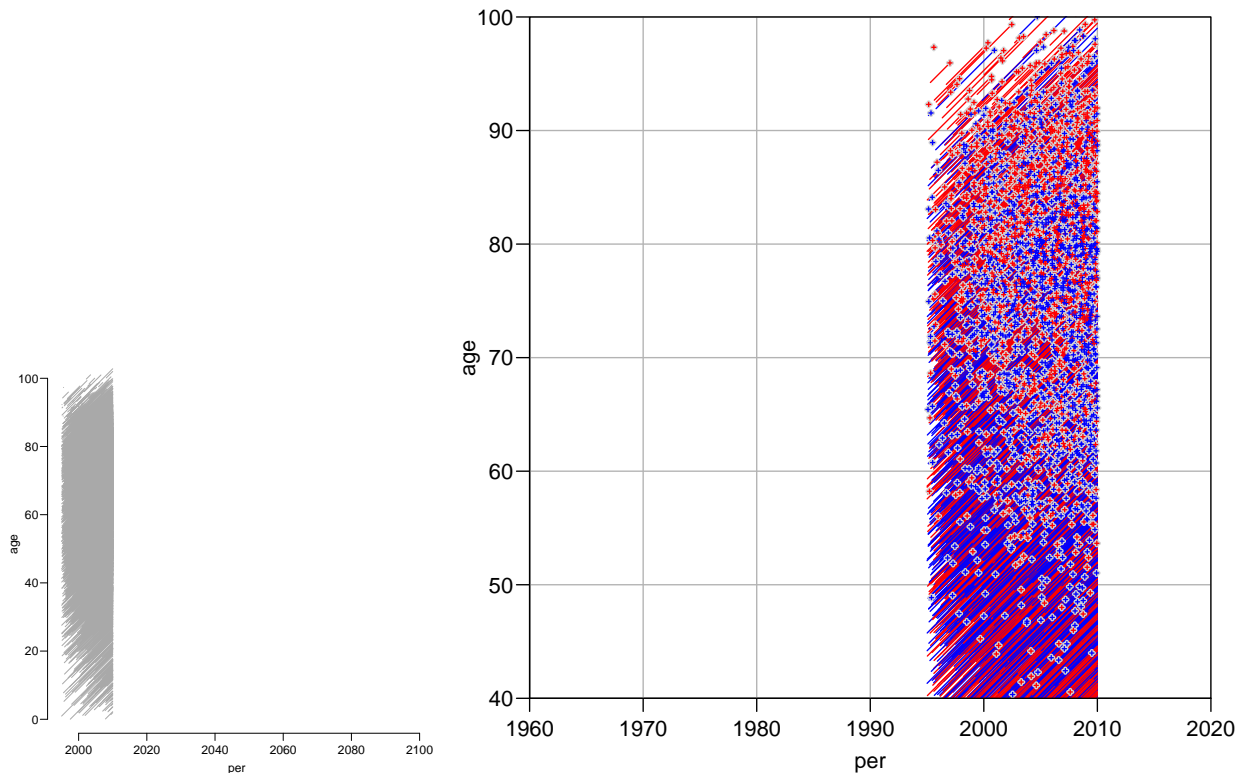


Figure 1.2: *Lexis diagram of the DMlate dataset; left panel is the default version, right panel: plot with some bells and whistles. The red lines are for women, blue for men, crosses indicate deaths.*

Transitions:

To

From DM Dead Records: Events: Risk time: Persons:
DM 18328 2499 20827 2499 54273.27 9996

We see that the number of persons and events and the amount of follow-up is the same in the two data sets; only the number of records differ — the extra records all have `lex.Cst=DM` and `lex.Xst=DM`.

To see how records are split for each individual, it is useful to list the results for a few individuals (whom we selected with a view to the illustrative usefulness):

```
> wh.id <- c(9,27,52,484)
> subset( dmL , lex.id %in% wh.id )[,1:10]
      per      age tfD  lex.dur lex.Cst lex.Xst lex.id sex  dobth  dodm
430048 1998.956 61.87269  0  9.508556      DM   Dead    9  F 1937.083 1998.956
22671  2000.042 52.71184  0  9.954825      DM      DM   27  M 1947.331 2000.042
338459 1998.249 61.85626  0 11.748118      DM      DM   52  F 1936.393 1998.249
274124 1998.260 62.37919  0 10.929500      DM   Dead  484  F 1935.881 1998.260

> subset( dmS1, lex.id %in% wh.id )[,1:10]
      lex.id      per      age      tfD  lex.dur lex.Cst lex.Xst sex  dobth  dodm
14         9 1998.956 61.87269 0.000000 3.127310      DM      DM  F 1937.083 1998.956
15         9 2002.083 65.00000 3.127310 5.000000      DM      DM  F 1937.083 1998.956
16         9 2007.083 70.00000 8.127310 1.381246      DM   Dead  F 1937.083 1998.956
54        27 2000.042 52.71184 0.000000 2.288159      DM      DM  M 1947.331 2000.042
55        27 2002.331 55.00000 2.288159 5.000000      DM      DM  M 1947.331 2000.042
```


56	27	2007.331	60.00000	7.288159	2.666667	DM	DM	M	1947.331	2000.042
108	52	1998.249	61.85626	0.000000	3.143737	DM	DM	F	1936.393	1998.249
109	52	2001.393	65.00000	3.143737	5.000000	DM	DM	F	1936.393	1998.249
110	52	2006.393	70.00000	8.143737	3.604381	DM	DM	F	1936.393	1998.249
1004	484	1998.260	62.37919	0.000000	2.620808	DM	DM	F	1935.881	1998.260
1005	484	2000.881	65.00000	2.620808	5.000000	DM	DM	F	1935.881	1998.260
1006	484	2005.881	70.00000	7.620808	3.308693	DM	Dead	F	1935.881	1998.260

The resulting object, `dmS1`, is again a `Lexis` object, and the follow-up may be split further along another time scale, for example diabetes duration, `tfD`. Subsequently we list the results for the chosen individuals:

```
> dmS2 <- splitLexis( dmS1, "tfD", breaks=c(0,1,2,5,10,20,30,40) )
> subset( dmS2, lex.id %in% wh.id )[,1:10]
```

	lex.id	per	age	tfD	lex.dur	lex.Cst	lex.Xst	sex	dobth	dodm
31	9	1998.956	61.87269	0.000000	1.0000000	DM	DM	F	1937.083	1998.956
32	9	1999.956	62.87269	1.000000	1.0000000	DM	DM	F	1937.083	1998.956
33	9	2000.956	63.87269	2.000000	1.1273101	DM	DM	F	1937.083	1998.956
34	9	2002.083	65.00000	3.127310	1.8726899	DM	DM	F	1937.083	1998.956
35	9	2003.956	66.87269	5.000000	3.1273101	DM	DM	F	1937.083	1998.956
36	9	2007.083	70.00000	8.127310	1.3812457	DM	Dead	F	1937.083	1998.956
111	27	2000.042	52.71184	0.000000	1.0000000	DM	DM	M	1947.331	2000.042
112	27	2001.042	53.71184	1.000000	1.0000000	DM	DM	M	1947.331	2000.042
113	27	2002.042	54.71184	2.000000	0.2881588	DM	DM	M	1947.331	2000.042
114	27	2002.331	55.00000	2.288159	2.7118412	DM	DM	M	1947.331	2000.042
115	27	2005.042	57.71184	5.000000	2.2881588	DM	DM	M	1947.331	2000.042
116	27	2007.331	60.00000	7.288159	2.6666667	DM	DM	M	1947.331	2000.042
229	52	1998.249	61.85626	0.000000	1.0000000	DM	DM	F	1936.393	1998.249
230	52	1999.249	62.85626	1.000000	1.0000000	DM	DM	F	1936.393	1998.249
231	52	2000.249	63.85626	2.000000	1.1437372	DM	DM	F	1936.393	1998.249
232	52	2001.393	65.00000	3.143737	1.8562628	DM	DM	F	1936.393	1998.249
233	52	2003.249	66.85626	5.000000	3.1437372	DM	DM	F	1936.393	1998.249
234	52	2006.393	70.00000	8.143737	1.8562628	DM	DM	F	1936.393	1998.249
235	52	2008.249	71.85626	10.000000	1.7481177	DM	DM	F	1936.393	1998.249
2084	484	1998.260	62.37919	0.000000	1.0000000	DM	DM	F	1935.881	1998.260
2085	484	1999.260	63.37919	1.000000	1.0000000	DM	DM	F	1935.881	1998.260
2086	484	2000.260	64.37919	2.000000	0.6208077	DM	DM	F	1935.881	1998.260
2087	484	2000.881	65.00000	2.620808	2.3791923	DM	DM	F	1935.881	1998.260
2088	484	2003.260	67.37919	5.000000	2.6208077	DM	DM	F	1935.881	1998.260
2089	484	2005.881	70.00000	7.620808	2.3791923	DM	DM	F	1935.881	1998.260
2090	484	2008.260	72.37919	10.000000	0.9295003	DM	Dead	F	1935.881	1998.260

A more efficient (and more intuitive) way of making this double split is to use the `splitMulti` function from the `popEpi` package:

```
> library( popEpi )
> dmM <- splitMulti( dmL, age = seq(0,100,5),
+                   tfD = c(0,1,2,5,10,20,30,40),
+                   drop = FALSE )
> summary( dmS2 )
```

Transitions:

From	To	DM	Dead	Records:	Events:	Risk time:	Persons:
DM	DM	40897	2499	43396	2499	54273.27	9996

```
> summary( dmM )
```

Transitions:

```

To
From    DM Dead  Records:  Events: Risk time:  Persons:
DM 40897 2499      43396      2499   54273.27      9996

```

Note we used the argument `drop=FALSE` which will retain follow-up also outside the window defined by the breaks. Otherwise the default for `splitMulti` would be to drop follow-up outside `age` [0,100] and `tfD` [0,40]. This clipping behaviour is not available in `splitLexis`, nevertheless this may be exactly what we want in some situations.

So we see that the two ways of splitting data yields the same amount of follow-up, but the results are not identical:

```

> identical( dmS2, dmM )
[1] FALSE
> class( dmS2 )
[1] "Lexis"      "data.frame"
> class( dmM )
[1] "Lexis"      "data.table" "data.frame"

```

As we see, this is because the `dmM` object also is a `data.table` object; the `splitMulti` uses the `data.table` machinery which makes the splitting substantially faster — this is of particular interest if you operate on large data sets (> 100,000 records).

Thus the recommended way of splitting follow-up time is by `splitMulti`. But you should be aware that the result is a `data.table` object, which in some circumstances behaves slightly different from `data.frames`. See the manual for `data.table`.

1.3 Cutting follow up time at dates of intermediate events

If we have a recording of the date of a specific event as for example recovery or relapse, we may classify follow-up time as being before or after this intermediate event, but it requires that follow-up records that straddle the event be cut in two and placed in separate records, one representing follow-up *before* the intermediate event, and another representing follow-up *after* the intermediate event. This is achieved with the function `cutLexis`, which takes three arguments: the time point of the intermediate event, the time scale that this point refers to, and the value of the (new) state following the date. Optionally, we may also define a new time scale with the argument `new.scale=`.

We are interested in the time before and after inception of insulin use, which occurs at the date `doins`:

```

> whc <- c(names(dmL)[1:7], "dodm", "doins") # Which Columns do we want to see?
> subset( dmL, lex.id %in% wh.id )[,whc]

```

	per	age	tfD	lex.dur	lex.Cst	lex.Xst	lex.id	dodm	doins
430048	1998.956	61.87269	0	9.508556	DM	Dead	9	1998.956	NA
22671	2000.042	52.71184	0	9.954825	DM	DM	27	2000.042	NA
338459	1998.249	61.85626	0	11.748118	DM	DM	52	1998.249	2004.804
274124	1998.260	62.37919	0	10.929500	DM	Dead	484	1998.260	2003.960

```

> dmC <- cutLexis( data = dmL,
+                 cut = dmL$doins,
+                 timescale = "per",
+                 new.state = "Ins",
+                 new.scale = "tfl",
+                 precursor.states = "DM" )
> whc <- c(names(dmL)[1:8], "doins") # Which Columns do we want to see?
> subset( dmC, lex.id %in% wh.id )[,whc]

```

	per	age	tfD	lex.dur	lex.Cst	lex.Xst	lex.id	sex	doins
9	1998.956	61.87269	0.000000	9.508556	DM	Dead	9	F	NA
27	2000.042	52.71184	0.000000	9.954825	DM	DM	27	M	NA
52	1998.249	61.85626	0.000000	6.554415	DM	Ins	52	F	2004.804
10048	2004.804	68.41068	6.554415	5.193703	Ins	Ins	52	F	2004.804
484	1998.260	62.37919	0.000000	5.700205	DM	Ins	484	F	2003.960
10480	2003.960	68.07940	5.700205	5.229295	Ins	Dead	484	F	2003.960

(The `precursor.states=` argument is explained below).

Note that the process of cutting time is simplified by having all types of events referred to the calendar time scale. This is a generally applicable advice in handling follow-up data: Get all event times as *dates*, location of events and follow-up on other time scales can then easily be derived from this.

Note that individual 52 has had his follow-up cut at 6.55 years from diabetes diagnosis and individual 484 at 5.70 years from diabetes diagnosis. This dataset could then be split along the time scales as we did before with `dmL`.

We can see which of the time scales that are defined as time since entry into an intermediate state:

```

> timeSince( dmC )
per    age    tfD    tfl
""      ""      "" "Ins"

```

The names of the vector are the time scales; each element is the name of the state entry into which defines the origin of the time scale. This resulting `Lexis` object can then be split along one or more time scales.

The result of this can however also be achieved by cutting the split dataset `dmS2` instead of `dmL`:

```

> dmS2C <- cutLexis( data = dmS2,
+                 cut = dmS2$doins,
+                 timescale = "per",
+                 new.state = "Ins",
+                 new.scale = "tfl",
+                 precursor.states = "DM" )
> subset( dmS2C, lex.id %in% wh.id )[,whc]

```

	per	age	tfD	lex.dur	lex.Cst	lex.Xst	lex.id	sex	doins
31	1998.956	61.87269	0.000000	1.0000000	DM	DM	9	F	NA
32	1999.956	62.87269	1.000000	1.0000000	DM	DM	9	F	NA
33	2000.956	63.87269	2.000000	1.1273101	DM	DM	9	F	NA
34	2002.083	65.00000	3.127310	1.8726899	DM	DM	9	F	NA
35	2003.956	66.87269	5.000000	3.1273101	DM	DM	9	F	NA
36	2007.083	70.00000	8.127310	1.3812457	DM	Dead	9	F	NA
111	2000.042	52.71184	0.000000	1.0000000	DM	DM	27	M	NA
112	2001.042	53.71184	1.000000	1.0000000	DM	DM	27	M	NA
113	2002.042	54.71184	2.000000	0.2881588	DM	DM	27	M	NA

114	2002.331	55.00000	2.288159	2.7118412	DM	DM	27	M	NA
115	2005.042	57.71184	5.000000	2.2881588	DM	DM	27	M	NA
116	2007.331	60.00000	7.288159	2.6666667	DM	DM	27	M	NA
229	1998.249	61.85626	0.000000	1.0000000	DM	DM	52	F	2004.804
230	1999.249	62.85626	1.000000	1.0000000	DM	DM	52	F	2004.804
231	2000.249	63.85626	2.000000	1.1437372	DM	DM	52	F	2004.804
232	2001.393	65.00000	3.143737	1.8562628	DM	DM	52	F	2004.804
233	2003.249	66.85626	5.000000	1.5544148	DM	Ins	52	F	2004.804
43629	2004.804	68.41068	6.554415	1.5893224	Ins	Ins	52	F	2004.804
43630	2006.393	70.00000	8.143737	1.8562628	Ins	Ins	52	F	2004.804
43631	2008.249	71.85626	10.000000	1.7481177	Ins	Ins	52	F	2004.804
2084	1998.260	62.37919	0.000000	1.0000000	DM	DM	484	F	2003.960
2085	1999.260	63.37919	1.000000	1.0000000	DM	DM	484	F	2003.960
2086	2000.260	64.37919	2.000000	0.6208077	DM	DM	484	F	2003.960
2087	2000.881	65.00000	2.620808	2.3791923	DM	DM	484	F	2003.960
2088	2003.260	67.37919	5.000000	0.7002053	DM	Ins	484	F	2003.960
45484	2003.960	68.07940	5.700205	1.9206023	Ins	Ins	484	F	2003.960
45485	2005.881	70.00000	7.620808	2.3791923	Ins	Ins	484	F	2003.960
45486	2008.260	72.37919	10.000000	0.9295003	Ins	Dead	484	F	2003.960

Thus it does not matter in which order we use `splitLexis` and `cutLexis`. Mathematicians would say that `splitLexis` and `cutLexis` are commutative.

Note in `lex.id=484`, that follow-up subsequent to the event is classified as being in state `Ins`, but that the final transition to state `Dead` is preserved. This is the point of the `precursor.states=` argument. It names the states (in this case `DM`) that will be over-written by `new.state` (in this case `Ins`), while the state `Dead` should not be updated even if it is after the time where the persons moves to state `Ins`. In other words, only state `DM` is a precursor to state `Ins`, state `Dead` is always subsequent to state `Ins`.

Note that we defined a new time scale, `tfI`, using the argument `new.scale=tfI`. This has a special status relative to the other three time scales, it is defined as time since entry into a state, namely `Ins`, this is noted in the time scale part of the summary of `Lexis` object — the information sits in the attribute `time.since` of the `Lexis` object, which can be accessed by the function `timeSince()` or through the `summary()`:

```
> summary( dmS2C, timeScales=TRUE )
Transitions:
      To
From    DM  Ins Dead  Records:  Events: Risk time:  Persons:
DM  35135 1694 2048    38877    3742  45885.49    9899
Ins      0 5762  451     6213     451   8387.77    1791
Sum 35135 7456 2499    45090    4193  54273.27    9996

Timescales:
  per   age   tfD   tfI
  ""    ""    ""  "Ins"
```

Finally we can get a quick overview of the states and transitions by using `boxes` — `scale.R` scales transition rates to rates per 1000 PY:

```
> boxes( dmC, boxpos=TRUE, scale.R=1000, show.BE=TRUE )
```

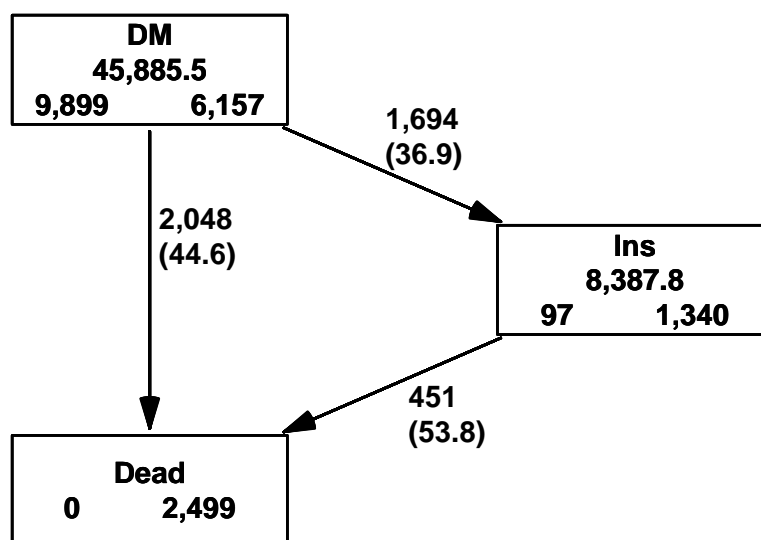


Figure 1.3: States, person years, transitions and rates in the cut dataset. The numbers in the boxes are person-years and the number of persons *Beginning*, resp. *Ending* their follow-up in each state (triggered by `show.BE=TRUE`). The numbers at the arrows are the number of transitions and transition rates per 1000 (triggered by `scale.R=1000`). ./flup-box1

Chapter 2

Modeling rates from Lexis objects

2.1 Covariates

In the dataset `dmS2C` there are three types of covariates that can be used to describe mortality rates:

1. time-dependent covariates
2. time scales
3. fixed covariates

There is only one time-dependent covariate here, namely `lex.Cst`, the current state of the person's follow up; it takes the values `DM` and `Ins` according to whether the person has ever purchased insulin at a given time of follow-up.

The time-scales are obvious candidates for explanatory variables for the rates, notably age and time from diagnosis (duration of diabetes) and insulin.

2.1.1 Time scales as covariates

If we want to model the effect of the time scale variables on occurrence rates, we will for each interval use either the value of the left endpoint in each interval or the middle. There is a function `timeBand` which returns either of these:

```
> timeBand( dmS2C, "age", "middle" )[1:10]
[1] 57.5 57.5 62.5 62.5 62.5 67.5 67.5 62.5 67.5 67.5

> # For nice printing and column labelling we use the data.frame() function:
> data.frame( dmS2C[,c("per", "age", "tfD", "lex.dur")],
+             mid.age=timeBand( dmS2C, "age", "middle" ),
+             mid.t=timeBand( dmS2C, "tfD", "middle" ),
+             left.t=timeBand( dmS2C, "tfD", "left" ),
+             right.t=timeBand( dmS2C, "tfD", "right" ),
+             fact.t=timeBand( dmS2C, "tfD", "factor" ) )[1:15,]
  per      age      tfD    lex.dur mid.age mid.t left.t right.t fact.t
1 1998.917 58.66119 0.0000000 1.0000000    57.5  0.5     0       1  (0,1]
2 1999.917 59.66119 1.0000000 0.3388090    57.5  1.5     1       2  (1,2]
3 2000.256 60.00000 1.3388090 0.66119097   62.5  1.5     1       2  (1,2]
4 2000.917 60.66119 2.0000000 3.0000000    62.5  3.5     2       5  (2,5]
5 2003.917 63.66119 5.0000000 1.33880903   62.5  7.5     5      10  (5,10]
```

6	2005.256	65.00000	6.3388090	3.66119097	67.5	7.5	5	10	(5,10]
7	2008.917	68.66119	10.0000000	1.08008214	67.5	15.0	10	20	(10,20]
8	2003.309	64.09035	0.0000000	0.90965092	62.5	0.5	0	1	(0,1]
9	2004.218	65.00000	0.9096509	0.09034908	67.5	0.5	0	1	(0,1]
10	2004.309	65.09035	1.0000000	1.00000000	67.5	1.5	1	2	(1,2]
11	2005.309	66.09035	2.0000000	3.00000000	67.5	3.5	2	5	(2,5]
12	2008.309	69.09035	5.0000000	0.90965092	67.5	7.5	5	10	(5,10]
13	2009.218	70.00000	5.9096509	0.77891855	72.5	7.5	5	10	(5,10]
14	2004.552	86.25051	0.0000000	1.00000000	87.5	0.5	0	1	(0,1]
15	2005.552	87.25051	1.0000000	1.00000000	87.5	1.5	1	2	(1,2]

Note that the values of these functions are characteristics of the intervals defined by `breaks=`, *not* the midpoints nor left or right endpoints of the actual follow-up intervals (which would be `tfD` and `tfD+lex.dur`, respectively).

These functions are intended for modeling time scale variables as factors (categorical variables) in which case the coding must be independent of the censoring and mortality pattern — it should only depend on the chosen grouping of the time scale. Modeling time scales as *quantitative* should not be based on these codings but directly on the values of the time-scale variables, notably the left endpoints of the intervals.

2.1.2 Differences between time scales

Apparently, the only fixed variable is `sex`, but formally the dates of birth (`dobth`), diagnosis (`dodm`) and first insulin purchase (`doins`) are fixed covariates too. They can be constructed as origins of time scales referred to the calendar time scale. Likewise, and possibly of greater interest, we can consider these origins on the age scale, calculated as the difference between age and another time scale.

These would then be age at birth (hardly relevant), age at diabetes diagnosis and age at insulin treatment.

2.1.3 Keeping the relation between time scales

The midpoint (as well as the right interval endpoint) should be used with caution if the variable age at diagnosis `dodm-dobth` is modeled too; the age at diabetes is logically equal to the difference between current age (`age`) and time since diabetes diagnosis (`tfD`):

```
> summary( (dmS2$age-dmS2$tfD) - (dmS2$dodm-dmS2$dobth) )
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0         0         0         0         0         0
```

This calculation refers to the *start* of each interval — which are in the time scale variables in the *Lexis* object. But when using the middle of the intervals, this relationship is not preserved:

```
> summary( timeBand( dmS2, "age", "middle" ) -
+           timeBand( dmS2, "tfD", "middle" ) - (dmS2$dodm-dmS2$dobth) )
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -7.4870 -2.0862 -0.3765     Inf  1.3641     Inf
```

If all three variables are to be included in a model, we must make sure that the *substantial* relationship between the variables be maintained. One way is to recompute age at diabetes

diagnosis from the two midpoint variables, but more straightforward would be to use the left endpoint of the intervals, that is the time scale variables in the `Lexis` object.

If we dissolve the relationship between the variables `age`, `tfD` and age at diagnosis by grouping we may obtain identifiability of the three separate effects, but it will be at the price of an arbitrary allocation of a linear trend between them.

For the sake of clarity, consider current age, a , duration of disease, d and age at diagnosis e , where

$$\text{current age} = \text{age at diagnosis} + \text{disease duration}, \quad \text{i.e.} \quad a = e + d \quad \Leftrightarrow \quad e + d - a = 0$$

If we model the effect of the quantitative variables a , e and d on the log-rates by three functions f , g and h : $\log(\lambda) = f(a) + g(d) + h(e)$ then for any κ :

$$\begin{aligned} \log(\lambda) &= f(a) + g(d) + h(e) + \kappa(e + d - a) \\ &= (f(a) - \kappa a) + (g(d) + \kappa d) + (h(e) + \kappa e) \\ &= \tilde{f}(a) + \tilde{g}(d) + \tilde{h}(e) \end{aligned}$$

In practical modeling this will turn up as a singular model matrix with one parameter aliased, corresponding to some arbitrarily chosen value of κ (depending on software conventions for singular models). This is well known from age-period-cohort models.

Thus we see that we can move any slope around between the three terms, so if we achieve identifiability by using grouping of one of the variables we will in reality have settled for a particular value of κ , without known why we chose just that. The solution is to resort to predictions which are independent of the particular parametrization or choose a particular parametrization with explicit constraints.

2.2 Modeling of rates

As mentioned, the purpose of subdividing follow-up data in smaller intervals is to be able to model effects of time scale variables as parametric functions. When we split along a time scale we can get intervals that are as small as we want; if they are sufficiently small, an assumption of constant rates in each interval becomes reasonable.

In a model that assumes a constant occurrence rate in each of the intervals the likelihood contribution from each interval is the same as the likelihood contribution from a Poisson variate D , say, with mean $\lambda\ell$ where λ is the rate and ℓ is the interval length, and where the value of the variate D is 1 or 0 according to whether an event has occurred or not. Moreover, the likelihood contributions from all follow-up intervals from a single person are *conditionally* independent (conditional on having survived till the start of the interval in question). This implies that the total contribution to the likelihood from a single person is a product of terms, and hence the same as the likelihood of a number of independent Poisson terms, one from each interval.

Note that variables are neither Poisson distributed (*e.g.* they can only ever assume values 0 or 1) nor independent — it is only the likelihood for the follow-up data that happens to be the same as the likelihood from independent Poisson variates. Different models can have the same likelihood, a model cannot be inferred from the likelihood.

Parametric modeling of the rates is obtained by using the *values* of the time scales for each interval as *quantitative* explanatory variables, using for example splines. And of

course also the values of the fixed covariates and the time-dependent variables for each interval. Thus the model will be one where the rate is assumed constant in each (small) interval, but where a parametric form of the *size* of the rate in each interval is imposed by the model, using the time scale as a quantitative covariate.

2.2.1 Interval length

In the first chapter we illustrated cutting and splitting by listing the results for a few individuals across a number of intervals. For illustrational purposes we used 5-year age bands to avoid excessive listings, but since the doubling time for mortality on the age scale is only slightly larger than 5 years, the assumption about constant rates in each interval would be pretty far fetched if we were to use 5 year intervals.

Thus, for modeling purposes we split the follow-up in 3 month intervals. When we use intervals of 3 month's length it is superfluous to split along multiple time scales — the precise location of tightly spaced splits will be irrelevant from any practical point of view. `splitLexis` and `splitMulti` will allocate the actual split times for all of the time scale variables, so these can be used directly in modeling.

So we split the cut dataset in 3-month intervals along the age scale:

```
> dmCs <- splitMulti( dmC, age = seq(0,110,1/4) )
> summary( dmCs, t=T )
```

Transitions:

	To						
From	DM	Ins	Dead	Records:	Events:	Risk time:	Persons:
DM	189669	1694	2048	193411	3742	45885.49	9899
Ins	0	34886	451	35337	451	8387.77	1791
Sum	189669	36580	2499	228748	4193	54273.27	9996

Timescales:

per	age	tfD	tfI
""	""	""	"Ins"

We see that we now have 228,748 records and 9996 persons, so about 23 records per person. The total risk time is 54,275 years, a bit less than 3 months per record as expected.

2.2.2 Practicalities for splines

In this study we want to look at how mortality depend on age (`age`) and time since start of insulin use (`tfI`). If we want to use splines in the description we must allocate knots for anchoring the splines at each of the time scales, either by some *ad hoc* method or by using some sort of penalized splines. The latter will not be treated here; it belongs in the realm of the `mgcv` package.

Here we shall use the former approach and allocate 5 knots on each of the time-scales. We allocate knots so that we have the events evenly distributed between the knots. Since the insulin state starts at 0 for all individuals we include 0 as the first knot, such that any set of natural splines with these knots will have the value 0 at 0 on the time scale.

```
> ( a.kn <- with( subset( dmCs, lex.Xst=="Dead" ),
+               quantile( age+lex.dur, (1:5-0.5)/5 ) ) )
      10%      30%      50%      70%      90%
60.29350 71.31937 77.72758 82.72745 89.86393
```

```
> ( i.kn <- c( 0,
+           with( subset( dmCs, lex.Xst=="Dead" & lex.Cst=="Ins" ),
+               quantile( tfl+lex.dur, (1:4)/5 ) ) ) )
                20%      40%      60%      80%
0.0000000 0.3093771 1.1307324 2.5489391 4.9117043
```

In the `Epi` package there is a convenience wrapper, `Ns`, for the natural spline generator `ns`, that takes the smallest and the largest of a set of supplied knots to be the boundary knots, so the explicit definition of the boundary knots becomes superfluous.

Note that it is a feature of the `Ns` (via the features of `ns`) that any generated spline function is 0 at the leftmost knot.

2.2.3 Poisson model

A model that describes mortality rates as only a function of age would then be:

```
> ma <- glm( (lex.Xst=="Dead") ~ Ns(age,knots=a.kn),
+           family = poisson,
+           offset = log(lex.dur),
+           data = dmCs )
> summary( ma )
Call:
glm(formula = (lex.Xst == "Dead") ~ Ns(age, knots = a.kn), family = poisson,
    data = dmCs, offset = log(lex.dur))
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-0.5883	-0.1688	-0.1144	-0.0766	4.5958

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-3.82830	0.03861	-99.16	<2e-16
Ns(age, knots = a.kn)1	1.36254	0.08723	15.62	<2e-16
Ns(age, knots = a.kn)2	1.49446	0.06845	21.83	<2e-16
Ns(age, knots = a.kn)3	2.63557	0.07058	37.34	<2e-16
Ns(age, knots = a.kn)4	1.94173	0.05769	33.66	<2e-16

(Dispersion parameter for poisson family taken to be 1)

```
Null deviance: 27719 on 228747 degrees of freedom
Residual deviance: 25423 on 228743 degrees of freedom
AIC: 30431
```

Number of Fisher Scoring iterations: 8

The offset, `log(lex.dur)` comes from the fact that the likelihood for the follow-up data during ℓ time is the same as that for independent Poisson variates with mean $\lambda\ell$, and that the default link function for the Poisson family is the log, so that we are using a linear model for the log-mean, $\log(\lambda) + \log(\ell)$. But when we want a model for the log-rate ($\log(\lambda)$), the term $\log(\ell)$ must still be included as a covariate, but with regression coefficient fixed to 1; a so-called *offset*. This is however a technicality; it just exploits that the likelihood of a particular Poisson model and that of the rates model is the same.

In the `Epi` package is a `glm` family, `poisreg` that has a more intuitive interface, where the response is a 2-column matrix of events and person-time, respectively. This is in

concert with the fact that the outcome variable in follow-up studies is bivariate: (event, risk time).

```
> Ma <- glm( cbind(lex.Xst=="Dead",lex.dur) ~ Ns(age,knots=a.kn),
+           family = poisreg, data = dmCs )
> summary( Ma )
```

Call:
 glm(formula = cbind(lex.Xst == "Dead", lex.dur) ~ Ns(age, knots = a.kn),
 family = poisreg, data = dmCs)

Deviance Residuals:

Min	1Q	Median	3Q	Max
-0.5883	-0.1688	-0.1144	-0.0766	4.5958

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-3.82830	0.03861	-99.15	<2e-16
Ns(age, knots = a.kn)1	1.36254	0.08723	15.62	<2e-16
Ns(age, knots = a.kn)2	1.49446	0.06845	21.83	<2e-16
Ns(age, knots = a.kn)3	2.63557	0.07058	37.34	<2e-16
Ns(age, knots = a.kn)4	1.94173	0.05769	33.66	<2e-16

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 27719 on 228747 degrees of freedom
 Residual deviance: 25423 on 228743 degrees of freedom
 AIC: 30431

Number of Fisher Scoring iterations: 7

Exploiting the multistate structure in the *Lexis* object there is a multistate convenience wrapper for `glm` with the `poisreg` family, that just requires specification of the transitions in terms of `from` and `to`. Although it is called `glm.Lexis` it is *not* an S3 method for *Lexis* objects:

```
> Xa <- glm.Lexis( dmCs, from="DM", to="Dead",
+               formula = ~ Ns(age,knots=a.kn) )
```

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

The result is a `glm` object but with an extra attribute, `Lexis`:

```
> attr( Xa, "Lexis" )
```

\$data
 [1] "dmCs"

\$trans
 [1] "DM->Dead"

\$formula
 ~Ns(age, knots = a.kn)
 <environment: 0xbe99468>

\$scale
 [1] 1

There are similar wrappers for `gam` and `coxph` models, `gam.Lexis` and `coxph.Lexis`, but these will not be elaborated in detail.

The `from=` and `to=` can even be omitted, in which case all possible transitions *into* any of the absorbing states is modeled:

```
> xa <- glm.Lexis( dmCs, formula = ~ Ns(age,knots=a.kn) )
stats::glm Poisson analysis of Lexis object dmCs with log link:
Rates for transitions: DM->Dead, Ins->Dead
```

We can check if the four models fitted are the same:

```
> c( deviance(ma), deviance(Ma), deviance(Xa), deviance(xa) )
[1] 25422.92 25422.92 20902.31 25422.92
```

Oops! the model `Xa` is apparently not the same as the other three? This is because the explicit specification `from="DM", to="Dead"`, omits modeling contributions from the `Ins → Dead` transition — the output actually said so. The other three models all use both transitions — and assume that the two transition rates are the same, *i.e.* that start of insulin has no effect on mortality. We shall relax this assumption later.

The parameters from the model do not have any direct interpretation *per se*, but we can compute the estimated mortality rates for a range of ages using `ci.pred` with a suitably defined prediction data frame.

Note that if we use the `poisson` family of models, we must specify all covariates in the model, including the variable in the offset, `lex.dur` (remember that this was a covariate with coefficient fixed at 1). We set the latter to 1000, because we want the mortality rates per 1000 person-years. Using the `poisreg` family, the prediction will ignore any value of `lex.dur` specified in the prediction data frame, the returned rates will be per unit in which `lex.dur` is recorded.

```
> nd <- data.frame( age=40:85, lex.dur=1000 )
> pr.0 <- ci.pred( ma, newdata = nd )      # mortality per 100 PY
> pr.a <- ci.pred( Ma, newdata = nd )*1000 # mortality per 100 PY
> summary(pr.0/pr.a)
      Estimate      2.5%      97.5%
Min.      :1  Min.      :1  Min.      :1
1st Qu.:1  1st Qu.:1  1st Qu.:1
Median :1  Median :1  Median :1
Mean      :1  Mean      :1  Mean      :1
3rd Qu.:1  3rd Qu.:1  3rd Qu.:1
Max.      :1  Max.      :1  Max.      :1
> matshade( nd$age, pr.a, plot=TRUE,
+           type="l", lty=1,
+           log="y", xlab="Age (years)",
+           ylab="DM mortality per 1000 PY")
```

2.3 Time dependent covariate

A Poisson model approach to mortality by insulin status, would be to assume that the rate-ratio between patients on insulin and not on insulin is a fixed quantity, independent of time since start of insulin, independent of age. This is commonly termed a proportional hazards assumption, because the rates (hazards) in the two groups are proportional along the age (baseline time) scale.

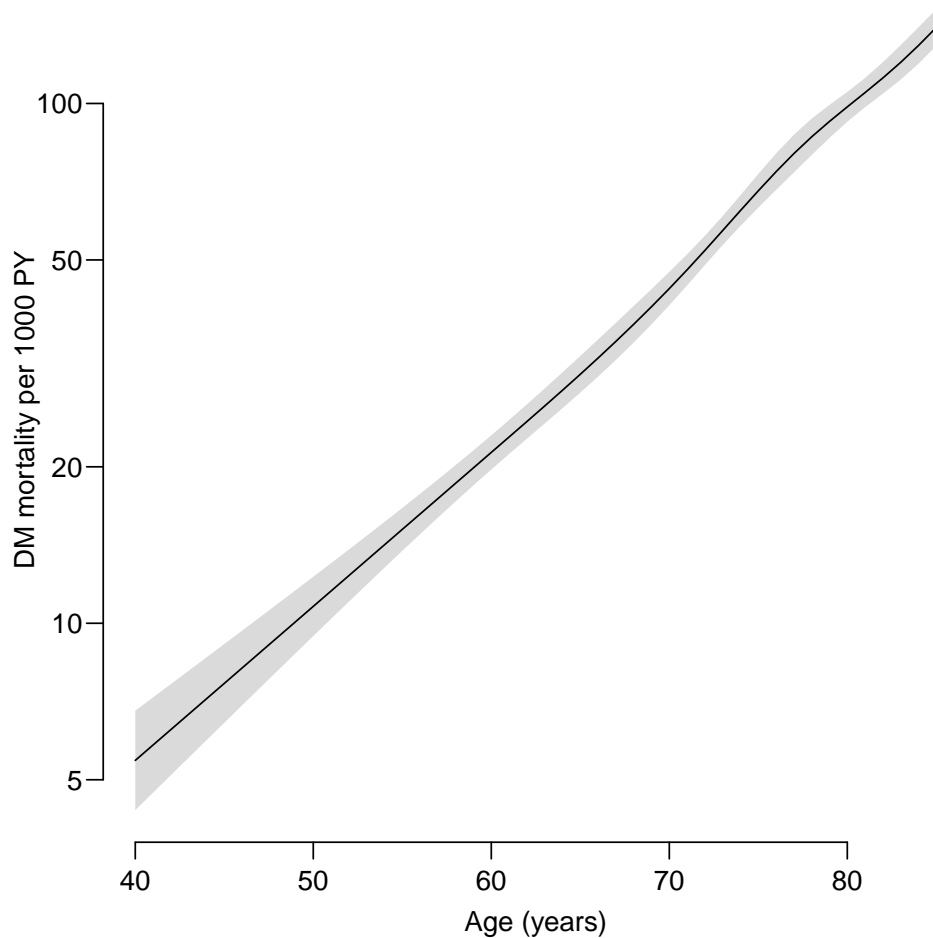


Figure 2.1: Mortality among Danish diabetes patients by age with 95% CI as shaded area. We see that the rates increase linearly on the log-scale, that is exponentially by age. `./flup-pr-a`

```
> pm <- glm( cbind(lex.Xst=="Dead",lex.dur) ~ Ns(age,knots=a.kn)
+                                     + lex.Cst + sex,
+                                     family=poisreg, data = dmCs )
> round( ci.exp( pm ), 3 )
```

	exp(Est.)	2.5%	97.5%
(Intercept)	0.022	0.021	0.024
Ns(age, knots = a.kn)1	4.248	3.581	5.040
Ns(age, knots = a.kn)2	5.008	4.376	5.731
Ns(age, knots = a.kn)3	16.832	14.624	19.373
Ns(age, knots = a.kn)4	7.994	7.126	8.968
lex.CstIns	1.985	1.791	2.200
sexF	0.668	0.617	0.724

So we see that persons on insulin have about twice the mortality of persons not on insulin and that women have 2/3 the mortality of men.

2.3.1 Time since insulin start

If we want to test whether the excess mortality depends on the time since start if insulin treatment, we can add a spline terms in `tfI`. But since `tfI` is a time scale defined as time

since entry into a new state (*Ins*), the variable *tfI* will be missing for those in the *DM* state, so before modeling we must set the *NAs* to 0, which we do with *tsNA20* (acronym for timescale *NAs* to zero):

```
> pm <- glm( cbind(lex.Xst=="Dead",lex.dur) ~ Ns(age,knots=a.kn)
+                                             + Ns(tfI,knots=i.kn)
+                                             + lex.Cst + sex,
+                                             family=poisreg, data = tsNA20(dmCs) )
```

As noted before we could do this simpler with *glm.Lexis*, even without the *from=* and *to=* arguments, because we are modeling all transitions *into* the absorbing state (*Dead*):

```
> Pm <- glm.Lexis( tsNA20(dmCs),
+                 form = ~ Ns(age,knots=a.kn)
+                       + Ns(tfI,knots=i.kn)
+                       + lex.Cst + sex )
stats::glm Poisson analysis of Lexis object tsNA20(dmCs) with log link:
Rates for transitions: DM->Dead, Ins->Dead
> c( deviance(Pm), deviance(pm) )
[1] 25096.33 25096.33
> identical( model.matrix(Pm), model.matrix(pm) )
[1] TRUE
```

The coding of the effect of *tfI* is so that the value is 0 at 0, so the meaning of the estimate of the effect of *lex.Cst* is the RR between persons with and without insulin, immediately after start of insulin:

```
> round( ci.exp( Pm, subset="ex" ), 3 )
           exp(Est.)  2.5% 97.5%
lex.CstIns      5.632 4.430  7.16
sexF            0.674 0.622  0.73
```

We see that the effect of *sex* is pretty much the same as before, but the effect of *lex.Cst* is much larger, it now refers to a different quantity, namely the RR at *tfI*=0. If we want to see the effect of time since insulin, it is best viewed jointly with the effect of *age*:

```
> ndI <- data.frame( expand.grid( tfI=c(NA,seq(0,15,0.1)),
+                               ai=seq(40,80,10) ),
+                   sex="M",
+                   lex.Cst="Ins" )
> ndI <- transform( ndI, age=ai+tfI )
> head( ndI )
  tfI ai sex lex.Cst age
1  NA 40  M      Ins  NA
2 0.0 40  M      Ins 40.0
3 0.1 40  M      Ins 40.1
4 0.2 40  M      Ins 40.2
5 0.3 40  M      Ins 40.3
6 0.4 40  M      Ins 40.4
> ndA <- data.frame( age= seq(40,100,0.1), tfI=0, lex.Cst="DM", sex="M" )
> pri <- ci.pred( Pm, ndI ) * 1000
> pra <- ci.pred( Pm, ndA ) * 1000
> matshade( ndI$age, pri, plot=TRUE, las=1,
+           xlab="Age (years)", ylab="DM mortality per 1000 PY",
+           log="y", lty=1, col="blue" )
> matshade( ndA$age, pra )
```

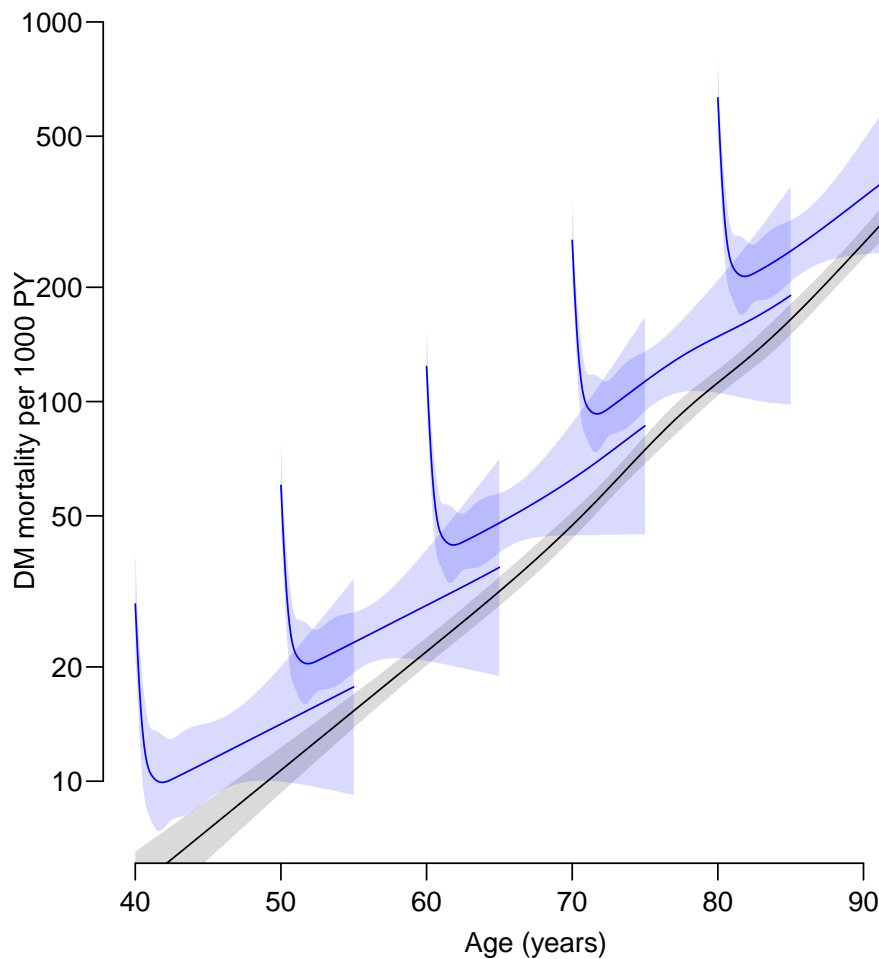


Figure 2.2: Mortality rates of persons on insulin, starting insulin at ages 40, 50, ..., 80 (blue), compared with persons not on insulin (black curve). Shaded areas are 95% CI./flup-ins-time

In figure 2.2, p. 21, we see that mortality is high just after insulin start, but falls by almost a factor 3 during the first year. Also we see that there is a tendency that mortality in a given age is smallest for those with the longest duration of insulin use.

2.4 The Cox model

Note that in the Cox-model the age is used as response variable, slightly counter-intuitive. Hence the age part of the linear predictors is not in that model:

```
> library( survival )
> cm <- coxph( Surv(age,age+lex.dur,lex.Xst=="Dead") ~
+             Ns(tfI,knots=i.kn) + lex.Cst + sex,
+             data = tsNA20(dmCs) )
```

There is also a multistate wrapper for Cox models, requiring a l.h.s. side for the `formula=` argument:

```
> Cm <- coxph.Lexis( tsNA20(dmCs),
+                   form= age ~ Ns(tfI,knots=i.kn) + lex.Cst + sex )
```

```

model survival::coxph analysis of Lexis object tsNA20(dmCs):
Rates for transitions DM->Dead, Ins->Dead
> cbind( ci.exp( cm ), ci.exp( Cm ) )

```

	exp(Est.)	2.5%	97.5%	exp(Est.)	2.5%	97.5%
Ns(tfI, knots = i.kn)1	0.2984062	0.19417148	0.4585960	0.2984062	0.19417148	0.4585960
Ns(tfI, knots = i.kn)2	0.3871151	0.29011380	0.5165495	0.3871151	0.29011380	0.5165495
Ns(tfI, knots = i.kn)3	0.1239128	0.06287008	0.2442238	0.1239128	0.06287008	0.2442238
Ns(tfI, knots = i.kn)4	0.4405121	0.34839015	0.5569932	0.4405121	0.34839015	0.5569932
lex.CstIns	5.6700284	4.45011220	7.2243623	5.6700284	4.45011220	7.2243623
lex.CstDead	1.0000000	1.00000000	1.0000000	1.0000000	1.00000000	1.0000000
sexF	0.6753202	0.62316569	0.7318397	0.6753202	0.62316569	0.7318397

We can compare the estimates from the Cox model with those from the Poisson model — we must add NAs because the Cox-model does not give the parameters for the baseline time scale (*age*), but also remove one of the parameters, because `coxph` parametrizes factors (in this case `lex.Cst`) by all defined levels and not only by the levels present in the dataset at hand (note the line of 1.0000000s in the print above):

```

> round( cbind( ci.exp( Pm ),
+               rbind( matrix(NA,5,3),
+               ci.exp( cm )[-6,] ) ), 3 )

```

	exp(Est.)	2.5%	97.5%	exp(Est.)	2.5%	97.5%
(Intercept)	0.022	0.021	0.024	NA	NA	NA
Ns(age, knots = a.kn)1	4.208	3.546	4.993	NA	NA	NA
Ns(age, knots = a.kn)2	5.012	4.380	5.736	NA	NA	NA
Ns(age, knots = a.kn)3	16.560	14.386	19.063	NA	NA	NA
Ns(age, knots = a.kn)4	7.921	7.061	8.885	NA	NA	NA
Ns(tfI, knots = i.kn)1	0.298	0.194	0.458	0.298	0.194	0.459
Ns(tfI, knots = i.kn)2	0.385	0.289	0.514	0.387	0.290	0.517
Ns(tfI, knots = i.kn)3	0.125	0.064	0.246	0.124	0.063	0.244
Ns(tfI, knots = i.kn)4	0.438	0.346	0.553	0.441	0.348	0.557
lex.CstIns	5.632	4.430	7.160	5.670	4.450	7.224
sexF	0.674	0.622	0.730	0.675	0.623	0.732

Thus we see that the Poisson and Cox gives pretty much the same results. You may argue that Cox requires a smaller dataset, because there is no need to subdivide data in small intervals *before* insulin use. But certainly the time *after* insulin inception need to be if the effect of this time should be modeled.

The drawback of the Cox-modeling is that it is not possible to show the absolute rates as we did in the graph above.

2.5 Marginal effect of time since insulin

When we plot the marginal effect of `tfI` from the two models we get pretty much the same; here we plot the RR relative to `tfI=2` years. Note that we are deriving the RR as the ratio of two sets of predictions, from the data frames `nd` and `nr` — for further details consult the help page for `ci.lin`, specifically the use of a list as the `ctr.mat` argument:

```

> nd <- data.frame( tfI=seq(0,15,,151), lex.Cst="Ins", sex="M" )
> nr <- data.frame( tfI=      2      , lex.Cst="Ins", sex="M" )
> ppr <- ci.exp( pm, list(nd,nr), xvars="age" )
> cpr <- ci.exp( cm, list(nd,nr) )

```



```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6, las=1, bty="n" )
> matshade( nd$tfI, cbind(ppr,cpr), plot=T,
+           lty=c(1,2), log="y",
+           xlab="Time since insulin (years)", ylab="Rate ratio")
> abline( h=1, lty=3 )
```

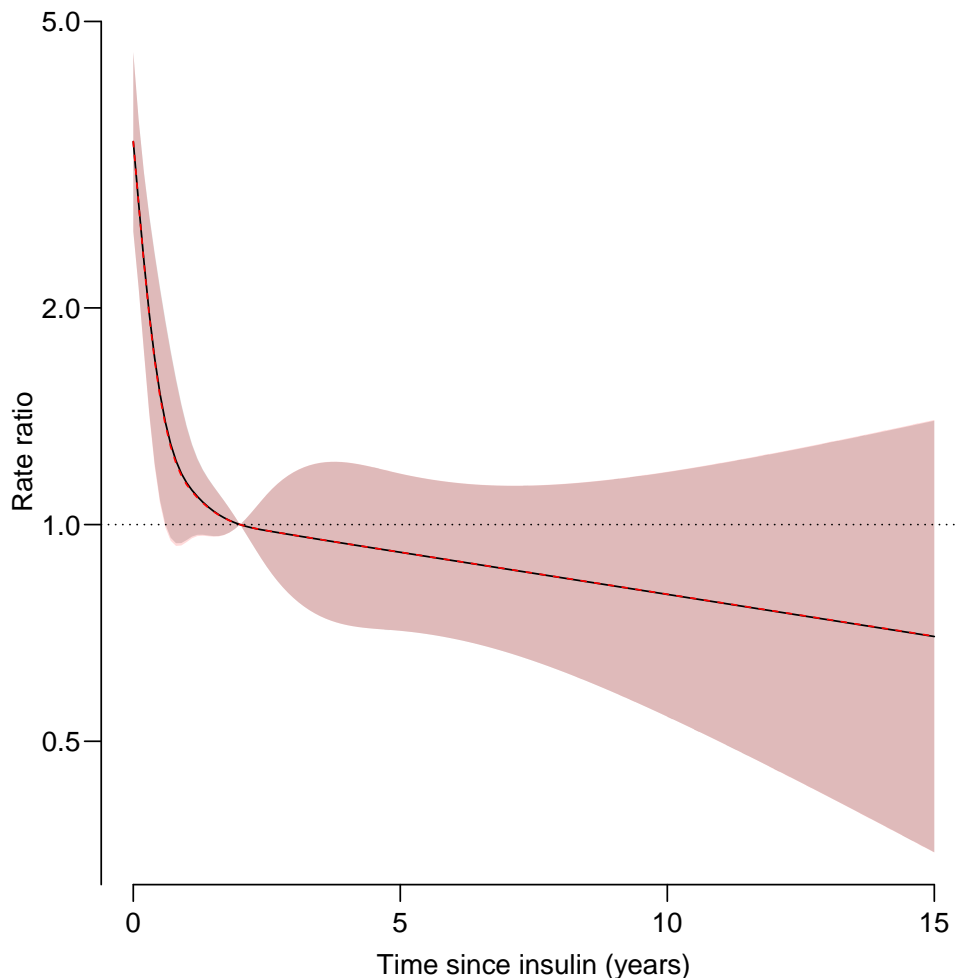


Figure 2.3: *The naked duration effects relative to 2 years of duration, black from Poisson model, red from Cox model. The two sets of estimates are identical, and so are the standard errors, so the two shaded areas overlap almost perfectly.*

./flup-Ieff

In figure 2.3, p. 23, we see that the duration effect is exactly the same from the two modeling approaches.

We will also want the RR relative to the non-insulin users — recall these are coded 0 on the `tfI` variable:

```
> nd <- data.frame( tfI=seq(0,15,,151), lex.Cst="Ins", sex="M" )
> nr <- data.frame( tfI= 0, lex.Cst="DM", sex="M" )
> ppr <- ci.exp( pm, list(nd,nr), xvars="age" )
> cpr <- ci.exp( cm, list(nd,nr) )
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6, las=1, bty="n" )
> matshade( nd$tfI, cbind(ppr,cpr),
+           xlab="Time since insulin (years)",
```

```
+      ylab="Rate ratio relative to non-Insulin",
+      lty=c(1,2), log="y", plot=T )
```

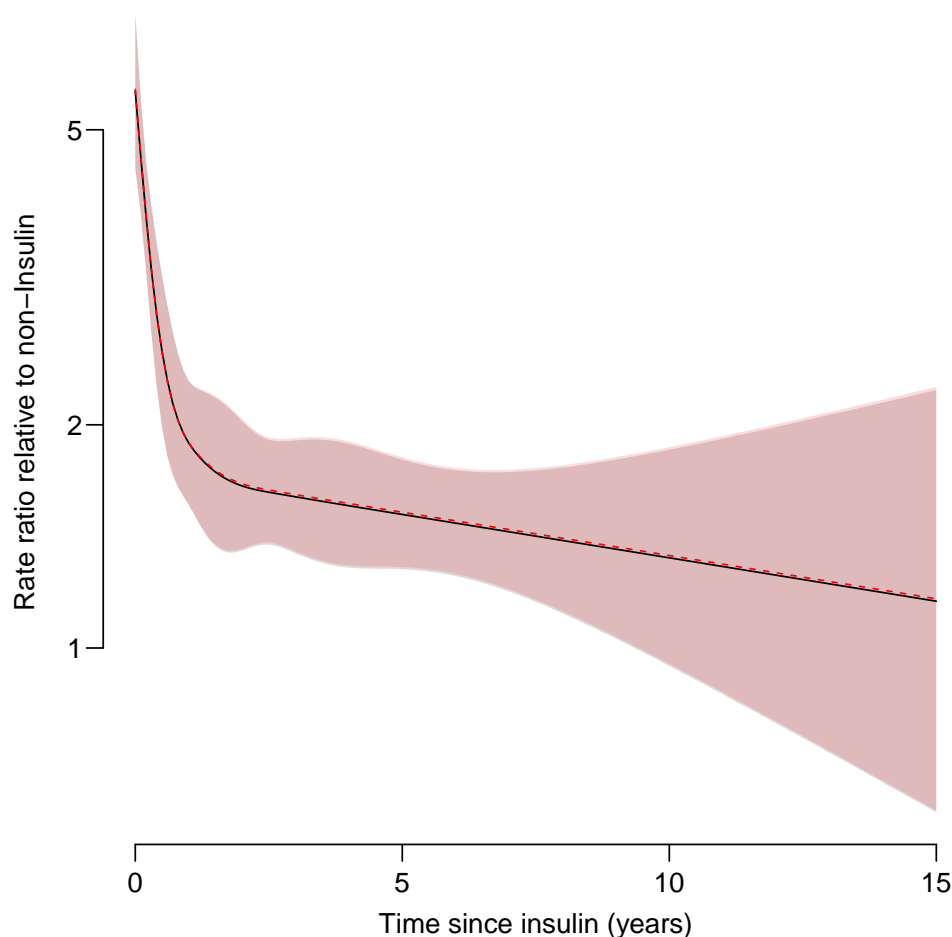


Figure 2.4: *Insulin duration effect (state **Ins**) relative to no insulin (state **DM**), black from Poisson model, red from Cox model. The shape is the same as the previous figure, but the RR is now relative to non-insulin, instead of relative to insulin users at 2 years duration. The two sets of estimates are identical, and so are the standard errors, so the two shaded areas overlap almost perfectly.*

`./flup-IeffR`

In figure 2.4, p. 24, we see the effect of increasing duration of insulin use *for a fixed age* which is a bit artificial, so we would like to see the *joint* effects of age and insulin duration. What we cannot see is how the duration affects mortality relative to **current** age (at the age attained at the same time as the attained **tfI**).

Another way of interpreting this curve is as the rate ratio relative to a person not on insulin, so we see that the RR (or hazard ratio, HR as some call it) is over 5 at the start of insulin (the `lex.Cst` estimate), and decreases to about 1.5 in the long term.

Both figure 2.3 and 2.4 indicate a declining RR by insulin duration, but only from figure 2.2 it is visible that mortality actually is *increasing* by age after some 2 years after insulin start. This point would not be available if we had only fitted a Cox model where we did not have access to the baseline hazard as a function of age.

2.6 Age×duration interaction

The model we fitted assumes that the RR is the same regardless of the age at start of insulin — the hazards are multiplicative. Sometimes this is termed the proportional hazards assumption: For *any* fixed age the HR is the same as a function of time since insulin, and vice versa.

A more correct term would be “main effects model” — there is no interaction between age (the baseline time scale) and other covariates. So there is really no need for the term “proportional hazards”; well defined and precise statistical terms for it has existed for aeons.

2.6.1 Age at insulin start

In order to check the consistency of the multiplicativity assumption across the spectrum of age at insulin inception, we can fit an interaction model. One approach to this would be using a non-linear effect of age at insulin use (for convenience we use the same knots as for age) — note that the prediction data frames are the same as we used above, because we do not compute age at insulin use as a separate variable, but rather enter it as the difference between current age (`age`) and insulin duration (`tfI`).

At first glance we might think of doing:

```
> imx <- glm.Lexis( tsNA20(dmCs),
+                   formula = ~ Ns(age      ,knots=a.kn)
+                               + Ns(      tfI,knots=i.kn)
+                               + Ns(age-tfI,knots=a.kn)
+                               + lex.Cst + sex )
stats::glm Poisson analysis of Lexis object tsNA20(dmCs) with log link:
Rates for transitions: DM->Dead, Ins->Dead
```

But this will fit a model with a rate-ratio between persons with and without insulin that depends only on age at insulin start for the time *after* insulin start, the RR at `tfI=0` will be the same at any age, which really is not the type of interaction we wanted.

We want the `age-tfI` term to be specific for the insulin exposed so we may use one of two other approaches, that are conceptually alike but mathematically different:

```
> Im <- glm.Lexis( tsNA20(dmCs),
+                 formula = ~ Ns(age      ,knots=a.kn)
+                               + Ns(      tfI,knots=i.kn)
+                               + Ns((age-tfI)*(lex.Cst=="Ins"),knots=a.kn)
+                               + lex.Cst + sex )
stats::glm Poisson analysis of Lexis object tsNA20(dmCs) with log link:
Rates for transitions: DM->Dead, Ins->Dead

> im <- glm.Lexis( tsNA20(dmCs),
+                 formula = ~ Ns(age      ,knots=a.kn)
+                               + Ns(      tfI,knots=i.kn)
+                               + lex.Cst:Ns(age-tfI,knots=a.kn)
+                               + lex.Cst + sex )
stats::glm Poisson analysis of Lexis object tsNA20(dmCs) with log link:
Rates for transitions: DM->Dead, Ins->Dead
```

The first model (`Im`) has a common age-effect (`Ns(age,...)`) for persons with and without diabetes and a RR depending on insulin duration `tfI` and age at insulin (`age-tfI`). Since

the linear effect of these two terms are in the model as well, a linear trend in the RR by current age (`age`) is accommodated as well.

The second model allows age-effects that differ non-linearly between person with and without insulin, because the interaction term `lex.Cst:Ns(age-tfI...` for persons not on insulin is merely an age term (since `tfI` is coded 0 for all follow-up not on insulin).

We can compare the models fitted:

```
> anova( imx, Im, im, test='Chisq')
Analysis of Deviance Table

Model 1: cbind(trt(Lx$lex.Cst, Lx$lex.Xst) %in% trnam, Lx$lex.dur) ~ Ns(age,
  knots = a.kn) + Ns(tfI, knots = i.kn) + Ns(age - tfI, knots = a.kn) +
  lex.Cst + sex
Model 2: cbind(trt(Lx$lex.Cst, Lx$lex.Xst) %in% trnam, Lx$lex.dur) ~ Ns(age,
  knots = a.kn) + Ns(tfI, knots = i.kn) + Ns((age - tfI) *
  (lex.Cst == "Ins"), knots = a.kn) + lex.Cst + sex
Model 3: cbind(trt(Lx$lex.Cst, Lx$lex.Xst) %in% trnam, Lx$lex.dur) ~ Ns(age,
  knots = a.kn) + Ns(tfI, knots = i.kn) + lex.Cst:Ns(age -
  tfI, knots = a.kn) + lex.Cst + sex
Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1      228734      25096
2      228733      25087  1    8.9631 0.002755
3      228730      25082  3    4.6804 0.196749
```

so we see that the models indeed are different, and moreover that the last model does not provide substantial further improvement, by allowing non-linear RR along the age-scale.

We can illustrate the different estimated rates from the three models in figure 2.5, p. 27:

```
> pxi <- ci.pred( imx, ndI )
> pxa <- ci.pred( imx, ndA )
> pli <- ci.pred( Im , ndI )
> pia <- ci.pred( Im , ndA )
> pii <- ci.pred( im , ndI )
> pia <- ci.pred( im , ndA )
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6, las=1, bty="n" )
> matshade( ndI$age, cbind( pxi, pli, pii)*1000, plot=T, log="y",
+           xlab="Age", ylab="Mortality per 1000 PY",
+           lty=1, lwd=2, col=c("blue","forestgreen","red"), alpha=0.1 )
> matshade( ndA$age, cbind( pxa, pia, pia)*1000,
+           lty=1, lwd=2, col=c("blue","forestgreen","red"), alpha=0.1 )
```

We can also plot the RRs only from these models (figure 2.6, p. 28); for this we need the reference frames, and the machinery from `ci.exp` allowing a list of two data frames:

```
> ndR <- transform( ndI, tfI=0, lex.Cst="DM" )
> cbind( head(ndI), head(ndR) )

  tfI ai sex lex.Cst  age tfI ai sex lex.Cst  age
1  NA 40  M    Ins   NA   0 40  M    DM   NA
2 0.0 40  M    Ins 40.0   0 40  M    DM 40.0
3 0.1 40  M    Ins 40.1   0 40  M    DM 40.1
4 0.2 40  M    Ins 40.2   0 40  M    DM 40.2
5 0.3 40  M    Ins 40.3   0 40  M    DM 40.3
6 0.4 40  M    Ins 40.4   0 40  M    DM 40.4
```

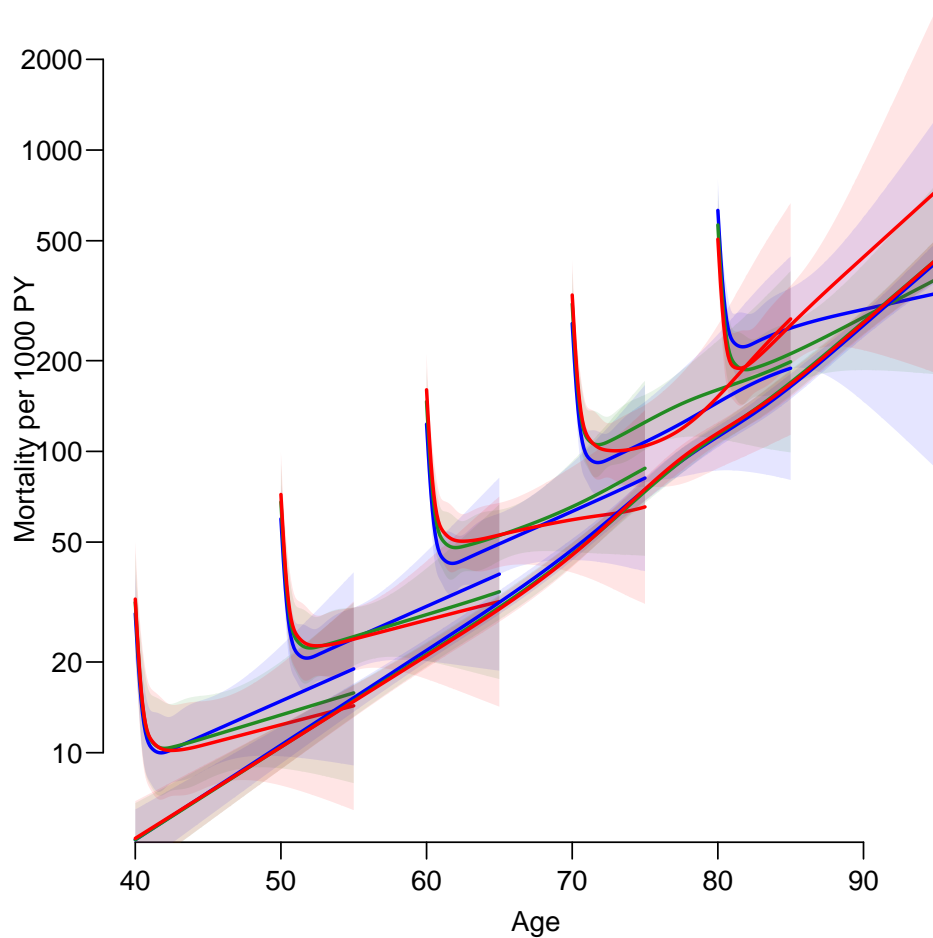


Figure 2.5: *Age at insulin as interaction between age and duration.* Blue curves are from the naive interaction model `imx` with identical RR at `tfI=0` at any age; green curves are from the interaction model with age at insulin, from the model `Im` with only linear differences by age, and red lines from the full interaction model `im`. ./flup-dur-int

```
> Rxi <- ci.exp( imx, list(ndI,ndR) )
> Rii <- ci.exp( im , list(ndI,ndR) )
> Rli <- ci.exp( Im , list(ndI,ndR) )
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6, las=1, bty="n" )
> matshade( ndI$age, cbind( Rxi, Rli, Rii), plot=T, log="y",
+           xlab="Age (years)", ylab="Rate ratio vs, non-Insulin",
+           lty=1, lwd=2, col=c("blue","forestgreen","red"), alpha=0.1 )
> abline( h=1 )
> abline( h=ci.exp(imx,subset="lex.Cst")[,1], lty="25", col="blue" )
```

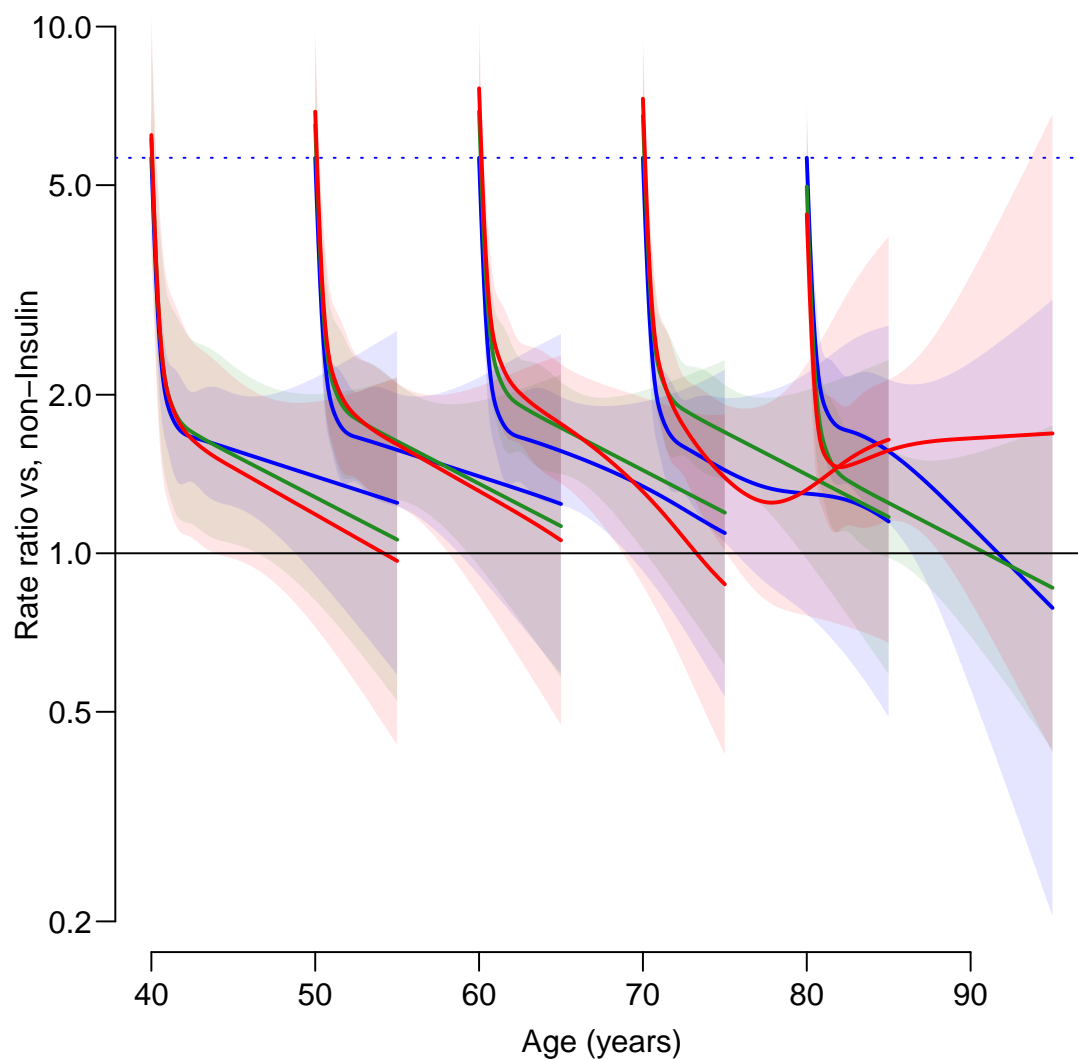


Figure 2.6: *RR from three different interaction models. The horizontal dotted line is at the estimated effect of `lex.Cst`, to illustrate that the first model (blue) constrains this initial HR to be constant across age. The green curves are the extended interaction model, and the red the full one.*

./flup-dur-int-RR

2.6.2 General interaction

As a final illustration we may want to explore a different kind of interaction, not defined from the duration — here we simplify the interaction by not using the second-last knot in the interaction terms — figure 2.7, p. 30. Note again that the prediction code is the same:

```
> gm <- glm.Lexis( tsNA20(dmCs),
+                 formula = ~ Ns(age,knots=a.kn)
+                           + Ns(tfI,knots=i.kn)
+                           + lex.Cst:Ns(age,knots=a.kn):Ns(tfI,knots=i.kn)
+                           + lex.Cst + sex )
stats::glm Poisson analysis of Lexis object tsNA20(dmCs) with log link:
Rates for transitions: DM->Dead, Ins->Dead

> pgi <- ci.pred( gm, ndI )
> pga <- ci.pred( gm, ndA )
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6, las=1, bty="n" )
> matshade( ndI$age, cbind( pgi, pii )*1000, plot=T,
+           lty=c("solid","2l"), lend="butt", lwd=2, log="y",
+           xlab="Age (years)", ylab="Mortality rates per 1000 PY",
+           alpha=c(0.2,0.1), col=c("black","red") )
> matshade( ndA$age, cbind( pga, pia )*1000,
+           lty=c("solid","2l"), lend="butt", lwd=2,
+           alpha=c(0.2,0.1), col=c("black","red") )
```

This is in figure 2.7, p. 30.

2.6.3 Evaluating interactions

Here we see that the interaction effect is such that in the older ages the length of insulin use has an increasing effect on mortality.

Even though there is no statistically significant interaction between age and time since start of insulin, it would be illustrative to show the RR as a function of age at insulin and age at follow-up:

```
> ndR <- transform( ndI, lex.Cst="DM", tfI=0 )
> iRR <- ci.exp( im, ctr.mat=list(ndI,ndR) )
> gRR <- ci.exp( gm, ctr.mat=list(ndI,ndR) )
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6, las=1, bty="n" )
> matshade( ndI$age, cbind(gRR,iRR), lty=1, log="y", plot=TRUE,
+           xlab="Age (years)", ylab="Rate ratio: Ins vs. non-Ins",
+           col=c("black","red") )
> abline( h=1 )
```

This is in figure 2.8, p. 31.

The advantage of the parametric modeling (be that with age at insulin or general spline interaction) is that it is straight-forward to *test* whether we have an interaction.

2.7 Separate models

In the above we insisted on making a joint model for the DM→Dead and the Ins→Dead transitions, but with the complications demonstrated it would actually have been more sensible to model the two transitions separately:

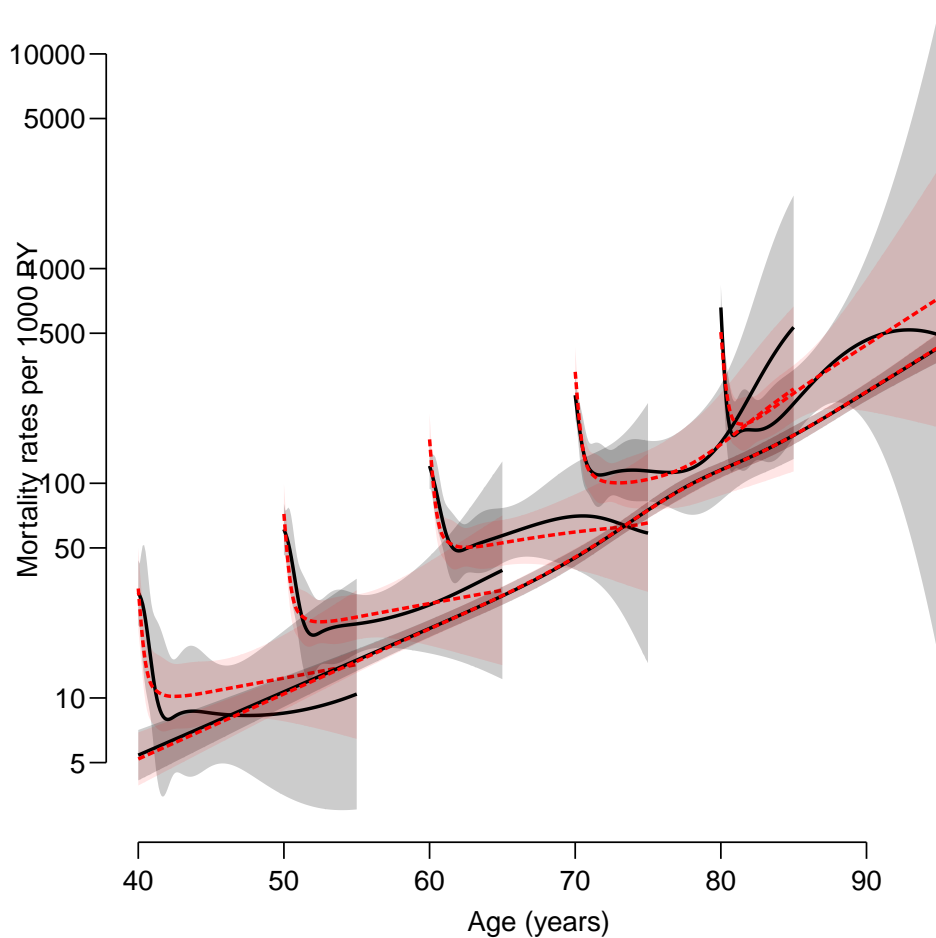


Figure 2.7: *Spline-by-spline interaction between age and duration (model `gm`, black), and the interaction using a non-linear effect of age at entry (model `im`, red), corresponding to the red curves in figure 2.5.*

`./flup-splint`

```
> dmd <- glm.Lexis( dmCs,
+                   from="DM", to="Dead",
+                   formula = ~ Ns(age,knots=a.kn)
+                   + sex )
stats::glm Poisson analysis of Lexis object dmCs with log link:
Rates for the transition: DM->Dead
> ind <- glm.Lexis( dmCs,
+                   from="Ins", to="Dead",
+                   formula = ~ Ns(age,knots=a.kn)
+                   + Ns(tfI,knots=i.kn)
+                   + Ns(age-tfI,knots=a.kn)
+                   + sex )
stats::glm Poisson analysis of Lexis object dmCs with log link:
Rates for the transition: Ins->Dead
> ini <- ci.pred( ind, ndI )
> dmi <- ci.pred( dmd, ndI )
> dma <- ci.pred( dmd, ndA )
```

The estimated mortality rates are shown in figure ??, p. ??, using:

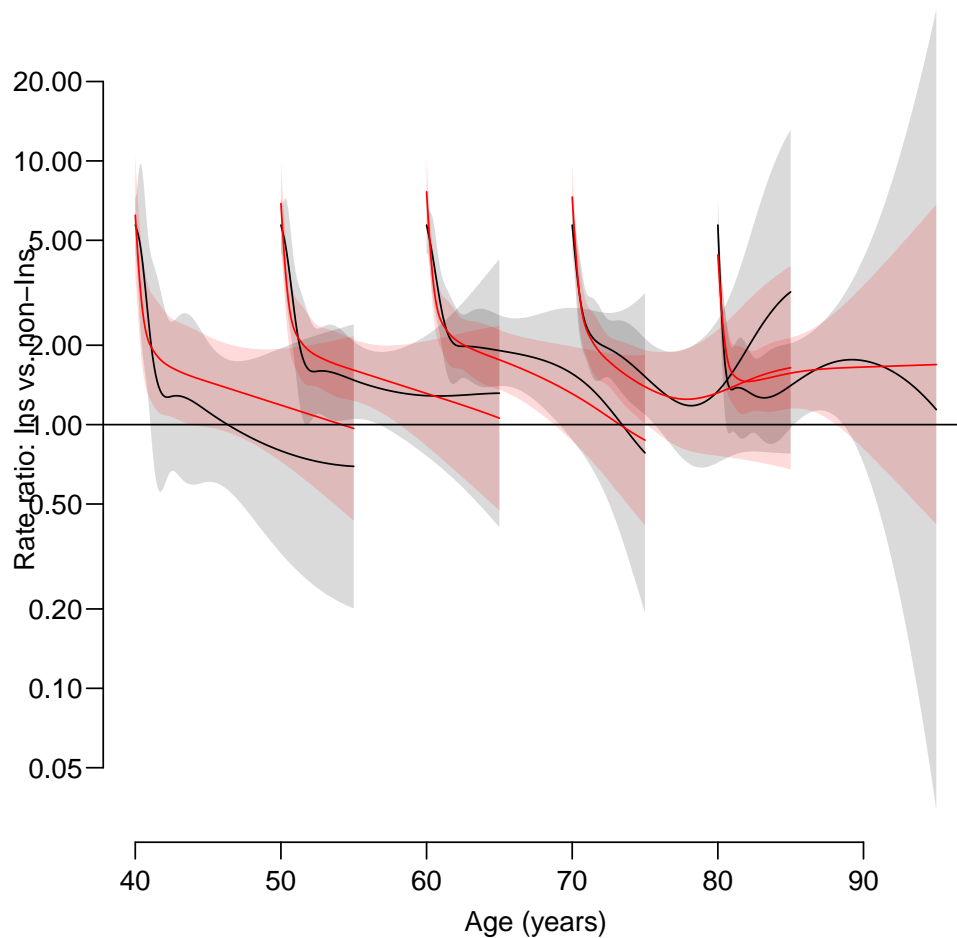


Figure 2.8: The effect of duration of insulin use at different ages of follow-up (and age at insulin start). Estimates are from the model with an interaction term using a non-linear effect of age at insulin start (model *im*, red) and using a general spline interactions (model *gm*, black). It appears that the general interaction over-models a bit. ./flup-RR-int

```
> par(mar=c(3,3,1,1),mgp=c(3,1,0)/1.6,las=1,bty="n")
> matshade( ndI$age, ini*1000, plot=TRUE, log="y",
+           xlab="Age (years)", ylab="Mortality rates per 1000 PY",
+           lwd=2, col="red" )
> matshade( ndA$age, dma*1000,
+           lwd=2, col="black" )
```

The estimated RRs are computed using that the estimates from the two models are uncorrelated, and hence qualify for `ci.ratio` (this and the previous graph appear in figure 2.9, p. 32)

```
> par(mar=c(3,3,1,1),mgp=c(3,1,0)/1.6,las=1,bty="n")
> matshade( ndI$age, ci.ratio(ini,dmi), plot=TRUE, log="y",
+           xlab="Age (years)", ylab="RR insulin vs. no insulin",
+           lwd=2, col="red" )
> abline( h=1 )
```

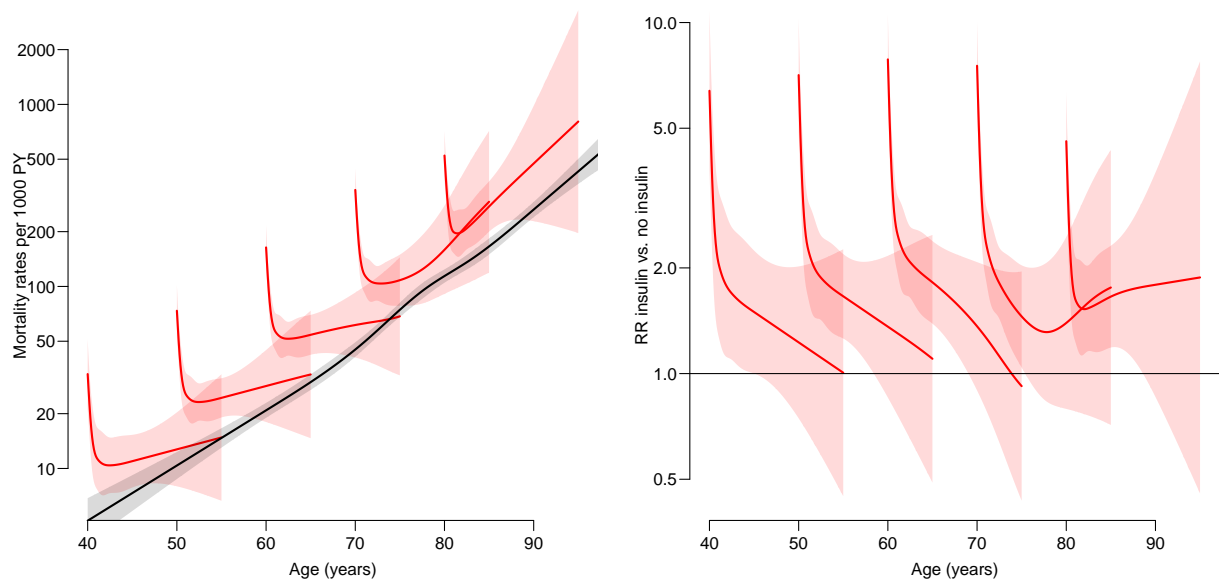


Figure 2.9: *Left panel: Mortality rates from separate models for the two mortality transitions; the DM→Dead transition modeled by age alone; Ins→Dead transition modeled with spline effects of current age, time since insulin and age at insulin. Right panel: Mortality HR of insulin vs. no insulin.*

Chapter 3

More states

3.1 Subdividing states

It may be of interest to subdivide the states following the intermediate event according to whether the event has occurred or not. This will enable us to address the question of the fraction of the patients that ever go on insulin.

This is done by the argument `split.states=TRUE`.

```
> dmCs <- cutLexis( data = dmS2,
+                   cut = dmS2$doins,
+                   timescale = "per",
+                   new.state = "Ins",
+                   new.scale = "tfl",
+                   precursor.states = "DM",
+                   split.states = TRUE )
> summary( dmCs )
```

Transitions:

	To								
From	DM	Ins	Dead	Dead(Ins)	Records:	Events:	Risk time:	Persons:	
DM	35135	1694	2048	0	38877	3742	45885.49	9899	
Ins	0	5762	0	451	6213	451	8387.77	1791	
Sum	35135	7456	2048	451	45090	4193	54273.27	9996	

We can illustrate the numbers and the transitions (figure 3.1, p. 34)

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

Note that it is only the mortality rates that we have been modeling, namely the transitions `DM→Dead` and `Ins→Dead(Ins)`. If we were to model the cumulative risk of using insulin we would also need a model for the `DM→Ins` transition. Subsequent to that we would then compute the probability of being in each state conditional on suitable starting conditions. With models where transition rates depend on several time scales this is not a trivial task. This is treated in more detail in the vignette on `simLexis`.

3.2 Multiple intermediate events

We may be interested in starting either insulin or OAD (oral anti-diabetic drugs), thus giving rise to more states and more time scales. This can be accomplished by the

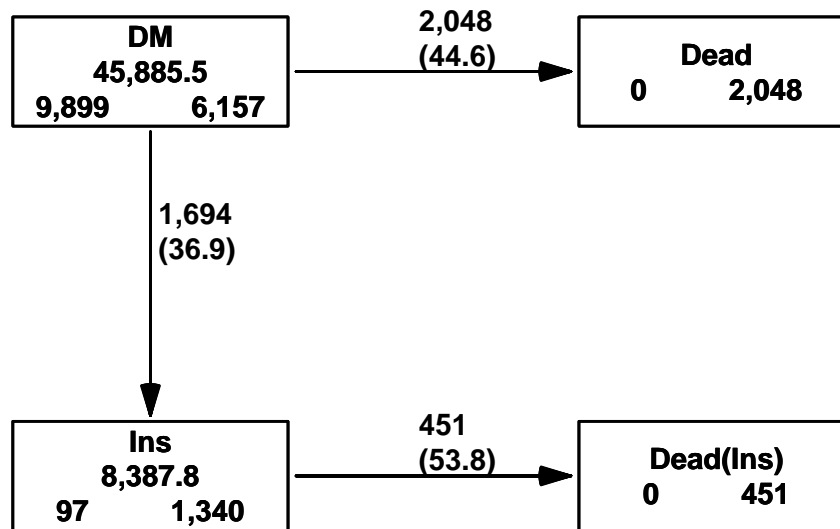


Figure 3.1: Transitions between 4 states: the numbers in the boxes are person-years (middle), and below the number of persons who start, respectively end their follow-up in each of the states.

./flup-box4

mcutLexis function, that generalizes cutLexis:

```
> dmM <- mcutLexis( dmL,
+                   timescale = "per",
+                   wh = c("doins", "doad"),
+                   new.states = c("Ins", "OAD"),
+                   new.scales = c("tfI", "tfO"),
+                   precursor.states = "DM",
+                   ties.resolve = TRUE )
```

NOTE: 9996 records with tied events times resolved.

Results only reproducible if the seed for the random number generator is set.

```
> summary( dmM, t=T )
```

Transitions:

From	To	DM	Dead	OAD	Ins	OAD-Ins	Ins-OAD	Records:	Events:	Risk time:	Persons:
DM		2830	1056	2957	689	0	0	7532	4702	22920.21	7532
OAD		0	992	3327	0	1005	0	5324	1997	22965.30	5324
Ins		0	152	0	462	0	172	786	324	3883.14	786
OAD-Ins		0	266	0	0	739	0	1005	266	3770.53	1005
Ins-OAD		0	33	0	0	0	139	172	33	734.09	172
Sum		2830	2499	6284	1151	1744	311	14819	7322	54273.27	9996

Timescales:

```
per age tfD tfI tfO
"" "" "" "Ins" "OAD"
```

We see that we now have two time scales defined as entry since into states.

```
> wh <- c(subset(dmM, lex.Cst=="Ins-OAD")$lex.id[1:2],
+         subset(dmM, lex.Cst=="OAD-Ins")$lex.id[1:2])
> options( width=110 )
```

```
> print( subset( dmM, lex.id %in% wh ),c('lex.id',names(dmM[1:8])),c("doins","doad")),
+       digits=6, row.names=FALSE )
```

lex.id	tfI	tfO	per	age	tfD	lex.dur	lex.Cst	lex.Xst	doins	doad
18	NA	NA	1996.75	61.7221	0.0000000	1.1690623	DM	OAD	2005.99	1997.
18	NA	0.00000	1997.92	62.8912	1.1690623	8.0793977	OAD	OAD-Ins	2005.99	1997.
18	0.0000000	8.07940	2005.99	70.9706	9.2484600	4.0027379	OAD-Ins	OAD-Ins	2005.99	1997.
25	NA	NA	2003.69	60.3422	0.0000000	1.8809035	DM	OAD	2008.64	2005.
25	NA	0.00000	2005.57	62.2231	1.8809035	3.0691307	OAD	OAD-Ins	2008.64	2005.
25	0.0000000	3.06913	2008.64	65.2923	4.9500342	1.3579740	OAD-Ins	OAD-Ins	2008.64	2005.
20	NA	NA	2009.25	53.2183	0.0000000	0.0316523	DM	Ins	2009.28	2009.
20	0.0000000	NA	2009.28	53.2500	0.0316523	0.0105728	Ins	Ins-OAD	2009.28	2009.
20	0.0105728	0.00000	2009.29	53.2606	0.0422251	0.7079461	Ins-OAD	Ins-OAD	2009.28	2009.
38	NA	NA	2008.37	63.9316	0.0000000	0.0930869	DM	Ins	2008.46	2008.
38	0.0000000	NA	2008.46	64.0246	0.0930869	0.2135524	Ins	Ins-OAD	2008.46	2008.
38	0.2135524	0.00000	2008.67	64.2382	0.3066393	1.3251198	Ins-OAD	Dead	2008.46	2008.

```
> summary( dmM, t=T )
```

Transitions:

From	To	DM	Dead	OAD	Ins	OAD-Ins	Ins-OAD	Records:	Events:	Risk time:	Persons:
DM		2830	1056	2957	689	0	0	7532	4702	22920.21	7532
OAD		0	992	3327	0	1005	0	5324	1997	22965.30	5324
Ins		0	152	0	462	0	172	786	324	3883.14	786
OAD-Ins		0	266	0	0	739	0	1005	266	3770.53	1005
Ins-OAD		0	33	0	0	0	139	172	33	734.09	172
Sum		2830	2499	6284	1151	1744	311	14819	7322	54273.27	9996

Timescales:

per	age	tfD	tfI	tfO
""	""	""	"Ins"	"OAD"

We can also illustrate the transitions to the different states, as in figure 3.2:

```
> boxes( dmM, boxpos=list(x=c(15,80,40,40,85,85),
+                           y=c(50,50,90,10,90,10)),
+       scale.R=1000, show.BE=TRUE )
```

We may not be interested in whether persons were prescribed insulin before OAD or vice versa, in which case we would combine the levels with both insulin and OAD to one, regardless of order (figure 3.3):

```
> summary( dmMr <- Relevel( dmM, list('OAD+Ins'=5:6), first=FALSE) )
```

	type	old	new
1	lex.Cst	DM	DM
2	lex.Cst	Dead	
3	lex.Cst	OAD	OAD
4	lex.Cst	Ins	Ins
5	lex.Cst	OAD-Ins	OAD+Ins
6	lex.Cst	Ins-OAD	OAD+Ins
7	lex.Xst	DM	DM
8	lex.Xst	Dead	Dead
9	lex.Xst	OAD	OAD
10	lex.Xst	Ins	Ins
11	lex.Xst	OAD-Ins	OAD+Ins
12	lex.Xst	Ins-OAD	OAD+Ins

Transitions:

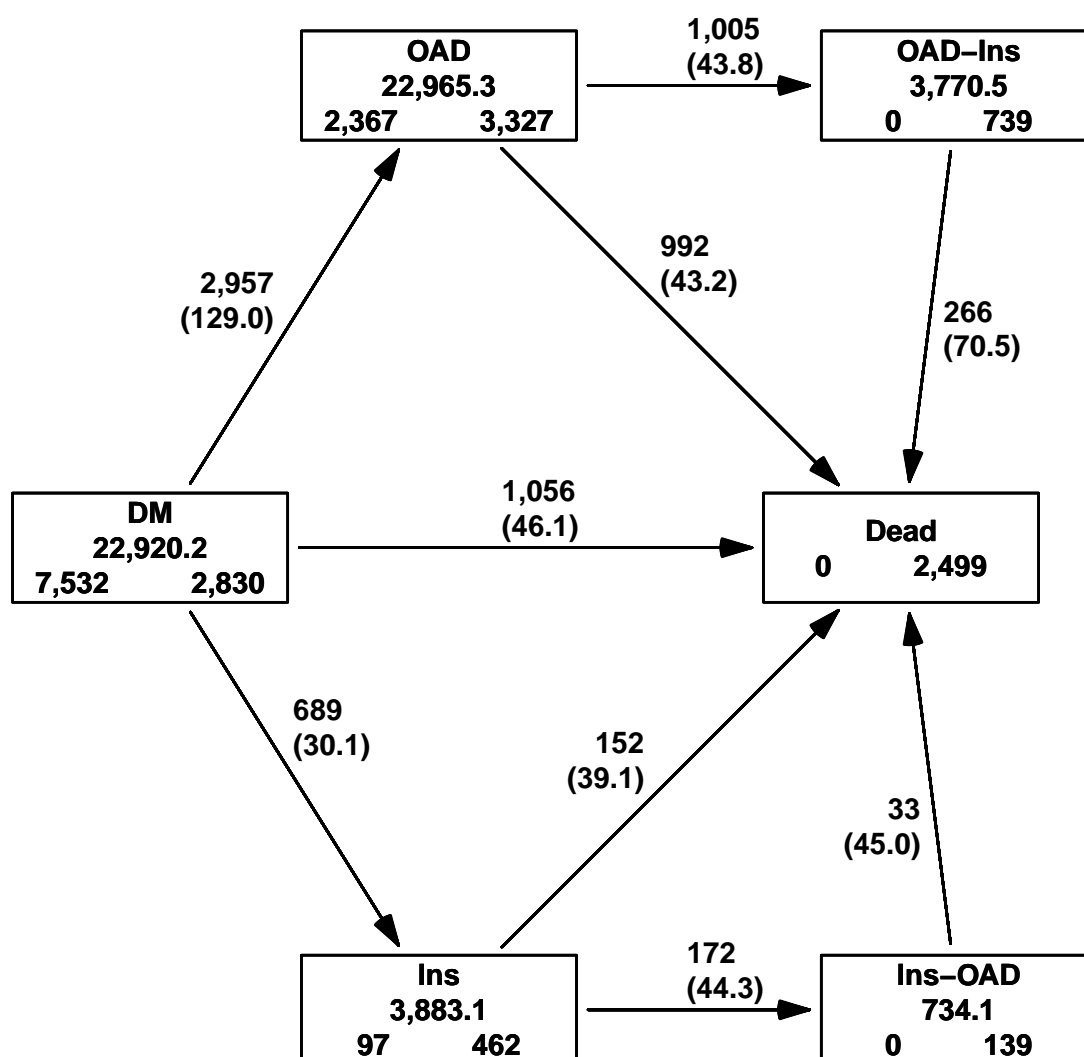


Figure 3.2: Boxes for the `dmM` object. The numbers in the boxes are person-years (middle), and below the number of persons who start, respectively end their follow-up in each of the states.

./flup-mbox

```

To
From    DM  Dead  OAD  Ins  OAD+Ins  Records:  Events:  Risk time:  Persons:
DM      2830 1056 2957 689      0       7532    4702    22920.21    7532
OAD      0   992 3327  0     1005    5324    1997    22965.30    5324
Ins      0   152  0   462     172     786     324     3883.14     786
OAD+Ins  0   299  0   0      878    1177     299     4504.62    1177
Sum      2830 2499 6284 1151    2055   14819    7322    54273.27   9996

> boxes( dmMr, boxpos=list(x=c(15,50,15,85,85),
+                             y=c(85,50,15,85,15)),
+        scale.R=1000, show.BE=TRUE )

```

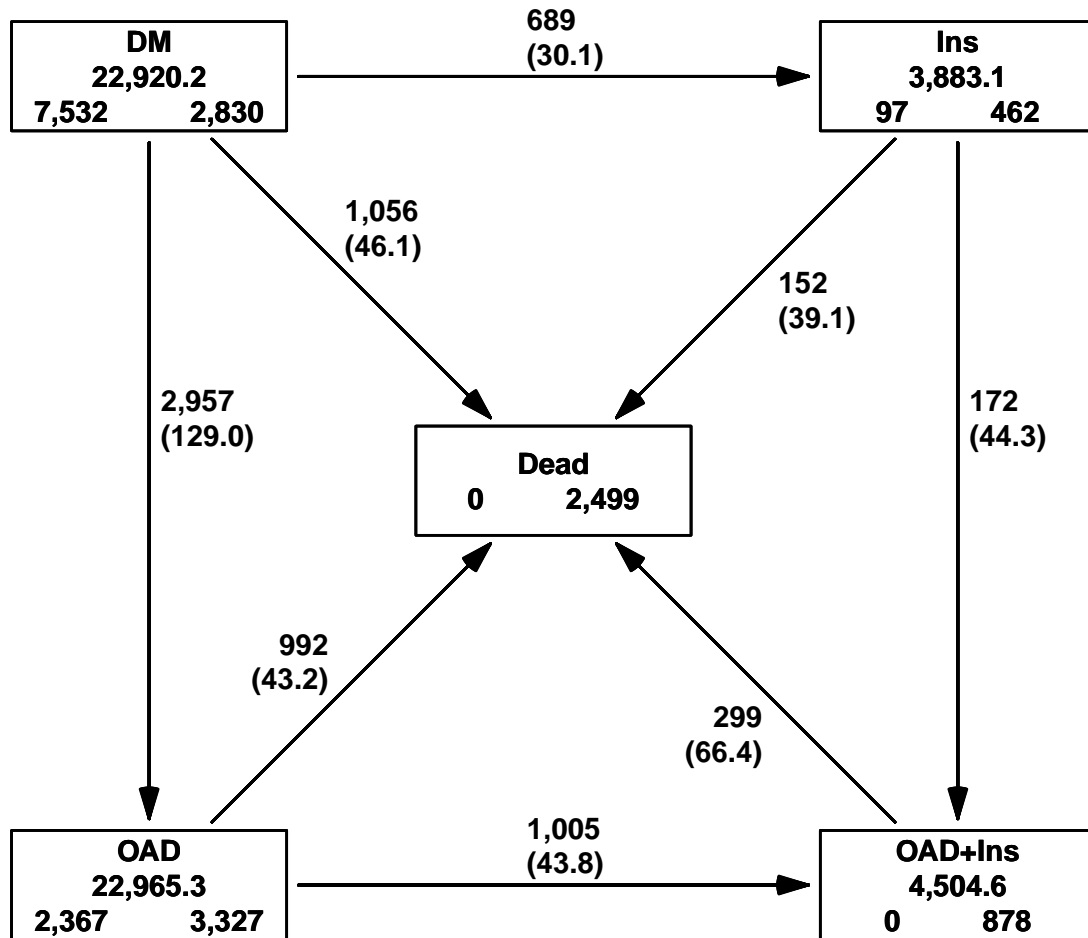


Figure 3.3: Boxes for the `dmMr` object with collapsed states. The numbers in the boxes are person-years (middle), and below the number of persons who start, respectively end their follow-up in each of the states. ./flup-mboxr

Diagrams as those in figures 3.2 and 3.3 gives an overview of the possible transitions, which states it might be relevant to collapse, and which transitions to model and how.

Chapter 4

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.

Over the years there have been additions of tools for handling multistate data. Here is a list of the current functions relating to `Lexis` objects with a very brief description; it does not replace the documentation. 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 intermediate event
`mcutLexis` cut follow-up at several intermediate events
`countLexis` cut follow-up at intermediate event count the no. events so far
`splitLexis` split follow up along a time scale
`splitMulti` split follow up along a time scale — from the `popEpi` package, faster and has simpler syntax than `splitLexis`
`addCov.Lexis` add clinical measurements at a given date to a `Lexis` object

Boxes and plots

`boxes.Lexis` draw a diagram of states and transitions
`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
`tsNA20` turn NAs to 0s for time scales
`Relevel.Lexis, factorize.Lexis` reorder and combine states
`bootLexis` bootstrap sample of *persons* (`lex.id`) in the `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

Interface to other packages

`msdata.Lexis` interface to `mstate` package
`etm.Lexis` interface to `etm` package
`crr.Lexis` interface to `cmprsk` package

Statistical models — these are *not* S3 methods

`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