Data management for epidemiological data analysis: Practical estimation of prevalence, mortality and survival in diabetes with R

Computer practicals

IDEG 2025, Bangkok
Thursday 3rd April 2025
http://bendixcarstensen.com/AdvCoh/courses/IDEG2025/
Version 3, March 2025

Compiled Friday 28th March, 2025, 13:46
from: C:\Bendix\teach\AdvCoh\courses\IDEG2025\pracs/pracs.tex

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

Preface

This workshop will provide an introduction to basic epidemiological data manipulation in population-based studies illustrated by prevalence and mortality measures

- The *target audience* is young epidemiologists starting work in diabetes epidemiological research
- The *prerequisites* are
  1. (desirable, but not necessary) a basic knowledge of R,
  2. a working installation of R (latest version, 4.4.3)
  3. a working installation of the latest version of the `Epi` package (2.59)
  4. a working installation of the latest version of the `popEpi` package (0.4.12)
- The *format* of the workshop will be short lectures closely aligned with the topics in the exercises. The exercises will be run in between the short lectures.
- The *mood* of the workshop will be relaxed, encouraging participants to ask questions and bring forward problems they consider relevant for the workshop. Fortunately, there will be the rest of the IDEG to interact.

Program of workshop

Each item on the program is a short(ish) lecture followed by computer practicals in R. The timing of the items is approximate.

Thursday, April 3rd, 2025	
10:15–10:20	Welcome and introdiction
10:20–10:40	Prevalence: tables
10:40–11:00	Prevalence: probability models
11:00–11:10	Mortality: concepts
11:10–11:30	Mortality: models for age
11:30–11:45	Mortality: models for duration
11:45–12:00	Mortality and survival

Practicalities

Exercises are given in green, and mostly the solutions will be included in this document too. You can get the exercise code chunks from the workshop website
http://bendixcarstensen.com/AdvCoh/courses/IDEG2025/R

In the exercises a certain R-lingo will be used, in particular:

"`<-`" is pronounced "gets"

"`fun(x)`" is pronounced "fun of x"—so when you hear "fun of…", you type "`fun( )`" and place the cursor between the brackets

If you want to know about the Rfunction `funk`, just type `?funk`

Chapter 1

Prevalence

. . . is the fraction of a population that suffer a particular condition, diabetes for example.

In this exercise we will use data from the 2023 National Health Interview Survey, a copy is available at https://bendixcarstensen.com/AdvCoh/courses/IDEG2025/data/?F=1 the data is the `.csv` file and the explanation of variables is the `.pdf` file.

The data set consists of persons interviewed, and this exercise is using information on age, sex and diabetes status.

You can learn more about the NHIS at https://www.cdc.gov/nchs/nhis/index.html
First load the R-packages needed:

```
> library(Epi)
> library(popEpi)
> library(survival)
> library(tidyverse)
```

```
     R  Epi popEpi
 4.4.2 2.59 0.4.12
```

1.1 Data

Read the NHIS data—it is in `.csv` format so use `read.csv`, you need the `header=TRUE` to indicate that the first line of data is the variable names:

```
> nhis <- read.csv(
+ "https://bendixcarstensen.com/AdvCoh/courses/IDEG2025/data/NHIS_IDEG.csv",
+ header = TRUE)
```

```
> str(nhis)
'data.frame':        29522 obs. of  10 variables:
 $ HHX        : chr  "H029691" "H028812" "H045277" "H021192" ...
 $ WTFA_A     : num  7371 3147 10808 4662 10930 ...
 $ SEX_A      : int  1 1 1 2 2 2 2 1 2 1 ...
 $ AGEP_A     : int  67 73 48 42 50 46 36 44 80 61 ...
 $ EDUCP_A    : int  1 8 5 9 7 8 8 10 8 1 ...
 $ DIBEV_A    : int  2 1 2 2 2 2 2 2 2 2 ...
 $ DIBAGETC_A : int  NA 61 NA NA NA NA NA NA NA NA ...
 $ DIFYRSTC1_A: int  NA 12 NA NA NA NA NA NA NA NA ...
 $ DIBTYPE_A  : int  NA 2 NA NA NA NA NA NA NA NA ...
 $ BMICAT_A   : int  3 3 4 3 2 3 2 4 4 3 ...
```

```
> newn <- tolower(gsub("_A", "", names(nhis)))
> cbind(names(nhis), newn)
```

```
                     newn
 [1,] "HHX"          "hhx"
 [2,] "WTFA_A"       "wtfa"
 [3,] "SEX_A"        "sex"
 [4,] "AGEP_A"       "agep"
 [5,] "EDUCP_A"      "educp"
 [6,] "DIBEV_A"      "dibev"
 [7,] "DIBAGETC_A"   "dibagetc"
 [8,] "DIFYRSTC1_A"  "difyrstc1"
 [9,] "DIBTYPE_A"    "dibtype"
[10,] "BMICAT_A"     "bmicat"
```

```
> names(nhis) <- newn
> str(nhis)
```

```
'data.frame':        29522 obs. of  10 variables:
 $ hhx      : chr  "H029691" "H028812" "H045277" "H021192" ...
 $ wtfa     : num  7371 3147 10808 4662 10930 ...
 $ sex      : int  1 1 1 2 2 2 2 1 2 1 ...
 $ agep     : int  67 73 48 42 50 46 36 44 80 61 ...
 $ educp    : int  1 8 5 9 7 8 8 10 8 1 ...
 $ dibev    : int  2 1 2 2 2 2 2 2 2 2 ...
 $ dibagetc : int  NA 61 NA NA NA NA NA NA NA NA ...
 $ difyrstc1: int  NA 12 NA NA NA NA NA NA NA NA ...
 $ dibtype  : int  NA 2 NA NA NA NA NA NA NA NA ...
 $ bmicat   : int  3 3 4 3 2 3 2 4 4 3 ...
```

CODE EXPLAINED: The variable names are a bit awkward, so we define a set of new ones by removing the `_A` and turning all to lower case. We use `cbind` to show the old and the new names juxtaposed before we replace the old with the new names. This is safer than using one of the `rename` functions, partly because we do not rely on our own correct typing of old and new names.

We also define category labels for readable tables

```
> nhis <- mutate(nhis, dibev = factor(dibev, labels = c("Y","N","R","U")),
+                      dibtype = factor(dibtype, labels = c("T1","T2","O","O","O")),
+                          agr = cut(agep, seq(0, 100, 10), right = FALSE),
+                          sex = factor(sex,  labels = c("M","W","U","U")))
> str(nhis)
```

```
'data.frame':        29522 obs. of  11 variables:
 $ hhx      : chr  "H029691" "H028812" "H045277" "H021192" ...
 $ wtfa     : num  7371 3147 10808 4662 10930 ...
 $ sex      : Factor w/ 3 levels "M","W","U": 1 1 1 2 2 2 2 1 2 1 ...
 $ agep     : int  67 73 48 42 50 46 36 44 80 61 ...
 $ educp    : int  1 8 5 9 7 8 8 10 8 1 ...
 $ dibev    : Factor w/ 4 levels "Y","N","R","U": 2 1 2 2 2 2 2 2 2 2 ...
 $ dibagetc : int  NA 61 NA NA NA NA NA NA NA NA ...
 $ difyrstc1: int  NA 12 NA NA NA NA NA NA NA NA ...
 $ dibtype  : Factor w/ 3 levels "T1","T2","O": NA 2 NA NA NA NA NA NA NA NA ...
 $ bmicat   : int  3 3 4 3 2 3 2 4 4 3 ...
 $ agr      : Factor w/ 10 levels "[0,10)","[10,20)",..: 7 8 5 5 6 5 4 5 9 7 ...
```

CODE EXPLAINED: `mutate` (re)defines variables in the data frame. Here we use `factor` to attach labels to the numerical variables `dibev` (ever diabetes) and `dibtype` (type of diabetes). The function `cut` groups the `agep` variable; the result is saved in a new variable, `agr`. `sex` is defined as having three levels, `Man`, `Woman`, `Unknown`—note that the original coding has 4 levels; the two last levels are combined as `U`.

1.2 Tables

Tabulate the diabetes status, and compute the prevalence of diabetes among those who have replied either yes or no:

```
> (tb <- with(nhis, table(dibev, exclude = NULL)))
dibev
    Y     N     R     U
 3294 26195    23    10
> tb["Y"] / (tb["Y"] + tb["N"]) * 100
       Y
11.17027
```

CODE EXPLAINED: The function `with` makes any variable mentioned after `nhis` refer to a variable in the `nhis` data frame. Putting brackets around an assignment will print the assigned value.

The square brackets ("`[ ]`") are used for indexing of tables (and arrays), so `[1:2, ]` selects the two first rows, `[, 1:2]` selects the two first columns (type 1 and type 2 diabetes) and `[, "N"]` selects the column labeled `"N"`.

The result is multiplied by 100 to get percentages.

A brief overview of persons' age (`agr`) and whether the person has diabetes or not—age-specific prevalence of diabetes:

```
> with(nhis, table(Age = agr,
+              Diabetes = dibev,
+               exclude = NULL)) |> addmargins() -> diab
> diab
          Diabetes
Age             Y     N     R     U   Sum
  [0,10)        0     0     0     0     0
  [10,20)       3   426     0     0   429
  [20,30)      44  3308     1     0  3353
  [30,40)     132  4534     0     1  4667
  [40,50)     281  3863     4     1  4149
  [50,60)     528  3953     4     0  4485
  [60,70)     976  4596     5     1  5578
  [70,80)     920  3556     5     3  4484
  [80,90)     408  1899     1     4  2312
  [90,100)      2    60     3     0    65
  Sum        3294 26195    23    10 29522
```

CODE EXPLAINED: `table` makes a table of `agr` versus `dibev`. `addmargins` is then by `|>` applied to the result forming margins. Also, note that the assignment operator can be used both ways: "`<-`" and "`->`". One could argue that the latter is more logical: first do the calculations, then assign.

We see that there are persons in the dataset with unknown diabetes status, so we compute the prevalence only among persons with known diabetes status:

```
> (diab <- addmargins(diab[, 1:2], 2))
          Diabetes
Age            Y     N   Sum
  [0,10)       0     0     0
  [10,20)      3   426   429
  [20,30)     44  3308  3352
  [30,40)    132  4534  4666
  [40,50)    281  3863  4144
  [50,60)    528  3953  4481
  [60,70)    976  4596  5572
  [70,80)    920  3556  4476
  [80,90)    408  1899  2307
  [90,100)     2    60    62
  Sum       3294 26195 29489
> cbind(round(diab["Y"] / diab["Sum"] * 100, 1))
         [,1]
[0,10)    NaN
[10,20)   0.7
[20,30)   1.3
[30,40)   2.8
[40,50)   6.8
[50,60)  11.8
[60,70)  17.5
[70,80)  20.6
[80,90)  17.7
[90,100)  3.2
Sum      11.2
```

CODE EXPLAINED: `addmargins` puts margins on a table, in this case named "`Sum`". We refer to colums of the table by the names, a way to make the code readable, using `1` and `3` would work equally well but be unreadable.

It would be useful to see the prevalence of type 1 and type 2 diabetes separately, so make a table of `dibtype` versus `agr`—remember to consider the orientation of the table.

```
> with(nhis, table(agr, dibtype, exclude = NULL)) |> addmargins() -> dtyp
> dtyp
          dibtype
agr            T1    T2     O  <NA>   Sum
  [0,10)        0     0     0     0     0
  [10,20)       0     2     1   426   429
  [20,30)      23    15     6  3309  3353
  [30,40)      36    80    16  4535  4667
  [40,50)      21   236    24  3868  4149
  [50,60)      43   447    38  3957  4485
  [60,70)      73   849    54  4602  5578
```

```
  [70,80)     49   818    53  3564  4484
  [80,90)     24   341    43  1904  2312
  [90,100)     0     1     1    63    65
  Sum        269  2789   236 26228 29522
```

CODE EXPLAINED: Using `exclude=NULL` causes the `table` to include `NA`s as a valid category. The default is to omit observations with `NA` in any of the tabulation variables.

In order to get the prevalences, divide the two first columns with the last:

```
> round(100 * dtyp[, 1:2] / dtyp[, "Sum"], 1)
          dibtype
agr          T1   T2
  [0,10)
  [10,20)   0.0  0.5
  [20,30)   0.7  0.4
  [30,40)   0.8  1.7
  [40,50)   0.5  5.7
  [50,60)   1.0 10.0
  [60,70)   1.3 15.2
  [70,80)   1.1 18.2
  [80,90)   1.0 14.7
  [90,100)  0.0  1.5
  Sum       0.9  9.4
```

What do you conclude about the prevalence of (known) T1 and T2 diabetes?

1.3 Probability

Above we defined prevalence as the fraction of a population that suffered from a given disease. In probabilistic terms prevalence can be formulated as the probability that a randomly selected person has the disease. This opens the possibility of statistical modeling to address the question on how the prevalence of diabetes depends on age and sex, for example. First restrict the dataset to those with known diabetes status and sex.

```
> nh <- subset(nhis, dibev %in% c("Y", "N") & sex %in% c("M", "W"))
> str(nh)
'data.frame':        29483 obs. of  11 variables:
 $ hhx      : chr  "H029691" "H028812" "H045277" "H021192" ...
 $ wtfa     : num  7371 3147 10808 4662 10930 ...
 $ sex      : Factor w/ 3 levels "M","W","U": 1 1 1 2 2 2 2 1 2 1 ...
 $ agep     : int  67 73 48 42 50 46 36 44 80 61 ...
 $ educp    : int  1 8 5 9 7 8 8 10 8 1 ...
 $ dibev    : Factor w/ 4 levels "Y","N","R","U": 2 1 2 2 2 2 2 2 2 2 ...
 $ dibagetc : int  NA 61 NA NA NA NA NA NA NA NA ...
 $ difyrstc1: int  NA 12 NA NA NA NA NA NA NA NA ...
 $ dibtype  : Factor w/ 3 levels "T1","T2","O": NA 2 NA NA NA NA NA NA NA NA ...
 $ bmicat   : int  3 3 4 3 2 3 2 4 4 3 ...
 $ agr      : Factor w/ 10 levels "[0,10)","[10,20)",..: 7 8 5 5 6 5 4 5 9 7 ...
```

CODE EXPLAINED: Here we used `subset`; another possibility would be `filter` from the `tidyverse` package. Note the `%in%` operator used to select specific values of `dibev` and `sex`.

Then we can model the presence of diabetes with a binomial regression model (the default is logistic regression):

```
> ma <- glm((dibev == "Y") ~ Ns(agep, knots = seq(30, 90,, 4)),
+           family = binomial,
+           data = nh)
> da <- data.frame(agep = 10:95)
> head(pa <- ci.pred(ma, da) * 100)
   Estimate      2.5%     97.5%
1 0.4034395 0.2770811 0.5870824
2 0.4375991 0.3041219 0.6292888
3 0.4746372 0.3337856 0.6745235
4 0.5147940 0.3663231 0.7230036
5 0.5583292 0.4020088 0.7749616
6 0.6055237 0.4411424 0.8306467
> matshade(da[,"agep"], pa, plot = TRUE, lwd = 3, ylim = c(0, 25), yaxs = "i")
```

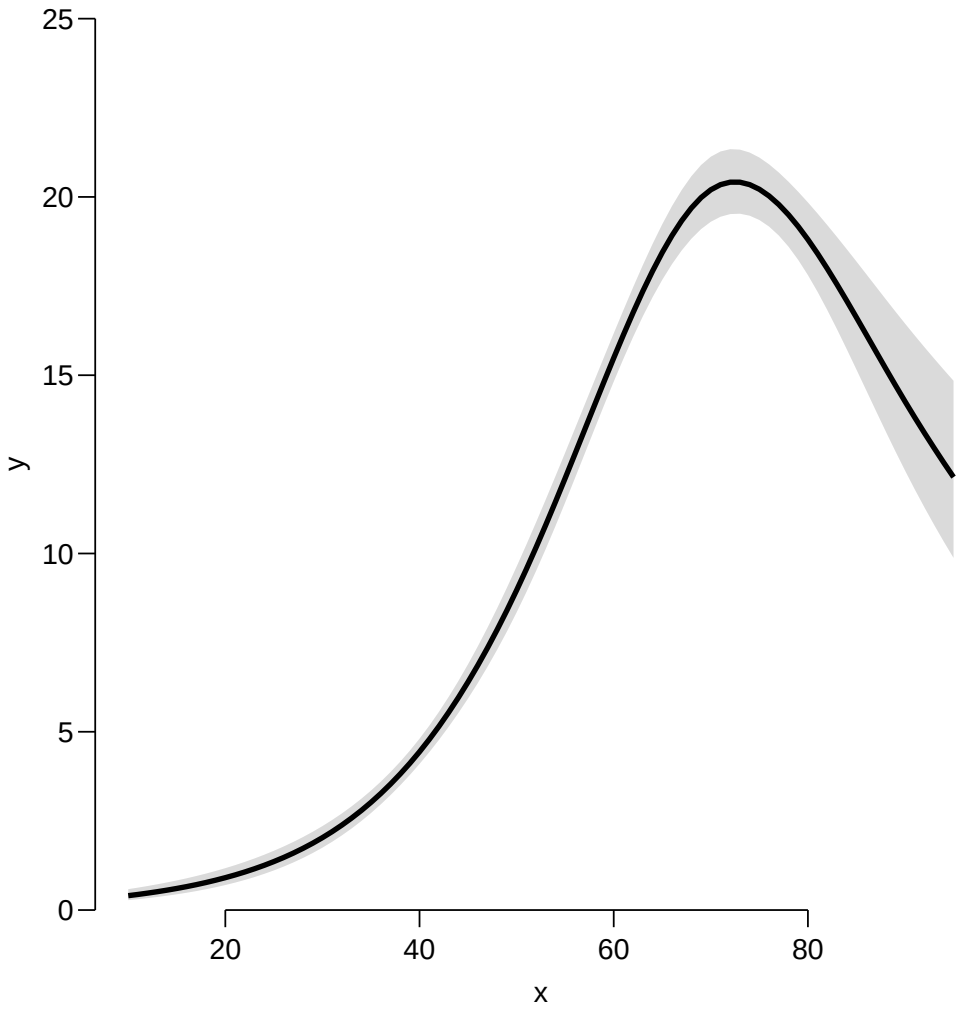

Figure 1.1: *Prevalence of diabetes in the NHIS.* ../graph/prev-both

CODE EXPLAINED: The response variable in a binomial regression model can be either numerical (0/1) or logical (`FALSE`/`TRUE`), but using a logical expression makes it clearer what outcome is modeled, namely P{`TRUE`}.

`Ns` is a function that represents a quantitative argument (here `agep`) as a restricted cubic spline, a function which is 3rd degree polynomials between each pair of the given knots, smoothly joined together, and linear at the ends. The resulting parameters do not have any interpretation, we need to make predictions for select values of the quantitative variable.

The predicted probabilities with confidence intervals are derived by `ci.pred`.

`matshade` is a function that draws a curve with a shaded area to represent a confidence interval. The first argument is the x-values, the second argument is a matrix with 3 columns: (estimate, lower, upper).

Annotate the axes with sensible labels (`xlab=`).

Fit the model separately for men and women, and draw the estimated prevalences in the same plot.

```
> Ma <- glm((dibev == "Y") ~ Ns(agep, knots = seq(30, 90,, 4)),
+           family = binomial,
+           data = subset(nh, sex == "M"))
> Wa <- update(Ma, data = subset(nh, sex == "W"))
> pM <- ci.pred(Ma, da)
> pW <- ci.pred(Wa, da)
> mw <- ci.ratio(pM, pW)
```

CODE EXPLAINED: `glm` fits the same logistic regression model as above, but now only for men, using `subset`. The same model can be fitted for women using `update`, a function that does the same as for `Ma` except for those parameters given. In this case only `data=`.

`ci.pred` returns the fitted values (that is prevalences) from the two models as functions of ages as given in `da`.

Since the two models are fitted to separate datasets the two sets of predictions are independent, hence we can use `ci.ratio` to compute the ratio of the predicted prevalences and its standard error.

```
> par(mar = c(3,3,1,3))
> matshade(da$agep, cbind(pM, pW) * 100, col = c("blue", "red"),
+          plot = TRUE, lwd = 3, ylim = c(0, 25), yaxs = "i",
+          xlab = "Age (years)", ylab = "Prevalence of diabetes (%)")
> axis(side = 1, at = seq(15, 95, 5), labels = NA, tcl = -0.3)
> axis(side = 1, at = seq(10, 90,10), labels = NA, tcl = -0.5)
> #
> matshade(da$agep, mw * 5, lwd =3)
> lines(c(10,100), c(5,5))
> axis(side = 4, at = c(1, 1.5, 2) * 5, labels = c(1, 1.5, 2))
> axis(side = 4, at = seq(0.8, 2, 0.1) * 5, labels = NA, tcl = -0.3)
```

CODE EXPLAINED: The first `matshade` draws the prevalences for men and women with shaded confidence intervals; `cbind(pM, pW)` puts the predictions side-by-side in a 6-column matrix, so `matshade` plots two curves with shadows. `plot=TRUE` starts a new plot. The `axis(side=1` statements adds tick marks to the x-axis.

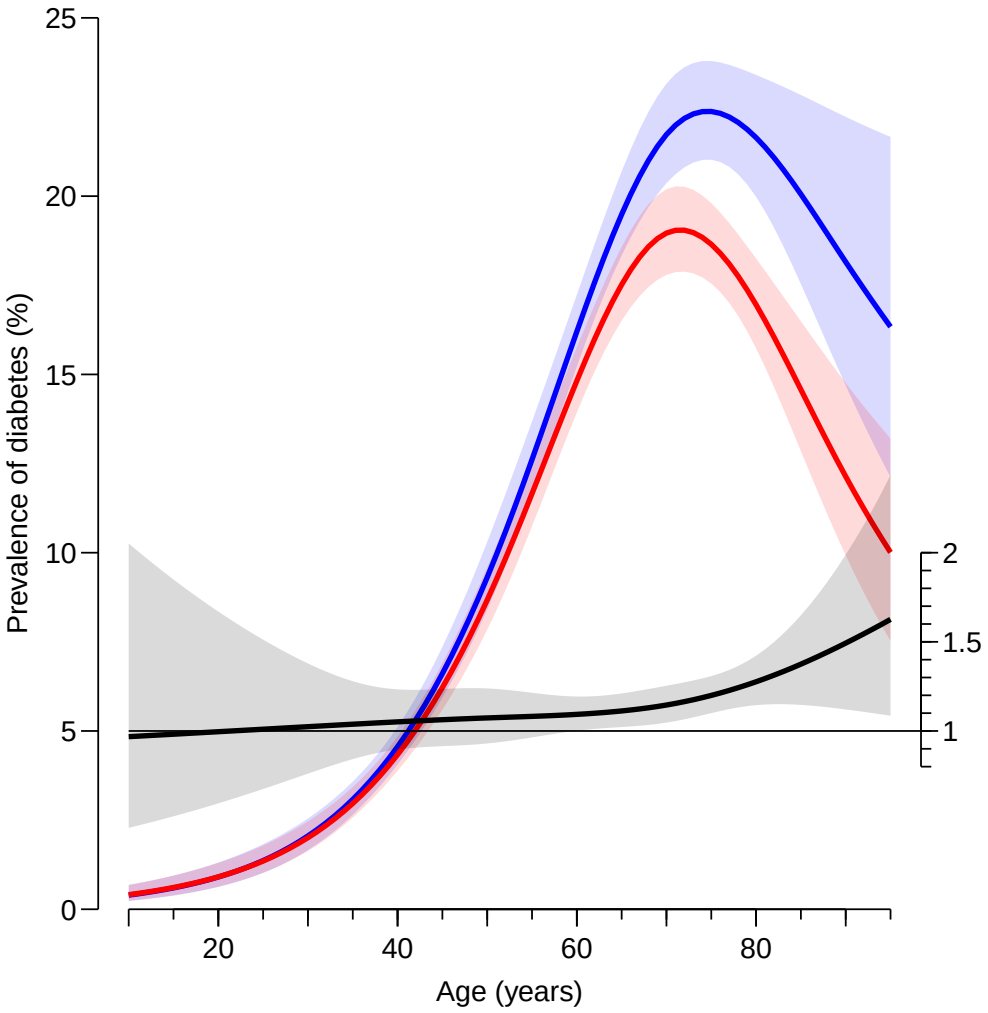

Figure 1.2: *Prevalence of diabetes in the NHIS. Red is women, blue is men, the black curve and the scale on the right is the ratio of prevalences among men versus women* `../graph/prev-m-w`

> The second `matshade adds` to the plot, namely the M/W prevalence ratio — multiplied by 5 so that the M/W ratio of 1 is at 5% on the prevalence scale.
>
> The `axis(side=4` adds a new axis at the r.h.s. where the M/W ratio is 1, 1.5 and 2, the second adds some tick marks.

What do you conclude from the graph?

Chapter 2

Mortality and survival

```
> library(Epi)
> library(popEpi)
> library(survival)
> library(tidyverse)
```

```
     R  Epi popEpi
 4.4.2 2.59 0.4.12
```

2.1 Concepts

A mortality rate is the "force of mortality", the rate with which deaths occur. It includes a time aspect—how *long* have people been at risk. In practice we need to known how many persons for how long time (the *risk time* or "*person-years*") and how many deaths (events) have occurred.

At the *individual* level we need to know how long time a person has been at risk of dying, and whether the person is dead at the end of the risk time. A person's risk time can only be included if the person would be counted as dead if dying during that time.

At the *theoretical* level we need a precise (probabilistic) definition of mortality: a mortality *rate* is defined as a conditional probability of death—conditional on being alive at a given time t—divided by the length of the risk interval, h:

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

—formally the limit of this as h gets smaller and smaller:

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

The t here is the time*scale*—*when* the person is at risk; the h is the *risk* time—*how long* the person has been at risk. The rate has dimension time^{-1}—events per time.

The mortality is a function of t, but one possibility for this function is that it is constant, the same at all times, $\lambda(t) = \lambda\ \forall t$.

2.2 Data

Get the `DMlate` data; data on a random sample of 10,000 persons the Danish diabetes register, of which we take a convenience sample of 2000:

```
> data(DMlate)
> set.seed(1952)
> DMlate <- DMlate[sample(1:nrow(DMlate), 2000),]
> rownames(DMlate) <- 1:2000
> str(DMlate)
'data.frame':        2000 obs. of  7 variables:
 $ sex  : Factor w/ 2 levels "M","F": 2 1 2 1 1 1 1 1 1 1 ...
 $ dobth: num  1964 1944 1957 1952 1952 ...
 $ dodm : num  2003 2006 2008 2007 2003 ...
 $ dodth: num  NA NA NA NA NA NA NA NA NA NA ...
 $ dooad: num  NA 2006 NA 2007 2006 ...
 $ doins: num  NA NA NA 2008 NA ...
 $ dox  : num  2010 2010 2010 2010 2010 ...
> head(DMlate)
  sex    dobth     dodm dodth    dooad    doins      dox
1   F 1963.591 2003.481    NA       NA       NA 2009.997
2   M 1944.127 2005.644    NA 2005.778       NA 2009.997
3   F 1956.790 2007.886    NA       NA       NA 2009.997
4   M 1952.355 2006.969    NA 2006.969 2008.026 2009.997
5   M 1952.240 2003.361    NA 2005.852       NA 2009.997
6   M 1978.758 2001.948    NA       NA 2001.967 2009.997
```

CODE EXPLAINED: `set.seed` sets the seed for the random number generator, so that the sequence of random numbers generated and used in `sample` will always be the same (also across different computers). The function `sample` returns a random sample of the first argument. The rownames are renamed for convenience.

How are the dates coded? (Use `?DMlate`)

The patients have been at risk of dying from date of diagnosis of diabetes, `dodm`, till the end of the register coverage or death, `dox`.

What is the total risk time among the 2000 patients (remember to state the units)?

```
> (y <- with(DMlate, sum(dox - dodm)))
[1] 10742.34
```

Approximately how long time per person?

How many deaths are there in total?

```
> (d <- with(DMlate, sum(!is.na(dodth))))
[1] 479
```

What is the overall mortality rate? (remember the scale).

```
> d / y
[1] 0.04458991
```

This is per 1 person-year, the rate per 100 person-years is

```
> d / y * 100
[1] 4.458991
```

so 4.5% per year.

2.3 Mortality by age

It is a bit of bold assumption to assume that mortality is constant over time. It likely depends on age. We can make a table of the mortality rates by age category:

```
> tt <- xtabs(cbind(D = !is.na(dodth),
+                   Y = dox - dodm) ~
+                   agr,
+             data = mutate(DMlate,
+                           agr = cut(dodm - dobth,
+                                     seq(0, 100, 10),
+                                     right = FALSE)))
> tt
agr                 D           Y
  [0,10)      0.00000   104.37235
  [10,20)     1.00000   146.11088
  [20,30)     0.00000   271.86037
  [30,40)     3.00000   710.50513
  [40,50)    14.00000  1503.36208
  [50,60)    55.00000  2323.28268
  [60,70)    99.00000  2942.04244
  [70,80)   188.00000  2027.45517
  [80,90)    98.00000   660.26557
  [90,100)   21.00000    53.08419
> cbind(mort = tt[, "D"] / tt[, "Y"] * 100)
               mort
[0,10)    0.0000000
[10,20)   0.6844117
[20,30)   0.0000000
[30,40)   0.4222348
[40,50)   0.9312461
[50,60)   2.3673400
[60,70)   3.3650092
[70,80)   9.2727081
[80,90)  14.8425125
[90,100) 39.5598019
```

CODE EXPLAINED: `xtabs` sums the first argument (left hand side of formula, `cbind(...)`, `D`, deaths and `Y`, person-years) in categories of the right hand side, `age`. `age` is defined in the `mutate` function, by using `cut` that classifies a continuous variable. (`dodm-dobth`)

Does the mortality rate depend on age?

Note that we could have written:

```
> cbind(tt[, 1] / tt[, 2] * 100)
               [,1]
[0,10)    0.0000000
[10,20)   0.6844117
[20,30)   0.0000000
[30,40)   0.4222348
[40,50)   0.9312461
[50,60)   2.3673400
[60,70)   3.3650092
[70,80)   9.2727081
[80,90)  14.8425125
[90,100) 39.5598019
```

but it would not have been as readable, you would have to backtrack the code to see what the 1st resp. 2nd columns were.

NOTE: It is only the *secondary* purpose of programming to get things right, the *primary* purpose is to demonstrate that you actually did with data what you claim to have done.

2.4 Age and age is not the same

What we have done is to classify follow-up (deaths and risk time) by the age *at diagnosis*. It would be more relevant to classify the follow-up by *current* age (also called *attained* age)—the age of the person as it changes during follow-up.

Now this would require that the follow-up for each person be classified according to current age, so persons would potentially contribute follow-up in more than one age class.

2.4.1 Current age

That is a bit of a book-keeping exercise, but there is a tool for this; the `Lexis` machinery. Set up the follow-up data as a `Lexis` data frame, defining age as a timescale:

```
> Lx <- Lexis(entry = list(age = dodm - dobth),
+              exit = list(age = dox  - dobth),
+       exit.status = factor(!is.na(dodth), labels = c("Alive","Dead")),
+              data = DMlate)
NOTE: entry.status has been set to "Alive" for all.
NOTE: Dropping  1  rows with duration of follow up < tol
```

```
> subset(DMlate, near(dodm, dox))
```

```
     sex    dobth     dodm    dodth dooad doins      dox
1895   F 1936.067 1996.984 1996.984    NA    NA 1996.984
```

```
> summary(Lx)
```

```
Transitions:
     To
From    Alive Dead  Records:  Events: Risk time:  Persons:
  Alive  1521  478      1999      478   10742.34      1999
```

```
> head(Lx)
```

```
 lex.id   age lex.dur lex.Cst lex.Xst sex   dobth    dodm dodth   dooad   doins  dox
      1 39.89    6.52   Alive   Alive   F 1963.59 2003.48    NA      NA      NA 2010
      2 61.52    4.35   Alive   Alive   M 1944.13 2005.64    NA 2005.78      NA 2010
      3 51.10    2.11   Alive   Alive   F 1956.79 2007.89    NA      NA      NA 2010
      4 54.61    3.03   Alive   Alive   M 1952.35 2006.97    NA 2006.97 2008.03 2010
      5 51.12    6.64   Alive   Alive   M 1952.24 2003.36    NA 2005.85      NA 2010
      6 23.19    8.05   Alive   Alive   M 1978.76 2001.95    NA      NA 2001.97 2010
```

CODE EXPLAINED: `Lexis` takes the data frame (here `DMlate`) and adds some variables that describes the follow-up: person id, `lex.id`; the timescale, `age`—age at start of follow-up; the risk time, `lex.dur`, the name of the current `st`ate the person is in,`lex.Cst`; the state the person e`X`its to after `lex.dur` risk time.

A number of attributes are also defined by `Lexis`.

The `summary` produces an overview of the follow-up.

It is still the same dataset, but amended with some extra variables.

What does `str(Lx)` tell you?

We can now subdivide the follow-up taking advantage of the `Lexis` structure:

```
> Sx <- splitLexis(Lx, breaks = seq(0, 100, 5))
> summary(Lx)
Transitions:
     To
From    Alive Dead  Records:  Events: Risk time:  Persons:
  Alive  1521  478      1999      478   10742.34      1999
> summary(Sx)

Transitions:
     To
From    Alive Dead  Records:  Events: Risk time:  Persons:
  Alive  3656  478      4134      478   10742.34      1999
```

CODE EXPLAINED: `seq` generates a sequence of equidistant numbers, and `splitLexis` subdivides the follow-up in 5-year age-classes, so now age in the split datset represents *current* age.

The two `summary` statements demonstrates that the follow-up (events and risk time) is the same, but distributed over a larger number of records in `Sx`.

We can then use `xtabs` to tabulate deaths and person-time by current age:

```
> tt <- xtabs(cbind(D = lex.Xst == "Dead",
+                   Y = lex.dur) ~
+                   I(floor(age / 5) * 5),
+             data = Sx)
> tt
I(floor(age/5) * 5)           D           Y
                  0    0.000000   13.258727
                  5    0.000000   44.838467
                 10    0.000000   81.636550
                 15    1.000000   65.104038
                 20    0.000000   75.470910
                 25    0.000000   97.076660
                 30    0.000000  203.124572
                 35    2.000000  281.568104
                 40    3.000000  448.275838
                 45    6.000000  654.642710
                 50    6.000000  879.850787
                 55   22.000000 1189.978097
                 60   33.000000 1318.171116
                 65   47.000000 1463.942505
                 70   67.000000 1396.142368
                 75   88.000000 1168.071184
                 80   87.000000  789.472964
                 85   66.000000  404.648871
                 90   39.000000  131.465435
                 95   10.000000   34.629021
                100    1.000000    0.971937
> (rt <- cbind(mort = tt[, "D"] / tt[, "Y"] * 100))
```

```
          mort
0    0.0000000
5    0.0000000
10   0.0000000
15   1.5360030
20   0.0000000
25   0.0000000
30   0.0000000
35   0.7103077
40   0.6692308
45   0.9165305
50   0.6819338
55   1.8487735
60   2.5034686
65   3.2105086
70   4.7989375
75   7.5337874
80  11.0200101
85  16.3104372
90  29.6655924
95  28.8775127
100 102.8873239
```

We can show these mortality rates graphically:

```
> plot(rownames(tt), rt, log ="y", type = "o", xlab = "Age", pch = 16)
```

CODE EXPLAINED: The `log="y"` defines a logarithmic y-axis, `type="o"` requests that lines and points are overplotted, and `pch=16` defines the plotting symbol as a blob.

The plot is a bit misleading, because the assumption underlying the calculations as that the mortality rates are constant in each interval, so we really have a step-function for mortality

```
> plot(rownames(tt), rt, log ="y", type = "s", xlab = "Age")
> points(rownames(tt), rt, pch = 16)
```

CODE EXPLAINED: The `type="s"` makes a step function of the curve instead of just connecting the points. The `points` command adds the points as blobs (`pch=16`).

2.5 Smooth model for age

From the plot we see that the curve is ragged; this would be even worse if we did the exercise in, say, 1-year classes. So far we estimated 21 parameters (one per 5 year age class), and estimating 101 would be even worse. But the assumption of constant rates in 5 year intervals is a bit coarse, 1 year would be a more reasonable approximation. But highly unrealistic that wee would need 101 parameters to describe mortality by age.

The solution is to put a parametric restriction on mortality rates in the 1-year classes, basically using the left endpoint of the intervals as a quantitative (metric, continuous) variable.

This parametric modeling also has the advantage that we do not need to tabulate data, we can directly fit a model for age to the split data.

Split the follow-up in `Lx` in 1-year intervals with `splitLexis`:

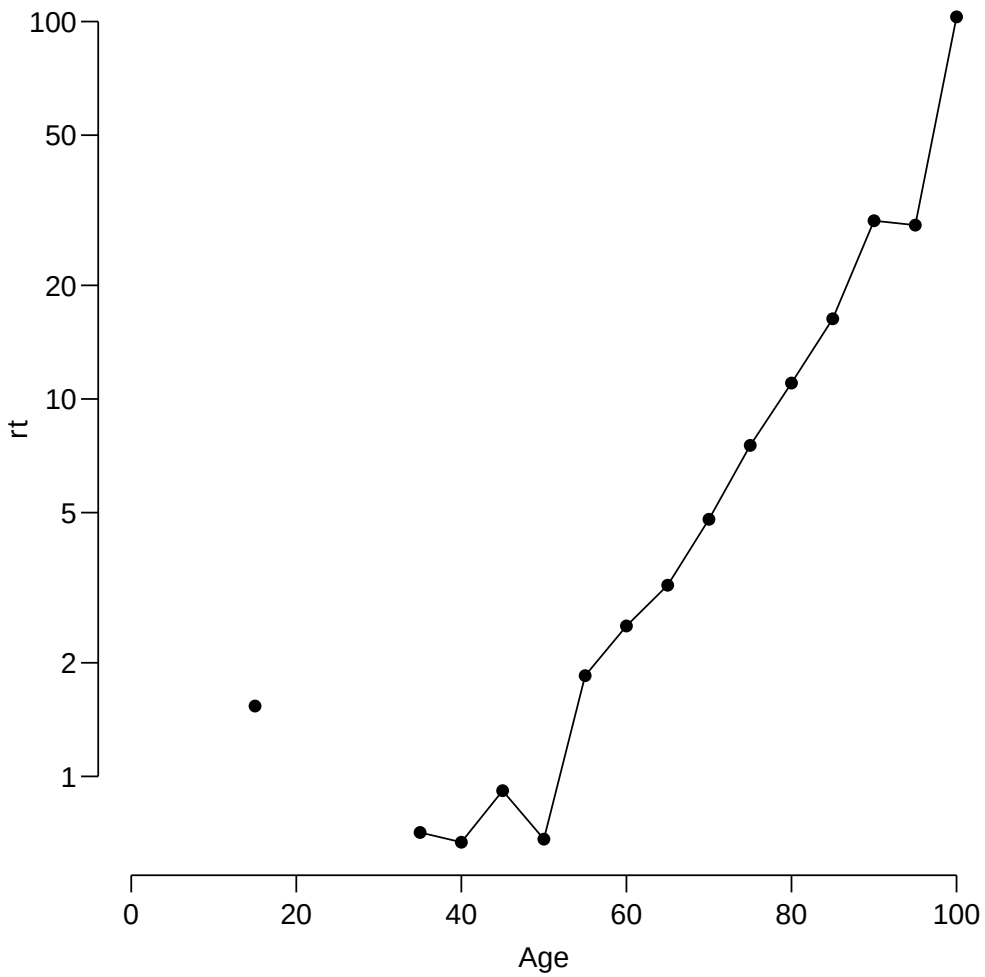

Figure 2.1: *Mortality rates among Danish diabetes patients by current age.../graph/mort-morta*

```
> Sx <- splitLexis(Lx, breaks = 0:100, time.scale = "age")
> summary(Lx)
Transitions:
     To
From    Alive Dead  Records:  Events: Risk time:  Persons:
  Alive  1521  478      1999      478   10742.34      1999
> summary(Sx)
Transitions:
     To
From    Alive Dead  Records:  Events: Risk time:  Persons:
  Alive 12201  478     12679      478   10742.34      1999
```

We see that instead of 2000 records as in `Lx` we now have some 13,000 records, but the same risk time. We see that the follow-up of person (`lex.id`) 2 has been split in 5 records, and person 3 in 3 records. And we see that the variable `age` now represents the `current` age—it changes as the person ages.

```
> subset(Lx, lex.id %in% 2:3)
 lex.id   age lex.dur lex.Cst lex.Xst sex   dobth    dodm dodth   dooad doins  dox
      2 61.52    4.35   Alive   Alive   M 1944.13 2005.64    NA 2005.78    NA 2010
      3 51.10    2.11   Alive   Alive   F 1956.79 2007.89    NA      NA    NA 2010
```

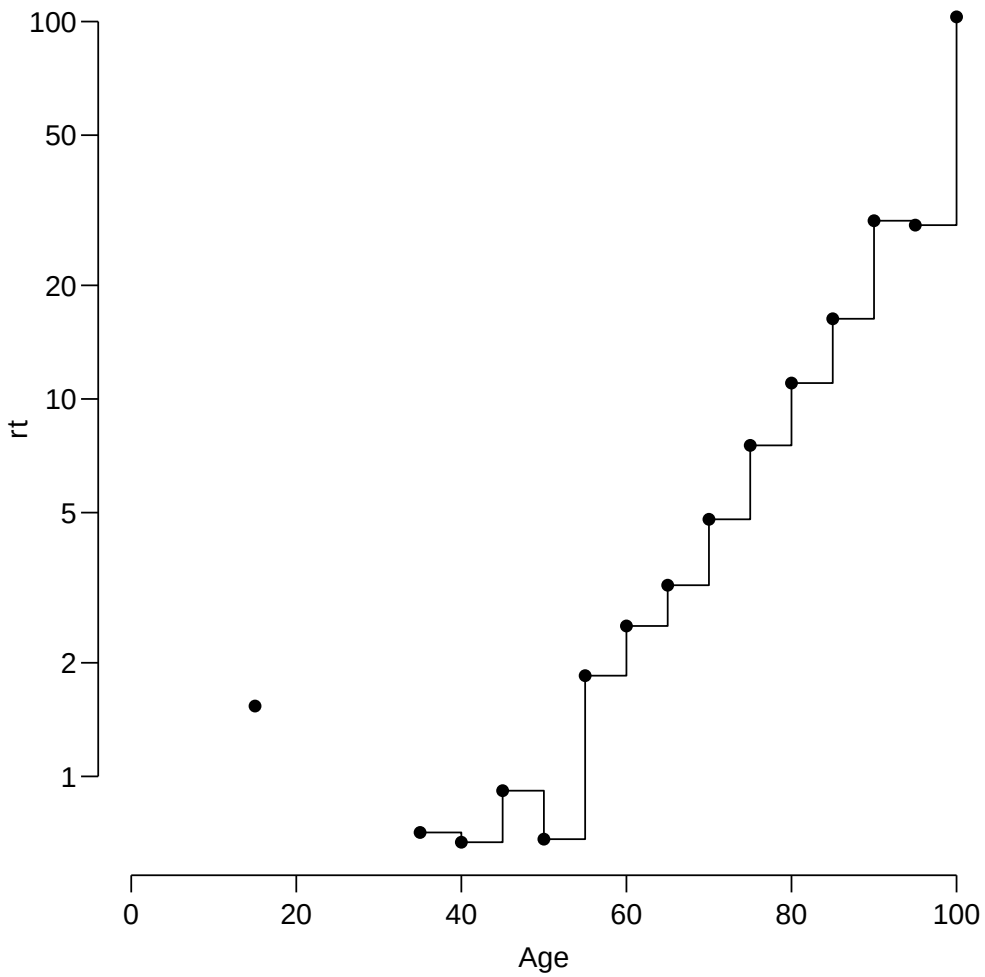

Figure 2.2: *Mortality rates among Danish diabetes patients by current age, showing constant rates in 5-year classes* ../graph/mort-mortas

```
> subset(Sx, lex.id %in% 2:3)
```

lex.id	age	lex.dur	lex.Cst	lex.Xst	sex	dobth	dodm	dodth	dooad	doins	dox
2	61.52	0.48	Alive	Alive	M	1944.13	2005.64	NA	2005.78	NA	2010
2	62.00	1.00	Alive	Alive	M	1944.13	2005.64	NA	2005.78	NA	2010
2	63.00	1.00	Alive	Alive	M	1944.13	2005.64	NA	2005.78	NA	2010
2	64.00	1.00	Alive	Alive	M	1944.13	2005.64	NA	2005.78	NA	2010
2	65.00	0.87	Alive	Alive	M	1944.13	2005.64	NA	2005.78	NA	2010
3	51.10	0.90	Alive	Alive	F	1956.79	2007.89	NA	NA	NA	2010
3	52.00	1.00	Alive	Alive	F	1956.79	2007.89	NA	NA	NA	2010
3	53.00	0.21	Alive	Alive	F	1956.79	2007.89	NA	NA	NA	2010

The `Lexis` machinery allows a simple way of modeling the mortality rates:

```
> mL <- glmLexis(Sx, ~ Ns(age, knots = seq(40, 80, 10)))
stats::glm Poisson analysis of Lexis object Sx with log link:
Rates for the transition:
Alive->Dead
```

CODE EXPLAINED: `glmLexis` fits a rate model using the events and person time as response and in this case age as explanatory variable. It uses the `Lexis` structure of `Sx` to find the outcome variables. It is designed to

The corresponding Poisson regression with `glm` would be:

```
> mP <- glm((lex.Xst == "Dead") ~ Ns(age, knots = seq(40, 80, 10)),
+           offset = log(lex.dur),
+           family = poisson,
+           data = Sx)
> round(cbind(ci.exp(mL),
+             ci.exp(mP)), 4)
                                  exp(Est.)   2.5%   97.5% exp(Est.)   2.5%   97.5%
(Intercept)                          0.0056 0.0032  0.0097    0.0056 0.0032  0.0097
Ns(age, knots = seq(40, 80, 10))1    3.9144 1.7492  8.7598    3.9144 1.7490  8.7608
Ns(age, knots = seq(40, 80, 10))2    6.2849 3.7176 10.6249    6.2849 3.7173 10.6258
Ns(age, knots = seq(40, 80, 10))3   18.3976 7.0794 47.8107   18.3976 7.0743 47.8456
Ns(age, knots = seq(40, 80, 10))4   13.2045 7.7908 22.3800   13.2045 7.7907 22.3803
```

The small differences are because the two methods use different algorithms.

But the parameter estimates are pretty useless; we need predicted rates.

2.6 Predicted mortality rates

With a model for how the mortality depends on age we can tease out the predicted rates and show how they look as a functions of age.

To that end is needed a *prediction data frame* (we call it `nd` for `new data`)—a data frame with all explanatory variables in the model set to values we want predictions for:

```
> nd <- data.frame(age = 30:95)
> pr.rates <- ci.pred(mL, nd) * 100
> head(pr.rates)
   Estimate      2.5%     97.5%
1 0.3582582 0.1275648 1.0061469
2 0.3743926 0.1413134 0.9919077
3 0.3912537 0.1563451 0.9791130
4 0.4088741 0.1727125 0.9679558
5 0.4272881 0.1904434 0.9586845
6 0.4465314 0.2095274 0.9516190
```

The risk time (in the variable `lex.dur`) is in units of years, so the units of the predicted rates is years^{-1}, a bit awkward, so we multiply by 100 to get the mortality rate in units per 100 PY:

```
> matshade(nd$age, pr.rates, plot = TRUE, log = "y", lwd = 3,
+          xlab = "Attained age", ylab = "Mortality rate per 100 PY")
```

2.7 Duration of diabetes

We will now explore how mortality depend on time since diabetes diagnosis:
Set up a different `Lexis` object, but now with time from diagnosis of diabetes as time scale:

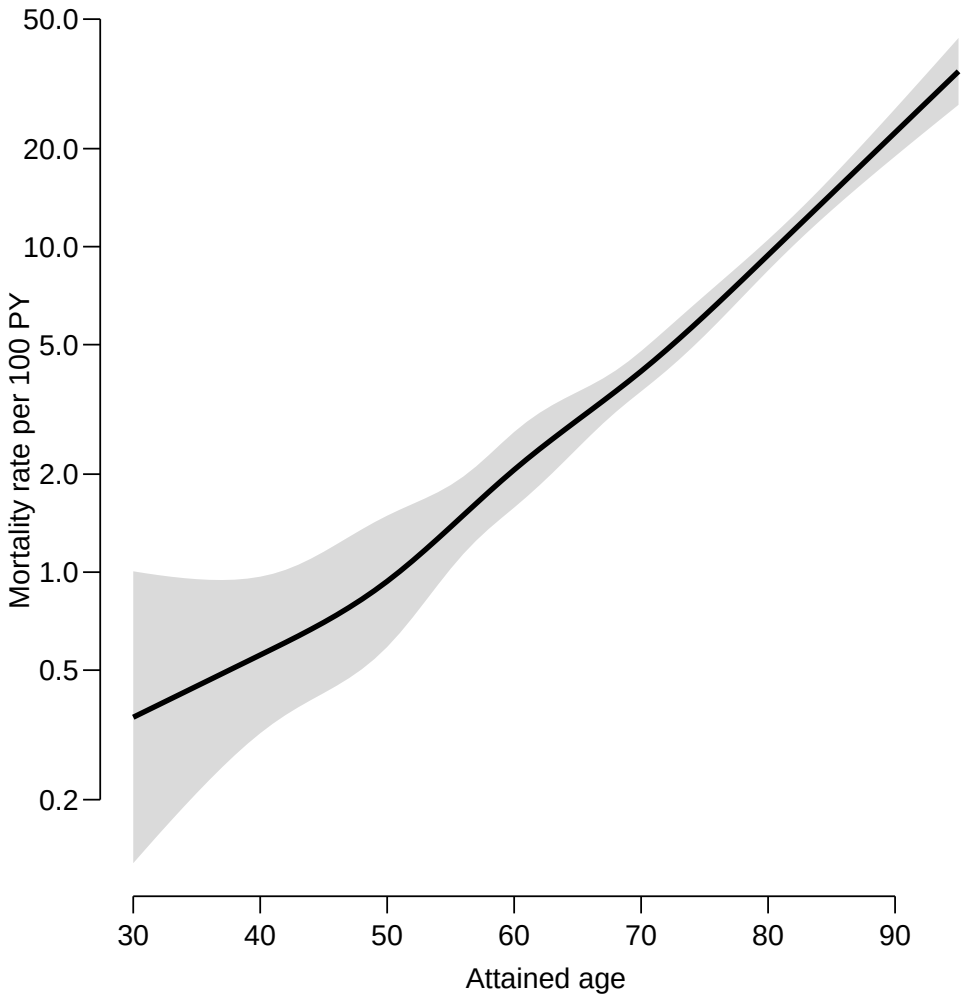

Figure 2.3: *Mortality rates among Danish diabetes patients by attained age.* ../graph/mort-mortaspl

```
> Lx <- Lexis(entry = list(tfd = dodm - dodm),
+             exit = list(tfd = dox  - dodm),
+       exit.status = factor(!is.na(dodth), labels = c("Alive", "Dead")),
+             data = DMlate)
NOTE: entry.status has been set to "Alive" for all.
NOTE: Dropping  1  rows with duration of follow up < tol
> summary(Lx)
Transitions:
     To
From    Alive Dead  Records:  Events: Risk time:  Persons:
  Alive  1521  478      1999      478   10742.34      1999
> head(Lx)
 lex.id tfd lex.dur lex.Cst lex.Xst sex   dobth    dodm dodth   dooad   doins  dox
      1   0    6.52   Alive   Alive   F 1963.59 2003.48    NA      NA      NA 2010
      2   0    4.35   Alive   Alive   M 1944.13 2005.64    NA 2005.78      NA 2010
      3   0    2.11   Alive   Alive   F 1956.79 2007.89    NA      NA      NA 2010
      4   0    3.03   Alive   Alive   M 1952.35 2006.97    NA 2006.97 2008.03 2010
      5   0    6.64   Alive   Alive   M 1952.24 2003.36    NA 2005.85      NA 2010
      6   0    8.05   Alive   Alive   M 1978.76 2001.95    NA      NA 2001.97 2010
```

> CODE EXPLAINED: If we want time from diabetes as the timescale (call it `tfd`), we must subtract date of diabetes (`dodm`) from the dates of entry and exit. We see that all persons are coded 0 for `tfd`—because follow-up starts at 0 after diagnosis of diabetes

Now split follow-up in intervals of 0.5 years:

```
> Sx <- splitLexis(Lx, breaks = seq(0, 20, 0.5))
> summary(Lx)
Transitions:
     To
From    Alive Dead  Records:  Events: Risk time:  Persons:
  Alive  1521  478      1999      478   10742.34      1999
> summary(Sx)
Transitions:
     To
From    Alive Dead  Records:  Events: Risk time:  Persons:
  Alive 22020  478     22498      478   10742.34      1999
```

We see that instead of 1999 records as in `Lx` we now have some 22,000 records, but again the same risk time and number of events. We see that the follow-up of person (`lex.id`) 3 has been split in 5 records, but still with a total of 2.11 person-years:

```
> subset(Lx, lex.id %in% 2:3)
 lex.id tfd lex.dur lex.Cst lex.Xst sex   dobth    dodm dodth   dooad doins  dox
      2   0    4.35   Alive   Alive   M 1944.13 2005.64    NA 2005.78    NA 2010
      3   0    2.11   Alive   Alive   F 1956.79 2007.89    NA       NA    NA 2010
> subset(Sx, lex.id %in% 2:3)
 lex.id tfd lex.dur lex.Cst lex.Xst sex   dobth    dodm dodth   dooad doins  dox
      2 0.0    0.50   Alive   Alive   M 1944.13 2005.64    NA 2005.78    NA 2010
      2 0.5    0.50   Alive   Alive   M 1944.13 2005.64    NA 2005.78    NA 2010
      2 1.0    0.50   Alive   Alive   M 1944.13 2005.64    NA 2005.78    NA 2010
      2 1.5    0.50   Alive   Alive   M 1944.13 2005.64    NA 2005.78    NA 2010
      2 2.0    0.50   Alive   Alive   M 1944.13 2005.64    NA 2005.78    NA 2010
      2 2.5    0.50   Alive   Alive   M 1944.13 2005.64    NA 2005.78    NA 2010
      2 3.0    0.50   Alive   Alive   M 1944.13 2005.64    NA 2005.78    NA 2010
      2 3.5    0.50   Alive   Alive   M 1944.13 2005.64    NA 2005.78    NA 2010
      2 4.0    0.35   Alive   Alive   M 1944.13 2005.64    NA 2005.78    NA 2010
      3 0.0    0.50   Alive   Alive   F 1956.79 2007.89    NA       NA    NA 2010
      3 0.5    0.50   Alive   Alive   F 1956.79 2007.89    NA       NA    NA 2010
      3 1.0    0.50   Alive   Alive   F 1956.79 2007.89    NA       NA    NA 2010
      3 1.5    0.50   Alive   Alive   F 1956.79 2007.89    NA       NA    NA 2010
      3 2.0    0.11   Alive   Alive   F 1956.79 2007.89    NA       NA    NA 2010
```

As we saw for age, the `Lexis` machinery allows a simple way of modeling the mortality rates as a function of time from diagnosis:

```
> tL <- glmLexis(Sx, ~ Ns(tfd, knots = c(0, 1, 3, 6, 10)))
stats::glm Poisson analysis of Lexis object Sx with log link:
Rates for the transition:
Alive->Dead
```

As before, derive the predicted mortality rates and plot the mortality, but now as a function of time since diagnosis:

```
> nd <- data.frame(tfd = seq(0, 12, 0.2))
> pr.rates <- ci.pred(tL, nd)
> head(pr.rates)
    Estimate       2.5%      97.5%
1 0.06056585 0.04788313 0.07660781
2 0.05424695 0.04494245 0.06547778
3 0.04879401 0.04150913 0.05735739
4 0.04426341 0.03771416 0.05194997
5 0.04066812 0.03413191 0.04845600
6 0.03800474 0.03132808 0.04610433
```

CODE EXPLAINED: The prediction data frame `nd` must have one column for each of the explanatory variables in the model, in this case only one, `tfd`.

The risk time (in the variable `lex.dur`) is in units of years, so the units of the predicted rates is years^{-1}, a bit awkward, so we multiply by 100 to get units per 100 PY:

```
> matshade(nd$tfd, pr.rates * 100,
+          plot = TRUE, log = "y", lwd = 3,
+          xlab = "Time since DM diagnosis",
+          ylab = "Mortality rate per 100 PY")
```

The curve shows a well-known phenomenon with high mortality at diagnosis and a decline during the initial time (approx. 2 years). This is most likely because newly diagnosed persons have an over-representation of frail persons with a high mortality that contribute to a high initial mortality. As these are removed from the population, the mortality declines, and eventually increases by age / duration.

2.8 Survival after diabetes

The data frame `DMlate` is follow-up of a random sample of diabetes patients from the date of diagnosis of diabetes, so it would be natural to ask for the survival probability as a function of time from diagnosis. The link between the mortality, $\lambda(t)$ and the origin $t = 0$ on one hand and the survival function $S(t)$ on the other hand is

$$S(t) = \exp\Big(-\int_0^t \lambda(u)\,\mathrm{d}u\Big)$$

So it is straight forward to derive the survival function by numerical integration of the mortality in `pr.rates[,1]`

```
> head(cbind(nd, pr.rates))
  tfd   Estimate       2.5%      97.5%
1 0.0 0.06056585 0.04788313 0.07660781
2 0.2 0.05424695 0.04494245 0.06547778
3 0.4 0.04879401 0.04150913 0.05735739
4 0.6 0.04426341 0.03771416 0.05194997
5 0.8 0.04066812 0.03413191 0.04845600
6 1.0 0.03800474 0.03132808 0.04610433
> surv <- exp(-cumsum(pr.rates[, 1]) * 0.2)
> plot(nd$tfd, surv, type = "l", ylim = c(0,1))
```

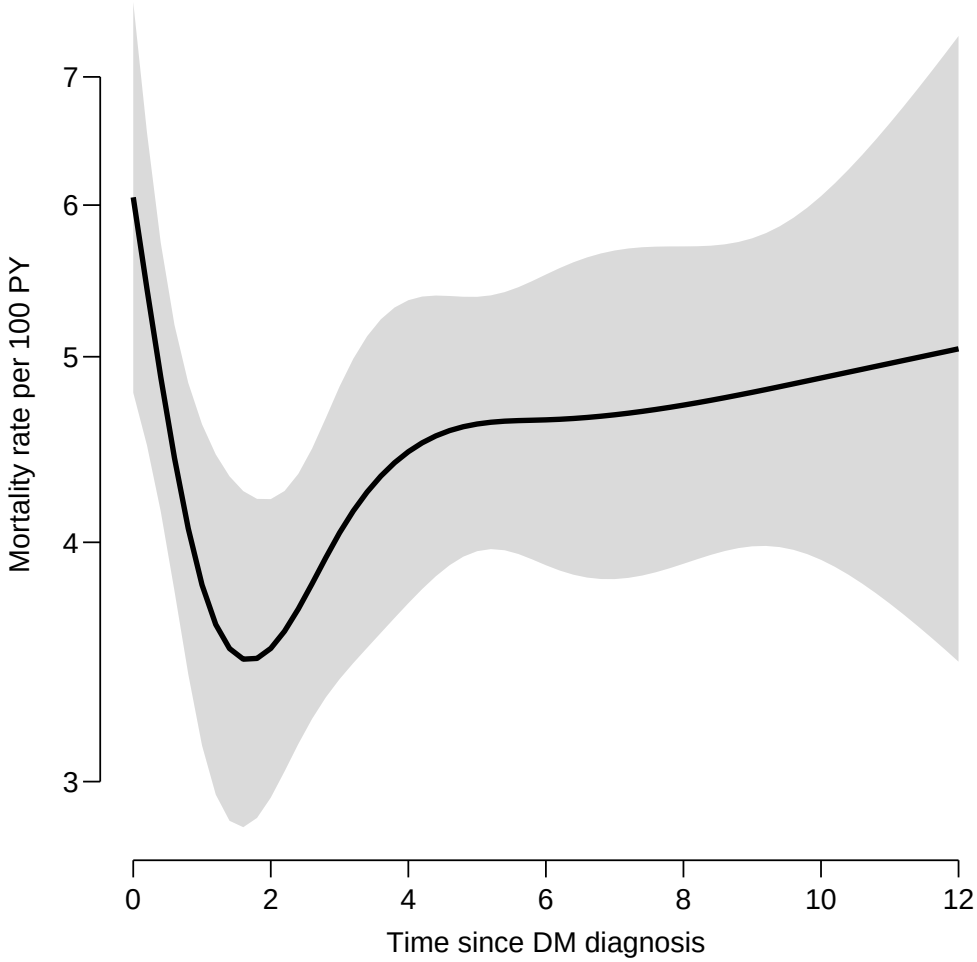

Figure 2.4: *Mortality rates among Danish diabetes patients by time from diagnosis.* `../graph/mort-mortdspl`

CODE EXPLAINED: The integral in the formula for the survival function is numerically approximated by the cumulative sum of the rates (in `pr.rates`) multiplied by the interval width—the area under the mortality curve.

The tricky thing is however to get a confidence interval for the survival function—it is quite convoluted. Fortunately this has been implemented in the function `ci.surv`. For comparison we overlay the Kaplan-Meier estimate of the survival function:

```
> matshade(nd$tfd, ci.surv(tL, nd), plot = TRUE,
+          lwd = 2, ylim = c(0.5,1), yaxs = "i",
+          xlab = "Time since diagnosis (years)",
+          ylab = "Survival probability")
NOTE: interval length chosen from  as tfd[2] - tfd[1]
> lines(survfit(Surv(dox - dodm, !is.na(dodth)) ~ 1, data = Lx), col = "red")
```

From the graph we see that the two estimates are almost indistinguishable, but the parametric estimate is more credible as a clear result from a proper statistical model.

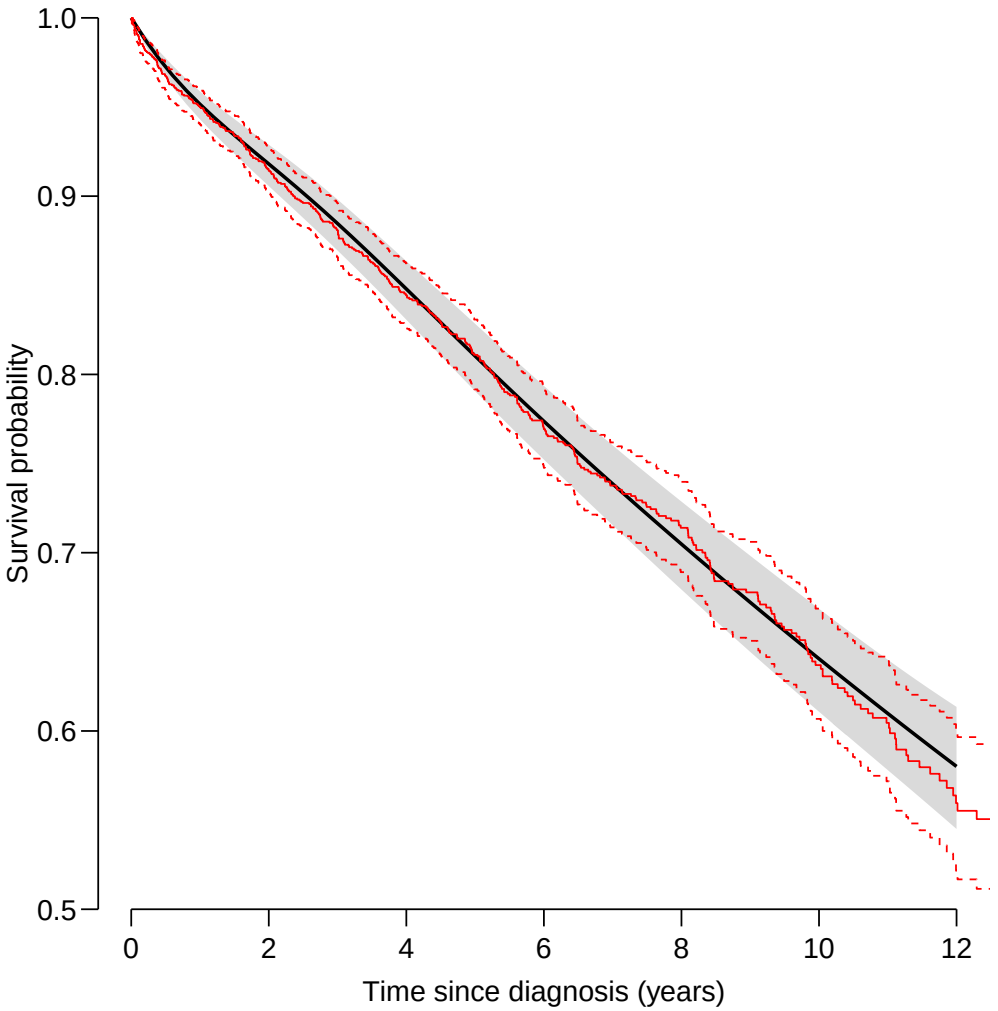

Figure 2.5: *Survival of Danish diabetes patients. Thick line and gray shade is based on the parametric mortality function, the thin red lines are the Kaplan-Meier estimator.../graph/mort-surv*

A thorough exposition of this type of analysis is in the document "Who needs the Cox model anyway?" at https://bendixcarstensen.com/WntCma.pdf. The document also explains the relationship between the Cox-model and the Poisson-model—how the Cox-model is a special case of the Poisson model.