A not so short introduction to R for Epidemiology

Notes for course at Université Bordeaux
January 2015
http://BendixCarstensen.com/Epi
Version 5

Compiled Monday 19th January, 2015, 17:55
from: /home/bendix/teach/Epi/Bordeaux2015-01/pracs/NSE

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

Edition 2014 by Bendix Carstensen

Contents

Preface

This set of notes and exercises have grown out of the course "Statistical Practise in Epidemiology with R" (http://BendixCarstensen.com/SPE) and of small workshops for my epidemiological colleagues in the Clinincal Epidemiology section at Steno Diabetes Center.

The first version of this document (mainly the first 7 chapters) were written jointly with Michael Hills (Highgate, London, retired) and Martyn Plummer (IARC, Lyon), to whom I owe much thanks for harsh comments. The major part of the document is however my responsibility, in particular the impenetrable parts that have not been scutinized thoroughly by others.

Bendix Carstensen
Steno Diabetes Center
January 2015.

Program for Bordeaux course

Time: Thursday 22nd January 2015, 14:00

Venue: Lecture Hall Pierre Alexandre Louis, ISPED, University of Bordeaux, 146, rue Leo-Saignat

Contents: There will be three time slots, the two first aprroximately equally divided between lecture and practicals. The timing will be:

14:00–15:15	Epidemiological study types and data types; simple analyses of epidemiological data. Practicals: ch. 5, 7.1, 7.2
15:30–16:15	Representation of follow-up data in the `Epi` package: The `Lexis` machinery Practicals: ch. 9
16:30–17:45	Cox-models and Poisson models: Example of identity and how to model and report multiple time scale models Discussion

Prerequisites: A working installation of:

- R, version 3.1.2 (any above 3.0.0 will do)
- the `Epi` package, version 1.1.67 (no other will do)
- You can check this by starting R and run the commands:

```
> library( Epi )
> sessionInfo()
  R version 3.1.2 (2014-10-31)
  Platform: x86_64-pc-linux-gnu (64-bit)

  locale:
   [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C                LC_TIME=en_US.UTF-8
   [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8     LC_MESSAGES=en_US.UTF-8
   [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                   LC_ADDRESS=C
  [10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

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

  other attached packages:
  [1] Epi_1.1.67
```

Chapter 1

Getting R running on your computer

1.1 What is R?

R is free program for data analysis and graphics. It contains all state of the art statistical methods, and has become the preferred analysis tool for most professional statisticians in the world. It can be used as simple calculator and as a very specialized statistical analysis and reporting machinery.

The special thing about R is that you enter commands from the keyboard into a console window, where you also see the results. This is an advantage because you end up with a script that you can use to *reproduce* your analyses—a requirement in any scientific endeavour.

The disadvantage is that you somehow have to find out what to type. The practicals will contain some hints, and you will mostly be using R as a calculator, as you just saw — type an expression, hit the return key and you get the result.

1.2 Getting R

You can obtain R, which is free, from CRAN (the **C**omprehensive **R** **A**rchive **N**etwork), at http://cran.r-project.org/. Under "`Download R for Windows`" click on "install R for the first time" and then on "Download R 3.0.2 for Windows", which is a self-extracting installer. This means that if you save it to your computer somewhere and click on it, it will install R for you.

Apart from what you have downloaded there are several thousand add-on packages to R dealing with all sorts of problems from ecology to fiance and incidentally, epidemiology. You must download these manually. In this course we shall only need the `Epi` package.

1.2.1 Starting R

You start R by clicking on the icon that the installer has put on your desktop. You should edit the properties of this, so that R starts in the folder that you have created on your computer for this course.

Once you have installed R, start it, and in the menu bar click on Packages → Install package(s)..., chose a mirror (this is just a server where you can get the stuff), and then the `Epi` package.

Once R (hopefully) has told you that it has been installed, you can type:

```
> library( Epi )
```

to get access to the `Epi` package. You can get an overview of the functions and datasets in the package by typing:

```
> library( help=Epi )
```

It should be apparent that you have version 1.1.49 of the `Epi` package. For documentauon purposes it is often useful to have the following at the beginning of your program:

```
> sessionInfo()
  R version 3.0.2 (2013-09-25)
  Platform: x86_64-pc-linux-gnu (64-bit)

  locale:
   [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
   [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
   [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
   [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
   [9] LC_ADDRESS=C               LC_TELEPHONE=C
  [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

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

  other attached packages:
  [1] Epi_1.1.67

  loaded via a namespace (and not attached):
  [1] tools_3.0.2
```

1.2.2 Quitting R

Type `q()` in the console, and answer "No" when asked whether you want to save workspace image.

1.3 Working with the script editor

If you click on File → New script, R will open a window for you which is a text-editor very much like Notepad.

If you write a command in it you can transfer it to the R console and have it executed by pressing CTRL-r. If nothing is highlighted, the line where the cursor is will be transmitted to the console and the cursor will move to the next line. If a part of the screen is highlighted the highlighted part will be transmitted to the console. Highlighting can also be used to transmit only a part of a line of code.

1.3.1 Rstudio

This is an interface that allows you to have a slithly more flexible script-editor than the built-in, R-studio har syntax coloriung which can be very nice. You can obtain it from http://rstudio.com.

1.3.2 Try!

Now, **either** open a script by File → New script, and type (omit the ">" in the beginning of the line), **or** fire up R-studio and type in the editor window:

```
> 5+7
> pi
> 1:10
> N <- c(27,33,81)
> N
```

Run the lines one at a time by pressing CTRL-r, (in R-studio it is CTRL-ENTER) and see what happens.

You can also type the commands in the console directly. But then you will not have a record of what you have done. Well, you can press File → Save History and save all you typed in the console (including the 73.6% commands with errors).

1.4 Changing the looks ...

1.4.1 ... of standard R

If you want R to start up with a different font, different colors etc., the go to the folder where R is installed — most likely `Program Files\R\R-2.13.1`, then to the folder `etc`, and open the file `Rconsole` with Notepad. In the file are specifications on how R will look when you start it, pretty self-explanatory, except perhaps for MDI.

MDI means "**M**ultiple **D**isplay **I**nterface", which means you get a single R-window, and within that sub-windows with the console, the script editor, graphs etc. If this is set to "no", you get SDI which means "**S**ingle **D**isplay **I**nterface", which means that R will open the console, script editor etc. in separate windows of their own.

A withe background can be trying to look at so on my (BxC) computer I use a bold font and the following colors:

```
> background = gray5
> normaltext = yellow2
> usertext = green
> pagerbg = gray5
> pagertext = yellow2
> highlight = red
> dataeditbg = gray5
> dataedittext = red
> datadituser = yellow2
> editorbg = gray5
> editortext = lightblue
```

(If you want to know which colors are available in R, just give the command `colors()`).

1.4.2 ... of Rstudio

Click on Tools→Global options...→Apperance and choose Consolas font, 16 pt, Editor theme `Cobalt`

1.5 Further reading

On the CRAN web-site the last menu-entry on the left is "Contributed" and will take you to a very long list of various introductions to R, including manuals in esoteric languages such as Danish, Finnish and Hungarian.

Chapter 2

Some basic commands in R

2.1 Preliminaries

The purpose of these notes is to describe a small subset of the Rlanguage, sufficient to allow someone new to R to get started. The exercises are important because they reinforce basic aspects of R. For further details about R we refer the reader to **An Introduction to R** by W.N.Venables, D.M.Smith, and the R development team. This can be downloaded from the R website at http://www.r-project.org.

To start R click on the R icon. To change your working directory click on File → Change dir... and select the directory you want to work in. Alternatively you can write:

```
> setwd("c:/where/alll/my/files/are")
```

To get out of R click on the File menu and select Exit, or simpler just type "q()". You will be offered the chance to save the work space, but at this stage just exit without saving, then start R again, and change the working directory, as before.

R is case sensitive, so that A is different from a. Commands in R are generally separated by a newline, although a semi-colon can also be used. When using R it makes sense to avoid as much typing as possible by recalling previous commands using the vertical arrow key and editing them.

2.2 Using R as a calculator

Typing 2+2 will return the answer 4, typing 2^3 will return the answer 8 (2 to the power of 3), typing log(10) will return the natural logarithm of 10, which is 2.3026, and typing sqrt(25) will return the square root of 25.

Instead of printing the result you can store it in an object, say

```
> a <- 2+2
```

which can be used in further calculations. The expression <-, pronounced "gets", is called the assignment operator, and is obtained by typing < and then -. The assignment operator can also be used in the opposite direction, as in

```
> 2+2 -> a
```

The contents of `a` can be printed by typing `a`.

Standard probability functions are readily available. For example, the probability below 1.96 in a standard normal (i.e. Gaussian) distribution is obtained with

```
> pnorm(1.96)
```

while

```
> pchisq(3.84,1)
```

will return the probability below 3.84 in a χ^2 distribution on 1 degree of freedom, and

```
> pchisq(3.84,1,lower.tail=FALSE)
```

will return the probability above 3.84.

Exercise 2.1.

1. Calculate $\sqrt{3^2 + 4^2}$.
2. Find the probability above 4.3 in a chi-squared distribution on 1 degree of freedom.

2.3 Objects and functions

All commands in R are *functions* which act on *objects*. One important kind of object is a *vector*, which is an ordered collections of numbers, or an ordered collection of character strings. Examples of vectors are 4, 6, 1, 2.2, which is a numeric vector with 4 components, and "Charles Darwin", "Alfred Wallace" which is a vector of character strings with 2 components. The components of a vector must be of the same type (numeric or character). The combine function `c()`, together with the assignment operator, is used to create vectors. Thus

```
> v <- c(4, 6, 1, 2.2)
```

creates a vector `v` with components 4, 6, 1, 2.2 by first combining the 4 numbers 4, 6, 1, 2.2 in order and then assigning the result to the vector `v`. Collections of components of different types are called *lists*, and are created with the `list()` function. Thus

```
> m <- list(4, 6, "name of company")
```

creates a list with 3 components. The main differences between the numbers 4, 6, 1, 2.2 and the vector `v` is that along with `v` is stored information about what sort of object it is and hence how it is printed and how it is combined with other objects. Try

```
> v
> 3+v
> 3*v
```

and you will see that R understands what to do in each case. This may seem trivial, but remember that unlike most statistical packages there are many different kinds of object in R.

You can get a description of the structure of any object using the function `str()`. For example, `str(v)` shows that `v` is numeric with 4 components.

2.4 Sequences

It is not always necessary to type out all the components of a vector. For example, the vector (15, 20, 25, ... ,85) can be created with

```
> seq(15, 85, by=5)
```

and the vector (5, 20, 25, ... ,85) can be created with

```
> c(5,seq(20, 85, by=5))
```

You can learn more about functions by typing ? followed by the function name. For example `?seq` gives information about the syntax and usage of the function `seq()`.

Exercise 2.2.

1. Create a vector `w` with components 1, -1, 2, -2
2. Print this vector (to the screen)
3. Obtain a description of `w` using `str()`
4. Create the vector `w+1`, and print it.
5. Create the vector (0, 1, 5, 10, 15, ... , 75) using c() and seq().

2.5 The births data

Table 2.1: *Variables in the births dataset*

Variable	Units or Coding	Type	Name
Subject number	–	categorical	`id`
Birth weight	grams	metric	`bweight`
Birth weight < 2500 g	1=yes, 0=no	categorical	`lowbw`
Gestational age	weeks	metric	`gestwks`
Gestational age < 37 weeks	1=yes, 0=no	categorical	`preterm`
Maternal age	years	metric	`matage`
Maternal hypertension	1=hypertensive, 0=normal	categorical	`hyp`
Sex of baby	1=male, 2=female	categorical	`sex`

The most important example of a vector in epidemiology is the data on a variable recorded for a group of subjects. To introduce R we use the births data which concern 500 mothers who had singleton births in a large London hospital. These data are available as an R object called `births` in the Epi package. You can get them into your workspace by:

```
> library( Epi )
> data( births )
```

Try

```
> objects()
```

to make sure that you have an object called births in your working directory. A more detailed overview of the objects in your workspace is obtained by:

```
> lls()
```

The function

```
> str(births)
```

shows that the object `births` is a data frame with 500 observations of 8 variables. The names and types of the variables are also shown together with the first 10 values of each variable.

Some of the variables which make up these data take integer values while others are numeric taking measurements as values. For most variables the integer values are just codes for different categories, such as `"male"` and `"female"` which are coded 1 and 2 for the variable `sex`.

Exercise 2.3.

1. The dataframe `"diet"` in the Epi package contains data from a follow-up study with coronary heart disease as the end-point. Load these data with:

   ```
   > data(diet)
   ```

   and print the contents of the data frame to the screen.
2. Check that you now have two objects, `births`, and `diet` in your work space.
3. Obtain a description of the object `diet`.
4. Remove the object `diet` with the command

   ```
   > rm(diet)
   ```

5. Check that you only have the object `births` left.

2.6 Referencing parts of the data frame

Typing `births` will list the entire data frame - not usually very helpful. Now try

```
> births[1,"bweight"]
```

This will list the value taken by the first subject for the `bweight` variable. Similarly

```
> births[2,"bweight"]
```

will list the value taken by the second subject for `bweight`, and so on. To list the data for the first 10 subject for the `bweight` variable, try

```
> births[1:10, "bweight"]
```

and to list all the data for this variable, try

```
> births[, "bweight"]
```

Exercise 2.4.

1. Print the data on the variable `gestwks` for subject 7 in the `births` data frame.
2. Print all the data for subject 7.
3. Print all the data on the variable `gestwks`.

2.7 Summaries

A good way to start an analysis is to ask for a summary of the data by typing

```
> summary(births)
```

To see the names of the variables in the data frame try

```
> names(births)
```

Variables in a data frame can be referred to by name, but to do so it is necessary also to specify the name of the data frame. Thus `births$hyp` refers to the variable `hyp` in the `births` data frame, and typing `births$hyp` will print the data on this variable. To summarize the variable `hyp` try

```
> summary(births$hyp)
```

In most datasets there will be some missing values. These are usually coded using tab delimited blanks to mark the values which are missing. R then codes the missing values using the `NA` (not available) symbol. The summary shows the number of missing values for each variable.

2.8 Turning a variable into a factor

In R categorical variables are known as *factors*, and the different categories are called the levels of the factor. Variables such as `hyp` and `sex` are originally coded using integer codes, and by default R will interpret these codes as numeric values taken by the variables. For R to recognize that the codes refer to categories it is necessary to convert the variables to be factors, and to label the levels. To convert the variable `hyp` to be a factor, try

```
> hyp <- factor(births$hyp)
> str(births)
> objects()
```

which shows that `hyp` is both in your work space (as a factor), and in in the `births` data frame (as a numeric variable). It is better to use the transform function on the data frame, as in

```
> births <- transform(births, hyp=factor(hyp))
> str(births)
```

which shows that `hyp`, in the births data frame, is now a factor with two levels, labeled `"0"` and `"1"` which are the original values taken by the variable. It is possible to change the labels to (say) `"normal"` and `"hyper"` with

```
> births <- transform( births, hyp=factor(hyp,labels=c("normal","hyper")) )
> str(births)
```

Exercise 2.5.

1. Convert the variable `sex` into a factor
2. Label the levels of `sex` as `"male"` and `"female"`.

2.9 Frequency tables

When starting to look at any new data frame the first step is to check that the values of the variables make sense and correspond to the codes defined in the coding schedule. For categorical variables (factors) this can be done by looking at one-way frequency tables and checking that only the specified codes (levels) occur. The most useful function for making tables is `stat.table`. This is currently part of the `Epi` package, so you will need to load this package first with

```
> library(Epi)
```

The distribution of the factors `hyp` and `sex` can be viewed by typing

```
> stat.table(hyp,data=births)
> stat.table(sex,data=births)
```

Their cross-tabulation is obtained by typing

```
> stat.table(list(hyp,sex),data=births)
```

Cross-tabulations are useful when checking for consistency, but because no distinction is drawn between the response variable and any explanatory variables, they are not useful as a way of presenting data.

2.10 Grouping the values of a metric variable

For a numeric variable like `matage` it is often useful to group the values and to create a new factor which codes the groups. For example we might cut the values taken by `matage` into the groups 20–29, 30–34, 35–39, 40–44, and then create a factor called `agegrp` with 4 levels corresponding to the four groups. The best way of doing this is with the function `cut`. Try

```
> births <- transform(births,agegrp=cut(matage, breaks=c(20,30,35,40,45),right=FALSE))
> stat.table(agegrp,data=births)
```

By default the factor levels are labeled [20-25), [25-30), etc., where [20-25) refers to the interval which includes the left hand end (20) but not the right hand end (25). This is the reason for `right=FALSE`. When `right=TRUE` (which is the default) the intervals include the right hand end but not the left hand.

It is important to realize that observations which are not inside the range specified in the `breaks()` part of the command result in missing values for the new factor. For example, try

```
> births <- transform(births,agegrp=cut(matage, breaks=c(20,30,35),right=FALSE))
> summary(births)
```

Only observations from 20 up to, but not including 35, are included. For the rest, `agegrp` is coded missing. You can specify that you want to cut a variable into a given number of intervals of equal length by specifying the number of intervals. For example

```
> births <- transform(births,agegrp=cut(matage,breaks=5,right=FALSE))
> stat.table(agegrp,data=births)
```

shows 5 intervals of width 4.

Exercise 2.6.

1. Summarize the numeric variable `gestwks`, which records the length of gestation for the baby, and make a note of the range of values.
2. Create a new factor `gest4` which cuts `gestwks` at 20, 35, 37, 39, and 45 weeks, including the left hand end, but not the right hand. Make a table of the frequencies for the four levels of `gest4`.
3. Create a new factor `gest5` which cuts `gestwks` into 5 equal intervals, and make a table of frequencies.

2.11 Tables of means and other things

To obtain the mean of `bweight` by `sex`, try

```
> stat.table(sex, mean(bweight), data=births)
```

The headings of the table can be improved with

```
> stat.table(sex,list("Mean birth weight"=mean(bweight)),data=births)
```

To make a two-way table of mean birth weight by sex and hypertension, try

```
> stat.table(list(sex,hyp),mean(bweight),data=births)
```

and to tabulate the count as well as the mean, try

```
> stat.table(list(sex,hyp),list(count(),mean(bweight)),data=births)
```

Available functions for the cells of the table are `count, mean, weighted.mean, sum, min, max, quantile,median, IQR`, and `ratio`. The last of these is useful for rates and odds. For example, to make a table of the odds of low birth weight by hypertension, try

```
> stat.table(hyp, list("odds"=ratio(lowbw,1-lowbw,100)),data=births)
```

The scale factor 100 makes the odds per 100. Margins can be added to the tables, as required. For example,

```
> stat.table(sex, mean(bweight),data=births,margins=TRUE)
```

for a one-way table, and

```
> stat.table(list(sex,hyp),mean(bweight),data=births,margins=c(TRUE,FALSE))
> stat.table(list(sex,hyp), mean(bweight),data=births,margins=c(FALSE,TRUE))
> stat.table(list(sex,hyp), mean(bweight),data=births,margins=c(TRUE,TRUE))
```

for a two-way table.

Exercise 2.7.

1. Make a table of median birth weight by `sex`.
2. Do the same for gestation time, but include `count` as a function to be tabulated along with `median`. Note that when there are missing values for the variable being summarized the count refers to the number of non-missing observations for the row variable, not the summarized variable.
3. Create a table showing the mean gestation time for the baby by `hyp` and `lowbw`, together with margins for both.
4. Make a table showing the odds of hypertension by sex of the baby.

2.11.1 Other tabulation functions

You may want to take a look at the help pages for the functions:

- `table`
- `ftable`
- `xtabs`
- `addmargins`
- `array`
- `tapply`

One way to do this is to simply type:

```
> example( table )
```

2.12 Generating new variables

New variables can be produced using assignment together with the usual mathematical operations and functions:

`+` `-` `*` `log` `exp` `^` `sqrt`

The sign `^` means "to the power of", `log` means "natural logarithm", and `sqrt` means "square root".

The `transform()` function allows you to transform or generate variables in a data frame. For example, try

```
> births <- transform(births,
+           num1=1,
+           num2=2,
+           logbw=log(bweight))
```

The variable `logbw` is the natural logarithm of birth weight. Logs base 10 are obtained with `log10( )`.

2.13 Logical variables

Logical variables take the values TRUE or FALSE, and behave like factors. New variables can be created which are logical functions of existing variables. For example

```
> births <- transform(births, low=bweight<2000)
> str(births)
```

creates a logical variable `low` with levels TRUE and FALSE, according to whether `bweight` is less than 2000 or not. The logical expressions which R allows are

```
==      <      <=      >      >=      !=
```

The first is logical equals and the last is not equals. One common use of logical variables is to restrict a command to a subset of the data. For example, to list the values taken by `bweight` for hypertensive women, try

```
> births$bweight[births$hyp=="hyper"]
```

If you want the entire dataframe restricted to hypertensive women try:

```
> births[births$hyp=="hyper",]
```

The `subset()` function also allows you to take a subset of a data frame. Try

```
> subset(births, hyp=="hyper")
```

Exercise 2.8.

1. Create a logical variable called `early` according to whether `gestwks` is less than 30 or not.Make a frequency table of `early`.
2. Print the `id` numbers of women with `gestwks` less than 30 weeks.

Chapter 3

Working with R

3.1 Saving the work space

When exiting from R you are offered the chance of saving all the objects in your current work space. If you do so, the work space is re-instated next time you start R. It can be useful to do this, but before doing so it is worth tidying things up, because the work space can fill up with temporary objects, and it is easy to forget what these are when you resume the session.

3.2 Saving output in a file

To save the output from an R command in a file, for future use, the `sink()` command is used. For example,

```
> sink("output.txt")
> summary(births)
```

first instructs R to re-direct output away from the R terminal to the file `"output.txt"` and then summarizes the births data frame, the output from which goes to the sink. While a sink is open all output will go to it, replacing what is already in the file. To append output to a file, use the `append=TRUE` option with `sink()`. To close a sink, use

```
> sink()
```

Exercise 3.9.

1. Sink output to a file called `"output1.txt"`.
2. Make frequency tables of `hyp` and `sex`
3. Make a table of mean birth weight by sex
4. Close the sink
5. From windows, have a look inside the file `output1.txt` and check that the output you expected is in the file.

3.3 Saving R objects in a file

The command `read.table()` is relatively slow because it carries out quite a lot of processing as it reads the data. To avoid doing this more than once you can save the data frame, which includes the R information, and read from this saved file in future. For example,

```
> save(births, file="births.Rdata")
```

will save the births data frame in the file `births.Rdata`. By default the data frame is saved as a binary file, but the option `ascii=TRUE` can be used to save it as a text file. To load the object from the file use

```
> load("births.Rdata")
```

The commands `save()` and `load()` can be used with any R objects, but they are particularly useful when dealing with large data frames.

Exercise 3.10.

1. Use `read.table()` to read the data in the file `diet.txt` into a data frame called `diet`.
2. Save this data frame in the file `"diet.Rdata"`
3. Remove the data frame
4. Load the data frame from the file `"diet.Rdata"`.

3.4 Using a text editor with R

When working with R it is best to use a text editor to prepare a batch file (or script) which contains R commands and then to run them from the script. This means you can use the cut and paste facilities of the editor to cut down on typing. For Windows we recommend using the text editor Tinn-R, but you can use your favorite text editor instead if you prefer, and copy-paste commands from it into the R-console.

Alternatively you can use the built-in script-editor: Click on `File`→`New script`, or `File`→`Open script`, according to whether you are using an old script. You can move the current line from the script-editor to the console by `CTRL-R`. If you have highlighted a section of the script the highlighted part will be moved to the console.

Now start up the editor and enter the following lines:

```
> births <- transform( births,
+                      lowbw = factor(lowbw, labels=c("normal","low")),
+                        hyp = factor(hyp, labels=c("normal","hyper")),
+                        sex = factor(sex, labels=c("male","female")) )
```

Now save the script as `mygetbirths.R` and run it. One major advantage of running all your R commands from a script is that you end up with a record of exactly what you did which can be repeated at any time.

This will also help you redo the analysis in the (highly likely) event that your data changes before you have finished all analyses.

Exercise 3.11.

1. Create a script called mytab.R which includes the lines

   ```
   > stat.table(hyp,data=births)
   > stat.table(sex,data=births)
   ```

   and run just these two lines.
2. Edit the script to include the lines

   ```
   > stat.table(sex,mean(bweight),data=births)
   > stat.table(hyp,mean(bweight),data=births)
   ```

   and run these two lines.
3. Edit the script to create a factor cutting `matage` at 20, 30, 35, 40, 45 years, and run just this part of the script.
4. Edit the script to create a factor cutting `gestwks` at 20, 35, 37, 39, 45 weeks, and run just this part of the script.
5. Save and run the entire script.

3.5 The search path

R organizes objects in different positions on a search path. The command

```
> search()
```

shows these positions. The first is the work space, or global environment, the second is the Epi package, the third is a package of commands called methods, the fourth is a package called stats, and so on. To see what is in the work space try

```
> objects()
```

You should see just the objects `births` and `diet`. The command `objects(1)` does the same as `objects()`. A shorther name for the same function is `ls()`. In the Epi package is a function that gives a more detailed picture, `lls()`; try:

```
> lls()
```

To see what is in the Epi package, try

```
> ls(2)
```

When you type the name of an object R looks for it in the order of the search path and will return the first object with this name that it finds. This is why it is best to start your session with a clean workspace, otherwise you might have an object in your workspace that masks another one later in the search path.

3.6 Attaching a data frame

The function `objects(1)` shows that the only objects in the workspace are `births` and `diet`. To refer to variables in the `births` data frame by name it is necessary to specify the name of the data frame, as in `births$hyp`. This is quite cumbersome, and provided you are working primarily with one data frame, it can help to put a copy of the variables from a data frame in their own position on the search path. This is done with the function

```
> attach(births)
```

which places a copy of the variables in the `births` data frame in position 2. You can verify this with

```
> objects(2)
```

which shows the objects in this position are the variables from the `births` data frame. Note that the methods package has now been moved up to position 3, as shown by the `search()` function.

When you type the command:

```
> hyp
```

R will look in the first position where it fails to find `hyp`, then the second position where it finds `hyp`, which now gets printed.

Although convenient, attaching a data frame can give rise to confusion. For example, when you create a new object from the variables in an attached data frame, as in

```
> subgrp <- bweight[hyp==1]
```

the object `subgrp` will be in your workspace (position 1 on the search path) not in position 2. To demonstrate this, try

```
> objects(1)
> objects(2)
```

Similarly, if you modify the data frame in the workspace the changes will not carry through to the attached version of the data frame. The best advice is to regard any operation on an attached data frame as temporary, intended only to produce output such as summaries and tabulations.

Beware of attaching a data frame more than once - the second attached copy will be attached in position 2 of the search path, while the first copy will be moved up to position 3. You can see this with

```
> attach(births)
> search()
```

Having several copies of the same data set can lead to great confusion. To detach a data frame, use the command

```
> detach(births)
```

which will detach the copy in position 2 and move everything else down one position. To detach the second copy repeat the command `detach(births)`.

Exercise 3.12.

1. Use `search()` to make sure you have no data frames attached.
2. Use `objects(1)` to check that you have the data frame `births` in your work space.
3. Verify that typing `births$hyp` will print the data on the variable `hyp` but typing `hyp` will not.
4. Attach the `births` data frame in position 2 and check that the variables from this data frame are now in position 2.
5. Verify that typing `hyp` will now print the data on the the variable `hyp`.
6. Summarize the variable `bweight` for hypertensive women.

```
> setwd(sweave.wd)
```

Chapter 4

Graphs in R

There are three kinds of plotting functions in R:

1. Functions that generate a new plot, e.g. `hist()` and `plot()`.
2. Functions that add extra things to an existing plot, e.g. `lines()` and `text()`.
3. Functions that allow you to interact with the plot, e.g. `locator()` and `identify()`.

The normal procedure for making a graph in R is to make a fairly simple initial plot and then add on points, lines, text etc., preferably in a script.

4.1 Simple plot on the screen

Load the births data and get an overview of the variables:

```
> library(Epi)
> data(births)
> str(births)
```

Now attach the dataframe and look at the birthweight distribution with

```
> attach(births)
> hist(bweight)
```

The histogram can be refined – take a look at the possible options with

```
> ?hist
```

and try some of the options, for example:

```
> hist(bweight, col="gray", border="white")
```

To look at the relationship between birthweight and gestational weeks, try

```
> plot(gestwks, bweight)
```

You can change the plot-symbol by the option `pch=`. If you want to see all the plot symbols try:

```
> plot(1:25, pch=1:25)
```

Exercise 4.13.

1. Make a plot of the birth weight versus maternal age with

```
> plot(matage, bweight)
```

2. Label the axes with

```
> plot(matage, bweight, xlab="Maternal age", ylab="Birth weight (g)")
```

4.2 Colours

There are many colours recognized by R. You can list them all by `colours()` or, equivalently, `colors()` (R allows you to use British or American spelling). To colour the points of birthweight versus gestational weeks, try

```
> plot(gestwks, bweight, pch=16, col="green")
```

This creates a solid mass of colour in the center of the cluster of points and it is no longer possible to see individual points. You can recover this information by overwriting the points with black circles using the `points()` function.

```
> points(gestwks, bweight)
```

4.3 Adding to a plot

The `points()` function is one of several functions that add elements to an existing plot. By using these functions, you can create quite complex graphs in small steps.

Suppose we wish to recreate the plot of birthweight *vs* gestational weeks using different colours for male and female babies. To start with an empty plot, try

```
> plot(gestwks, bweight, type="n")
```

Then add the points with the `points` function.

```
> points(gestwks[sex==1], bweight[sex==1], col="blue")
> points(gestwks[sex==2], bweight[sex==2], col="red")
```

To add a legend explaining the colours, try

```
> legend("topleft", pch=1, legend=c("Boys","Girls"), col=c("blue","red"))
```

which puts the legend in the top left hand corner.

Finally we can add a title to the plot with

```
> title("Birth weight vs gestational weeks in 500 singleton births")
```

4.3.1 Using indexing for plot elements

One of the most powerful features of R is the possibility to index vectors, not only to get subsets of them, but also for repeating their elements in complex sequences.

Putting separate colours on males and female as above would become very clumsy if we had a 5 level factor instead.

Instead of specifying one color for all points, we may specify a vector of colours of the same length as the `gestwks` and `bweight` vectors. This is rather tedious to do directly, but R allows you to specify an expression anywhere, so we can use the fact that `sex` takes the values 1 and 2, as follows:

First create a colour vector with two colours, and take look at `sex`:

```
> c("blue","red")
> sex
```

Now see what happens if you index the colour vector by sex:

```
> c("blue","red")[sex]
```

For every occurrence of a `1` in `sex` you get `"blue"`, and for every occurrence of `2` you get `"red"`, so the result is a long vector of `"blue"`s and `"red"`s corresponding to the males and females. This can now be used in the plot:

```
> plot( gestwks, bweight, pch=16, col=c("blue","red")[sex] )
```

The same trick can be used if we want to have a separate symbol for mothers over 40 say. We first generate the indexing variable:

```
> oldmum <- ( matage >= 40 ) + 1
```

Note we add 1 because `( matage >= 40 )` generates a logic variable, so by adding 1 we get a numeric variable with values 1 and 2, suitable for indexing:

```
> plot( gestwks, bweight, pch=c(16,3)[oldmum], col=c("blue","red")[sex] )
```

so where `oldmum` is 1 we get `pch=16` (a dot) and where `oldmum` is 2 we get `pch=3` (a cross).

R will accept any kind of complexity in the indexing as long as the result is a valid index, so you don't need to create the variable `oldmum`, you can create it on the fly:

```
> plot( gestwks, bweight, pch=c(16,3)[(matage>=40 )+1], col=c("blue","red")[sex] )
```

Exercise 4.14.

1. Make a three level factor for maternal age with cutpoints at 30 and 40 years.
2. Use this to make the plot of gestational weeks with three different plotting symbols. (Hint: Indexing with a factor automatically gives indexes 1,2,3 etc.).

4.3.2 Generating colours

R has functions that generate a vector of colours for you. For example,

```
> rainbow(4)
```

produces a vector with 4 colours (not immediately human readable, though). There are a few other functions that generates other sequences of colours, type `?rainbow` to see them.

Gray-tones are produced by the function `gray` (or `grey`), which takes a numerical argument between 0 and 1; `gray(0)` is black and `gray(1)` is white. Try:

```
> plot( 0:10, pch=16, cex=3, col=gray(0:10/10) )
> points( 0:10, pch=1, cex=3 )
```

4.4 Interacting with a plot

The `locator()` function allows you to interact with the plot using the mouse. Typing `locator(1)` shifts you to the graphics window and waits for one click of the left mouse button. When you click, it will return the corresponding coordinates.

You can use `locator()` inside other graphics functions to position graphical elements exactly where you want them. Recreate the birth-weight plot,

```
> plot( gestwks, bweight, pch=c(16,3)[(matage>=40 )+1], col=c("blue","red")[sex] )
```

and then add the legend where you wish it to appear by typing

```
> legend(locator(1), pch=1, legend=c("Boys","Girls"), col=c("blue","red") )
```

The `identify()` function allows you to find out which records in the data correspond to points on the graph. Try

```
> identify( gestwks, bweight )
```

When you click the left mouse button, a label will appear on the graph identifying the row number of the nearest point in the data frame `births`. If there is no point nearby, R will print a warning message on the console instead. To end the interaction with the graphics window, right click the mouse: the `identify` function returns a vector of identified points.

Exercise 4.15.

1. Use `identify()` to find which records correspond to the smallest and largest number of gestational weeks.
2. View all the variables corresponding to these records with:

   ```
   > births[identify(gestwks,bweight), ]
   ```

4.5 Saving your graphs for use in other documents

Once you have a graph on the screen you can click on File → Save as, and choose the format you want your graph in. The `PDF` (Acrobat reader) format is normally the most economical, and Acrobat reader has good options for viewing in more detail on the screen. The `Metafile` format will give you an enhanced metafile `.emf`, which can be imported into a Word document by Insert → Picture → From File. Metafiles can be resized and edited inside Word.

If you want exact control of the size of your plot you can start a graphics device *before* doing the plot. Instead of appearing on the screen, the plot will be written directly to a file. After the plot has been completed you will need to close the device again in order to be able to access the file. Try:

```
> win.metafile(file="plot1.emf", height=3, width=4)
> plot(gestwks, bweight)
> dev.off()
```

This will give you a enhanced metafile `plot1.emf` with a graph which is 3 inches tall and 4 inches wide.

4.6 The `par()` command

It is possible to manipulate any element in a graph, by using the graphics options. These are collected on the help page of `par()`. For example, if you want axis labels always to be horizontal, use the command `par(las=1)`. This will be in effect until a new graphics device is opened.

Look at the typewriter-version of the help-page with

```
> ?par
```

or better, use the the html-version through Help → Html help → Packages → base → P → par.

It is a good idea to take a print of this (having set the text size to "smallest" because it is long) and carry it with you at any time to read in buses, cinema queues, during boring lectures etc. Don't despair, few R-users can understand what all the options are for.

`par()` can also be used to ask about the current plot, for example `par("usr")` will give you the exact extent of the axes in the current plot.

If you want more plots on a single page you can use the command

```
>  par( mfrow=c(2,3) )
```

This will give you a layout of 2 rows by 3 columns for the next 6 graphs you produce. The plots will appear by row, i.e. in the top row first. If you want the plots to appear column-wise, use `par( mfcol=c(2,3) )` (you still get 2 rows by 3 columns). To restore the layout to a single plot per page use

```
>  par( mfrow=c(1,1) )
```

Finally for more complex graphical lay-outs you can use the functions `layout()`, take a look:

```
> ?layout
```

4.7 Population pyramid

First, load the `Epi` package and then the Danish population data, `N.dk`

```
> library( Epi )
> data( N.dk )
> head( N.dk )
```

For the sake of simplicity we turn the data into a 3-way array, classified by sex, age and calendar year:

```
> pp <- xtabs( N ~ sex + A + P, data=N.dk )
> str( pp )
```

A simple histogram (well, box-plot) of the male population in 1980:

```
> barplot( pp["1",,"1980"] )
> barplot( pp["1",,"1980"], horiz=TRUE, col="blue", border="blue" )
```

Now you want the females back to back with this. We plot makes and females stacked, by entering a 2×100 matrix instead of a vector:

```
> dim( pp[,,"1980"] )
> barplot( pp[,,"1980"], horiz=TRUE, col=c("blue","red"), border="transparent" )
```

That is not quite what we want, but there is a hidden (*i.e.* undocumented) feature in `barplot`; namely that if you give a first row of negative numbers, then this will determine the starting point of the bars, so we should just add a row of the male population with a minus:

```
> barplot( rbind(-pp["1",,"1980"], pp[,,"1980"] ),
+          horiz=TRUE, col=c("red","blue"), border="transparent", space=0 )
```

Finally we want to see all the pyramids, on the same scale so we fix the extent of the x-axis:

```
> barplot( rbind(-pp["1",,"1980"], pp[,,"1980"] )/1000,
+          xlim=c(-1,1)*50, xlab="Persons (1000s)", yaxt="n",  xaxt="n",
+          horiz=TRUE, col=c("red","blue"), border="transparent", space=0 )
> axis( side=2, at=0:10*10, las=1 )
> axis( side=1, at=c(-5:0,1:5)*10, labels=c(5:0,1:5)*10 )
> text( -48, 100, "1980", font=2, adj=0 )
```

If we want to see it for all years, we print to a pdf-file, and make a loop over the years inside the plot. Note the differences here:

```
> dimnames(pp)[3]
> dimnames(pp)[[3]]
```

The first is a list of length 1, namely a subset of the list `dimnames(pp)`, the second is the *content* of the 3rd list element if `dimnames(pp)`. It is the latter we use in the loop:

```
> pdf( "./graph/DKpyr.pdf", height=8, width=8 )
> for( yy in dimnames(pp)[[3]] )
+    {
+ barplot( rbind(-pp["1",,yy], pp[,,yy] )/1000,
+          xlim=c(-1,1)*50, xlab="Persons (1000s)", yaxt="n",  xaxt="n",
+          horiz=TRUE, col=c("red","blue"), border="transparent", space=0 )
+ axis( side=2, at=0:10*10, las=1 )
+ axis( side=1, at=c(-5:0,1:5)*10, labels=c(5:0,1:5)*10 )
+ text( -48, 100, yy, font=2, adj=0 )
+    }
> dev.off()
```

4.7.0.0.1 Exercise: Add an axis on the right showing the year of birth. Remember to restrict it properly — you may need the functions `pmin` and `pmax`.

Chapter 5

The `effx` function for effects estimation

Identifying the response variable correctly is the key to analysis. The main types are:

- Metric (a measurement taking many values, usually with units)
- Binary (two values coded 0/1)
- Failure (does the subject fail at end of follow-up, and how long was follow-up)
- Count (aggregated failure data)

The response variable must be numeric.

Variables on which the response may depend are called explanatory variables. They can be factors or numeric. A further important aspect of explanatory variables is the role they will play in the analysis.

- Primary role: exposure
- Secondary role: confounder

The word effect is a general term referring to ways of comparing the values of the response variable at different levels of an explanatory variable. The main measures of effect are:

- Differences in means for a metric response.
- Ratios of odds for a binary response.
- Ratios of rates for a failure or count response.

What other measures of effects might be used?

5.1 The function effx

The function `effx` is intended to introduce the estimation of effects in epidemiology, together with the related ideas of stratification and controlling, without the need for familiarity with statistical modelling.

We shall use the births data in the Epi package, which can be loaded and inspected with

```
> library(Epi)
> data(births)
> help(births)
```

The variables we shall be interested in are `bweight` (birth weight) and `hyp` (hypertension). An alternative way of characterizing birth weight is shown in `lowbw` which is coded 1 for babies with low birth weight, and 0 otherwise. Other variables of interest are `sex` (of the baby) and `gestwks`, the gestation time.

All variables are numeric, so first we need first to do a little housekeeping:

```
> births$hyp <- factor(births$hyp,labels=c("normal","hyper"))
> births$sex <- factor(births$sex,labels=c("M","F"))
> births$agegrp <- cut(births$matage,breaks=c(20,25,30,35,40,45),right=FALSE)
> births$gest4 <- cut(births$gestwks,breaks=c(20,35,37,39,45),right=FALSE)
```

Now try

```
> effx(response=bweight,typ="metric",exposure=sex,data=births)
```

The effect of sex on birth weight, measured as a difference in means, is -197. The command

```
> stat.table(sex,mean(bweight), data=births)
```

verifies this ($3032.8 - 3229.9 = -197.1$). The p-value refers to the test that there is no effect of `sex` on birth weight. Use `effx` to find the effect of `hyp` on `bweight`.

For another example, consider the effect of `sex` on the binary response `lowbw`.

```
> effx(response=lowbw,typ="binary",exposure=sex,data=births)
```

The effect of `sex` on `lowbw`, measured as an odds ratio, is 1.43. The command

```
> stat.table(sex,list(odds=ratio(lowbw,1-lowbw,100)),data=births)
```

can be used to verify this ($16.26/11.39 = 1.427$). Use `effx` to find the effect of `hyp` on `lowbw`.

5.2 Factors on more than two levels

The variable `gest4` is the result of cutting `gestwks` into 4 groups with boundaries [20,35) [35,37) [37,39) [39,45). We shall find the effects of `gest4` on the metric response `bweight`.

```
> effx(response=bweight,typ="metric",exposure=gest4,data=births)
```

There are now 3 effects

```
[35,37) vs [20,35)  856.6
[37,39) vs [20,35) 1360.0
[39,45) vs [20,35) 1668.0
```

The command

```
> stat.table(gest4,mean(bweight),data=births)
```

verifies that the effect of `agegrp` (level 2 vs level 1) is $2590 - 1733 = 857$, etc. Find the effects of `gest4` on `lowbw`. Use the option `base=4` to change the baseline for `gest4` from 1 to 4.

5.3 Stratified effects

As an example we shall stratify the effects of `hyp` on `bweight` by `sex` with

```
> effx(bweight, type="metric", exposure=hyp, strata=sex,data=births)
```

The effects of `hyp` in the different strata defined by `sex` are −496 and −380.

Use `effx` to stratify the effect of `hyp` on `lowbw` first by `sex` and then by `gest4`.

5.4 Controlling the effect of hyp for sex

The effect of `hyp` is controlled for `sex` by first looking at the effects of `hyp` in the two strata defined by `sex`, and then combining these effects if they are similar. In this case the effcts were −496 and −380 which look similar (the test for effect modification is a test of whether they differ significantly) so we can combine them, and control for `sex`.

The combining is done by declaring `sex` as a control variable:

```
> effx(bweight, type="metric", exposure=hyp, control=sex,data=births)
```

The effect of `hyp` on `bweight` controlled for `sex` is −448. Note that it is the name of the control variable which is passed, not the variable itself. There can be more than one control variable, `control=list(sex,agegrp)`.

Many people go straight ahead and control for variables which are likely to confound the effect of exposure without bothering to stratify first, but there are times when it is useful to stratify first.

5.5 Numeric exposures

If we wished to study the effect of gestation time on the baby's birth weight then `gestwks` is a numeric exposure. Assuming that the relationship of the response with `gestwks` is roughly linear (for a metric response) or log-linear (for a binary response) we can find the linear effect of `gestwks`.

```
> effx(response=bweight, type="metric", exposure=gestwks,data=births)
```

The linear effect of `gestwks` is 197 g per extra week of gestation. The linear effect of `gestwks` on `lowbw` can be found similarly

```
> effx(response=lowbw, type="binary", exposure=gestwks,data=births)
```

The linear effect of `gestwks` on `lowbw` is a reduction by a factor of 0.408 per extra week of gestation, i.e. the odds of a baby having a low birth weight is reduced by a factor of 0.408 per one week increase in gestation.

You cannot stratify by a numeric variable, but you can study the effects of a numeric exposure stratified by (say) `agegrp` with

```
> effx(lowbw, type="binary",exposure=gestwks,strata=agegrp,data=births)
```

You can control for a numeric variable by putting it in `control=`.

5.6 Checking on linearity

At this stage it will be best to make a visual check using `plot`. For example, to check whether `bweight` goes up linearly with `gestwks` try

```
> with(births, plot(gestwks,bweight))
```

Is the relationship roughly linear? It is not possible to check graphically whether log odds of a baby being low birth weight goes down linearly with gestation because the individual odds are either 0 or ∞. Instead we use the grouped variable `gest4`:

```
> tab<-stat.table(gest4,ratio(lowbw,1-lowbw,100),data=births)
> str(tab)
> #Extract the odds from tab, and plot the logodds against 1:4
> odds<-tab[1,1:4]
> plot(1:4,log(odds),type="b")
```

The relationship is remarkably linear, but remember this is quite crude because it takes no account of unequal gestation intervals. More about checking for linearity later.

5.7 Frequency data

Data from very large studies are often summarized in the form of frequency data, which records the frequency of all possible combinations of values of the variables in the study. Such data are sometimes presented in the form of a contingency table, sometimes as a data frame in which one variable is the frequency. As an example, consider the `UCBAdmissions` data, which is one of the standard R data sets, and refers to the outcome of applications to 6 departments by gender. The command

```
> UCBAdmissions
```

shows that the data are in the form of a $2 \times 2 \times 6$ contingency table for the three variables `Admit` (admitted/rejected), `Gender` (male/female), and `Dept` (A/B/C/D/E/F). Thus in department A 512 males were admitted while 312 were rejected, and so on. The question of interest is whether there is any bias against admitting female applicants.

The command

```
> ucb <- as.data.frame(UCBAdmissions)
> head(ucb)
```

coerces the contingency table to a data frame, and shows the first 10 lines. The relationship between the contingency table and the data frame should be clear. The command

```
> ucb$Admit <- as.numeric(ucb$Admit)-1
```

turns `Admit` into a numeric variable coded 1 for rejection, 0 for admission, so

```
> effx(Admit,type="binary",exposure=Gender,weights=Freq,data=ucb)
```

shows the odds of rejection for female applicants to be 1.84 times the odds for males (note the use of `weights` to take account of the frequencies). A crude analysis therefore suggests there is a strong bias against admitting females. Continue the analysis by stratifying the crude analysis by department - does this still support a bias against females? What is the effect of gender controlled for department?

Chapter 6

Dates in R

Epidemiological studies often contain date variables which take values such as 2/11/1962. We shall use the diet data to illustrate how to deal with variables whose values are dates.

The important variables in the dataset are `chd`, which takes the value 1 if the subject develops coronary heart disease during the study the value 0 if the observation is censored, and the three date variables which are date of birth (`dob`), date of entry (`doe`) and date of exit (`dox`). The command

```
> str(diet)
```

shows that these three variables are `Date` variables.

You will also see that the values are just numbers, but if you try

```
> head( diet )
```

you will see these variables printed as "real" dates. The variables are internally stored as number of days since 1/1/1970.

To convert a character string (or a character variable) to date format try:

```
> as.Date( "14/07/1952", format="%d/%m/%Y" )
> as.numeric( as.Date( "14/07/1952", format="%d/%m/%Y" ) )
```

The first form shows the date form and the latter the number of days since 1/1/1970, which is a negative number for dates prior to 1/1/1970.

The format parts, "`%d`" etc., identify elements of the dates, whereas the "`/`"s are just the separator characters that are in the character string. There are other possibilities for formats, see `?strftime` or the section on dates and times in the R command sheet at the end of this document.

Reading dates from an external file is done by reading the fields as character variables and then transforming them to date variables by the function `as.Date`

If you want to enter a fixed date, for example if you want to terminate follow-up at 1st April 1975 you could say:

```
>   newx <- pmin( diet$dox, as.Date( "1975-4-1", format="%F" ) )
```

The format `%F` is shorthand for the ISO-standard date representation `%Y-%m-%d`, which is the default, so it can be omitted altogether:

```
>   newx <- pmin( diet$dox, as.Date("1975-4-1") )
```

You can print dates in the format you like by using the function `format.Date()`, try for example:

```
>   bdat <- as.Date( "1952-7-14", format="%F" )
>   format.Date( bdat, format="%A %d %B %Y" )
```

Exercise 6.16.

1. Convert `doe` and `dox` to date variables.
2. Generate a new variable `y` which is the elapsed time in years between the date of entry and the date of exit.
3. The file `getdiet.R` reads the diet data, converts all three date variables to standard form using the `transform` function, and generates the variable `y`. Run this script and check the results are what you want.
4. Enter your own birtday as a date. Print it using `format.Date()` with the format `"%A %d %B %Y"`. Did you learn anything new?
5. Enter the birthday of your husband/wife/... as a date too. When will you be (were you) 100 years old together? (Hint: `mean()` works on vectors of dates as well.)

In the Epi package is also a function `cal.yr` which converts dates to fractional years:

```
> as.Date( "1952-7-14" )
> cal.yr( as.Date("1952-7-14") )
> cal.yr( "1952-7-14" )
```

The function will also find all date-variabels in a dataframe and convert them; try:

```
> data( diet )
> str( diet )
> str( cal.yr(diet) )
```

Chapter 7

Epidemiological calculations

7.1 Calculation of rates, RR and RD

This exercise is `very` prescriptive, so you should make an effort to really understand everything you type into R.

Recall that the standard error of log-rate is $1/\sqrt{D}$, so that a 95% confidence interval for the log of a rate is:

$$\hat{\theta} \pm 1.96/\sqrt{D} = \log(\lambda) \pm 1.96/\sqrt{D}$$

If we take the exponential, we get the confidence interval for the rate:

$$\lambda \overset{\times}{\div} \underbrace{\exp(1.96/\sqrt{D})}_{\text{error factor,erf}}$$

1. Now, suppose you have 15 events during 5532 person-years. Now use R as a simple desk calculator to derive the rate and a confidence interval:

```
> options(width=90,show.signif.stars=FALSE)
> library( Epi )
```

```
> D <- 15
> Y <- 5532
> rate <- D / Y
> erf <- exp( 1.96 / sqrt(D) )
> c( rate, rate/erf, rate*erf )
```

```
[1] 0.002711497 0.001634654 0.004497720
```

You can explore the function `ci.mat()`, which lets you use matrix multiplication to produce confidence interval from an estimate and a standard error (or columns of such):

```
> ci.mat()
```

```
     Estimate      2.5%    97.5%
[1,]        1  1.000000 1.000000
[2,]        0 -1.959964 1.959964
```

```
> exp( c( log(D/Y), 1/sqrt(D) ) %*% ci.mat() )
```

```
        Estimate       2.5%       97.5%
[1,] 0.002711497 0.001634669 0.004497678
```

2. Now try to achieve this estimate and c.i. using a Poisson model. Use the number of events as the response and the log-person-years as offset:

```
> mm <- glm( D ~ 1, offset=log(Y), family=poisson )
> summary( mm )

  Call:
  glm(formula = D ~ 1, family = poisson, offset = log(Y))

  Deviance Residuals:
  [1]  0

  Coefficients:
              Estimate Std. Error z value Pr(>|z|)
  (Intercept)  -5.9103     0.2582  -22.89   <2e-16

  (Dispersion parameter for poisson family taken to be 1)

      Null deviance: -8.8818e-16  on 0  degrees of freedom
  Residual deviance: -8.8818e-16  on 0  degrees of freedom
  AIC: 6.557

  Number of Fisher Scoring iterations: 3
```

   What is the interpretation of the parameter in this model?

3. You can extract a confidence interval directly from the model with the `ci.lin()` and `ci.exp`-functions from `Epi`:

```
> ci.lin( mm )

               Estimate    StdErr         z             P      2.5%     97.5%
  (Intercept) -5.910254 0.2581989 -22.89032 5.801722e-116 -6.416315 -5.404194

> ci.lin( mm, E=T)[,5:7]

    exp(Est.)        2.5%       97.5%
  0.002711497 0.001634669 0.004497678

> ci.exp( mm )

                exp(Est.)        2.5%       97.5%
  (Intercept) 0.002711497 0.001634669 0.004497678
```

4. There is an alternative way to fit a Poisson model, using the rates a the Poisson response, and the person-years as weights instead (albeit it will give you a warning about non-integer response in a Poisson model):

```
> mmx <- glm( D/Y ~ 1, weight=Y, family=poisson )
> ci.exp( mmx )

                exp(Est.)        2.5%       97.5%
  (Intercept) 0.002711497 0.001634669 0.004497678
```

   Verify that this give the same results as above.

5. The advantage of this latter approach is that it will also make sense to use an identity link — the response is the same but the parameter estimated is now the rate, not the log-rate:

```
> ma <- glm( D/Y ~ 1, weight=Y, family=poisson(link=identity) )
```

   What is the meaning of the intercept in this model?

   Verify that you actually get the same rate estimate as before.

6. Now use `ci.lin` to produce the estimate and the confidence intervals from this model:

```
> ci.lin( ma )
```

```
                Estimate       StdErr        z            P        2.5%       97.5%
(Intercept) 0.002711497 0.0007001054 3.872983 0.0001075112 0.001339315 0.004083678
```

```
> ci.lin( ma )[,c(1,5,6)]
```

```
   Estimate        2.5%       97.5%
0.002711497 0.001339315 0.004083678
```

```
> ci.exp( ma, Exp=FALSE )
```

```
                Estimate        2.5%       97.5%
(Intercept) 0.002711497 0.001339315 0.004083678
```

   Why are the confidence limits not the same as from the multiplicative model?

   Derive the formula for the standard error of this estimated rate.

7. Now, suppose the events and person years are collected over three periods:

```
> Dx <- c(3,7,5)
> Yx <- c(1412,2783,1337)
> Px <- 1:3
> cbind( Px, Dx, Yx )
```

```
     Px Dx   Yx
[1,]  1  3 1412
[2,]  2  7 2783
[3,]  3  5 1337
```

   Try to fit the same model as before to the data from the separate periods.

```
> m1 <- glm( Dx ~ 1, offset=log(Yx), family=poisson )
```

8. Now test whether the rates are the same in the three periods: Try to fit a model with the period as a factor in the model:

```
> mp <- glm( Dx ~ factor(Px), offset=log(Yx), family=poisson )
```

   and compare the two models using `anova` with the argument `test="Chisq"`:

```
> anova( m1, mp, test="Chisq" )
```

```
Analysis of Deviance Table

Model 1: Dx ~ 1
Model 2: Dx ~ factor(Px)
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1         2    0.70003
2         0    0.00000  2  0.70003   0.7047
```

Compare the test statistic to the deviance of the model `mp`.

What is the deviance good for?

9. If we have observations of two rates λ_1 and λ_0, based on (D_1, Y_1) and (D_0, Y_0) the variance of the difference of the log of the rates, that is the log(RR), is:

$$\begin{aligned} \text{var}(\log(\text{RR})) &= \text{var}(\log(\lambda_1/\lambda_0)) \\ &= \text{var}(\log(\lambda_1)) + \text{var}(\log(\lambda_0)) \\ &= 1/D_1 + 1/D_0 \end{aligned}$$

As before a 95% c.i. for the RR is then:

$$\text{RR} \overset{\times}{\div} \exp\left(1.96\sqrt{\frac{1}{D_1} + \frac{1}{D_0}}\right)$$

Suppose you have 15 events during 5532 person-years in an unexposed group and 28 events during 4783 person-years in an exposed group:

Compute the the rate-ratio and c.i. by:

```
> D0 <- 15   ; D1 <- 28
> Y0 <- 5532 ; Y1 <- 4783
> RR <- (D1/Y1)/(D0/Y0)
> erf <- exp( 1.96 * sqrt(1/D0+1/D1) )
> c( RR, RR/erf, RR*erf )

[1] 2.158980 1.153146 4.042153

> exp( c( log(RR), sqrt(1/D0+1/D1) ) %*% ci.mat() )

     Estimate    2.5%    97.5%
[1,]  2.15898 1.15316 4.042106
```

10. Now achieve this using a Poisson model:

```
> D <- c(D0,D1) ; Y <- c(Y0,Y1); xpos <- 0:1
> mm <- glm( D ~ factor(xpos), offset=log(Y), family=poisson )
```

What does the parameters mean in this model?

You can extract the exponentiated parameters by:

```
> ci.exp( mm )

                  exp(Est.)        2.5%       97.5%
(Intercept)    0.002711497 0.001634669 0.004497678
factor(xpos)1 2.158979720 1.153159560 4.042106222
```

11. If we instead want the rate-difference, we just subtract the rates, and the variance of the difference is (since the rates are based on independent samples) just the sum of the variances:

$$\begin{aligned} \text{var}(\text{RD}) &= \text{var}(\lambda_1) + \text{var}(\lambda_0) \\ &= D_1/Y_1^2 + D_0/Y_0^2 \end{aligned}$$

Use this formula to compute the rate difference and a 95% confidence interval for it. Note that we use the `%*%` for matrix multiplication:

```
> rd <- diff( D/Y )
> sd <- sqrt( sum( D/Y^2 ) )
> c( rd, sd ) %*% ci.mat()

        Estimate         2.5%       97.5%
[1,] 0.00314257 0.0005765288 0.005708611
```

12. Verify that this is the confidence interval you get when you fit an additive model with exposure as factor:

```
> ma <- glm( D/Y ~ factor(xpos), weight=Y,
+                  family=poisson(link=identity) )
> ci.lin( ma )[,c(1,5,6)]

                  Estimate         2.5%       97.5%
(Intercept)   0.002711497 0.0013393153 0.004083678
factor(xpos)1 0.003142570 0.0005765288 0.005708611
```

13. Normally one would like to get both the rates and the ratio between them. This can be achieved in one go using the `ctr.mat` argument to `ci.lin`. Try:

```
> CM <- rbind( c(1,0), c(1,1), c(0,1) )
> rownames( CM ) <- c("rate 0","rate 1","RR 1 vs. 0")
> CM

            [,1] [,2]
rate 0         1    0
rate 1         1    1
RR 1 vs. 0     0    1

> mm <- glm( D ~ factor(xpos),
+                offset=log(Y), family=poisson )
> ci.lin( mm, ctr.mat=CM, E=T)[,5:7]

              exp(Est.)        2.5%       97.5%
rate 0     0.002711497 0.001634669 0.004497678
rate 1     0.005854066 0.004041994 0.008478512
RR 1 vs. 0 2.158979720 1.153159560 4.042106222

> round( ci.lin( mm, ctr.mat=CM, E=T)[,5:7], 3 )

            exp(Est.)  2.5% 97.5%
rate 0          0.003 0.002 0.004
rate 1          0.006 0.004 0.008
RR 1 vs. 0      2.159 1.153 4.042
```

14. Refit the model with `Y/1000` as the person time, so you get the estimated rates in units of cases per 1000.

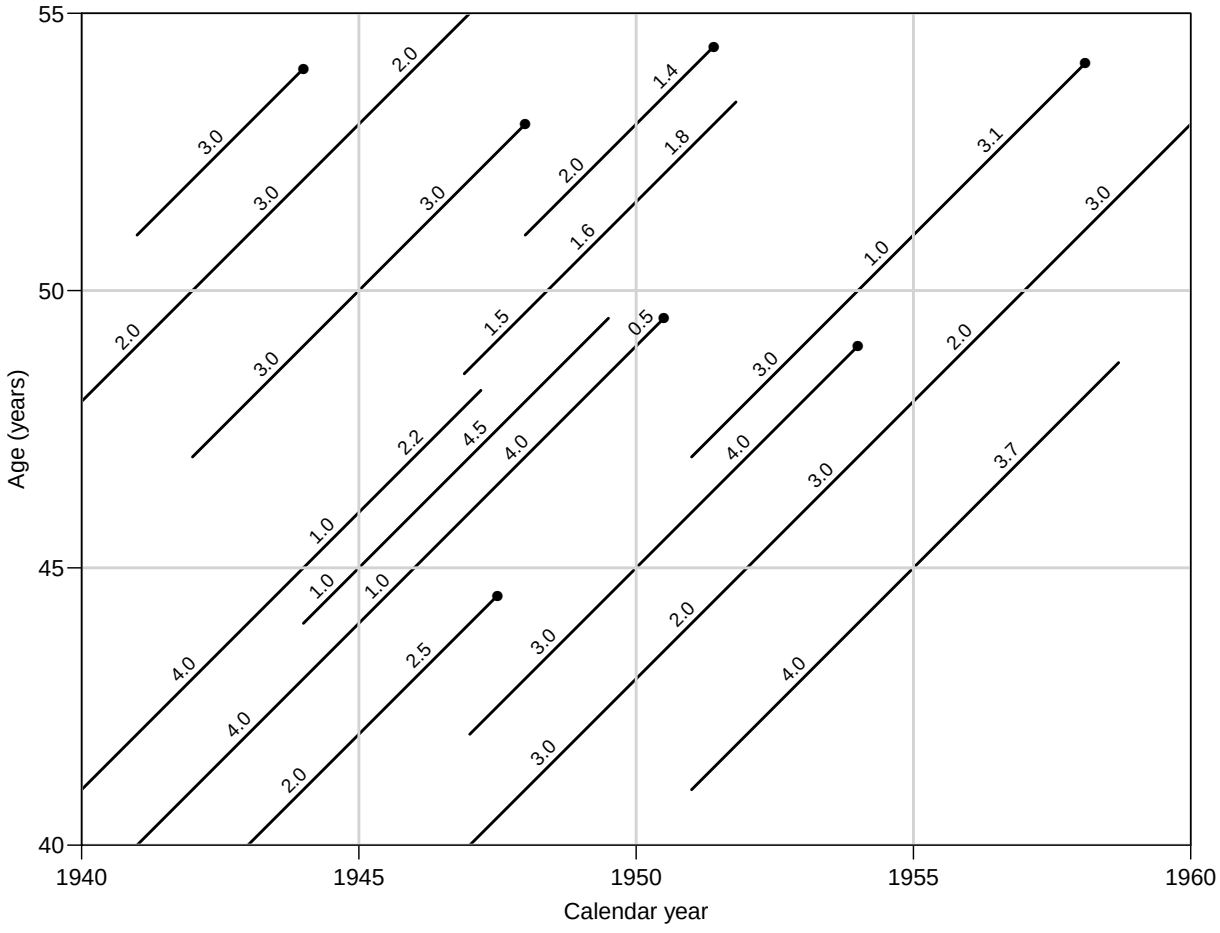
15. Use the same machinery to the additive model to get the rates and the rate-difference in one go. Note that the annotation of the resulting estimates are via the column-names of the contrast matrix.

```
> rownames( CM ) <- c("rate 0","rate 1","RD 1 vs. 0")
> ma <- glm( D/Y ~ factor(xpos), weight=Y,
+                  family=poisson(link=identity) )
> ci.lin( ma, ctr.mat=CM )[,c(1,5,6)]

               Estimate         2.5%       97.5%
rate 0     0.002711497 0.0013393153 0.004083678
rate 1     0.005854066 0.0036857298 0.008022403
RD 1 vs. 0 0.003142570 0.0005765288 0.005708611
```

7.2 Lexis diagram

In the Lexis diagram below displayed follow-up times of a small occupational cohort over the years 1940-1959 and the age range 40-54 years (this example is from **B&D**). Each line runs from the entry to follow-up until either the diagnosis of cancer (•), or censoring or withdrawal (no symbol) due to death from other causes or migration.



1. Calculate the numbers of new cases of cancer, and person-years at risk in all the three 5-year agebands: 40-44, 45-49, and 50-54 years for each of the 5-year calendar periods 1940-44, 1945-49, and 1950-54 separately.

Hint 1: Execute some division of labour in your group, so that not everybody is calculating these items for all periods.

Hint 2: The data set is avaialable as an example dataset, `occup`, in the `Epi` package. Try:

```
> library( Epi )
> data( occup )
> str( occup )
> occup
> example( occup )
```

2. Calculate the numbers of new cases of cancer, person-years at risk in the three 5-year age groups: 40-44, 45-49, and 50-54 years for a *birth cohort* born in 1902-11.

3. Now estimate the cumulative rate and the cumulative risk over the whole 15-year age range for the chosen birth cohort.

4. The age-specific incidences (per 100,000 person-years) in the three 5-year age-groups during 1940–60 in the whole population of the country were 100, 200, and 400, respectively, so there was no variation between the subperiods. Assuming that this is an appropriate reference population, calculate the expected number of cases for the index occupational cohort for the same period. Compare the observed and expected number of cases by standardised incidence ratio, SIR.

   Comment on the result.

7.2.1 Lexis diagram — solution

1. You can load the dataset from the `Epi` package by:

```
> library( Epi )
> data( occup )
> occup
```

   In order to compute the cases and person-years we set up a `Lexis` object:

```
> oL <- Lexis( entry = list( age=AoE, per=DoE ),
+               exit = list(          per=DoX ),
+       entry.status = factor( rep("W",nrow(occup)) ),
+        exit.status = factor( Xst ),
+               data = occup )
> summary( oL )
```

   Exit status `X` and `W` are synonymous. If we want to classify the follow-up (person-years and events) by age and calendar time we must first subdivide by the two timescales, this is done by `splitLexis`:

```
> oL <- splitLexis( oL, time="age", breaks=seq(0,100,5) )
> oL <- splitLexis( oL, time="per", breaks=seq(0,100,5)+1900 )
> oL[order(oL$lex.id,oL$age),]
```

   Having split the follow-up we can make a tabulation of the follow-up using the utility function `timeBand`:

```
> table( timeBand(oL,"age","left"), timeBand(oL,"per","left"))
```

However we do not want the number of observations (lines) in the dataset, we want the number of person-yeras (`lex.dur`) and the number of deaths (`lex.Xst=="D"`), so we set up a matrix with these as columns, and define the two clssification variables:

```
> FU <- with( oL, cbind(lex.Xst=="D",lex.dur) )
> colnames(FU) <- c("D","Y")
> Age    <- timeBand(oL,"age","left")
> Period <- timeBand(oL,"per","left")
```

This enables us to use `xtabs` to simultaneously tabulate person-years and deaths

```
> FUtab <- xtabs( FU ~ Age + Period )
> ftable(FUtab,col.vars=2:3)
```

2. If we want the tabulation by age for the birth cohort 1902–11, we simply restrict the dataset to his group, i.e. the persons where `per` − `age` is betwwen 1029 and 1912:

```
> BC <- subset(oL,per-age>1902 & per-age<1912)
> FU <- with( BC, cbind(lex.Xst=="D",lex.dur) )
> colnames(FU) <- c("D","Y")
> Age   <- timeBand(BC,"age","left")
> FUctab <- xtabs( FU ~ Age )
> FUctab
```

3. The cumulative rate for the cohort:

$$5 \times \left(\frac{1}{16.5} + \frac{1}{15.7} + \frac{1}{7.1} \right) = 1.32, \quad 1 - \exp(-1.32) = 0.73$$

or in terms of the just computed:

```
> sum(FUctab[,1]/FUctab[,2]*5)
```

4. Occupational cohort. Expected number of cases

$$E = \frac{100}{10^5\text{y}} \times (11+9.5+6+0)\ \text{y} + \frac{200}{10^5\text{y}} \times (6+12.2+10.5+5.7)\ \text{y} + \frac{400}{10^5\text{y}} \times (6+8.5+4.2+6.1)\ \text{y} = 0.1949$$

Observed $O = 7$, standardised incidence ratio $7/0.1949 = 35.9$. Quite a risky occupation!

Note that the point of subdividing the follow-up by age and calendar time is to make it possible to apply population rates to the follow-up — the population rates vary by age and calendar time. So what is done is to match the population rates to the follow-up dataset:

```
> p.rates <- data.frame( rate=c(100,200,400), Age=c(40,45,50) )
> oL$Age <- timeBand(oL,"age","left")
> oL <- merge(oL,p.rates)
> oL
```

With this we can now compute the observed and expected cases:

```
> O <- with( oL, sum( lex.Xst=="D" ) )
> E <- with( oL, sum( lex.dur*rate/10^5 ) )
> c( O, E, O/E )
```

Usually, we will use smaller intervals, as well as population rates that actually *do* vary by calendar time, but that would require more complicated computing:

Chapter 8

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 etc.

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"). Implicitly is also assumed a status *during* the follow-up (in survival studies this is "alive").

8.1 Timescales

A timescale is a variable that varies deterministicly *within* each person during follow-up, *e.g.*:

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

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

- Age at entry.
- Date of entry.
- Time since treatment (*at* treatment this is 0).
- Time since relapse (*at* relapse this is 0, before relapse it is `NA`).

In the `Epi` package, follow-up in a cohort is represented in a `Lexis` object. A `Lexis` object is a data frame, where each record represents a piece of follow-up time and with a bit of extra structure representing the follow-up. Normally we will start by setting up a Lexis object with one record per person, each representing the entire follow-up of a person.

For the `nickel` data we would construct a `Lexis` object by:

```
> data( nickel )
> nicL <- Lexis( entry = list( per=agein+dob,
+                              age=agein,
+                              tfh=agein-age1st ),
+                exit = list( age=ageout ),
+          exit.status = ( icd %in% c(162,163) )*1,
+                data = nickel )
```

The `entry` argument is a *named* list with the entry points on each of the timescales we want to use. It defines the names of the timescales and the entry points. The `exit` argument gives the exit time on *one* of the timescales, 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 `ageout - agein`. Now take a look at the result:

```
> str( nickel )
> str( nicL )
> head( nicL )
> summary( nicL )
```

The `Lexis` object `nicL` has a variable for each timescale, the value of which is the entry point on this timescale. The follow-up time is in the variable `lex.dur` (`dur`ation).

We defined the exit status to be death from lung cancer (ICD7 162,163), i.e. this variable is 1 if follow-up ended with a death from this cause. If follow-up ended alive or by death from another cause, the exit status is coded 0, i.e. as a censoring.

Note that the exit status is in the variable `lex.Xst` (e`X`it `st`atus. The variable `lex.Cst` is the state where the follow-up takes place (`C`urrent `st`atus), in this case 0 (alive) for all persons.

It is possible to get a visualization of the follow-up along the timescales chosen by using the `plot` method for `Lexis` objects. `nicL` 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( nicL )
```

The function allows a lot of control over the output, and a `points.Lexis` function allows plotting of the endpoints of follow-up, so here is a more elaborate plot:

```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( nicL, 1:2, lwd=1, col=c("blue","red")[(nicL$exp>0)+1],
+       grid=TRUE, lty.grid=1, col.grid=gray(0.7),
+       xlim=1900+c(0,90), xaxs="i",
+       ylim=  10+c(0,90), yaxs="i", las=1 )
> points( nicL, 1:2, pch=c(NA,3)[nicL$lex.Xst+1],
+         col="lightgray", lwd=3, cex=1.2 )
> points( nicL, 1:2, pch=c(NA,3)[nicL$lex.Xst+1],
+         col=c("blue","red")[(nicL$exp>0)+1], lwd=1, cex=1.2 )
```

8.2 Splitting the follow-up time along a timescale

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

```
> nicS1 <- splitLexis( nicL, "age", breaks=seq(0,100,10) )
> str( nicL )
> str( nicS1 )
> round( subset( nicS1, id %in% 8:10 ), 2 )
```

The resulting object is again a `Lexis` object, and so follow-up may be split further along another timescale. Try this and list the result for individuals 4 and 6:

```
> nicS2 <- splitLexis( nicS1, "tfh", breaks=c(0,1,5,10,20,30,100) )
> round( subset( nicS2, id %in% 8:10 ), 2 )
```

If we want to model the effect of these timescales 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 these. Try:

```
> timeBand( nicS2, "age", "middle" )[1:10]
```

Note that these are the midpoints of the intervals defined by `breaks=`, *not* the midpoints of the actual follow-up intervals. This is because the variable to be used in modeling must be independent of the censoring and mortality pattern — it should only depend on the chosen grouping of the timescale.

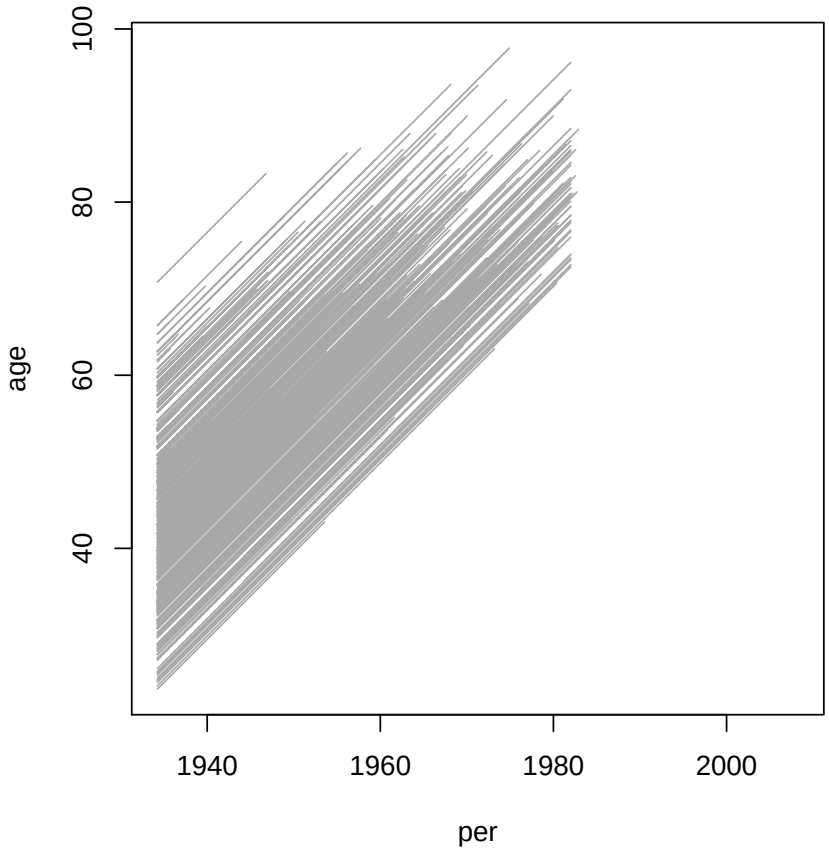


Figure 8.1: *Lexis diagram of the* **nickel** *data set.*

8.3 Cutting time at a specific date

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. This is achieved with the function `cutLexis`, which takes three arguments: the time point, the timescale, and the name of the (new) state following the date.

Now we define the age for the nickel workers where the cumulative exposure exceeds 50 exposure years:

```
> subset( nicL, id %in% 8:10 )
> agehi <- nicL$age1st + 50/nicL$exposure
> nicC <- cutLexis( data=nicL, cut=agehi, timescale="age",
+                   new.state=2, precursor.states=0 )
> subset( nicC[order(nicC$id,nicC$age),], id %in% 8:10 )
```

(The `precursor.states=` argument is explained below). Note that individual 6 has had his follow-up split at age 25 where 50 exposure-years were attained. This could also have been achieved in the split data set `nicS2` instead of `nicL`, try:

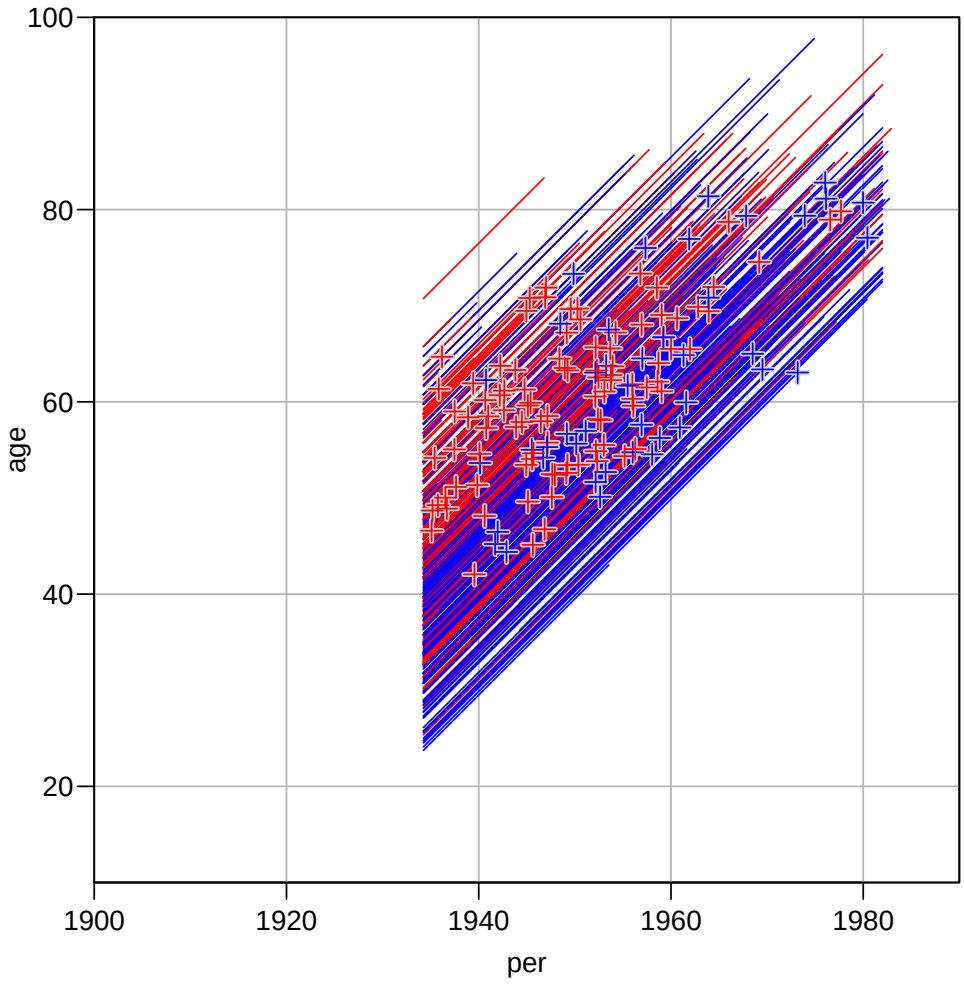


Figure 8.2: *Lexis diagram of the* **nickel** *data set, with bells and whistles. The red lines are for persons with exposure*> 0*, so it is pretty evident that the oldest ones are the exposed part of the cohort.*

```
> subset( nicS2, id %in% 8:10 )
> agehi <- nicS2$age1st + 50/nicS2$exposure
> nicS2C <- cutLexis( data=nicS2, cut=agehi, timescale="age",
+                     new.state=2, precursor.states=0 )
> subset( nicS2C[order(nicS2C$id,nicS2C$age),], id %in% 8:10 )
> summary( nicS2C )
```

Note that follow-up subsequent to the event is classified as being in state 2, but that the final transition to state 1 (death from lung cancer) is preserved. This is the point of the `precursor.states=` argument. It names the states (in this case 0, "Alive") that will be over-written by `new.state` (in this case 2, "High exposure"). Clearly, state 1 ("Dead") should not be updated even if it is after the time where the persons moves to state 2. On other words, only state 0 is a precursor to state 2, state 1 is always subsequent to state 2.

Note if the intermediate event is to be used as a time-dependent variable in a Cox-model, then `lex.Cst` should be used as the time-dependent variable, and `lex.Xst==1` as the event.

It is possible to illustrate the transitions between the different states by the command `boxes.Lexis` — if you omit `boxpos=TRUE`, you will be asked to click on the screen to locate the boxes.

```
> boxes( nicS2C, boxpos=TRUE )
```

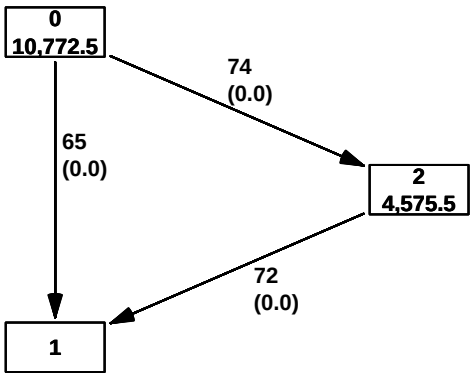


Figure 8.3: *The person years (in the boxes) and number of transitions between the states.*

8.4 Competing risks — multiple types of events

If we want to consider death from lung cancer and death from other causes as separate events we can code these as for example 1 and 2.

```
> data( nickel )
> nicL <- Lexis( entry = list( per=agein+dob,
+                              age=agein,
+                              tfh=agein-age1st ),
+                exit = list( age=ageout ),
+          exit.status = ( icd > 0 ) + ( icd %in% c(162,163) ),
```

```
+                 data = nickel )
> str( nicL )
> head( nicL )
> subset( nicL, id %in% 8:10 )
```

If we want to label the states, we can enter the names of these in the `states` parameter, try for example:

```
> nicL <- Lexis( entry = list( per=agein+dob,
+                              age=agein,
+                              tfh=agein-age1st ),
+                exit = list( age=ageout ),
+          exit.status = ( icd > 0 ) + ( icd %in% c(162,163) ),
+                data = nickel,
+              states = c("Alive","D.oth","D.lung") )
> str( nicL )
```

You can get an overview of the number of records by state and transitions between states as well as the person-years in each state by using `summary.Lexis()`, and computing rates:

```
> summary( nicL, scale=1000 )
```

When we cut at a date as in this case, the date where cumulative exposure exceeds 50 exposure-years, we get the follow-up *after* the date classified as being in the new state if the exit (`lex.Xst`) was to a state we defined as one of the `precursor.states`:

```
> nicL$agehi <- nicL$age1st + 50/nicL$exposure
> nicC <- cutLexis( data=nicL, cut=nicL$agehi, "age",
+                   new.state="HiExp", precursor.states="Alive" )
> subset( nicC, id %in% 8:10 )
> summary( nicC, scale=1000 )
```

Note that the persons-years is the same, but that the number of events has changed. This is because events are now defined as any transition from alive, including the transitions to `HiExp`.

As before we can illustrate the different states with little boxes:

```
> boxes( nicC, boxpos=TRUE, scale.R=1000, pos.arr=c(5,7,5,5,7)/10 )
```

8.5 Multiple events of the same type (recurrent events)

Sometimes more events of the same type are recorded for each person and one would then like to count these and put follow-up time in states accordingly. So states must be *numbered*. Essentially, each set of cut-points represents progressions from one state to the next. Therefore the states should be numbered, and the numbering of states subsequently occupied be increased accordingly.

This is a behaviour different from the one outlined above, and it is achieved by the argument `count=TRUE` to `cutLexis`. When `count` is set to `TRUE`, the value of the arguments `new.state` and `precursor.states` are ignored. Actually, when using the argument `count=TRUE`, the function `countLexis` is called, so an alternative is to use this directly.

If we record when persons pass thresholds of exposure we have this situation. But if we at the same time want to keep track of when people die, we must code death by a sufficiently large number, because all states will be increased by one for each event:

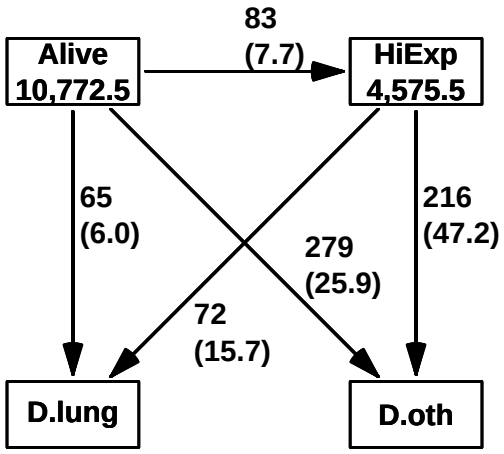


Figure 8.4: *The persons years (in the boxes) and number of transitions and rates per 1000 PY between states in the competing risks model.*

```
> nicL <- Lexis( entry = list( per=agein+dob,
+                              age=agein,
+                              tfh=agein-age1st ),
+                exit = list( age=ageout ),
+          exit.status = ( icd > 0 )*100,
+                data = nickel )
> summary( nicL )
```

We now cut the follow-up at successive exposure thresholds — note that we go through the levels (*i.e.* the times at which they are crossed) by going through them in random order (`sample.int(x)` returns a random permutation of the numbers $1, \ldots, x$).

```
> nicC <- nicL
> exlev <- seq(20,140,40)
> for( level in exlev[sample.int(length(exlev))] )
+    {
+    agehi <- nicC$age1st + level/nicC$exposure
+    nicC <- cutLexis( data=nicC, cut=agehi, "age", count=TRUE )
+    }
> summary( nicC )
```

We can now plot these:

```
> nc <- length( table( nicC$lex.Cst ) )
> boxes( nicC, boxpos=list( x=rep( seq(5,95,,nc), 2 ),
+                           y=rep( c(80,20), each=nc) ),
+              scale.R=100 )
```

We can put a few extra bells and whistles on the graph, by redefining the names of the names of the states by first making them factors (using `factorize`), then by pasting the relevant pieces of text to it. Moreover we also ask that rates instead of no. transitions be shown.

```
> nicF <- factorize( nicC )
> xlev <- paste( c("<",rep("",nc-1)),
+                c(exlev[1],exlev),
+                c("",rep("-",nc-1)), sep="" )
> levels( nicF$lex.Cst ) <-
+ levels( nicF$lex.Xst ) <-
+ c( paste( "Cum.ex.\n", xlev, "\n" ),
+    paste( "Dead\n", xlev ) )
> levels( nicF$lex.Cst )
> boxes( nicF, boxpos=list( y=rep( c(80,20), each=nc),
+                           x=rep( seq(5,95,,nc), 2 ) ),
+              eq.ht=FALSE, hmult=1.5, wmult=1.5, scale.R=1000, pos=0.3 )
```

The resulting two graphs are shown in figure 8.5.

A more thorough explanation of the `Lexis` machinery and its practical use in modeling is given in the papers by Plummer & Carstensen [1, 2].

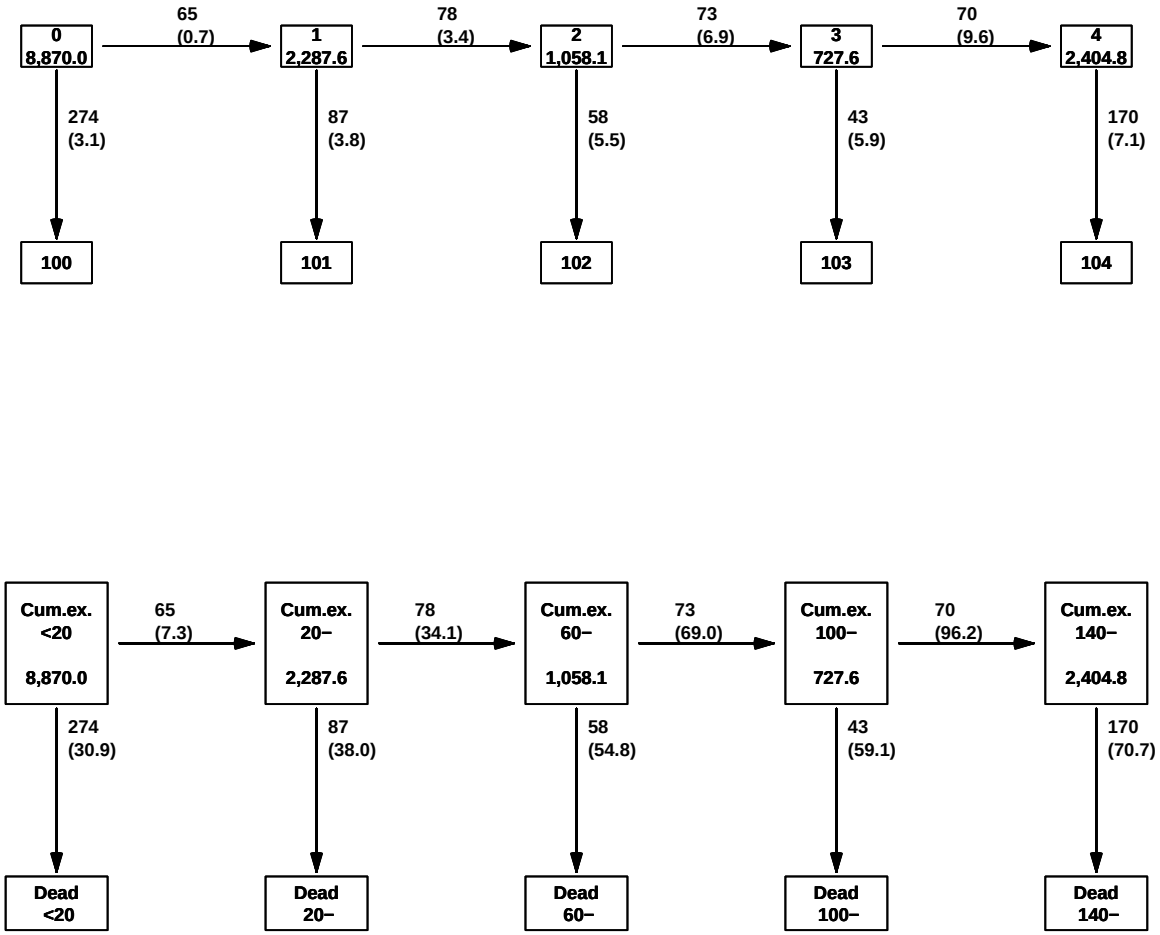


Figure 8.5: *The person years (in the boxes) and number of transitions between states in the counting model. The bottom display is enhanced by labeling of exposure levels, and showing the transition* rates *rather than the no. of transitions.*

8.6 Cox-models and Poisson regression

This exercise demonstrates the equivalence of the Cox-model and an absurd limit of the Poisson model.

To illustrate this we use the lung cancer survival data set from the `survival` package:

```
> options( width=95 )
> library( splines )
> library( Epi )
> library( survival )
> data( lung )
> str( lung )
'data.frame':        228 obs. of  10 variables:
 $ inst     : num  3 3 3 5 1 12 7 11 1 7 ...
 $ time     : num  306 455 1010 210 883 ...
 $ status   : num  2 2 1 2 2 1 2 2 2 2 ...
 $ age      : num  74 68 56 57 60 74 68 71 53 61 ...
 $ sex      : num  1 1 1 1 1 1 2 2 1 1 ...
 $ ph.ecog  : num  1 0 0 1 0 1 2 2 1 2 ...
 $ ph.karno : num  90 90 90 90 100 50 70 60 70 70 ...
 $ pat.karno: num  100 90 90 60 90 80 60 80 80 70 ...
 $ meal.cal : num  1175 1225 NA 1150 NA ...
 $ wt.loss  : num  NA 15 15 11 0 0 10 1 16 34 ...
> lung[1:5,]
  inst time status age sex ph.ecog ph.karno pat.karno meal.cal wt.loss
1    3  306      2  74   1       1       90       100     1175      NA
2    3  455      2  68   1       0       90        90     1225      15
3    3 1010      1  56   1       0       90        90       NA      15
4    5  210      2  57   1       1       90        60     1150      11
5    1  883      2  60   1       0      100        90       NA       0
```

Now take a look at the number of censorings (`status`=1) and deaths (`status`=1) as well as the number of ties:

```
> table( lung$status )
  1   2
 63 165
> table( table( lung$time ) )
  1   2   3
146  38   2
```

Then take a look at the censorings (+) and event times and the summary of the Kaplan-Meier estimator of the overall survival function:

```
> with( lung, Surv( time, status==2 ) )
  [1]  306   455  1010+  210   883  1022+  310   361   218   166   170   654   728    71   567
 [16]  144   613   707    61    88   301    81   624   371   394   520   574   118   390    12
 [31]  473    26   533   107    53   122   814   965+   93   731   460   153   433   145   583
 [46]   95   303   519   643   765   735   189    53   246   689    65     5   132   687   345
 [61]  444   223   175    60   163    65   208   821+  428   230   840+  305    11   132   226
 [76]  426   705   363    11   176   791    95   196+  167   806+  284   641   147   740+  163
 [91]  655   239    88   245   588+   30   179   310   477   166   559+  450   364   107   177
[106]  156   529+   11   429   351    15   181   283   201   524    13   212   524   288   363
[121]  442   199   550    54   558   207    92    60   551+  543+  293   202   353   511+  267
[136]  511+  371   387   457   337   201   404+  222    62   458+  356+  353   163    31   340
[151]  229   444+  315+  182   156   329   364+  291   179   376+  384+  268   292+  142   413+
[166]  266+  194   320   181   285   301+  348   197   382+  303+  296+  180   186   145   269+
[181]  300+  284+  350   272+  292+  332+  285   259+  110   286   270    81   131   225+  269
[196]  225+  243+  279+  276+  135    79    59   240+  202+  235+  105   224+  239   237+  173+
[211]  252+  221+  185+   92+   13   222+  192+  183   211+  175+  197+  203+  116   188+  191+
[226]  105+  174+  177+
```

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

  records   n.max n.start  events  median 0.95LCL 0.95UCL
      228     228     228     165     310     285     363
```

8.6.1 Cox-model

We then fit a Cox-model with age and sex as regression variables, but first we make sex a factor:

```
> lung$sex <- factor( lung$sex, labels=c("M","F") )
> system.time(
+ c.as <- coxph( Surv( time, status==2 ) ~ age + sex,
+                 method="breslow", eps=10^-8, iter.max=25, data=lung )
+             )
     user  system elapsed
    0.001   0.003   0.006
> summary( c.as )
  Call:
  coxph(formula = Surv(time, status == 2) ~ age + sex, data = lung,
      method = "breslow", eps = 10^-8, iter.max = 25)

    n= 228, number of events= 165

            coef exp(coef)  se(coef)      z Pr(>|z|)
  age   0.017013  1.017158  0.009222  1.845  0.06506
  sexF -0.512565  0.598957  0.167462 -3.061  0.00221

       exp(coef) exp(-coef) lower .95 upper .95
  age      1.017     0.9831    0.9989    1.0357
  sexF     0.599     1.6696    0.4314    0.8316

  Concordance= 0.603  (se = 0.026 )
  Rsquare= 0.06   (max possible= 0.999 )
  Likelihood ratio test= 14.08  on 2 df,   p=0.0008741
  Wald test            = 13.44  on 2 df,   p=0.001208
  Score (logrank) test = 13.69  on 2 df,   p=0.001067
```

We can then plot the predicted survival curves for a 60 year old man and woman:

```
> plot( survfit( c.as, newdata=data.frame(age=60,sex=c("M","F")) ),
+       col=c("blue","red"), lwd=3 )
```

Note that the two curves are *estimates* of survival curves; the jumps are at the same places for both curves.

8.6.2 The Poisson model

IN order to fit a Poisson-model to the same follow-up we define the follow-up data as a Lexis object:

```
> Lx <- Lexis( exit = list( tfd=time),
+              exit.status = factor(status,labels=c("Alive","Dead")),
+              data=lung )
  NOTE: entry.status has been set to "Alive" for all.
  NOTE: entry is assumed to be 0 on the tfd timescale.
> summary( Lx )
```

```
  Transitions:
       To
  From    Alive Dead  Records:  Events: Risk time:  Persons:
    Alive    63  165       228      165      69593       228
```

So everyone starts as "Alive" at time 0. The Cox-model is obtained if we fit a Poisson model to data where we have one interval per event, and moreover assign one parameter to each interval. So first we cut data at all entry and exit times to form a data frame with one record per follow-up interval between event times:

```
> dx <- splitLexis( Lx, "tfd", breaks=c(0,unique(Lx$time)) )
> summary( dx )
  Transitions:
       To
  From    Alive Dead  Records:  Events: Risk time:  Persons:
    Alive 19857  165     20022      165      69593       228

> subset( dx, lex.id==19 )[,1:13]

      lex.id tfd lex.dur lex.Cst lex.Xst inst time status age sex ph.ecog ph.karno pat.karno
  2139     19   0       5   Alive   Alive    1   61      2  56   F       2       60         60
  2140     19   5       6   Alive   Alive    1   61      2  56   F       2       60         60
  2141     19  11       1   Alive   Alive    1   61      2  56   F       2       60         60
  2142     19  12       1   Alive   Alive    1   61      2  56   F       2       60         60
  2143     19  13       2   Alive   Alive    1   61      2  56   F       2       60         60
  2144     19  15      11   Alive   Alive    1   61      2  56   F       2       60         60
  2145     19  26       4   Alive   Alive    1   61      2  56   F       2       60         60
  2146     19  30       1   Alive   Alive    1   61      2  56   F       2       60         60
  2147     19  31      22   Alive   Alive    1   61      2  56   F       2       60         60
  2148     19  53       1   Alive   Alive    1   61      2  56   F       2       60         60
  2149     19  54       5   Alive   Alive    1   61      2  56   F       2       60         60
  2150     19  59       1   Alive   Alive    1   61      2  56   F       2       60         60
  2151     19  60       1   Alive    Dead    1   61      2  56   F       2       60         60
```

So we see that the time-split data frame has substantially more records, but exactly the same amount of risk time and no. of events.

We can now fit the detailed Poisson model — pretty daft endeavour, takes quite some computing time because of the excessively large number of parameters:

```
> system.time(
+ p.as <- glm( (dx$lex.Xst=="Dead") ~ factor(tfd) - 1 + age + factor( sex ),
+               offset = log(lex.dur),
+               family=poisson, data=dx, eps=10^-8, maxit=25 )
+             )
     user  system elapsed
    9.915   0.021   9.933

> length( coef(p.as) )

  [1] 188
```

We can now verify that we actually have the same regression parameters as in the Cox-model:

```
> ci.lin(p.as,subset=c("age","sex"))
                  Estimate      StdErr         z          P         2.5%       97.5%
  age           0.01701289 0.009221954  1.844825 0.06506302 -0.001061808  0.03508759
  factor(sex)F -0.51256479 0.167462060 -3.060782 0.00220760 -0.840784401 -0.18434519

> ci.lin(c.as)

          Estimate      StdErr         z           P         2.5%       97.5%
  age   0.01701289 0.009221954  1.844825 0.065063023 -0.001061808  0.03508759
  sexF -0.51256479 0.167462063 -3.060782 0.002207601 -0.840784404 -0.18434518
```

```
> ci.lin(p.as,subset=c("age","sex"))/
+ ci.lin(c.as)
              Estimate StdErr z         P      2.5% 97.5%
  age                1      1 1 0.9999999 0.9999997     1
  factor(sex)F       1      1 1 0.9999998 1.0000000     1
```

Thus the Cox-model is the same as the Poisson model with 188 parameters to describe the baseline intensity.

8.6.3 A more sensible Poisson model

The model above (Cox/Poisson) models the baseline by one parameter per interval between event times. Thus, the ordering and location of the event times (as well as the location of the risk time) is not used in the model; the rates are allowed to vary unrestricted between intervals. This does not seem sensible, so instead of assigning a separate rate-parameter to each interval, we could model the rates as a smooth function of the location of the interval.

To this end we define internal and boundary knots for a spline basis and fit the spline model. The `Epi` package has a function `Ns` that allows specification of a natural spline (restricted cubic spline) by only supplying all knots without specifying which ones are the boundary knots:

```
> kn <- c(0,25,75,150,250,500,1000)
> system.time(
+ s.as <- glm( (lex.Xst=="Dead") ~ Ns( tfd, knots=kn )
+                                 + age + sex,
+              offset = log(lex.dur),
+              family = poisson, data=dx, eps=10^-8, maxit=25 )
+             )
     user  system elapsed
    0.310   0.004   0.315
> length( coef( s.as ) )
  [1] 9
```

This model is much quicker to fit because the no. of parameters is so much smaller, but it still takes much longer than to fit the Cox-model.

We can now compare the parameters from the model with those from the Cox-model:

```
> ci.lin(s.as,subset=c("age","sex"))
         Estimate      StdErr         z           P         2.5%       97.5%
  age   0.01636881 0.009204915  1.778268 0.075359829 -0.001672495  0.03441011
  sexF -0.51200146 0.167452705 -3.057588 0.002231258 -0.840202726 -0.18380019
> ci.lin(c.as)
         Estimate      StdErr         z           P         2.5%       97.5%
  age   0.01701289 0.009221954  1.844825 0.065063023 -0.001061808  0.03508759
  sexF -0.51256479 0.167462063 -3.060782 0.002207601 -0.840784404 -0.18434518
> ci.lin(s.as,subset=c("age","sex"))/
+ ci.lin(c.as)
        Estimate    StdErr         z        P      2.5%     97.5%
  age  0.9621415 0.9981524 0.9639225 1.158259 1.5751390 0.9806918
  sexF 0.9989009 0.9999441 0.9989568 1.010716 0.9993082 0.9970436
```

From the parametric model we can extract the intensity as a function of time since diagnosis, and from that we compute the estimated cumulative intensity over 10-day periods for 60 year old men, and then the survival function:

```
> new <- data.frame( tfd = 1:100*10,
+                    sex = "M",
+                    age = 60,
+                lex.dur = 10 )
> s.pr <- ci.pred( s.as, newdata=new )
> s.surv <- exp( -cumsum( c(0,s.pr[,1]) ) )
```

We can now ploy *both* the estimated underlying mortality (which is the one you never see from the fitted Cox-model):

```
> par( mfrow=c(1,2), mar=c(3,3,1,1), mgp=c(3,1,0)/1.5 )
> matplot( new$tfd, s.pr*300,
+          log="y", xlab="Time since diagnosis (days)", ylab="Mortality (%/month)",
+          type="l", lty=1, lwd=c(4,1,1), col="black" )
> plot( survfit( c.as, newdata=data.frame(age=60,sex="M") ),
+       conf.int=FALSE, mark.time=FALSE, lwd=3,
+       xlab="Time since diagnosis (days)", ylab="Survival probability" )
> lines( c(1,new$tfd), s.surv, col="red", lwd=4 )
```

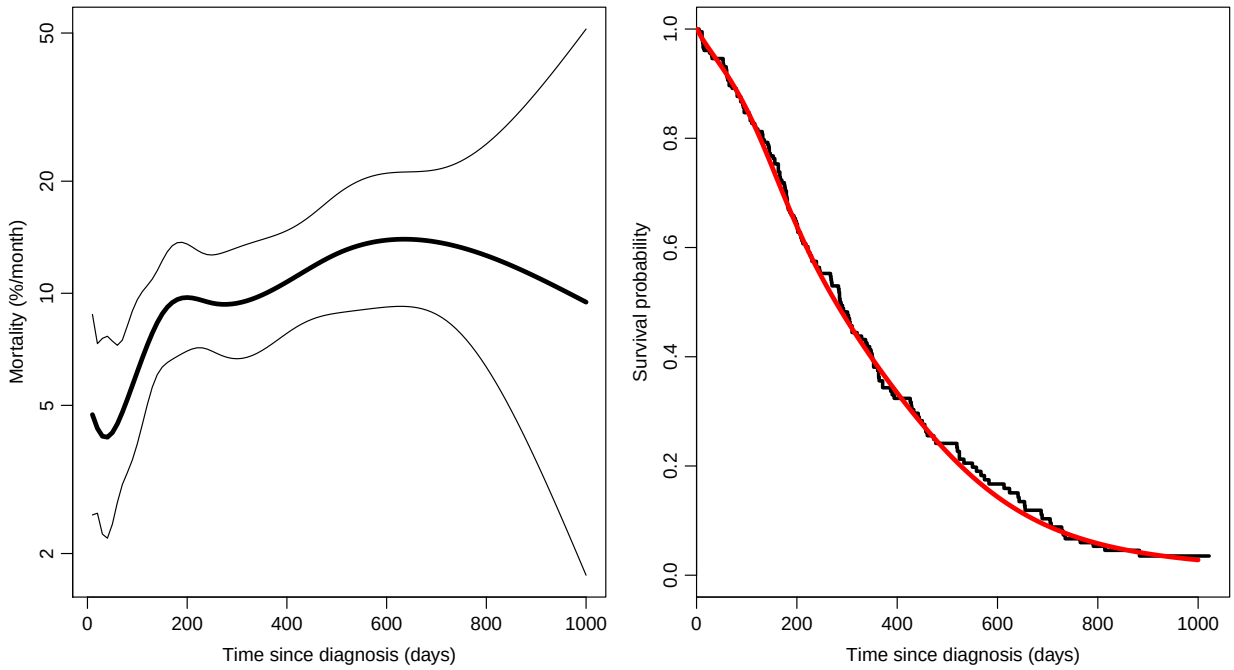


Figure 8.6: *Left: Mortality among 60 year old male lung cancer patients as estimated from a Poisson model with a spline in time. Right: Estimated survival from the Cox-model (black) and the Poisson model (red), 60 year old male lung cancer patients.*

This is the simple way to get the survival function from the parametric Poisson model, but the standard errors does not come with it this way.

In order to get the predicted survival with confidence intervals from the spline model we need to devise a contrast matrix to produce the predictions directly, including the covariance between the point estimates for log-incidence rates at the prediction points. We then use the delta-method to derive standard errors of the cumulative incidences.

This is implemented in the function `ci.cum` in `Epi`, which need the contrast matrix to be multiplied to the parameter vector as one argument

First the time points where we compute the incidences

```
> T.pt <- seq(10,1000,10)
> CM = cbind( 1, Ns( T.pt, knots=kn ), 60, 0 )
> ci.lin( s.as )
                          Estimate      StdErr           z             P         2.5%       97.5%
  (Intercept)          -7.33249005 0.780497057 -9.39464151 5.741300e-21 -8.862236173 -5.80274393
  Ns(tfd, knots = kn)1 -0.01216061 0.590516380 -0.02059317 9.835702e-01 -1.169551443  1.14523023
  Ns(tfd, knots = kn)2  0.73013494 0.620231686  1.17719710 2.391168e-01 -0.485496824  1.94576671
  Ns(tfd, knots = kn)3  0.43351613 0.590979501  0.73355528 4.632198e-01 -0.724782404  1.59181467
  Ns(tfd, knots = kn)4  1.37818863 0.588420963  2.34218140 1.917139e-02  0.224904739  2.53147253
  Ns(tfd, knots = kn)5  0.55990396 1.283944678  0.43608107 6.627779e-01 -1.956581362  3.07638929
  Ns(tfd, knots = kn)6  0.77107474 0.908692852  0.84855377 3.961296e-01 -1.009930519  2.55208001
  age                   0.01636881 0.009204915  1.77826812 7.535983e-02 -0.001672495  0.03441011
  sexF                 -0.51200146 0.167452705 -3.05758845 2.231258e-03 -0.840202726 -0.18380019

> head( CM )
                    1            2 3           4         5          6
  [1,] 1 0.003555556 0.000000e+00 0 -0.09162891 0.2290723 -0.1374434 60 0
  [2,] 1 0.028444444 0.000000e+00 0 -0.16647084 0.4161771 -0.2497063 60 0
  [3,] 1 0.095111111 8.888889e-05 0 -0.20830538 0.5207635 -0.3124581 60 0
  [4,] 1 0.203555556 2.400000e-03 0 -0.21394301 0.5348575 -0.3209145 60 0
  [5,] 1 0.333333333 1.111111e-02 0 -0.19322508 0.4830627 -0.2898376 60 0
  [6,] 1 0.463111111 3.048889e-02 0 -0.15655953 0.3913988 -0.2348393 60 0

> head( Lambda <- ci.cum( s.as, ctr.mat=CM, intl=10 ) )
         Estimate       2.5%      97.5%    Std.err.
  [1,] 0.01573572 0.00597744 0.02549399 0.004978803
  [2,] 0.03018553 0.01409597 0.04627510 0.008209113
  [3,] 0.04395497 0.02175695 0.06615300 0.011325733
  [4,] 0.05764164 0.02890228 0.08638099 0.014663206
  [5,] 0.07172113 0.03665723 0.10678504 0.017890076
  [6,] 0.08658460 0.04604430 0.12712490 0.020684205

> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( survfit( c.as, newdata=data.frame(age=60,sex="M") ),
+       conf.int=TRUE, mark.time=FALSE, lwd=1,
+       ylim=0:1, yaxs="i", bty="n",
+       xlab="Time since diagnosis (days)", ylab="Survival proability")
> lines( survfit( c.as, newdata=data.frame(age=60,sex="M") ),
+       conf.int=FALSE, mark.time=FALSE, lwd=3 )
> lines( c(1,new$tfd), s.surv, col="red", lwd=4 )
> matlines( 1:100*10, exp(-Lambda[,1:3]), lty=1,
+           col=c("white","red","red") )
```

It is not necessary to split time at all event times; it would suffice to split in say 5-day intervals equidistantly; the results would be pretty much the same.

8.6.4 Interaction: testing the proportionality assumption

8.6.4.1 Categorical interaction

If we test the proportionality by `cox.zph` we find a weak indication of a sex-effect, that is an interaction between sex and time:

```
> ( pr.as <- cox.zph( c.as ) )
            rho chisq     p
  age    -0.0274 0.128 0.720
  sexF    0.1235 2.444 0.118
  GLOBAL      NA 2.642 0.267

> par( mfrow=c(1,2) )
> plot( pr.as )
```

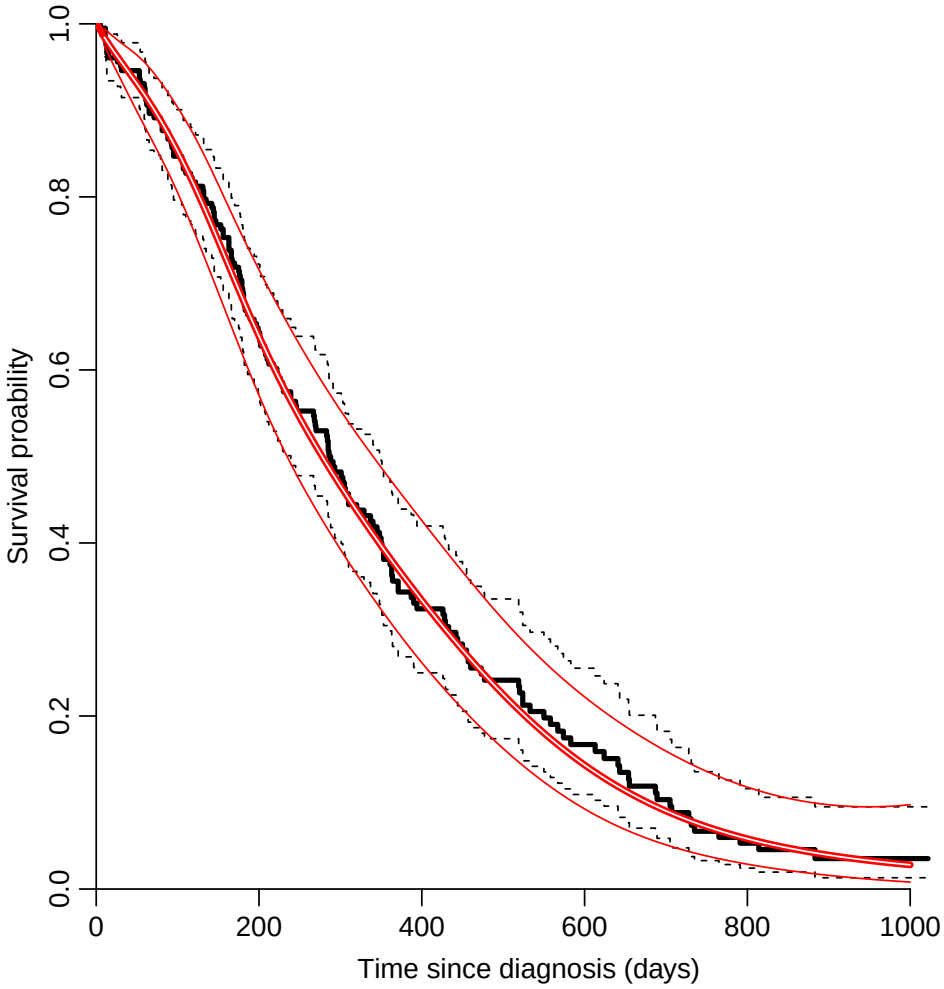


Figure 8.7: *Estimated survival for 60 year old men; black curves are the Breslow-estimator from the Cox-model; red curves estimates from Poisson-model with smooth effects of time since diagnosis.*

Another way of estimating the interaction is to do it explicitly in the Poisson-model; then we both get a likelihood-ratio test and an estimate of the male-female mortality ratio as a function of time. In this setup we can test the *shape* of the interaction; we test a linear and a more extended:

```
> i.as <- glm( (lex.Xst=="Dead") ~ Ns( tfd, knots=kn ) + age +
+                                  sex*tfd,
+               offset = log(lex.dur),
+               family = poisson, data=dx, eps=10^-8, maxit=25 )
> I.as <- glm( (lex.Xst=="Dead") ~ Ns( tfd, knots=kn ) + age +
+                                  sex + sex:Ns( tfd, knots=kn ),
+               offset = log(lex.dur),
+               family = poisson, data=dx, eps=10^-8, maxit=25 )
> anova( s.as, i.as, I.as, test="Chisq" )
Analysis of Deviance Table

Model 1: (lex.Xst == "Dead") ~ Ns(tfd, knots = kn) + age + sex
Model 2: (lex.Xst == "Dead") ~ Ns(tfd, knots = kn) + age + sex * tfd
Model 3: (lex.Xst == "Dead") ~ Ns(tfd, knots = kn) + age + sex + sex:Ns(tfd,
    knots = kn)
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
```

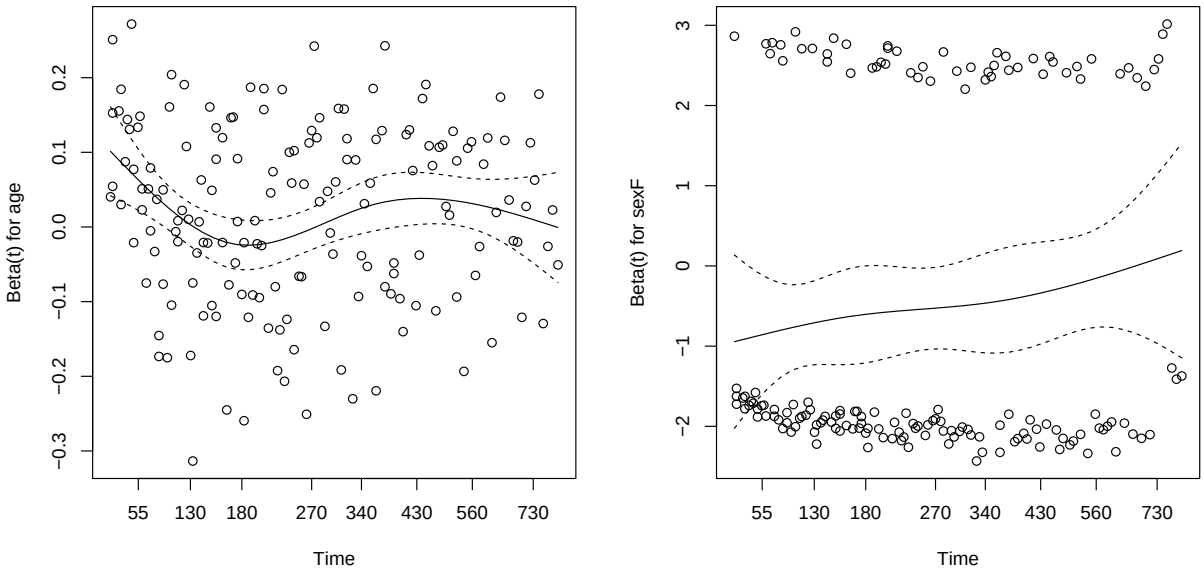


Figure 8.8: *Scoenfeld residuals from the Cox model with age and sex as predictors.*

```
  1     20013     1618.1
  2     20012     1615.1  1   2.9503  0.08586
  3     20007     1613.8  5   1.3683  0.92775
```

We see that there is a borderline significant effect, pretty much with the same p-value as the Schoenfeld test. However, it would also be useful to see the magnitude of the effect, so we set up the matrix needed to extract it:

```
> tpt <- 0:100*10
> ci.exp( i.as, subset="sex" )
            exp(Est.)      2.5%     97.5%
  sexF      0.3924498 0.2159101 0.7133376
  sexF:tfd  1.0014594 0.9998030 1.0031186
> RR <- ci.exp( i.as, subset="sex", ctr.mat=-cbind(1,tpt) )
> matplot( tpt, RR, type="l", lty=1, lwd=c(4,1,1), col="black",
+          log="y", ylab="M/F mortality RR",
+          xlab="Time since diagnosis (days)" )
> abline( h=1 )
```

We can also show the estimate of the curved interaction:

```
> ci.exp( I.as, subset="sex" )
                              exp(Est.)        2.5%        97.5%
  sexF                        0.2119284 0.010758941     4.174543
  Ns(tfd, knots = kn)1:sexF   4.2515736 0.182089006    99.269464
  Ns(tfd, knots = kn)2:sexF   1.7836462 0.059666558    53.319545
  Ns(tfd, knots = kn)3:sexF   4.1326768 0.167130783   102.189539
  Ns(tfd, knots = kn)4:sexF   1.9484053 0.128262624    29.597736
  Ns(tfd, knots = kn)5:sexF  10.3902777 0.008567524 12600.825082
  Ns(tfd, knots = kn)6:sexF  22.6497755 0.463848067  1105.992172
> CM <- cbind( 1, Ns(tpt, knots=kn) )
> RRi <- ci.exp( I.as, subset="sex", ctr.mat=-CM )
> matplot( tpt, cbind(RR,RRi), type="l", lty=1, lwd=c(4,1,1),
+          col=rep(c("black","blue"),each=3),
```

```
+          log="y", ylab="M/F mortality RR", ylim=c(1/5,5),
+          xlab="Time since diagnosis (days)" )
> abline( h=1 )
> abline( h=1/ci.exp( s.as, subset="sex" ), col="gray" )
> abline( h=1/ci.exp( s.as, subset="sex" )[1], lwd=3, col="gray" )
```

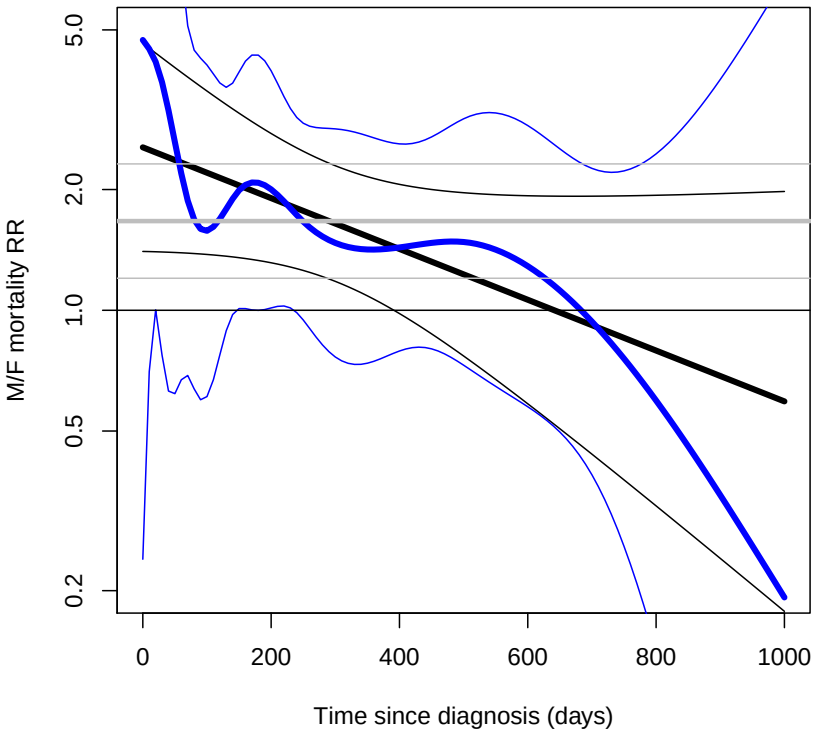


Figure 8.9: *Estimated interaction between sex and time since diagnosis.*

From figure 8.9 it is pretty clear that if anything there is a decreasing M/F rate ratio of quite substantial magnitude, albeit not significantly different from the horizontal gray line. There is a tendency (presumably a selection phenomenon) that men have a higher mortality shortly after diagnosis.

8.6.4.2 Continuous interaction

If we expand the Cox-model with the physician-derived Karnofsky index, `ph.karnof` and test whether there is a proportionality we get a significant non-proportionality:

```
> c.as
  Call:
  coxph(formula = Surv(time, status == 2) ~ age + sex, data = lung,
      method = "breslow", eps = 10^-8, iter.max = 25)


         coef exp(coef) se(coef)     z      p
  age   0.017     1.017  0.00922  1.84 0.0650
  sexF -0.513     0.599  0.16746 -3.06 0.0022

  Likelihood ratio test=14.1  on 2 df, p=0.000874  n= 228, number of events= 165
```

```
> ( c.ask <- update( c.as, ~ . + ph.karno ) )

  Call:
  coxph(formula = Surv(time, status == 2) ~ age + sex + ph.karno,
      data = lung, method = "breslow", eps = 10^-8, iter.max = 25)


               coef exp(coef) se(coef)     z      p
  age        0.0124     1.012  0.00940  1.31 0.1900
  sexF      -0.4966     0.609  0.16771 -2.96 0.0031
  ph.karno -0.0133     0.987  0.00588 -2.26 0.0240

  Likelihood ratio test=18.8  on 3 df, p=0.000306  n= 227, number of events= 164
     (1 observation deleted due to missingness)

> ( cz <- cox.zph( c.ask ) )

                rho    chisq       p
  age       0.00711  0.00896 0.92460
  sexF      0.12238  2.41657 0.12006
  ph.karno 0.23152  8.24624 0.00408
  GLOBAL         NA 11.54038 0.00914

> par( mfrow=c(1,3), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( cz )
```

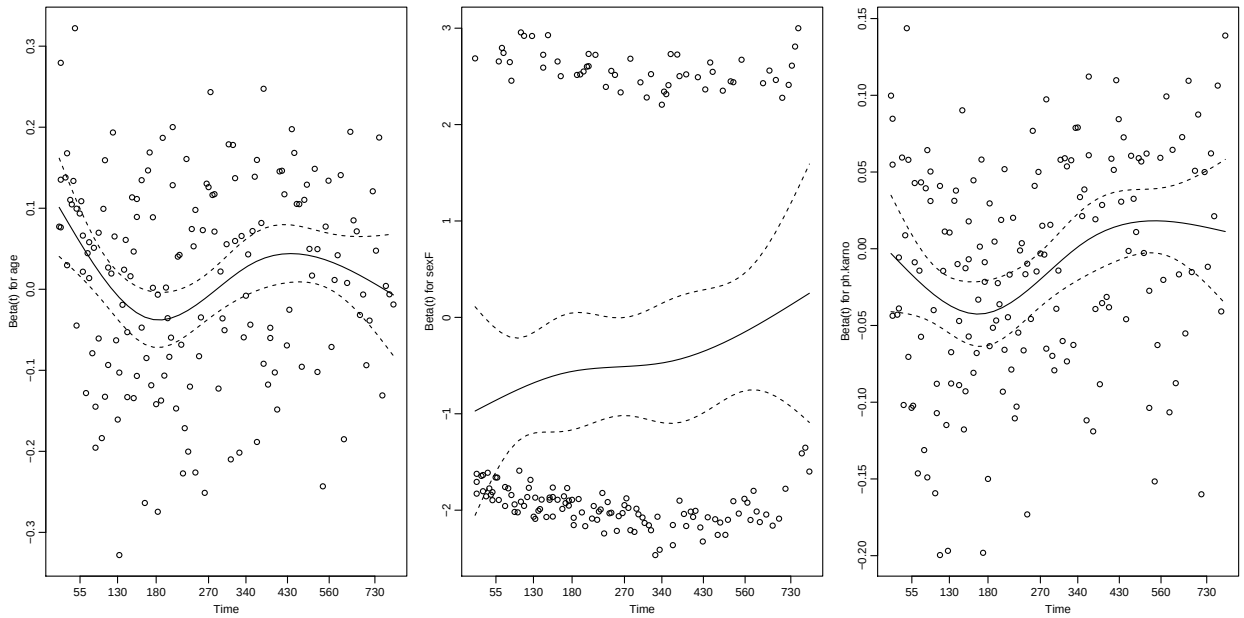


Figure 8.10: *Schoenfeld residuals and smoothed means for the three covariates in the model.*

If we instead use the Poisson model we can explicitly incorporate a parametric interaction between time and the Karnofsky index, and we can graph the mortality RR by time for persons with different values of the Karnofsky index relative to persons with some reference index.

Also we can plot the estimated absolute mortality rates for say 60-year old men with different values of the Karnofsky index. So first we expand the model with the Karnofsky index and continuous interactions.

```
> s.ask <- update( s.as, . ~ . + ph.karno )
> round( ci.exp( s.ask ), 3 )
```

```
                      exp(Est.)  2.5%  97.5%
  (Intercept)             0.002 0.000  0.017
  Ns(tfd, knots = kn)1    0.908 0.282  2.922
  Ns(tfd, knots = kn)2    2.220 0.656  7.519
  Ns(tfd, knots = kn)3    1.561 0.488  4.991
  Ns(tfd, knots = kn)4    4.057 1.287 12.792
  Ns(tfd, knots = kn)5    1.705 0.136 21.315
  Ns(tfd, knots = kn)6    1.830 0.313 10.690
  age                     1.012 0.993  1.031
  sexF                    0.608 0.438  0.845
  ph.karno                0.987 0.976  0.999
```

```
> si.ask <- update( s.ask, . ~ . + I(100/ph.karno):tfd )
> round( ci.exp( si.ask ), 3 )
```

```
                      exp(Est.)   2.5%      97.5%
  (Intercept)             0.018  0.002      0.170
  Ns(tfd, knots = kn)1    1.667  0.485      5.726
  Ns(tfd, knots = kn)2    6.533  1.585     26.929
  Ns(tfd, knots = kn)3   12.260  2.110     71.219
  Ns(tfd, knots = kn)4  150.153 10.892   2069.934
  Ns(tfd, knots = kn)5  527.564  7.428  37469.374
  Ns(tfd, knots = kn)6 1475.427 20.815 104584.667
  age                     1.014  0.995      1.033
  sexF                    0.579  0.417      0.804
  ph.karno                0.960  0.941      0.979
  I(100/ph.karno):tfd     0.995  0.992      0.998
```

```
> sI.ask <- update( s.ask, . ~ . + I(100/ph.karno):Ns(tfd, knots = kn ) )
> round( ci.exp( sI.ask ), 3 )
```

```
                                      exp(Est.)  2.5%        97.5%
  (Intercept)                             0.032 0.001 1.502000e+00
  Ns(tfd, knots = kn)1                    0.272 0.002 3.841900e+01
  Ns(tfd, knots = kn)2                    3.793 0.028 5.146880e+02
  Ns(tfd, knots = kn)3                  458.051 1.852 1.133167e+05
  Ns(tfd, knots = kn)4                    5.804 0.019 1.734744e+03
  Ns(tfd, knots = kn)5                 9173.667 0.424 1.986010e+08
  Ns(tfd, knots = kn)6                 1752.350 0.130 2.369029e+07
  age                                     1.014 0.996 1.033000e+00
  sexF                                    0.578 0.415 8.050000e-01
  ph.karno                                0.953 0.910 9.990000e-01
  Ns(tfd, knots = kn)1:I(100/ph.karno)    2.429 0.073 8.107600e+01
  Ns(tfd, knots = kn)2:I(100/ph.karno)    0.747 0.022 2.595100e+01
  Ns(tfd, knots = kn)3:I(100/ph.karno)    0.013 0.000 7.980000e-01
  Ns(tfd, knots = kn)4:I(100/ph.karno)    0.792 0.009 6.733500e+01
  Ns(tfd, knots = kn)5:I(100/ph.karno)    0.002 0.000 2.386000e+00
  Ns(tfd, knots = kn)6:I(100/ph.karno)    0.006 0.000 1.297700e+01
```

```
> anova( s.ask, si.ask, sI.ask, s.ask, test="Chisq" )
```

```
  Analysis of Deviance Table

  Model 1: (lex.Xst == "Dead") ~ Ns(tfd, knots = kn) + age + sex + ph.karno
  Model 2: (lex.Xst == "Dead") ~ Ns(tfd, knots = kn) + age + sex + ph.karno +
      I(100/ph.karno):tfd
  Model 3: (lex.Xst == "Dead") ~ Ns(tfd, knots = kn) + age + sex + ph.karno +
      Ns(tfd, knots = kn):I(100/ph.karno)
  Model 4: (lex.Xst == "Dead") ~ Ns(tfd, knots = kn) + age + sex + ph.karno
    Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
  1     19989     1607.3
  2     19988     1594.7  1  12.5835 0.0003892
  3     19983     1588.9  5   5.8045 0.3257085
  4     19989     1607.3 -6 -18.3880 0.0053324
```

Here is a clearly significant linear interaction with the Karnofsky index, and the further non-linear extension is not significant relative to the linear. Interestingly, if we had tested

the non-linear interaction directly against the main-effects model, we would have had a significant difference.

Hence we shall look at the distribution of the Karnofsky index in the dataset:

```
> with( lung, pctab( table( status, ph.karno ) ) )
        ph.karno
status     50    60    70    80    90   100   All     N
     1    1.6   4.8   4.8  31.7  39.7  17.5 100.0  63.0
     2    3.0   9.8  17.7  28.7  29.9  11.0 100.0 164.0
```

We see that the relevant reference value for the index is 80, so we will for both interaction models show:

- mortality rates as a function of time for 60-year old men, with Karnofsky indices 60, 70, . . . , 100
- mortality RR as a function of time relative to index 80 for Karnofsky indices 60, 70, . . . , 100

We extract the mortality rates and the RRs for different values of Karnofsky index in the three different models and graph them separately.

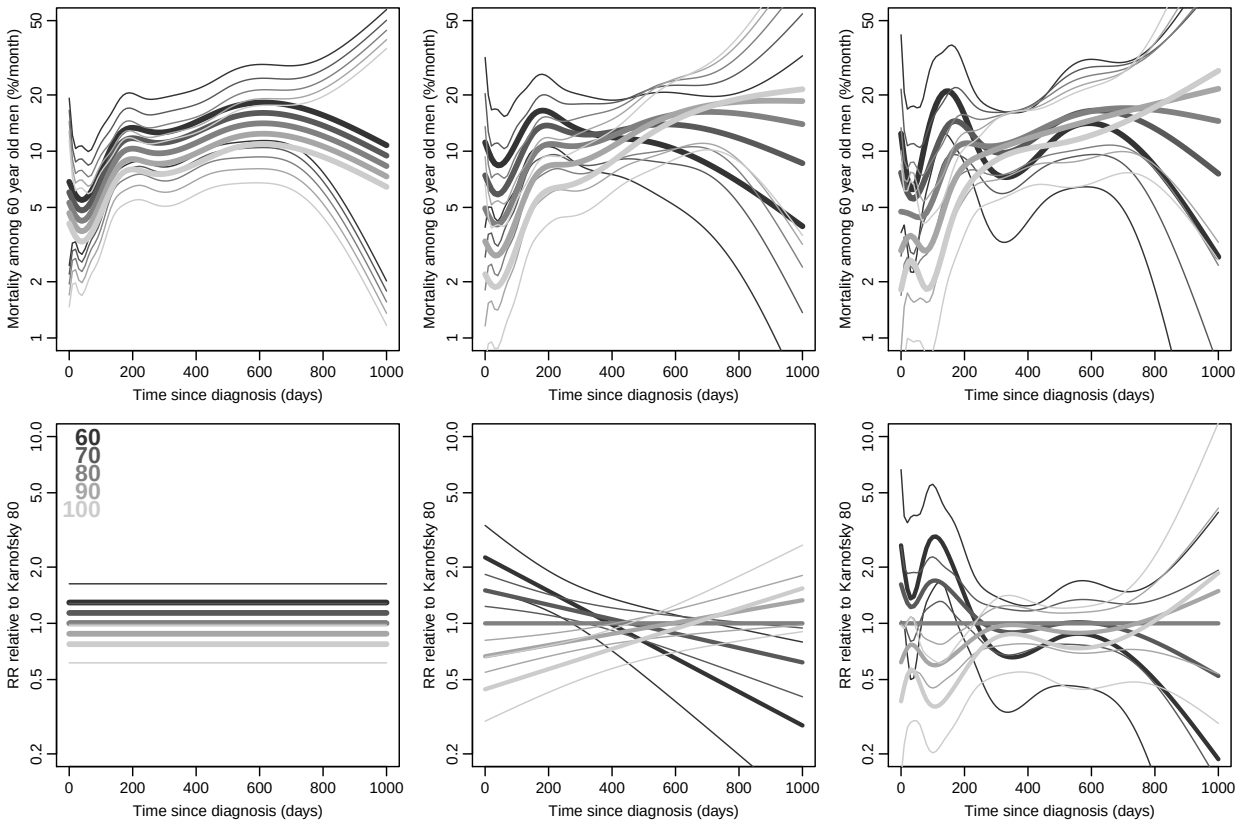


Figure 8.11: *The effect of Karnofsky index in three models (left to right) linear effect, linear-inverse interaction with time and spline-inverse interaction with time. Top panels are predicted mortality rates for a 60-year old man, bottom panels are RR relative Karnofsky index 80.*

```
> new <- data.frame( tfd=0:100*10, sex="M", age=60, lex.dur=3000 )
> rr.k <- rr.i <- rr.I <- NULL
> pr.k <- pr.i <- pr.I <- NULL
> kr <- 80
> for( ki in 6:10*10 )
+ {
+ ndat <- cbind( new, ph.karno=ki )
+ kk <- ndat$ph.karno
+ tt <- ndat$tfd
+ c0 <- cbind( kk-kr )
+ ci <- cbind( kk, (100/kk)*tt )
+ cr <- cbind( kr, (100/kr)*tt )
+ cI <- cbind( kk, Ns(tt,knots=kn)*(100/kk) )
+ cR <- cbind( kr, Ns(tt,knots=kn)*(100/kr) )
+ pr.k <- cbind( pr.k, ci.pred(  s.ask, newdata = ndat ) )
+ pr.i <- cbind( pr.i, ci.pred( si.ask, newdata = ndat ) )
+ pr.I <- cbind( pr.I, ci.pred( sI.ask, newdata = ndat ) )
+ rr.k <- cbind( rr.k, ci.exp(  s.ask, subset="karno", ctr.mat=c0 ) )
+ rr.i <- cbind( rr.i, ci.exp( si.ask, subset="karno", ctr.mat=ci-cr ) )
+ rr.I <- cbind( rr.I, ci.exp( sI.ask, subset="karno", ctr.mat=cI-cR ) )
+ }
> round( head( pr.i ), 3 )
  Estimate  2.5%  97.5% Estimate  2.5%  97.5% Estimate  2.5%  97.5% Estimate  2.5% 97.5%
1   11.142 3.910 31.748    7.422 2.708 20.344    4.944 1.806 13.538    3.294 1.159 9.358
2   10.009 5.025 19.936    6.747 3.590 12.680    4.535 2.419  8.501    3.042 1.536 6.023
3    9.133 5.008 16.655    6.230 3.646 10.646    4.225 2.483  7.188    2.853 1.579 5.156
4    8.596 4.392 16.825    5.934 3.205 10.985    4.059 2.204  7.478    2.761 1.426 5.345
5    8.386 4.212 16.699    5.858 3.101 11.065    4.043 2.154  7.589    2.769 1.410 5.436
6    8.434 4.463 15.941    5.961 3.336 10.652    4.151 2.344  7.353    2.863 1.545 5.305
  Estimate  2.5% 97.5%
1    2.194 0.719 6.693
2    2.038 0.931 4.461
3    1.922 0.953 3.875
4    1.870 0.878 3.982
5    1.886 0.878 4.048
6    1.960 0.965 3.982
> par( mfrow=c(2,3), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( new$tfd, pr.k, log="y", ylim=c(1,50),
+          type="l", lty=1, lwd=c(4,1,1),
+          col=rep( gray(seq(0.2,0.8,,5)), each=3 ),
+          xlab="Time since diagnosis (days)",
+          ylab="Mortality among 60 year old men (%/month)" )
> matplot( new$tfd, pr.i, log="y", ylim=c(1,50),
+          type="l", lty=1, lwd=c(4,1,1),
+          col=rep( gray(seq(0.2,0.8,,5)), each=3 ),
+          xlab="Time since diagnosis (days)",
+          ylab="Mortality among 60 year old men (%/month)" )
> matplot( new$tfd, pr.I, log="y", ylim=c(1,50),
+          type="l", lty=1, lwd=c(4,1,1),
+          col=rep( gray(seq(0.2,0.8,,5)), each=3 ),
+          xlab="Time since diagnosis (days)",
+          ylab="Mortality among 60 year old men (%/month)" )
> matplot( new$tfd, rr.k, log="y", ylim=c(0.2,10),
+          type="l", lty=1, lwd=c(4,1,1),
+          col=rep( gray(seq(0.2,0.8,,5)), each=3 ),
+          xlab="Time since diagnosis (days)",
+          ylab="RR relative to Karnofsky 80" )
> text( rep(100,5), 10*(0.8^(0:4)), 6:10*10,
+       col=gray(seq(0.2,0.8,,5)), font=2, cex=1.5, adj=1 )
> matplot( new$tfd, rr.i, log="y", ylim=c(0.2,10),
+          type="l", lty=1, lwd=c(3,1,1),
+          col=rep( gray(seq(0.2,0.8,,5)), each=3 ),
+          xlab="Time since diagnosis (days)",
+          ylab="RR relative to Karnofsky 80" )
> matplot( new$tfd, rr.I, log="y", ylim=c(0.2,10),
+          type="l", lty=1, lwd=c(3,1,1),
```

```
+          col=rep( gray(seq(0.2,0.8,,5)), each=3 ),
+          xlab="Time since diagnosis (days)",
+          ylab="RR relative to Karnofsky 80" )
```

It appears from the interactions in figure 8.11 that patients with a low Karnofsky index have a higher mortality, particularly in the beginning, and that the relationship either goes away or reverts by time.

The exploration of the non-proportionality as done in the two previous examples is not easily achieved in the Cox-model.

Chapter 9

Diabetes mortality in Denmark

This chapter illustrates the assessment of how mortality among Danish Diabetes patients depend on age, calendar time and duration of diabetes. And how to understand and compute SMR, and assess how it depends on these factors as well.

We are using data from the National Danish Diabetes register. There is a sample of 10,000 records from this in the `Epi` package. Actually there are two, we shall use the one with only cases of diabetes diagnosed after 1995. This is of interest because it is only for these where the data of diagnosis is certain, and hence for whom we can compute the duration of diabetes during follow-up.

9.1 Mortality in Danish diabetes patients

First, we load the `Epi` package and the dataset, and take a look at it:

```
> options( width=120 )
> library( Epi )
> data( DMlate )
> str( DMlate )
  'data.frame':        10000 obs. of  7 variables:
   $ sex  : Factor w/ 2 levels "M","F": 2 1 2 2 1 2 1 1 2 1 ...
   $ dobth: num  1940 1939 1918 1965 1933 ...
   $ dodm : num  1999 2003 2005 2009 2009 ...
   $ dodth: num  NA NA NA NA NA ...
   $ dooad: num  NA 2007 NA NA NA ...
   $ doins: num  NA NA NA NA NA NA NA NA NA NA ...
   $ dox  : num  2010 2010 2010 2010 2010 ...
> 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
> summary( DMlate )
   sex          dobth           dodm          dodth          dooad          doins          dox
   M:5185   Min.   :1898   Min.   :1995   Min.   :1995   Min.   :1995   Min.   :1995   Min.   :1995
   F:4815   1st Qu.:1930   1st Qu.:2000   1st Qu.:2002   1st Qu.:2001   1st Qu.:2001   1st Qu.:2010
            Median :1941   Median :2004   Median :2005   Median :2004   Median :2005   Median :2010
            Mean   :1942   Mean   :2003   Mean   :2005   Mean   :2004   Mean   :2004   Mean   :2009
            3rd Qu.:1951   3rd Qu.:2007   3rd Qu.:2008   3rd Qu.:2007   3rd Qu.:2007   3rd Qu.:2010
            Max.   :2008   Max.   :2010   Max.   :2010   Max.   :2010   Max.   :2010   Max.   :2010
                                          NA's   :7497   NA's   :4503   NA's   :8209
```

We then set up the dataset as a `Lexis` object with age, calendar time and duration of diabetes as timescales, and date of death as event.

9.2 A `Lexis` object

In the dataset we have a date of exit `dox` which is either the day of censoring or the date of death:

```
> with( DMlate, table( dead=!is.na(dodth),
+                      same=(dodth==dox), exclude=NULL ) )
         same
  dead    TRUE <NA>
    FALSE    0 7497
    TRUE  2503    0
    <NA>     0    0
```

So we can set up the `Lexis` object by specifying the timescales and the exit status:

```
> LL <- Lexis( entry = list( A = dodm-dobth,
+                            P = dodm,
+                          dur = 0 ),
+              exit = list( P = dox ),
+       exit.status = factor( !is.na(dodth),
+                             labels=c("Alive","Dead") ),
+              data = DMlate )
  NOTE: entry.status has been set to "Alive" for all.
```

We can get an overview of the data by using the `summary` function on the object:

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

A very crude picture of the mortality by sex can be obtained by the `stat.table` function:

```
> stat.table( sex,
+             list( D=sum( lex.Xst=="Dead" ),
+                   Y=sum( lex.dur ),
+                rate=ratio( lex.Xst=="Dead", lex.dur, 1000 ) ),
+             data=LL )
   --------------------------------
   sex          D        Y    rate
   --------------------------------
   M      1343.00 27614.21   48.63
   F      1156.00 26659.05   43.36
   --------------------------------
```

So not surprisingly, we see that men have a higher mortality than women.

9.2.1 Time-splitting

We now want to assess how mortality depends on age, calendar time and duration. In principle we could split the follow-up along all three time scales, but in practice it would be sufficient to split it along one of the time-scales and then just use the value of each of the time-scales at the left endpoint of the intervals.

We note that the total follow-up time was some 54,000 person-years, so if we split the follow-up in 12-month intervals we get a bit more than 50,000 records:

```
> SL <- splitLexis( LL, breaks=seq(0,125,1/2), time.scale="A" )
> summary( SL )
  Transitions:
       To
  From     Alive Dead  Records:  Events: Risk time:  Persons:
    Alive 115974 2499    118473     2499   54273.27      9996
```

9.3 Mortality models

With this in place we can start by making a crude age-specific mortality curve for men and women separately, using natural splines:

```
> library( splines )
> r.m <- glm( (lex.Xst=="Dead") ~ ns( A, df=10, intercept=TRUE ) - 1,
+             offset = log( lex.dur ),
+             family = poisson,
+               data = subset( SL, sex=="M" ) )
> r.f <- update( r.m, data = subset( SL, sex=="F" ) )
```

With these objects we can get the estimated log-rates by using `predict`, and supplying a data frame of prediction points, and finally use the wrapper `ci.pred` to get the rates with CIs:

```
> nd <-  data.frame( A = seq(10,90,0.5),
+              lex.dur = 1000)
> p.m <- ci.pred( r.m, newdata = nd )
> p.f <- ci.pred( r.f, newdata = nd )
```

and then we can plot the two sets of estimated rates:

```
> matplot( seq(10,90,0.5), cbind(p.m,p.f),
+          type="l", lty=1, lwd=c(3,1,1), las=1,
+          col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(0.1,200),
+          xlab="Age", ylab="Mortality rates per 1000 PY" )
```

9.3.1 Graphical comparison with the population rates

We can compare with the mortality rates from the general population; they are available in the data frame `M.dk`

```
> data( M.dk )
> head( M.dk )
    A sex    P   D        Y      rate
  1 0   1 1974 459 35963.33 12.762999
  2 0   2 1974 303 34382.83  8.812537
  3 0   1 1975 435 36099.00 12.050195
  4 0   2 1975 311 34652.17  8.974908
  5 0   1 1976 405 34965.00 11.583012
  6 0   2 1976 258 33278.33  7.752792
```

So we just plot the mortality rates from 2005 on top of this:

```
> with( subset( M.dk, sex==1 & P==2005 ), lines( A, rate, col="blue", lty="12", lwd=3 ) )
> with( subset( M.dk, sex==2 & P==2005 ), lines( A, rate, col="red" , lty="12", lwd=3 ) )
```

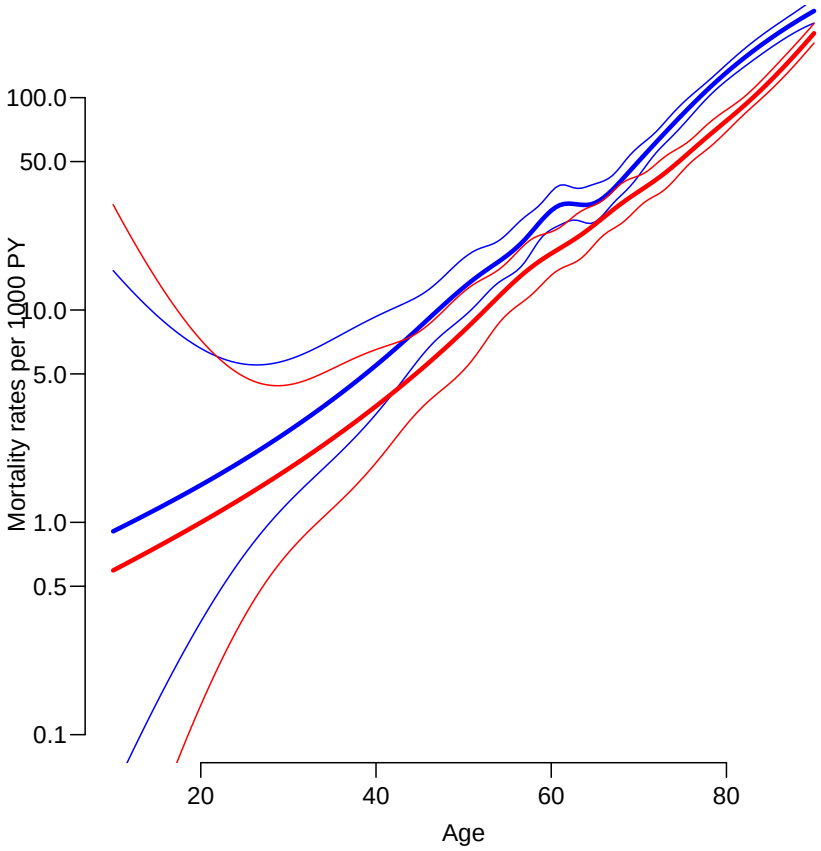


Figure 9.1: *Age-specific mortality rates for Danish diabetes patients as estimated from a model with only age. Blue: men, red: women.*

9.3.2 Modeling population mortality

It would however be more prudent to model these rates in a similar fashion as the diabetes mortality:

```
> R.m <- glm( D ~ ns( A, df=10, intercept=TRUE ) - 1,
+             offset = log( Y ),
+             family = poisson,
+               data = subset( M.dk, sex==1 & P>1994 ) )
> R.f <- update( R.m, data = subset( M.dk, sex==2 & P>1994 ) )
> nd <-  data.frame( A = seq(10,90,0.5),
+                    Y = 1000)
> P.m <- ci.pred( R.m, newdata = nd )
> P.f <- ci.pred( R.f, newdata = nd )
```

Once we have the predicted rates from a smoothing model we can redo the plot with these overlaid:

```
> matplot( seq(10,90,0.5), cbind(p.m,p.f),
+          type="l", lty=1, lwd=c(3,1,1),
+          col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(0.1,200),
+          xlab="Age", ylab="Mortality rates per 1000 PY" )
> matlines( seq(10,90,0.5), cbind(P.m,P.f), lty="12",
+           col=c("blue","red"), lwd=c(3,1,1) )
```

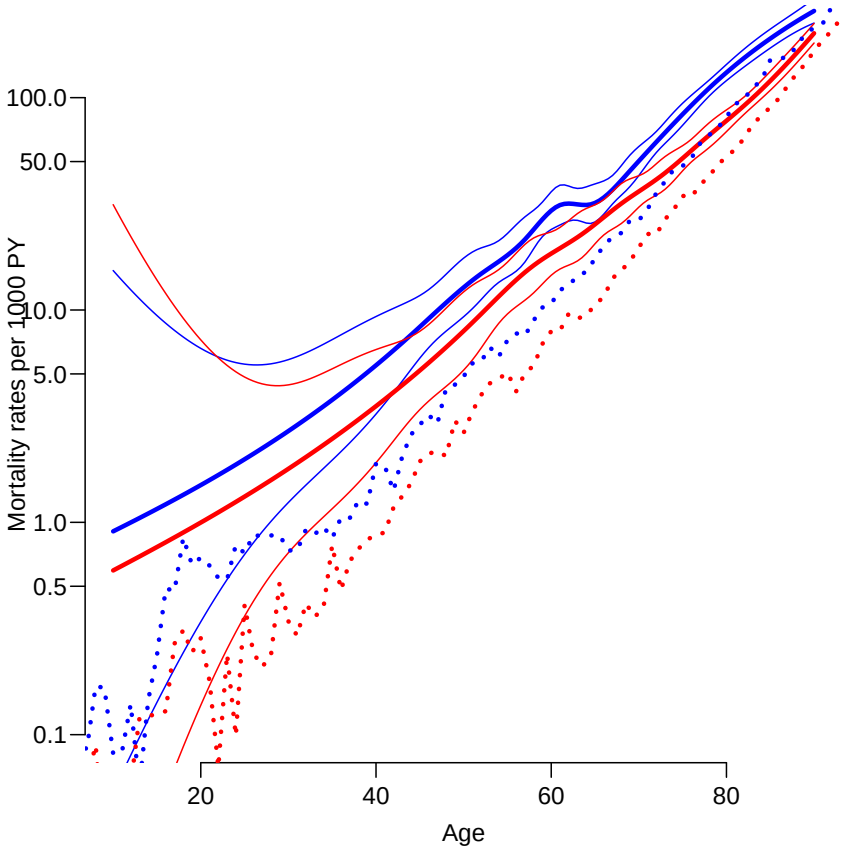


Figure 9.2: *Age-specific mortality rates for Danish diabetes patients as estimated from a model with only age. Broken lines are empirical rates from 2005. Blue: men, red: women.*

9.4 Period and duration effects

We now want to model the mortality rates among diabetes patients also including current date and duration of diabetes. However, we shall not just use the positioning of knots for the splines as provided by `ns`, because this is based on the allocating knots so that the number of observations (lines in the dataset), is the same between knots. However the information in a follow-up study is in the number of events, so it would be better to allocate knots so that number of events were the same between knots.

We will be using so-called *natural splines* that are linear beyond the boundary knots, and hence we take the 5th and 95th percentile of deaths as the boundary knots for age (`A`) and calendar time (`P`) but for duration where we actually have follow-up from tine 0 on the timescale we use 0 as the first knot.

So we start out by placing knots so that the number of events is the same between each pair of knots (strictly speaking we should do this separately for men and women, but we pass on that one here):

```
> ( kn.A <- with( subset( SL, lex.Xst=="Dead" ),
+                 quantile( A+lex.dur, probs=seq(5,95,10)/100 ) ) )
       5%      15%      25%      35%      45%      55%      65%      75%      85%      95%
56.02519 63.67995 69.06092 73.25311 76.29021 79.03847 81.42094 84.27242 87.66598 92.27406
> ( kn.P <- with( subset( SL, lex.Xst=="Dead" ),
+                 quantile( P+lex.dur, probs=seq(5,95,30)/100 ) ) )
```

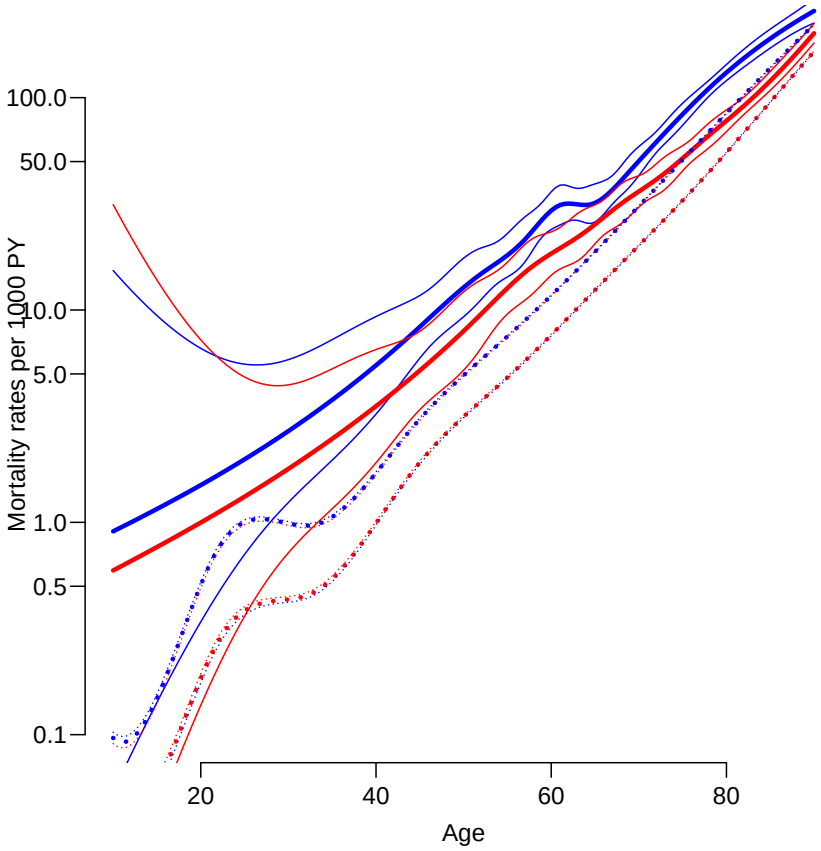


Figure 9.3: *Age-specific mortality rates for Danish diabetes patients as estimated from a model with only age. Broken lines are modeled population rates 1995–2010. Blue: men, red: women.*

```
       5%      35%      65%      95%
1998.117 2003.490 2006.826 2009.658
> ( kn.dur <- c(0,with( subset( SL, lex.Xst=="Dead" ),
+                 quantile( dur+lex.dur, probs=seq(5,95,10)/100 ) ) ) )
                    5%        15%        25%        35%        45%        55%        65%        75%
  0.0000000  0.1065024  0.5549624  1.2210815  1.9783710  2.9568789  3.9411362  5.0770705  6.366872
         95%
 10.6789870
```

With these we can now model mortality rates (separately for men and women), as functions of age, calendar time and duration:

```
> mm <- glm( (lex.Xst=="Dead") ~ Ns( A, kn=kn.A ) +
+                                Ns( P, kn=kn.P ) +
+                                Ns( dur, kn=kn.dur ),
+            offset = log( lex.dur ),
+            family = poisson,
+              data = subset( SL, sex=="M" ) )
> summary( mm )

  Call:
  glm(formula = (lex.Xst == "Dead") ~ Ns(A, kn = kn.A) + Ns(P,
      kn = kn.P) + Ns(dur, kn = kn.dur), family = poisson, data = subset(SL,
      sex == "M"), offset = log(lex.dur))

  Deviance Residuals:
```

```
    Min       1Q   Median       3Q      Max
-0.8209  -0.2257  -0.1637  -0.1113   4.4781

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)             -3.07100    0.11525 -26.647  < 2e-16
Ns(A, kn = kn.A)1        0.68137    0.18401   3.703 0.000213
Ns(A, kn = kn.A)2        1.23017    0.16609   7.407 1.30e-13
Ns(A, kn = kn.A)3        1.45987    0.18737   7.791 6.63e-15
Ns(A, kn = kn.A)4        1.93749    0.17576  11.024  < 2e-16
Ns(A, kn = kn.A)5        2.14289    0.18795  11.401  < 2e-16
Ns(A, kn = kn.A)6        1.88278    0.19994   9.417  < 2e-16
Ns(A, kn = kn.A)7        2.39331    0.16835  14.216  < 2e-16
Ns(A, kn = kn.A)8        3.15094    0.13231  23.814  < 2e-16
Ns(A, kn = kn.A)9        2.46528    0.12507  19.711  < 2e-16
Ns(P, kn = kn.P)1       -0.27974    0.11518  -2.429 0.015150
Ns(P, kn = kn.P)2       -0.46891    0.17475  -2.683 0.007288
Ns(P, kn = kn.P)3       -0.29893    0.09597  -3.115 0.001841
Ns(dur, kn = kn.dur)1   -0.80244    0.23296  -3.444 0.000572
Ns(dur, kn = kn.dur)2   -0.70553    0.23032  -3.063 0.002190
Ns(dur, kn = kn.dur)3   -0.85622    0.21821  -3.924 8.72e-05
Ns(dur, kn = kn.dur)4   -0.65603    0.21720  -3.020 0.002524
Ns(dur, kn = kn.dur)5   -0.99130    0.21039  -4.712 2.46e-06
Ns(dur, kn = kn.dur)6   -0.20911    0.19712  -1.061 0.288775
Ns(dur, kn = kn.dur)7   -1.05815    0.20146  -5.252 1.50e-07
Ns(dur, kn = kn.dur)8   -0.44781    0.16043  -2.791 0.005249
Ns(dur, kn = kn.dur)9   -0.85253    0.26511  -3.216 0.001301
Ns(dur, kn = kn.dur)10  -0.24127    0.13095  -1.842 0.065414

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 12999  on 60346  degrees of freedom
Residual deviance: 11708  on 60324  degrees of freedom
AIC: 14440

Number of Fisher Scoring iterations: 8
```

```
> mf <- update( mm, data = subset( SL, sex=="F" ) )
```

These models fit substantially better than the model with only age as we can see from this comparison:

```
> anova( mm, r.m, test="Chisq" )
Analysis of Deviance Table

Model 1: (lex.Xst == "Dead") ~ Ns(A, kn = kn.A) + Ns(P, kn = kn.P) + Ns(dur,
    kn = kn.dur)
Model 2: (lex.Xst == "Dead") ~ ns(A, df = 10, intercept = TRUE) - 1
  Resid. Df Resid. Dev  Df Deviance  Pr(>Chi)
1     60324      11708
2     60337      11810 -13  -102.33 5.868e-16
```

```
> anova( mf, r.f, test="Chisq" )
Analysis of Deviance Table

Model 1: (lex.Xst == "Dead") ~ Ns(A, kn = kn.A) + Ns(P, kn = kn.P) + Ns(dur,
    kn = kn.dur)
Model 2: (lex.Xst == "Dead") ~ ns(A, df = 10, intercept = TRUE) - 1
  Resid. Df Resid. Dev  Df Deviance  Pr(>Chi)
1     58103      10199
2     58116      10266 -13  -67.312 2.499e-09
```

The models are not formally nested since the location of the knots are different, so from a formal point of view these test are not valid, but is is clear that the more extensive modeling provides a much better description of the rates.

The model fitted separately for men and women has three terms: age (`A`), calendar time (`P`) and diabetes duration (`dur`). Since the outcome is a rate with dimension time^{-1} we must put the rate dimension on one of these terms and leave the two others as rate-ratios. In order to do this we must fix reference values for the two rate-ratio terms. The natural variable for the rate-dimension is age, so that we get estimated age-specific rate-ratios for a specific calendar time, 1.1.2008, say, and a specific duration of diabetes, 2 years, say.

In order to extract these terms from the model we need contrast matrices, that is matrices where each row corresponds to a set of values for age or period or duration, and the columns correspond to the columns in the spline basis as used in the model *i.e.* the parameters.

This is one reason for explicitly fixing the knots in the spline definitions; when we extract the effects we `must` use the same set of knots as in the model specification in order to get the right predictions.

We will need matrices for specified set of values for age, calendar time and duration, but also matrices where all rows refer to the chosen reference values for calendar time and duration.

We begin by specifying the prediction points for the time scales and the reference points. There is formally no reason to require that the matrices all have the same number of rows, but it makes the handling of the reference points much easier.

```
> N <- 100
> pr.A <- seq(10,90,,N)
> pr.P <- seq(1995,2010,,N)
> pr.d <- seq(0,15,,N)
> rf.P <- 2009
> rf.d <- 2
```

With these in place we generate the matrices we shall multiply to the parameter estimates:

```
> AC <- Ns( pr.A, knots=kn.A )
> PC <- Ns( pr.P, knots=kn.P )
> dC <- Ns( pr.d, knots=kn.dur )
> PR <- Ns( rep(rf.P,N), knots=kn.P )
> dR <- Ns( rep(rf.d,N), knots=kn.dur )
```

Note that the rows of `AC` refer to `N` points on the age-scale, `PC` to N points on the calendar time scale, etc.

These matrices are the necessary input for extracting the effects; this is done by the function `ci.exp` — remember to take a look at the help page for this.

Note that we make use of *all* parameters when extracting the age-effect — this is the effect where we have the dimension of the response (rate), and hence the intercept, and where we have fixed the values of date and duration at their reference values.

The rate-ratios for calendar time and duration are estimated exclusively from the parameters for these terms, but note that we subtract the values at the reference point:

```
> m.A <- ci.exp( mm, ctr.mat=cbind(1,AC,PR,dR) ) * 1000
> f.A <- ci.exp( mf, ctr.mat=cbind(1,AC,PR,dR) ) * 1000
> m.P <- ci.exp( mm, subset="P"  , ctr.mat=PC-PR )
> f.P <- ci.exp( mf, subset="P"  , ctr.mat=PC-PR )
> m.d <- ci.exp( mm, subset="dur", ctr.mat=dC-dR )
> f.d <- ci.exp( mf, subset="dur", ctr.mat=dC-dR )
```

We now plot the three effects in three panels beside each other:

```
> par( mfrow=c(1,3), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( pr.A, cbind(m.A,f.A),
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", xlab="Age", ylab="Mortality rate per 1000 PY" )
> matplot( pr.P,  cbind(m.P,f.P),
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", xlab="Date of follow-up", ylab="Mortality rate ratio" )
> matplot( pr.d,  cbind(m.d,f.d),
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", xlab="Diabetes duration", ylab="Mortality rate ratio" )
```

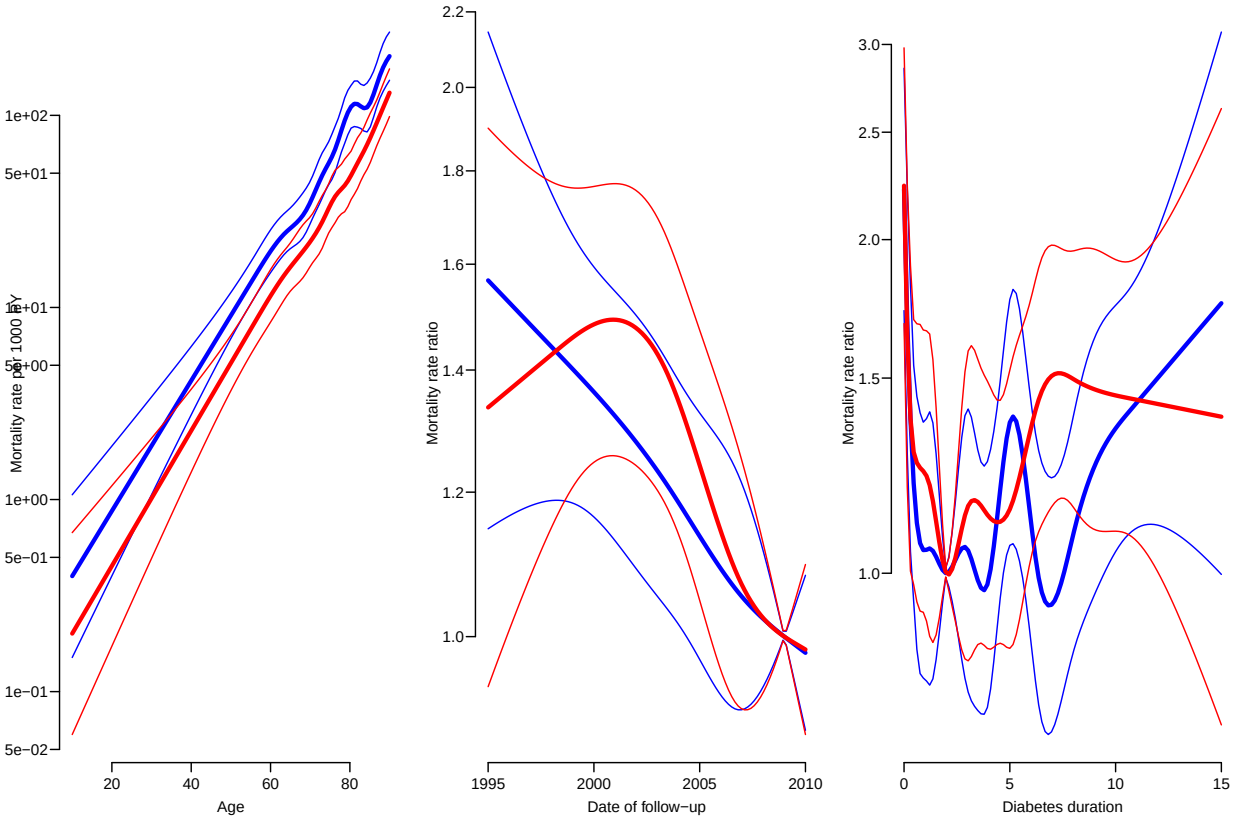


Figure 9.4: *Estimates from model for mortality of Danish diabetes patients. The duration is modeled with 10 parameters, which is clearly way too much.*

Figure 9.4 clearly shows that the duration effect is grossly over-modeled, and that the rate-ratios have a much smaller variability than the mortality rates.

Moreover the y-axis for mortality rates should be from about 0.1 to 200, and the y-axes for the rate-ratios should be on approximately the same scale. To make the RR-axes symmetric, from 1/30 to 30, that is a factor $30^2 = 900$, and the the rate-axis from 0.2 to 180, also a factor of 900 between endpoints of the axes.

So we redefine the duration knots, refit the models, re-extract parameters and plot using pre-specified axis ranges:

```
> kn.dur <- c(0,with( subset( SL, lex.Xst=="Dead" ),
+                 quantile( dur+lex.dur, probs=seq(5,95,30)/100 ) ))
> dC <- Ns( pr.d, knots=kn.dur )
> dR <- Ns( rep(rf.d,N), knots=kn.dur )
> mm <- glm( (lex.Xst=="Dead") ~ Ns( A, kn=kn.A ) +
+                                Ns( P, kn=kn.P ) +
```

```
+                                Ns( dur, kn=kn.dur ),
+             offset = log( lex.dur ),
+             family = poisson,
+               data = subset( SL, sex=="M" ) )
> mf <- update( mm, data = subset( SL, sex=="F" ) )
> m.A <- ci.exp( mm, ctr.mat=cbind(1,AC,PR,dR) ) * 1000
> f.A <- ci.exp( mf, ctr.mat=cbind(1,AC,PR,dR) ) * 1000
> m.P <- ci.exp( mm, subset="P"  , ctr.mat=PC-PR )
> f.P <- ci.exp( mf, subset="P"  , ctr.mat=PC-PR )
> m.d <- ci.exp( mm, subset="dur", ctr.mat=dC-dR )
> f.d <- ci.exp( mf, subset="dur", ctr.mat=dC-dR )
> par( mfrow=c(1,3), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( pr.A, cbind(m.A,f.A),
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(0.2,180),
+          xlab="Age", ylab="Mortality rate per 1000 PY" )
> matplot( pr.P,  cbind(m.P,f.P),
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(1/30,30),
+          xlab="Date of follow-up", ylab="Mortality rate ratio" )
> abline( h=1 )
> matplot( pr.d,  cbind(m.d,f.d),
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(1/30,30),
+          xlab="Diabetes duration", ylab="Mortality rate ratio" )
> abline( h=1 )
```

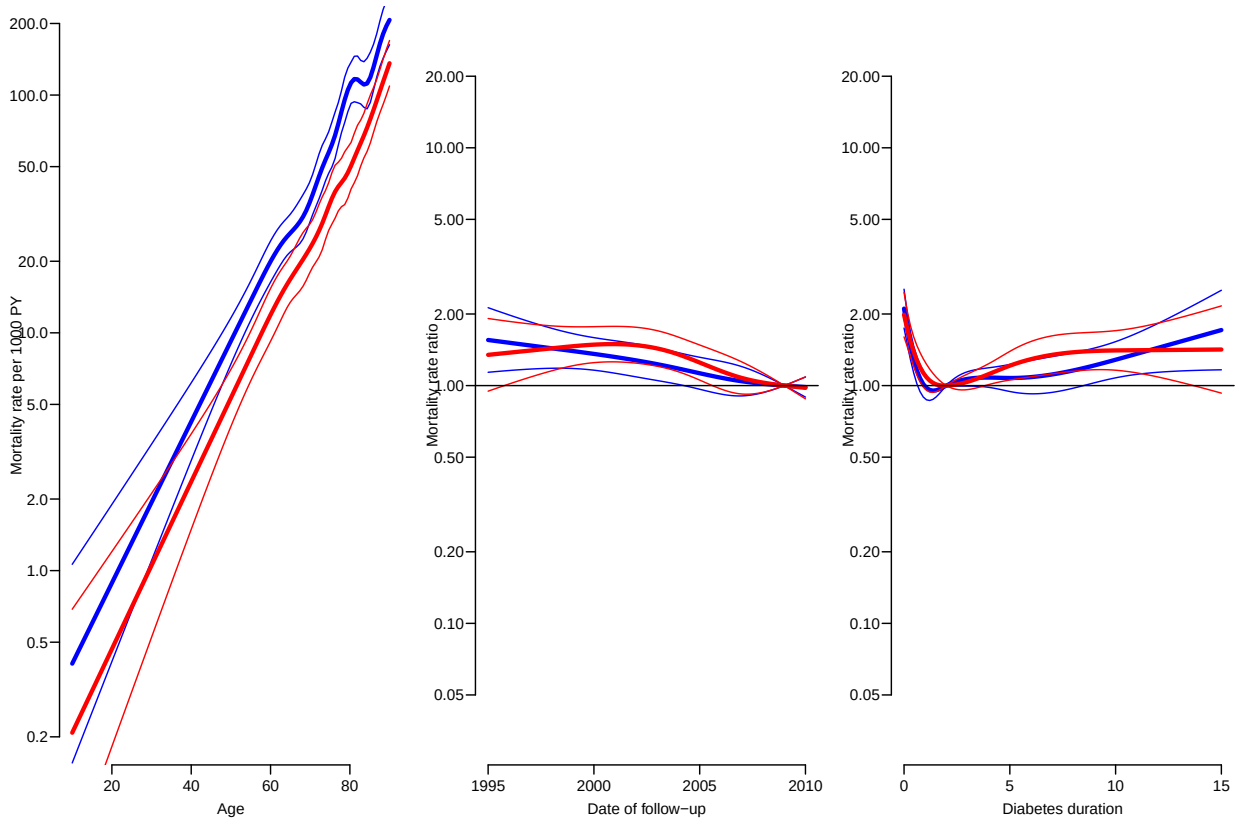


Figure 9.5: *Estimates from the model for mortality of Danish diabetes patients with only 5 knots (corresponding to 4 parameters) for duration.*

We might argue that we do not need the same scale for the y-axes for rates and RRs:

```
> par( mfrow=c(1,3), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( pr.A, cbind(m.A,f.A),
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(0.2,180),
+          xlab="Age", ylab="Mortality rate per 1000 PY" )
> matplot( pr.P,  cbind(m.P,f.P),
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(1/3,3),
+          xlab="Date of follow-up", ylab="Mortality rate ratio" )
> abline( h=1 )
> matplot( pr.d,  cbind(m.d,f.d),
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(1/3,3),
+          xlab="Diabetes duration", ylab="Mortality rate ratio" )
> abline( h=1 )
```

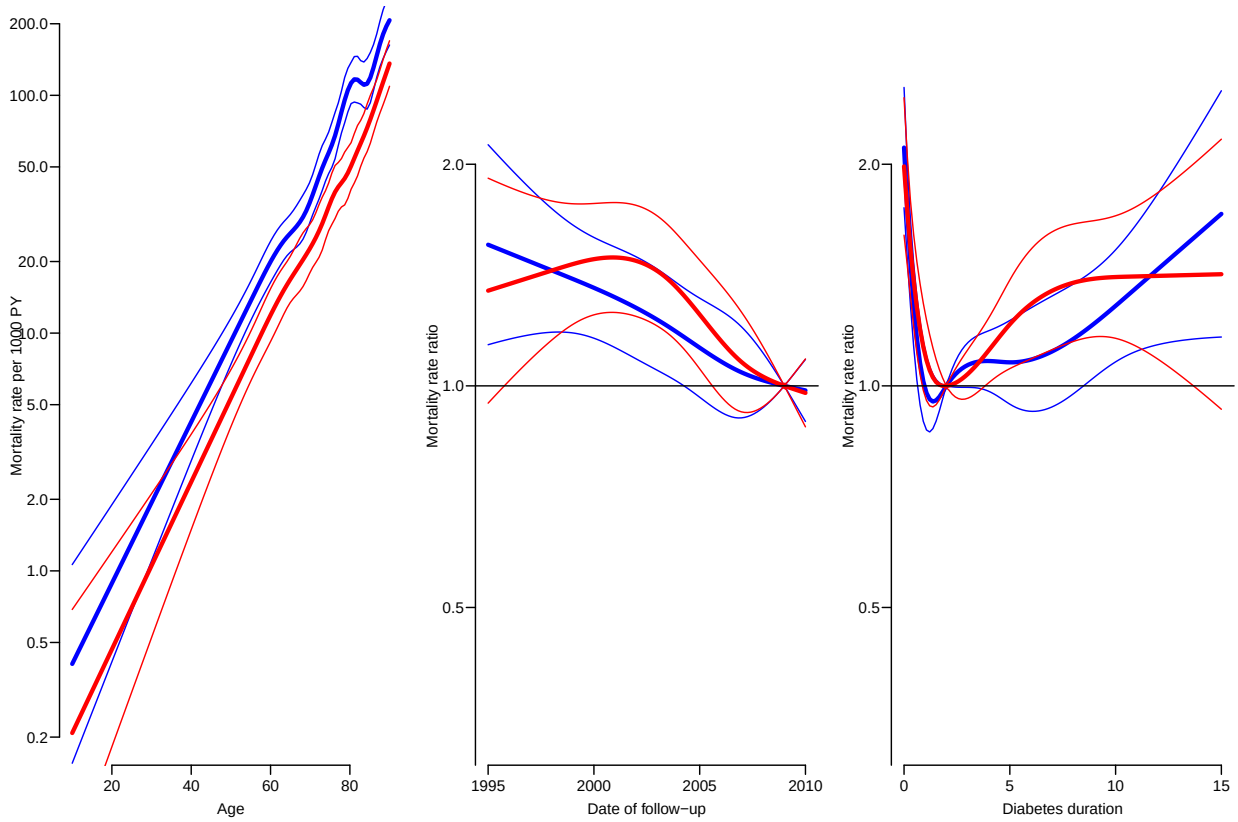


Figure 9.6: *Estimates from model for mortality of Danish diabetes patients.*

9.4.1 Common parameters for men and women

We have so far fitted models separately for men and women, but judging from the display of the parameters in figure 9.6, the period and duration effects are the same, so we might fit a model for the entire dataset with common period and duration effects, but different age-effect for the two sexes:

```
> m2 <- glm( (lex.Xst=="Dead") ~ sex +
+                              sex:Ns( A, kn=kn.A ) +
+                                  Ns( P, kn=kn.P ) +
+                                  Ns( dur, kn=kn.dur ),
```

```
+            offset = log( lex.dur ),
+            family = poisson,
+              data = SL )
> ci.exp(m2)
                            exp(Est.)        2.5%       97.5%
  (Intercept)             0.04119274  0.03438381  0.04935001
  sexF                    0.66279727  0.52988985  0.82904063
  Ns(P, kn = kn.P)1       0.73073343  0.61936177  0.86213158
  Ns(P, kn = kn.P)2       0.68819843  0.53300611  0.88857722
  Ns(P, kn = kn.P)3       0.69848011  0.60798995  0.80243837
  Ns(dur, kn = kn.dur)1   0.50506934  0.41989411  0.60752230
  Ns(dur, kn = kn.dur)2   0.74203629  0.62822061  0.87647214
  Ns(dur, kn = kn.dur)3   0.32632124  0.23932256  0.44494573
  Ns(dur, kn = kn.dur)4   0.99090716  0.84957330  1.15575313
  sexM:Ns(A, kn = kn.A)1  1.98512038  1.38417401  2.84697075
  sexF:Ns(A, kn = kn.A)1  2.39621511  1.49756936  3.83411082
  sexM:Ns(A, kn = kn.A)2  3.40484122  2.45876781  4.71494042
  sexF:Ns(A, kn = kn.A)2  3.18475083  2.13514511  4.75032719
  sexM:Ns(A, kn = kn.A)3  4.30809709  2.98415003  6.21942609
  sexF:Ns(A, kn = kn.A)3  4.72130255  3.03591495  7.34233276
  sexM:Ns(A, kn = kn.A)4  6.94917213  4.92438942  9.80649360
  sexF:Ns(A, kn = kn.A)4  5.04812724  3.33296803  7.64591450
  sexM:Ns(A, kn = kn.A)5  8.47754594  5.86563417 12.25251749
  sexF:Ns(A, kn = kn.A)5  6.56708550  4.38443826  9.83629131
  sexM:Ns(A, kn = kn.A)6  6.57840684  4.44759052  9.73008561
  sexF:Ns(A, kn = kn.A)6  8.39810642  5.83749263 12.08193243
  sexM:Ns(A, kn = kn.A)7 10.97880122  7.89479441 15.26753831
  sexF:Ns(A, kn = kn.A)7 10.70794349  7.91116788 14.49344214
  sexM:Ns(A, kn = kn.A)8 23.24296856 17.93627190 30.11972557
  sexF:Ns(A, kn = kn.A)8 27.59059972 21.11490857 36.05230826
  sexM:Ns(A, kn = kn.A)9 11.67404685  9.13826544 14.91348339
  sexF:Ns(A, kn = kn.A)9 14.86040520 11.52591094 19.15958261
```

We can formally test this model against the separate models; the deviance and degrees of freedom from the separate models for men and women add up to that of a joint model with interaction between all terms and sex. Note that we add 1 to the degrees of freedom for the joint model; this is because the degrees of freedom is equal to the number of parameters *minus 1*, so the sum of the degrees of freedom from the two models is 1 too small — loosely speaking the intercepts from the two separate models correspond to the overall intercept and the main effect of sex in a joint model, and the sex parameter should be counted too.

```
> j.dev <- mm$dev + mf$dev
> j.df  <- mm$df.r + mf$df.r + 1
> 1 - pchisq( m2$dev - j.dev, m2$df.r - j.df )
  [1] 0.347409
```

So there is indeed no evidence of different period and duration effects.

We might from a purely technical point of view contemplate a model where the difference in age-specific mortality between men and women were either constant or exponentially increasing or decreasing by age. And we might even accept a model of that sort by a statistical test, but given the different biology of men and women over their life span, it would make little sense. And therefore we have not done it here.

We can now extract the parameters from the model. Note that the sequence (and hence meaning) of the parameters depend on how the model is specified. The age-specific rates for men and women at the reference time and reference duration will need parameters extracted by the following subset-argument to `ci.exp`:

```
> ci.exp( m2, subset=c("Int","sexM","P","dur") )
```

```
                          exp(Est.)        2.5%       97.5%
  (Intercept)             0.04119274  0.03438381  0.04935001
  sexM:Ns(A, kn = kn.A)1  1.98512038  1.38417401  2.84697075
  sexM:Ns(A, kn = kn.A)2  3.40484122  2.45876781  4.71494042
  sexM:Ns(A, kn = kn.A)3  4.30809709  2.98415003  6.21942609
  sexM:Ns(A, kn = kn.A)4  6.94917213  4.92438942  9.80649360
  sexM:Ns(A, kn = kn.A)5  8.47754594  5.86563417 12.25251749
  sexM:Ns(A, kn = kn.A)6  6.57840684  4.44759052  9.73008561
  sexM:Ns(A, kn = kn.A)7 10.97880122  7.89479441 15.26753831
  sexM:Ns(A, kn = kn.A)8 23.24296856 17.93627190 30.11972557
  sexM:Ns(A, kn = kn.A)9 11.67404685  9.13826544 14.91348339
  Ns(P, kn = kn.P)1       0.73073343  0.61936177  0.86213158
  Ns(P, kn = kn.P)2       0.68819843  0.53300611  0.88857722
  Ns(P, kn = kn.P)3       0.69848011  0.60798995  0.80243837
  Ns(dur, kn = kn.dur)1   0.50506934  0.41989411  0.60752230
  Ns(dur, kn = kn.dur)2   0.74203629  0.62822061  0.87647214
  Ns(dur, kn = kn.dur)3   0.32632124  0.23932256  0.44494573
  Ns(dur, kn = kn.dur)4   0.99090716  0.84957330  1.15575313
> ci.exp( m2, subset=c("Int","sexF","P","dur") )
                          exp(Est.)        2.5%       97.5%
  (Intercept)             0.04119274  0.03438381  0.04935001
  sexF                    0.66279727  0.52988985  0.82904063
  sexF:Ns(A, kn = kn.A)1  2.39621511  1.49756936  3.83411082
  sexF:Ns(A, kn = kn.A)2  3.18475083  2.13514511  4.75032719
  sexF:Ns(A, kn = kn.A)3  4.72130255  3.03591495  7.34233276
  sexF:Ns(A, kn = kn.A)4  5.04812724  3.33296803  7.64591450
  sexF:Ns(A, kn = kn.A)5  6.56708550  4.38443826  9.83629131
  sexF:Ns(A, kn = kn.A)6  8.39810642  5.83749263 12.08193243
  sexF:Ns(A, kn = kn.A)7 10.70794349  7.91116788 14.49344214
  sexF:Ns(A, kn = kn.A)8 27.59059972 21.11490857 36.05230826
  sexF:Ns(A, kn = kn.A)9 14.86040520 11.52591094 19.15958261
  Ns(P, kn = kn.P)1       0.73073343  0.61936177  0.86213158
  Ns(P, kn = kn.P)2       0.68819843  0.53300611  0.88857722
  Ns(P, kn = kn.P)3       0.69848011  0.60798995  0.80243837
  Ns(dur, kn = kn.dur)1   0.50506934  0.41989411  0.60752230
  Ns(dur, kn = kn.dur)2   0.74203629  0.62822061  0.87647214
  Ns(dur, kn = kn.dur)3   0.32632124  0.23932256  0.44494573
  Ns(dur, kn = kn.dur)4   0.99090716  0.84957330  1.15575313
```

Note that the two subsets of parameters have different length; the parameters for the women (sex="F") has one more column:

```
> mi.A <- ci.exp( m2, subset=c("Int","sexM","P","dur"), ctr.mat=cbind(1  ,AC,PR,dR) ) * 1000
> fi.A <- ci.exp( m2, subset=c("Int","sexF","P","dur"), ctr.mat=cbind(1,1,AC,PR,dR) ) * 1000
> b.P  <- ci.exp( m2, subset="P"  , ctr.mat=PC-PR )
> b.d  <- ci.exp( m2, subset="dur", ctr.mat=dC-dR )


> par( mfrow=c(1,3), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( pr.A, cbind(m.A,f.A,mi.A,fi.A),
+          type="l", lty=rep(c(3,1),each=6), lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(0.2,180),
+          xlab="Age", ylab="Mortality rate per 1000 PY" )
> matplot( pr.P,  cbind(m.P,f.P,b.P),
+          type="l", lty=rep(c(3,1),c(6,3)), lwd=c(3,1,1), col=rep(c("blue","red","black"),each=3),
+          log="y", ylim=c(1/3,3),
+          xlab="Date of follow-up", ylab="Mortality rate ratio" )
> abline( h=1 )
> matplot( pr.d,  cbind(m.d,f.d,b.d),
+          type="l", lty=rep(c(3,1),c(6,3)), lwd=c(3,1,1), col=rep(c("blue","red","black"),each=3),
+          log="y", ylim=c(1/3,3),
+          xlab="Diabetes duration", ylab="Mortality rate ratio" )
> abline( h=1 )
```

We shall return to the set-up with separate effects for men and women.

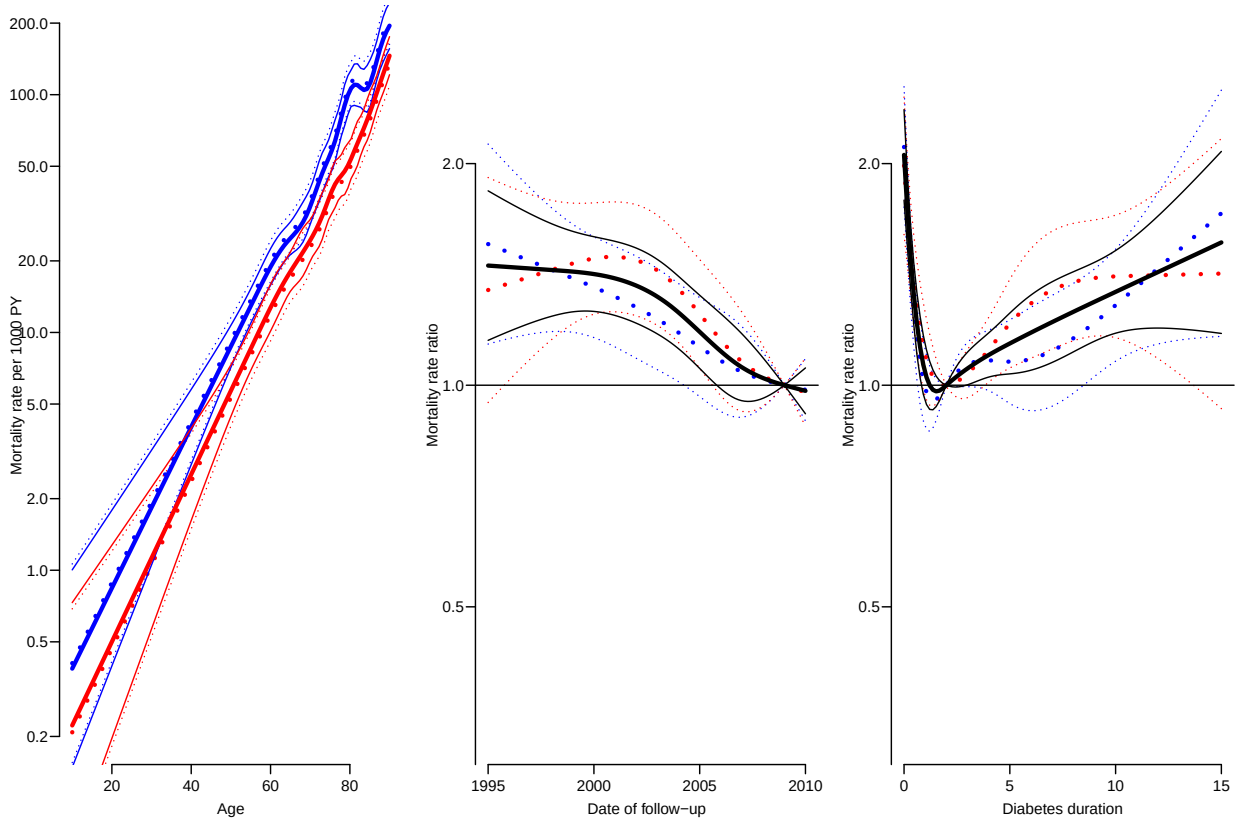


Figure 9.7: *Estimates from models for mortality of Danish diabetes patients. The broken lines are from the full interaction model, full lines with common effects of date and duration. Men:blue, women:red, both (i.e. common): black.*

9.5 Accounting for multiple time scales

The model we fitted has three time-scales: current age, current date and current duration of diabetes, so the effects that we report are not immediately interpretable, as they are (as in all multiple regression) to be interpreted as "all else equal" which they are not, as the three time scales advance by the same pace.

The reporting would therefore more naturally be *only* on the mortality scale, but showing the mortality for persons diagnosed in different ages, using separate displays for separate years of diagnosis.

Incidentally, this is most easily done using the `ci.pred` function with the `newdata=` argument. So a person diagnosed in age 50 will have a (log-)mortality measure in cases per 1000 PY as:

```
> pts <- seq(0,20,1)
> nd <- data.frame( A=  50+pts,
+                   P=1995+pts,
+                 dur=     pts,
+             lex.dur=1000 )
> ci.pred( mm, newdata=nd )
     Estimate     2.5%     97.5%
1  30.26381 21.77308  42.06563
2  14.97557 11.23766  19.95680
3  15.91802 12.50610  20.26078
4  17.87916 14.37698  22.23446
```

```
  5  19.06408 15.79295  23.01274
  6  19.96428 16.70839  23.85462
  7  21.13109 17.51102  25.49955
  8  22.67632 18.65497  27.56451
  9  24.53447 20.21171  29.78177
  10 26.61244 21.98992  32.20667
  11 28.86802 23.55884  35.37366
  12 31.33564 24.82984  39.54605
  13 34.10668 26.23335  44.34300
  14 37.28204 27.88749  49.84135
  15 40.74629 29.14142  56.97253
  16 44.39407 29.59358  66.59666
  17 48.33981 29.52353  79.14830
  18 52.88668 29.36325  95.25518
  19 58.45978 29.45660 116.01970
  20 65.65161 30.04833 143.44007
  21 75.17049 31.21620 181.01508
```

We can wrap this so that we get the predicted rates with confidence intervals: This can be nicely wrapped in a function that takes age and date of diagnosis as input and returns the estimated mortality rates for a male and a female diagnosed this age and date:

```
> DMm <-
+ function( A, P, pts=seq(0,25,0.1) )
+ {
+ nd <- data.frame( A=A+pts,
+                   P=P+pts,
+                 dur=  pts,
+               lex.dur=1000 )
+ cbind( nd$A, ci.pred( mm, newdata=nd ),
+               ci.pred( mf, newdata=nd ) )
+ }
> DMm( 50, 1996, pts=0:10 )
        Estimate     2.5%    97.5%  Estimate      2.5%    97.5%
  1   50 29.46588 21.97066 39.51808 14.542246 10.213136 20.70637
  2   51 14.58073 11.30879 18.79933  8.920489  6.575060 12.10257
  3   52 15.49833 12.55439 19.13262  8.976858  6.923712 11.63884
  4   53 17.40561 14.31189 21.16809 10.302383  8.080729 13.13484
  5   54 18.54146 15.54911 22.10967 12.222651  9.792516 15.25585
  6   55 19.37755 16.27207 23.07570 14.458964 11.629029 17.97757
  7   56 20.44634 16.97085 24.63358 16.526631 13.154646 20.76297
  8   57 21.84970 18.08634 26.39612 18.101043 14.344029 22.84210
  9   58 23.51919 19.59123 28.23469 18.981532 15.077545 23.89637
  10 59 25.43706 21.14450 30.60105 19.249329 15.204291 24.37053
  11 60 27.65893 22.55955 33.91098 19.272483 14.924008 24.88799
```

With this in place we can now plot the mortality rates for persons diagnosed at different ages and different dates:

```
> DMm.1996 <-
+ rbind(
+ DMm( 30, 1996 ), NA,
+ DMm( 40, 1996 ), NA,
+ DMm( 50, 1996 ), NA,
+ DMm( 60, 1996 ), NA,
+ DMm( 70, 1996 ), NA,
+ DMm( 80, 1996 ), NA,
+ DMm( 90, 1996 ) )
> DMm.2005 <-
+ rbind(
+ DMm( 30, 2005 ), NA,
+ DMm( 40, 2005 ), NA,
+ DMm( 50, 2005 ), NA,
+ DMm( 60, 2005 ), NA,
```

```
+ DMm( 70, 2005 ), NA,
+ DMm( 80, 2005 ), NA,
+ DMm( 90, 2005 ) )
> par( mfrow=c(1,2), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( DMm.1996[,1], DMm.1996[,-1],
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(1,1000), xlim=c(30,95), las=1,
+          xlab="Age", ylab="Mortality rate per 1000 PY" )
> text( 30, 1000, "DM diagnosed 1996", adj=c(0,1) )
> matplot( DMm.2005[,1], DMm.2005[,-1],
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(1,1000), xlim=c(30,95), las=1,
+          xlab="Age", ylab="Mortality rate per 1000 PY" )
> text( 30, 1000, "DM diagnosed 2005", adj=c(0,1) )
```

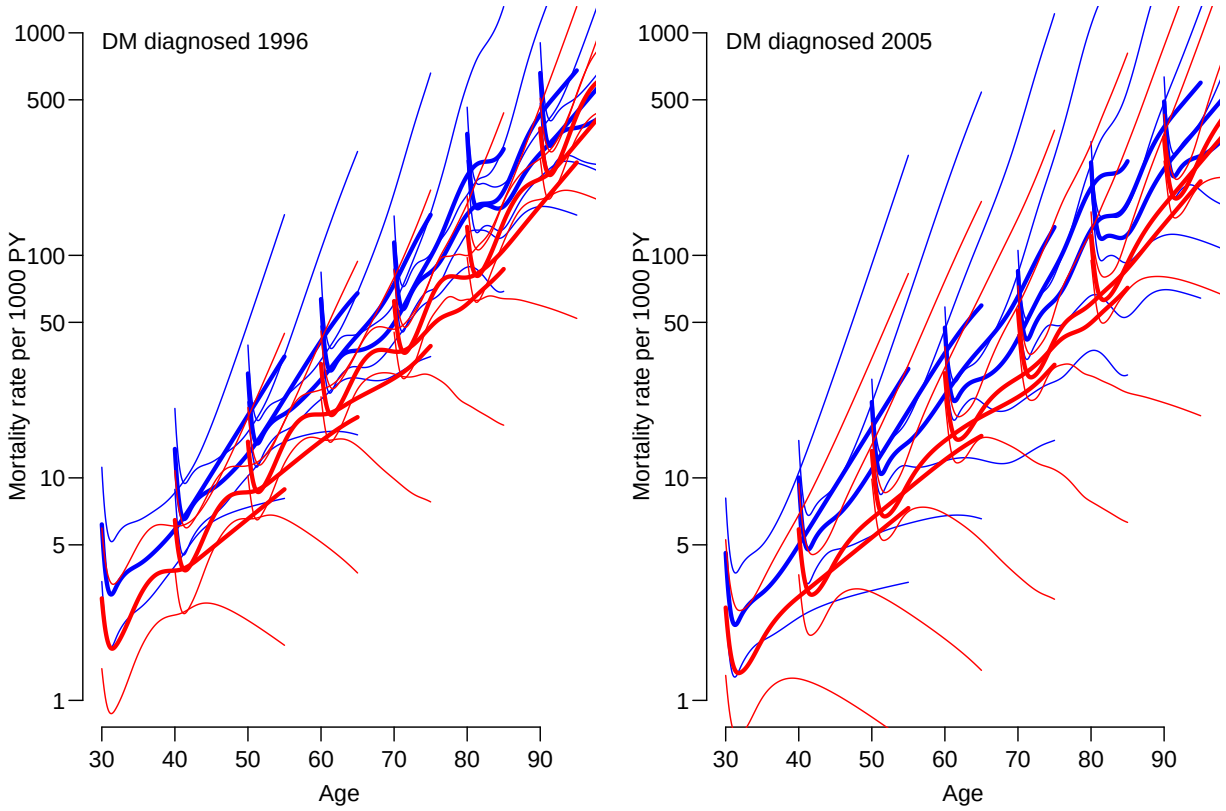


Figure 9.8: *Estimates of mortality of Danish diabetes patients for patients diagnosed in ages 30, 40, . . . , 90.*

Note from figure 9.8 that it seems that mortality among men is higher the younger age at diagnosis, but not for women. But also note that we predicted from 0 to 25 years of diabetes duration, which is a bit bold, given that we only have 15 years of observation, and thus no one with diabetes duration longer than that. Also the rightmost boundary knot for the duration effect is at 10 years, so we are effectively assuming that the duration effect is (log-)linear beyond this — for 15 years, out which we have data for the first 5!

The model we used for the mortality rates used three time-scales: age, calendar time and duration of diabetes.

It would be of interest to see whether we would get the same (or better) description by adding age at diagnosis and date of diagnosis to the model.

Now, age at diagnosis = current age − duration of diabetes, and date of diagnosis = current date − duration of diabetes, so the terms we might add only constitute the *non-linear* effects of these variables.

We add the effects one at at time and test whether age at diagnosis or current age is the better predictor, but we want to use a set of knots which is aligned to the new variables we consider:

```
> kn.Ad <- with( subset( SL, lex.Xst=="Dead" ),
+                quantile( A-dur, probs=seq(5,95,10)/100 ) )
> kn.Pd <- with( subset( SL, lex.Xst=="Dead" ),
+                quantile( P-dur, probs=seq(5,95,20)/100 ) )
```

We can now make on-the-fly tests of the non-linear effects of these fixed effects using `anova`:

```
> anova( mm,
+        update( mm, . ~ . + Ns(A-dur,knots=kn.Ad) ),
+        update( mm, . ~ . + Ns(A-dur,knots=kn.Ad) - Ns(A,knots=kn.A) ),
+        test = "Chisq" )
  Analysis of Deviance Table

  Model 1: (lex.Xst == "Dead") ~ Ns(A, kn = kn.A) + Ns(P, kn = kn.P) + Ns(dur,
      kn = kn.dur)
  Model 2: (lex.Xst == "Dead") ~ Ns(A, kn = kn.A) + Ns(P, kn = kn.P) + Ns(dur,
      kn = kn.dur) + Ns(A - dur, knots = kn.Ad)
  Model 3: (lex.Xst == "Dead") ~ Ns(P, kn = kn.P) + Ns(dur, kn = kn.dur) +
      Ns(A - dur, knots = kn.Ad)
    Resid. Df Resid. Dev Df Deviance Pr(>Chi)
  1     60330      11719
  2     60322      11710  8   9.2579   0.3210
  3     60330      11720 -8 -10.1280   0.2562
> anova( mm,
+        update( mm, . ~ . + Ns(P-dur,knots=kn.Pd) ),
+        update( mm, . ~ . + Ns(P-dur,knots=kn.Pd) - Ns(P,knots=kn.P) ),
+        test = "Chisq" )
  Analysis of Deviance Table

  Model 1: (lex.Xst == "Dead") ~ Ns(A, kn = kn.A) + Ns(P, kn = kn.P) + Ns(dur,
      kn = kn.dur)
  Model 2: (lex.Xst == "Dead") ~ Ns(A, kn = kn.A) + Ns(P, kn = kn.P) + Ns(dur,
      kn = kn.dur) + Ns(P - dur, knots = kn.Pd)
  Model 3: (lex.Xst == "Dead") ~ Ns(A, kn = kn.A) + Ns(dur, kn = kn.dur) +
      Ns(P - dur, knots = kn.Pd)
    Resid. Df Resid. Dev Df Deviance Pr(>Chi)
  1     60330      11719
  2     60327      11714  3   4.3186    0.229
  3     60329      11715 -2  -0.6486    0.723
> anova( mf,
+        update( mf, . ~ . + Ns(A-dur,knots=kn.Ad) ),
+        update( mf, . ~ . + Ns(A-dur,knots=kn.Ad) - Ns(A,knots=kn.A) ),
+        test = "Chisq" )
  Analysis of Deviance Table

  Model 1: (lex.Xst == "Dead") ~ Ns(A, kn = kn.A) + Ns(P, kn = kn.P) + Ns(dur,
      kn = kn.dur)
  Model 2: (lex.Xst == "Dead") ~ Ns(A, kn = kn.A) + Ns(P, kn = kn.P) + Ns(dur,
      kn = kn.dur) + Ns(A - dur, knots = kn.Ad)
  Model 3: (lex.Xst == "Dead") ~ Ns(P, kn = kn.P) + Ns(dur, kn = kn.dur) +
      Ns(A - dur, knots = kn.Ad)
    Resid. Df Resid. Dev Df Deviance Pr(>Chi)
  1     58109      10203
  2     58101      10193  8  10.1731   0.2531
  3     58109      10198 -8  -5.2649   0.7289
```

```
> anova( mf,
+        update( mf, . ~ . + Ns(P-dur,knots=kn.Pd) ),
+        update( mf, . ~ . + Ns(P-dur,knots=kn.Pd) - Ns(P,knots=kn.P) ),
+        test = "Chisq" )
  Analysis of Deviance Table

  Model 1: (lex.Xst == "Dead") ~ Ns(A, kn = kn.A) + Ns(P, kn = kn.P) + Ns(dur,
      kn = kn.dur)
  Model 2: (lex.Xst == "Dead") ~ Ns(A, kn = kn.A) + Ns(P, kn = kn.P) + Ns(dur,
      kn = kn.dur) + Ns(P - dur, knots = kn.Pd)
  Model 3: (lex.Xst == "Dead") ~ Ns(A, kn = kn.A) + Ns(dur, kn = kn.dur) +
      Ns(P - dur, knots = kn.Pd)
    Resid. Df Resid. Dev Df Deviance Pr(>Chi)
  1     58109      10203
  2     58106      10202  3    1.591  0.66142
  3     58108      10208 -2   -5.820  0.05447
```

From this it is pretty clear that there is not much difference between using current age or age at diagnosis, and likewise for date of diagnosis, except possibly for period for women, where it seems more appropriate to use current age (since the p-value for removing this from the model is 0.033). But since the tests concerning the age-effects are insignificant, we could argue that an equally good description of data could be obtained using age at diagnosis and duration of diabetes.

In conclusion, there does not seem to be much need to change the model we fitted.

But we try to fit the models with age at diagnosis and date of diagnosis as explanatory variables instead. To this end we also need new contrast matrices, because the deaths are distributed differently along these "entry"-variables, and we therefor placed the knots differently.

```
> AC <- Ns( pr.A, knots=kn.Ad )
> PC <- Ns( pr.P, knots=kn.Pd )
> PR <- Ns( rep(rf.P,N), knots=kn.Pd )
> Mm <- glm( (lex.Xst=="Dead") ~ Ns( A-dur, kn=kn.Ad ) +
+                                Ns( P-dur, kn=kn.Pd ) +
+                                Ns(   dur, kn=kn.dur ),
+            offset = log( lex.dur ),
+            family = poisson,
+              data = subset( SL, sex=="M" ) )
> Mf <- update( Mm, data = subset( SL, sex=="F" ) )
> M.A <- ci.exp( Mm, ctr.mat=cbind(1,AC,PR,dR) ) * 1000
> F.A <- ci.exp( Mf, ctr.mat=cbind(1,AC,PR,dR) ) * 1000
> M.P <- ci.exp( Mm, subset="P"  , ctr.mat=PC-PR )
> F.P <- ci.exp( Mf, subset="P"  , ctr.mat=PC-PR )
> M.d <- ci.exp( Mm, subset="kn.dur", ctr.mat=dC-dR )
> F.d <- ci.exp( Mf, subset="kn.dur", ctr.mat=dC-dR )
```

Once the models are fitted, we can plot the estimated effects, as seen in figure 9.9

```
> par( mfrow=c(1,3), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( pr.A, cbind(M.A,F.A),
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(0.2,180),
+          xlab="Age at diagnosis", ylab="Mortality rate at 2 years duration per 1000 PY" )
> matplot( pr.P,  cbind(M.P,F.P),
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(1/3,3),
+          xlab="Date of diagnosis", ylab="Mortality rate ratio" )
> abline( h=1 )
> matplot( pr.d,  cbind(M.d,F.d),
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(1/3,3),
+          xlab="Diabetes duration", ylab="Mortality rate ratio" )
> abline( h=1 )
```

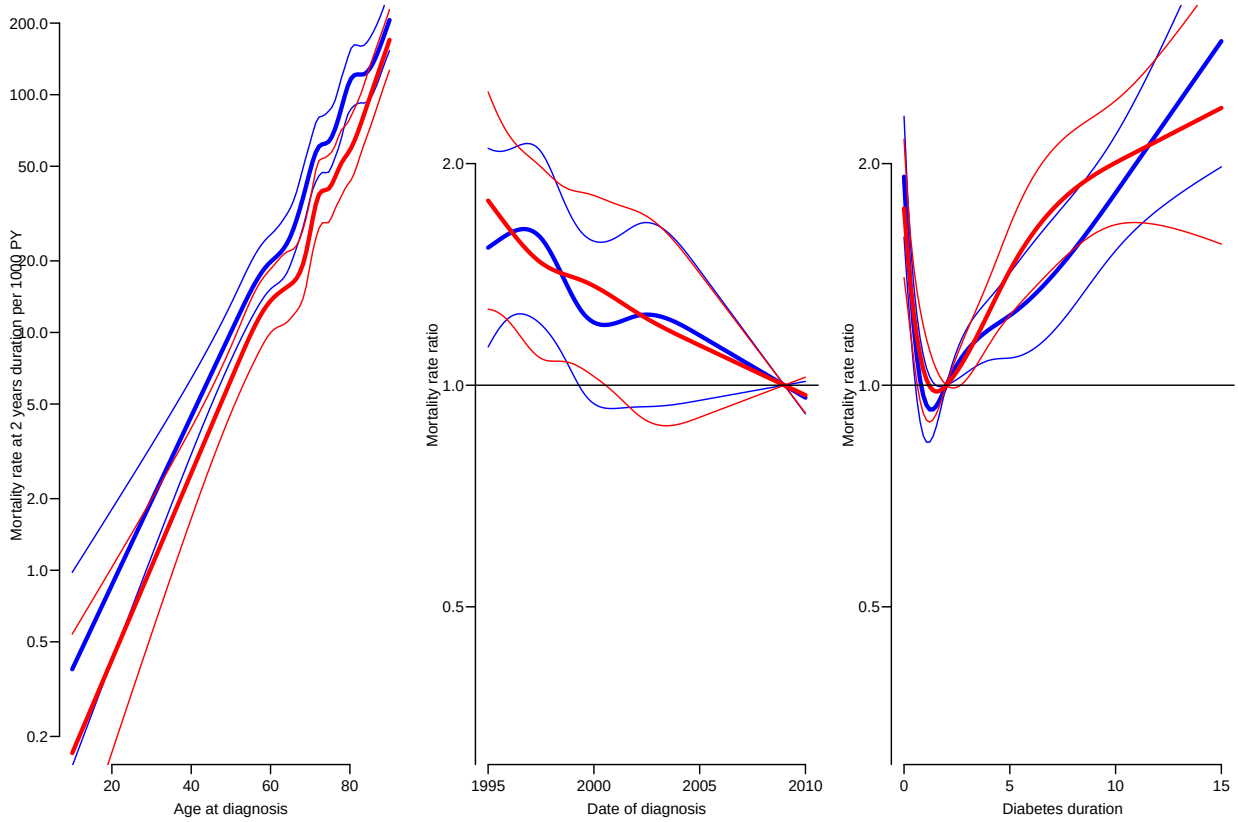


Figure 9.9: *Model for diabetes patient mortality using age and date at diagnosis.*

The effects shown in figure are shown in a slightly counter-intuitive way; the age-effect is the effect of age *at diagnosis*, the period effect is the effect of date *at diagnosis*, and the duration effect is the only time-scale in the model, the effect of time *since* diagnosis.

In order to see how the effects from the two approaches using age/date at diagnosis/follow-up relate to each other we can plot them on top of each other:

```
> par( mfrow=c(1,3), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( pr.A, cbind(M.A,F.A,m.A,f.A),
+          type="l", lty=rep(c(1,2),each=6), lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(0.2,180),
+          xlab="Age at diagnosis/follow-up", ylab="Mortality rate at 2 years duration per 1000 PY"
> matplot( pr.P,  cbind(M.P,F.P,m.P,f.P),
+          type="l", lty=rep(c(1,2),each=6), lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(1/3,3),
+          xlab="Date of diagnosis/follow-up", ylab="Mortality rate ratio" )
> abline( h=1 )
> matplot( pr.d,  cbind(M.d,F.d,m.d,f.d),
+          type="l", lty=rep(c(1,2),each=6), lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(1/3,3),
+          xlab="Diabetes duration", ylab="Mortality rate ratio" )
> abline( h=1 )
```

From figure 9.10 we see that the age and duration curves from the model with two time scales have smaller slopes than those from the model with the age and calendar time as fixed effects. This is because in the latter all the time effect (that is the effect of the clock advancing) is in the duration effect. The *sum* of the average slopes are however the same.

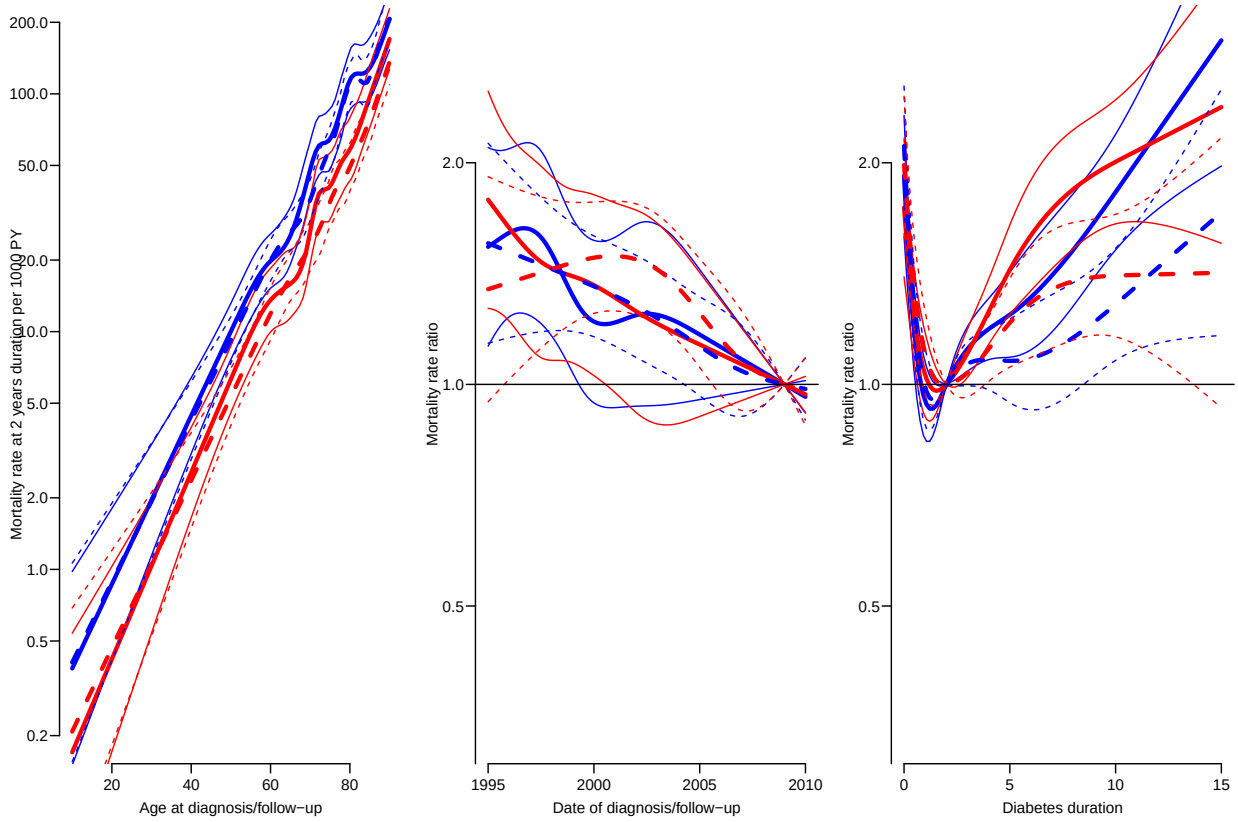


Figure 9.10: *Comparison of estimates from two different models; the full lines give the estimates from the model where age and date are included as fixed variables with the value at diabetes diagnosis, whereas the broken lines are estimates from the model where age and calendar time are included as time scales.*

9.6 SMR

The SMR is the standardized mortality ratio, which is mortality rate-ratio between the diabetes patients and the general population. In real studies we would subtract the deaths and the person-years among the diabetes patients from those of the general population, but since we do not have access to these (recall that we only have a random sample of 10,000 diabetes patients), we make the comparison to the general population at large, *i.e.* also including the diabetes patients.

There are two ways to make the comparison to the population mortality; one is toe amend the diabetes patient dataset with the population mortality dataset, the other (classical) one is to include the population mortality rates as a fixed variable in the calculations.

The latter requires that each analytical unit in the diabetes patient dataset is amended with a variable with the population mortality for the corresponding sex, age and calendar time.

This can be achieved in two ways: Either we just use the current split of follow-up time and allocate the population mortality rates for some suitably chosen (mid-)point of the follow-up in each, or we make a second split by date, so that follow-up in the diabetes patients is in the same classification of age and data as the population mortality table.

We will use the second approach, that is include as an extra variable the population mortality as available from the data set `M.dk`.

First we create the variables in the diabetes dataset that we need for matching with the population mortality data, that is age, date and sex at the midpoint of each of the intervals (or rater at a point 3 months after the left end point of the interval — recall we split the follow-up in 6 month intervals).

We need to have variables with the same names in both datasets, moreover, they should be of the same type, so we must transform the sex variable in `M.dk` to a factor:

```
> str( SL )
  Classes âLexisâ and 'data.frame':        118473 obs. of  14 variables:
   $ lex.id : int  1 1 1 1 1 1 1 1 1 1 ...
   $ A      : num  58.7 59 59.5 60 60.5 ...
   $ P      : num  1999 1999 2000 2000 2001 ...
   $ dur    : num  0 0.339 0.839 1.339 1.839 ...
   $ lex.dur: num  0.339 0.5 0.5 0.5 0.5 ...
   $ lex.Cst: Factor w/ 2 levels "Alive","Dead": 1 1 1 1 1 1 1 1 1 1 ...
   $ lex.Xst: Factor w/ 2 levels "Alive","Dead": 1 1 1 1 1 1 1 1 1 1 ...
   $ sex    : Factor w/ 2 levels "M","F": 2 2 2 2 2 2 2 2 2 2 ...
   $ dobth  : num  1940 1940 1940 1940 1940 ...
   $ dodm   : num  1999 1999 1999 1999 1999 ...
   $ dodth  : num  NA NA NA NA NA NA NA NA NA NA ...
   $ dooad  : num  NA NA NA NA NA NA NA NA NA NA ...
   $ doins  : num  NA NA NA NA NA NA NA NA NA NA ...
   $ dox    : num  2010 2010 2010 2010 2010 ...
   - attr(*, "breaks")=List of 3
    ..$ A  : num  0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 ...
    ..$ P  : NULL
    ..$ dur: NULL
   - attr(*, "time.scales")= chr  "A" "P" "dur"
   - attr(*, "time.since")= chr  "" "" ""

> SL$Am <- floor( SL$A+0.5 )
> SL$Pm <- floor( SL$P+0.5 )
> data( M.dk )
> str( M.dk )
```

```
  'data.frame':        7800 obs. of  6 variables:
   $ A   : num  0 0 0 0 0 0 0 0 0 0 ...
   $ sex : num  1 2 1 2 1 2 1 2 1 2 ...
   $ P   : num  1974 1974 1975 1975 1976 ...
   $ D   : num  459 303 435 311 405 258 332 205 312 233 ...
   $ Y   : num  35963 34383 36099 34652 34965 ...
   $ rate: num  12.76 8.81 12.05 8.97 11.58 ...
   - attr(*, "Contents")= chr "Number of deaths and risk time in Denmark"
> M.dk <- transform( M.dk, Am = A,
+                          Pm = P,
+                         sex = factor( sex, labels=c("M","F") ) )
> str( M.dk )
  'data.frame':        7800 obs. of  8 variables:
   $ A   : num  0 0 0 0 0 0 0 0 0 0 ...
   $ sex : Factor w/ 2 levels "M","F": 1 2 1 2 1 2 1 2 1 2 ...
   $ P   : num  1974 1974 1975 1975 1976 ...
   $ D   : num  459 303 435 311 405 258 332 205 312 233 ...
   $ Y   : num  35963 34383 36099 34652 34965 ...
   $ rate: num  12.76 8.81 12.05 8.97 11.58 ...
   $ Am  : num  0 0 0 0 0 0 0 0 0 0 ...
   $ Pm  : num  1974 1974 1975 1975 1976 ...
```

Then we can match up the rates from `M.dk`:

```
> SLr <- merge( SL, M.dk[,c("Am","Pm","sex","rate")] )
> dim( SL )
  [1] 118473     16
> dim( SLr )
  [1] 118448     17
```

This merge only takes rows that have information from both datasets, hence the slightly fewer rows in `SLr` than in SL. There is no point in including observations where there is no risk time among the diabetes patients; the computed expected numbers will be 0, and hence crash the analysis.

We can now compute the SMR as the observed divided by the expected numbers by say age and sex:

```
> stat.table( list( Age=floor(A/10)*10,
+                   Sex=sex ),
+             list( D=sum(lex.Xst=="Dead"),
+                   E=sum(lex.dur*rate/1000),
+                 SMR=ratio(lex.Xst=="Dead",lex.dur*rate/1000) ),
+             margins = TRUE,
+                data = SLr )
   ---------------------------------
          ----------Sex-----------
   Age           M       F   Total
   ---------------------------------
   0          0.00    0.00    0.00
              0.02    0.01    0.02
              0.00    0.00    0.00

   10         1.00    1.00    2.00
              0.14    0.04    0.18
              7.27   23.98   11.15

   20         0.00    0.00    0.00
              0.36    0.18    0.54
              0.00    0.00    0.00

   30         5.00    4.00    9.00
```

```
            1.49    1.06    2.55
            3.37    3.77    3.53

40         32.00   15.00   47.00
            9.91    5.24   15.16
            3.23    2.86    3.10

50        119.00   62.00  181.00
           50.49   22.92   73.41
            2.36    2.71    2.47

60        275.00  157.00  432.00
          148.12   77.20  225.32
            1.86    2.03    1.92

70        486.00  331.00  817.00
          288.07  214.03  502.10
            1.69    1.55    1.63

80        348.00  423.00  771.00
          266.89  336.16  603.05
            1.30    1.26    1.28

90         76.00  159.00  235.00
           65.20  127.34  192.54
            1.17    1.25    1.22


Total    1342.00 1152.00 2494.00
          830.68  784.18 1614.86
            1.62    1.47    1.54
---------------------------------
```

We see that the overall SMR is 1.6, but strongly varying with age and to some extent by sex. Moreover, it may seem that the variation with age is not the same for the two sexes.

We can now model the SMR by including the log-expected numbers instead of the log-person-years as offset, using separate models for men and women. Also note that we exclude those units where no deaths in the population occur. Also we compute the expected numbers, `E`:

```
> SLr <- subset( SLr, rate>0)
> SLr$E <- SLr$lex.dur * SLr$rate / 1000
> Sm <- glm( (lex.Xst=="Dead") ~ Ns( A-dur, kn=kn.Ad ) +
+                                Ns( P-dur, kn=kn.Pd ) +
+                                Ns( dur, kn=kn.dur ),
+            offset = log( E ),
+            family = poisson,
+              data = subset( SLr, sex=="M" ) )
> Sf <- update( Sm, data = subset( SLr, sex=="F" ) )
```

The estimates are extracted exactly as for the mortality model; but the results are not mortality rates but rather SMRs (rate-ratios):

```
> sM.A <- ci.exp( Sm, ctr.mat=cbind(1,AC,PR,dR) )
> sF.A <- ci.exp( Sf, ctr.mat=cbind(1,AC,PR,dR) )
> sM.P <- ci.exp( Sm, subset="P"     , ctr.mat=PC-PR )
> sF.P <- ci.exp( Sf, subset="P"     , ctr.mat=PC-PR )
> sM.d <- ci.exp( Sm, subset="kn.dur", ctr.mat=dC-dR )
> sF.d <- ci.exp( Sf, subset="kn.dur", ctr.mat=dC-dR )
```

— plotted using the same code (with obvious adjustments of the axes:

```
> par( mfrow=c(1,3), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( pr.A, cbind(sM.A,sF.A),
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(1/3,3),
+          xlab="Age at follow-up", ylab="SMR" )
> abline( h=1 )
> matplot( pr.P,  cbind(sM.P,sF.P),
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(1/3,3),
+          xlab="Date of follow-up", ylab="SMR ratio" )
> abline( h=1 )
> matplot( pr.d,  cbind(sM.d,sF.d),
+          type="l", lty=1, lwd=c(3,1,1), col=rep(c("blue","red"),each=3),
+          log="y", ylim=c(1/3,3),
+          xlab="Diabetes duration", ylab="SMR ratio" )
> abline( h=1 )
```

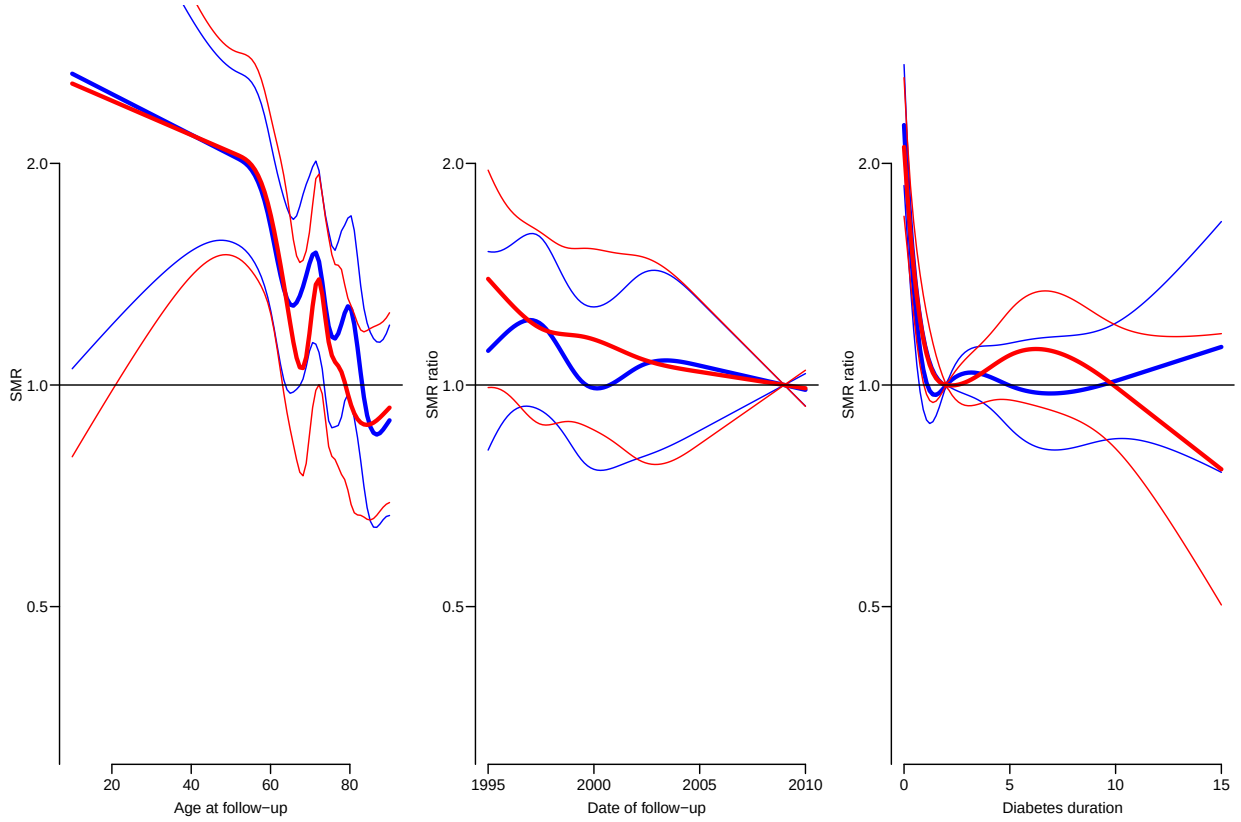


Figure 9.11: *SMR in the diabetic population relative to the (entire) Danish population. Clearly the effect of age is over-modeled.*

It seems reasonably from figure 9.11 clear that there is very little difference between SMR for males and females once we controlled for age, date and duration of diabetes. This can be formally tested by fitting models with and without sex-interaction and also a model with no overall effect of sex:

```
> Sb   <- update( Sm, data = SLr )
> Sb.s <- update( Sb, . ~. + sex )
> Sb.i <- update( Sb, . ~. + sex:( Ns( A-dur, kn=kn.Ad  ) +
+                                  Ns( P-dur, kn=kn.Pd  ) +
+                                  Ns(   dur, kn=kn.dur ) ) )
> anova( Sb, Sb.s, Sb.i, test="Chisq" )
```

```
Analysis of Deviance Table

Model 1: (lex.Xst == "Dead") ~ Ns(A - dur, kn = kn.Ad) + Ns(P - dur, kn = kn.Pd) +
    Ns(dur, kn = kn.dur)
Model 2: (lex.Xst == "Dead") ~ Ns(A - dur, kn = kn.Ad) + Ns(P - dur, kn = kn.Pd) +
    Ns(dur, kn = kn.dur) + sex
Model 3: (lex.Xst == "Dead") ~ Ns(A - dur, kn = kn.Ad) + Ns(P - dur, kn = kn.Pd) +
    Ns(dur, kn = kn.dur) + Ns(A - dur, kn = kn.Ad):sex + Ns(P -
    dur, kn = kn.Pd):sex + Ns(dur, kn = kn.dur):sex
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1    118403      21916
2    118402      21916  1   0.0122   0.9121
3    118386      21903 16  13.2846   0.6518
```

So we see there is absolutely no difference between the SMR between the sexes.

We therefore extract the parameters from the model with common SMR for the two sexes.

```
> Sb.A <- ci.exp( Sb, ctr.mat=cbind(1,AC,PR,dR) )
> Sb.P <- ci.exp( Sb, subset="P"     , ctr.mat=PC-PR )
> Sb.d <- ci.exp( Sb, subset="kn.dur", ctr.mat=dC-dR )
> par( mfrow=c(1,3), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( pr.A, Sb.A,
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          log="y", ylim=c(1/3,3),
+          xlab="Age at diagnosis", ylab="SMR" )
> abline( h=1 )
> matplot( pr.P, Sb.P,
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          log="y", ylim=c(1/3,3),
+          xlab="Date of diagnosis", ylab="SMR ratio" )
> abline( h=1 )
> matplot( pr.d, Sb.d,
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          log="y", ylim=c(1/3,3),
+          xlab="Diabetes duration", ylab="SMR ratio" )
> abline( h=1 )
```

We can simplify the model to one the is easier to convey to users by using a linear effect of date of diagnosis, and using only knots at 0,1,and 2 years for duration, giving an estimate of the change in SMR as duration increases beyond 2 years. At the same time we also limit the number of knots for the age-effect:

```
> kn.Ad <- with( subset( SL, lex.Xst=="Dead" ),
+                quantile( A-dur, probs=seq(5,95,20)/100 ) )
> kn.dur <- 0:2
> AC <- Ns( pr.A, knots=kn.Ad )
> dC <- Ns( pr.d, knots=kn.dur )
> dR <- Ns( rep(rf.d,N), knots=kn.dur )
> Sx <- glm( (lex.Xst=="Dead") ~ Ns( A-dur, kn=kn.Ad ) +
+                                  I( P-dur ) +
+                                 Ns(   dur, kn=kn.dur ),
+            offset = log( E ),
+            family = poisson,
+              data = SLr )
```

Having fitted the model, we can then plot the estimates from it:

```
> Sx.A <- ci.exp( Sx, ctr.mat=cbind(1,AC,rf.P,dR) )
> Sx.P <- ci.exp( Sx, subset="P"     , ctr.mat=cbind(pr.P-rf.P) )
> Sx.d <- ci.exp( Sx, subset="kn.dur", ctr.mat=dC-dR )
> par( mfrow=c(1,3), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( pr.A, Sx.A,
```

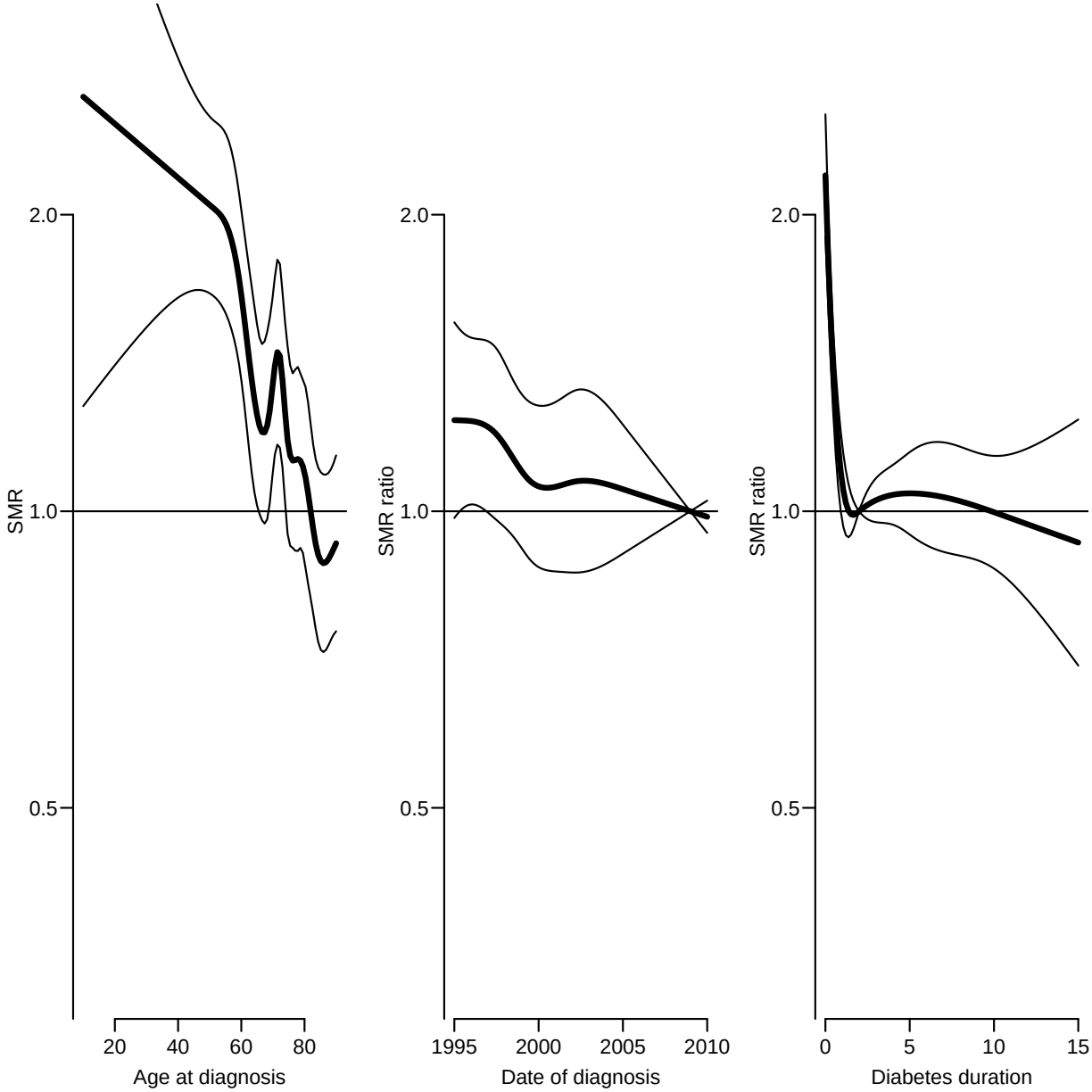


Figure 9.12: *SMR in the diabetic population for both sexes, relative to the (entire) Danish population.*

```
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          log="y", ylim=c(1/2,4),
+          xlab="Age at diagnosis", ylab="SMR" )
> abline( h=1 )
> abline( v=4:8*10, col="gray" )
> matplot( pr.P, Sx.P,
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          log="y", ylim=c(1/2,4),
+          xlab="Date of diagnosis", ylab="SMR ratio" )
> abline( h=1 )
> matplot( pr.d, Sx.d,
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          log="y", ylim=c(1/2,4),
+          xlab="Diabetes duration", ylab="SMR ratio" )
```

```
> abline( h=1,v=2 )
```

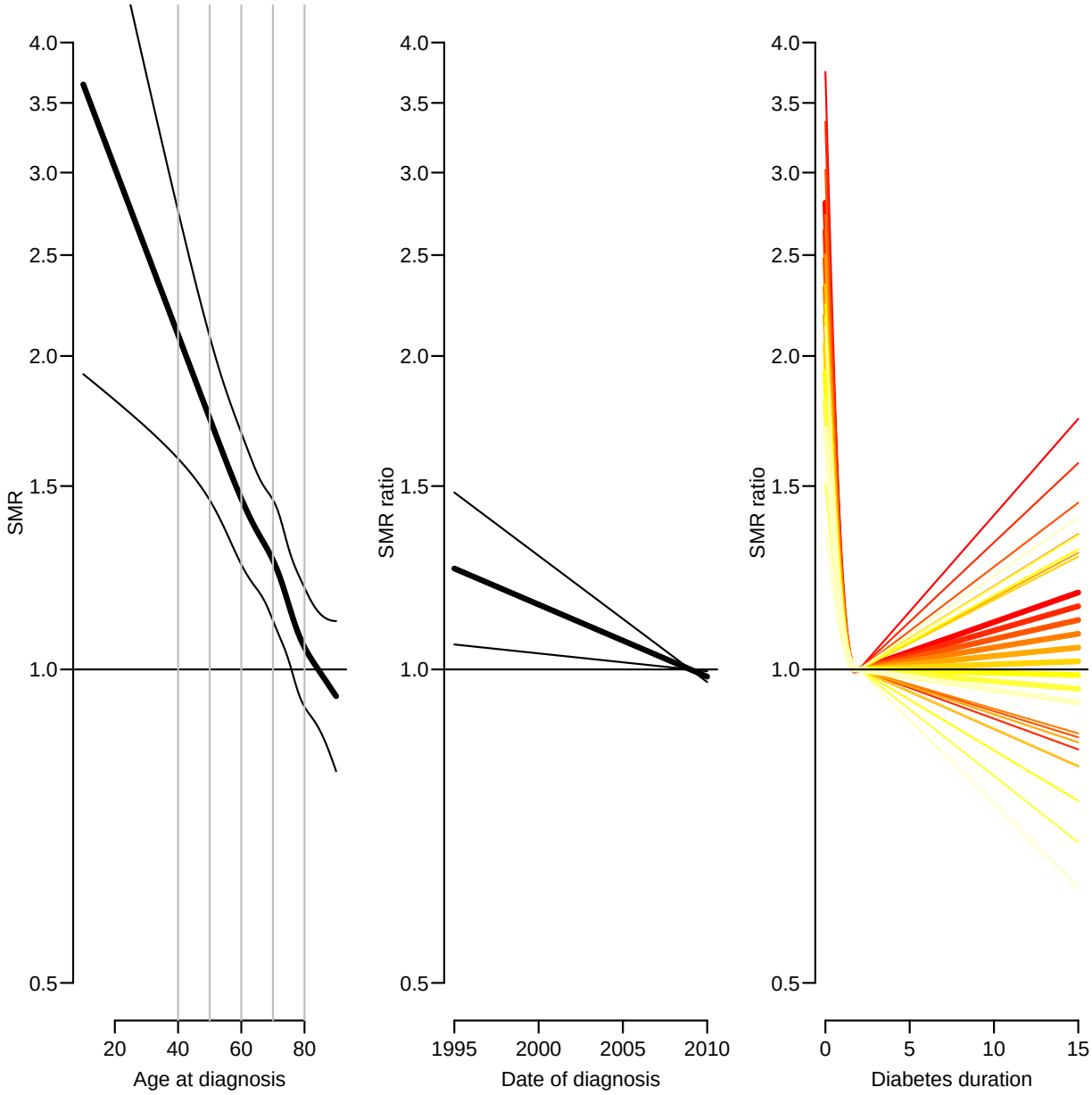


Figure 9.13: *SMR in the diabetic population for both sexes, relative to the (entire) Danish population — simplified model.*

We can formulate the period and duration effects by looking at the estimated parameters:

```
> 100*( 1 - ci.exp( Sx, subset="P" ) )
             exp(Est.)     2.5%   97.5%
I(P - dur)   1.59215 2.766214 0.40391
```

If we want to assess the annual change in SMR by duration of diabetes we can calculate the duration effects at say 5 and 6 years and subtract them:

```
> d6 <- Ns( 6, knots=kn.dur )
> d5 <- Ns( 5, knots=kn.dur )
> 100*( ci.exp( Sx, subset="kn.dur", ctr.mat=d6-d5 ) - 1 )
       exp(Est.)      2.5%    97.5%
  [1,] 0.2222163 -1.365779 1.835778
```

Thus the estimate is an annual increase in SMR of 0.3% (-1.3–1.9)%, thus no evidence of any increasing SMR after 2 years of diabetes duration.

The conclusion is that SMR for diabetes patients diagnosed at age 50 is about 2 after two years of duration and does not change, whereas it for patients aged 70 is about 1.4 after 2 years of diabetes and does not change. The SMR is initially (just after diagnosis) about twice as high, and does not change.

9.6.1 Interaction models

We may explore whether there is an interaction between age and duration by including a product of the (linear) duration effects and age at diagnosis:

```
> Slx <- update( Sx, . ~. + I(A-dur):Ns(dur,knots=kn.dur) )
> anova( Slx, Sx, test="Chisq" )
  Analysis of Deviance Table

  Model 1: (lex.Xst == "Dead") ~ Ns(A - dur, kn = kn.Ad) + I(P - dur) +
      Ns(dur, kn = kn.dur) + Ns(dur, kn = kn.dur):I(A - dur)
  Model 2: (lex.Xst == "Dead") ~ Ns(A - dur, kn = kn.Ad) + I(P - dur) +
      Ns(dur, kn = kn.dur)
    Resid. Df Resid. Dev Df Deviance Pr(>Chi)
  1    118411      21936
  2    118413      21941 -2  -4.9395  0.08461
> ci.exp( Slx )
                                      exp(Est.)         2.5%        97.5%
  (Intercept)                      3.820872e+14 1.393779e+04 1.047444e+25
  Ns(A - dur, kn = kn.Ad)1         5.796186e-01 4.542913e-01 7.395205e-01
  Ns(A - dur, kn = kn.Ad)2         5.016243e-01 3.968090e-01 6.341260e-01
  Ns(A - dur, kn = kn.Ad)3         3.060510e-01 2.100009e-01 4.460324e-01
  Ns(A - dur, kn = kn.Ad)4         4.835756e-01 3.826729e-01 6.110842e-01
  I(P - dur)                       9.841822e-01 9.724406e-01 9.960657e-01
  Ns(dur, kn = kn.dur)1            5.988204e-02 1.451733e-02 2.470054e-01
  Ns(dur, kn = kn.dur)2            4.090076e-01 2.651628e-01 6.308847e-01
  Ns(dur, kn = kn.dur)1:I(A - dur) 1.021670e+00 1.002343e+00 1.041369e+00
  Ns(dur, kn = kn.dur)2:I(A - dur) 1.006681e+00 1.000835e+00 1.012562e+00
```

Even if the effect is not statistically significant, we would still want to explore the shape of it:

```
> Slx.A <- ci.exp( Slx, ctr.mat=cbind(1,AC,rf.P,dR,dR*pr.A) )
> Slx.P <- ci.exp( Slx, subset="P"     , ctr.mat=cbind(pr.P-rf.P) )
> Slx.d <- ci.exp( Slx, subset="kn.dur", ctr.mat=cbind(dC-dR,(dC-dR)*50) )
> for( a in seq(55,90,5) ) Slx.d <- cbind( Slx.d,
+                  ci.exp( Slx, subset="kn.dur", ctr.mat=cbind(dC-dR,(dC-dR)*a) ) )
> dim( Slx.d )
  [1] 100  27
> par( mfrow=c(1,3), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( pr.A, Slx.A,
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          log="y", ylim=c(1/2,4),
+          xlab="Age at diagnosis", ylab="SMR" )
> abline( h=1 )
> abline( v=4:8*10, col="gray" )
```

```
> matplot( pr.P, Slx.P,
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          log="y", ylim=c(1/2,4),
+          xlab="Date of diagnosis", ylab="SMR ratio" )
> abline( h=1 )
> matplot( pr.d, Slx.d,
+          type="l", lty=1, lwd=c(3,1,1), col=rep(heat.colors(9),each=3),
+          log="y", ylim=c(1/2,4),
+          xlab="Diabetes duration", ylab="SMR ratio" )
> abline( h=1 )
```

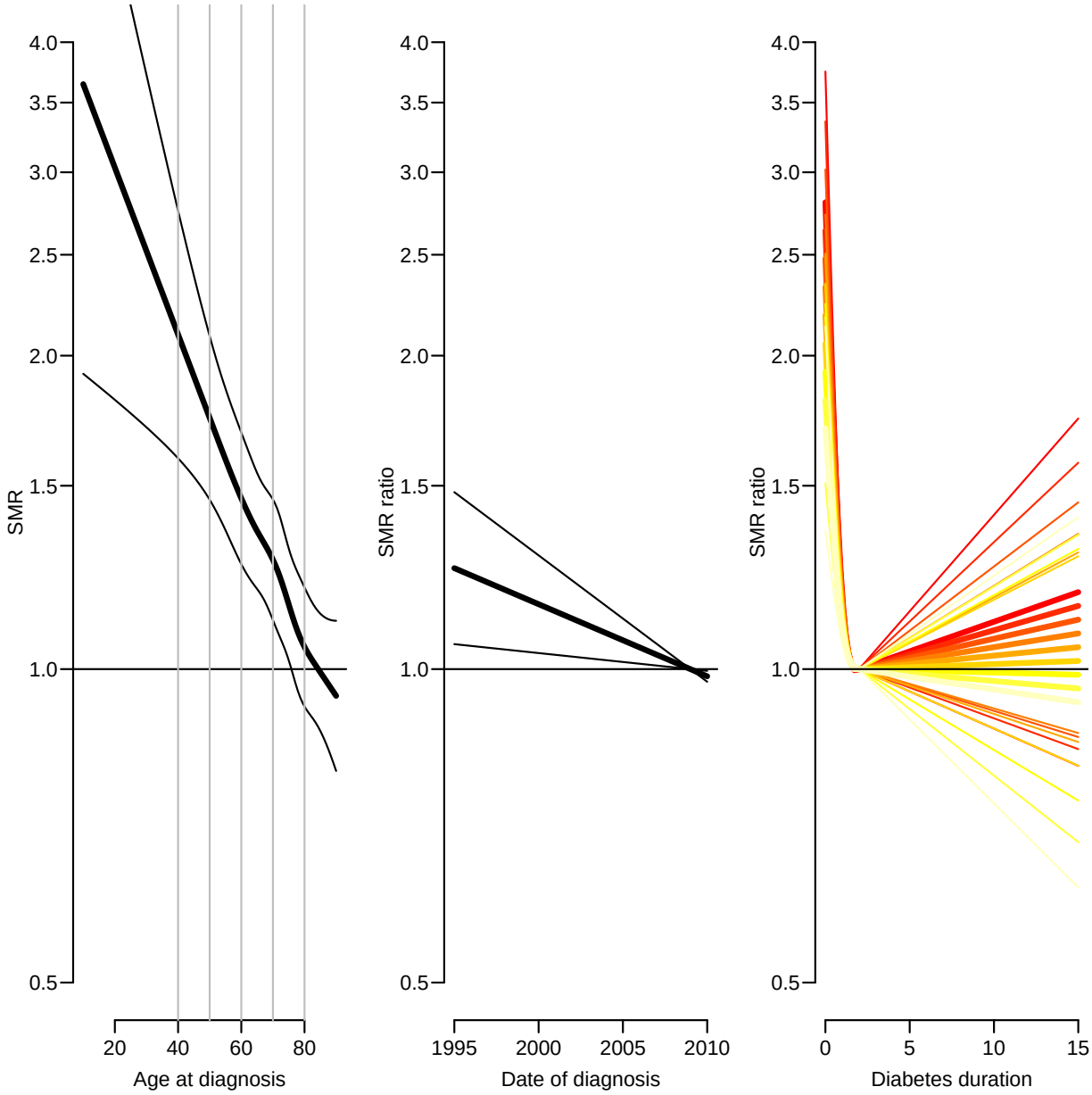


Figure 9.14: *SMR in the diabetic population for both sexes, relative to the (entire) Danish population — interaction model with age-specific duration effects.*

This approach is however a bit artificial, because we have fixed the duration effects to be 1 at duration 2 years. It would be appropriate to combine the effects of age at diagnosis and duration to show how the SMR looks as a function of current age.

```
> pts <- c(seq(0,15,0.1),NA)
> np <- length( pts )
> nd <- data.frame( A=rep(seq(50,90,5),each=np)+pts,
+                   P=rf.P+pts,
+                 dur=     pts,
+                   E=1 )
> A.sx <- ci.pred( Sx , newdata=nd )
> A.sl <- ci.pred( Slx, newdata=nd )


> matplot( NA, NA,
+          log="y", ylim=c(1/2,5), xlim=c(50,100),
+          xlab="Age at follow-up", ylab="SMR" )
> abline( h=c(5:19/10,seq(2,5,0.5)), v=seq(50,100,5), col=gray(0.8) )
> matlines( nd$A, cbind(A.sx,A.sl),
+           type="l", lty=rep(c(1,3),each=3), lwd=c(3,1,1), col="forestgreen" )
> abline( h=1 )
```

From figure **??** it is clear that the interaction means that the patients diagnosed at young age (50–60, that is) do not experience a declining SMR, on the contrary, they have a relative mortality that is close to what it is a year or so after diagnosis, which is about 2 for 50-year old , 1.4 for 70 year old and 1.1 for 80 year old

This interaction machinery with linear age easily generalizes to more complex age-effects, it is just a question of choosing another age-effect:

```
> Six <- update( Sx, . ~. + Ns(A-dur,knots=kn.Ad):Ns(dur,knots=kn.dur) )
> anova( Six, Slx, Sx, test="Chisq" )
  Analysis of Deviance Table

  Model 1: (lex.Xst == "Dead") ~ Ns(A - dur, kn = kn.Ad) + I(P - dur) +
      Ns(dur, kn = kn.dur) + Ns(A - dur, kn = kn.Ad):Ns(dur, kn = kn.dur)
  Model 2: (lex.Xst == "Dead") ~ Ns(A - dur, kn = kn.Ad) + I(P - dur) +
      Ns(dur, kn = kn.dur) + Ns(dur, kn = kn.dur):I(A - dur)
  Model 3: (lex.Xst == "Dead") ~ Ns(A - dur, kn = kn.Ad) + I(P - dur) +
      Ns(dur, kn = kn.dur)
    Resid. Df Resid. Dev Df Deviance Pr(>Chi)
  1    118405      21929
  2    118411      21936 -6  -6.7561  0.34400
  3    118413      21941 -2  -4.9395  0.08461
> A.si <- ci.pred( Six, newdata=nd )
```

And we can use the exact same code to show the interaction and plot it along the others in a similar plot:

```
> matplot( NA, NA,
+          log="y", ylim=c(1/2,5), xlim=c(50,100),
+          xlab="Age at follow-up", ylab="SMR" )
> abline( h=c(5:19/10,seq(2,5,0.5)), v=seq(50,100,5), col=gray(0.8) )
> matlines( nd$A, cbind(A.si,A.sl,A.sx),
+           type="l", lty=rep(c(1,3),c(6,3)), lwd=c(3,1,1),
+           col=rep(c("magenta","limegreen"),c(3,6)) )
> abline( h=1 )
```

From figure **??** it is seen that the interaction chosen was way too complex; the long-term variations in the SMR as estimated here do not seem believable. Although the general pattern is pretty much the same; it is the age at diagnosis that determines the SMR.

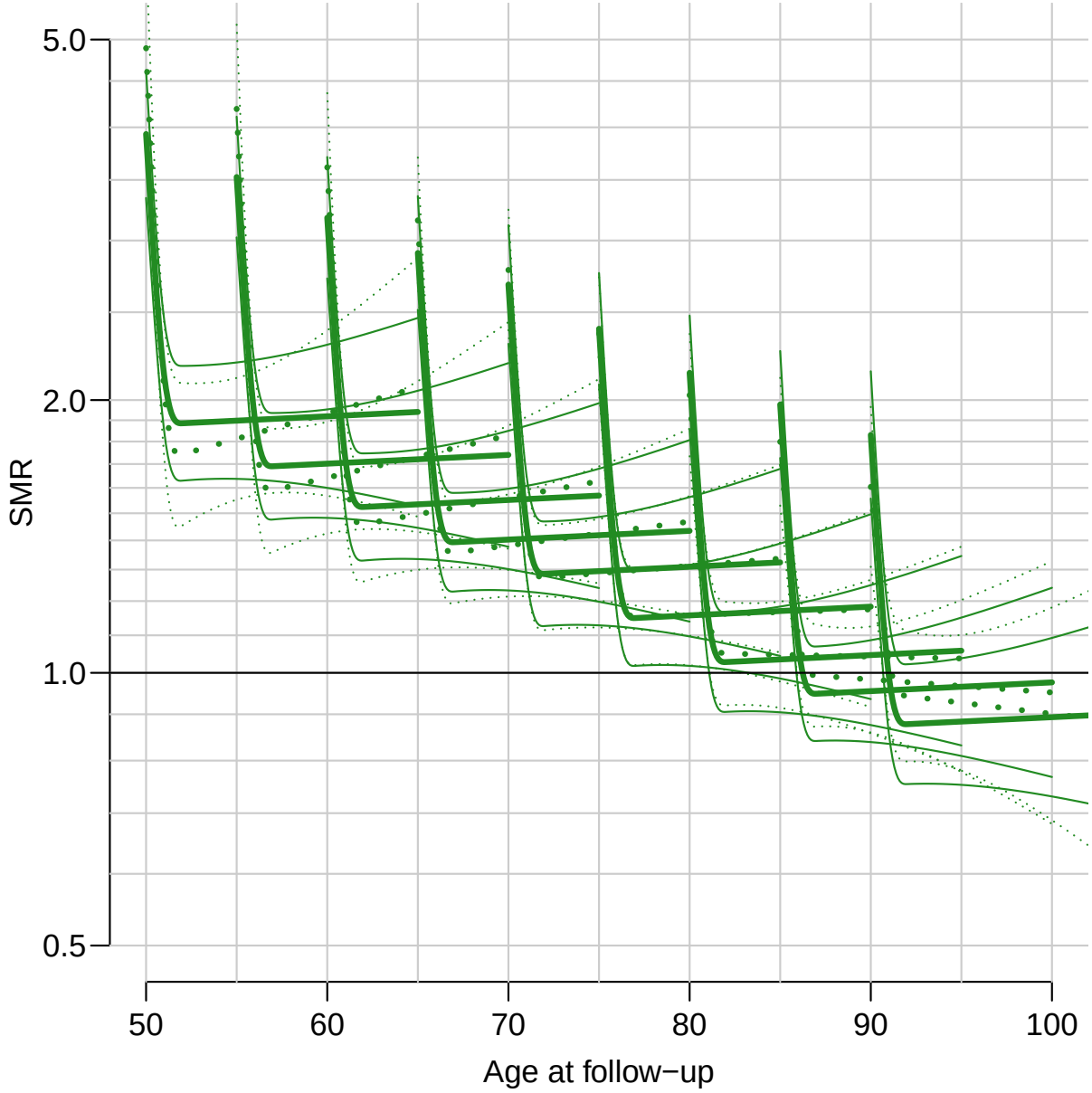


Figure 9.15: *SMR in the diabetic population for both sexes, relative to the (entire) Danish population — interaction model with age-specific duration effects, shown for patients diagnosed at ages 50 to 90.*

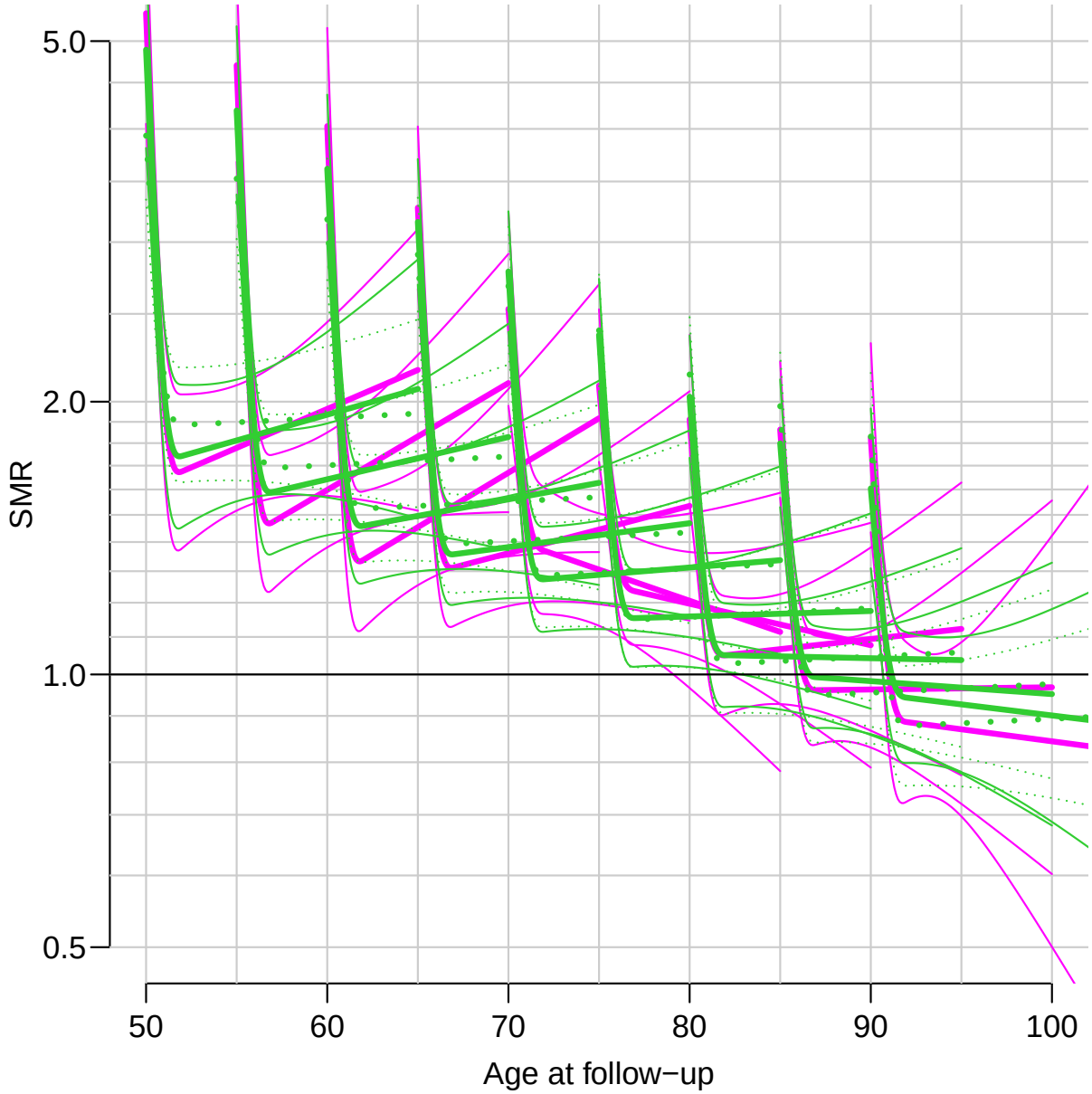


Figure 9.16: *SMR in the diabetic population for both sexes, relative to the (entire) Danish population — interaction model with age-specific duration effects, shown for patients diagnosed at ages 50, 60, 70, 80 and 90. The bright green curves are from the simple interaction model, while magenta curves are from the more complex interaction model.*

Chapter 10

General calculations

10.1 Modelling linear and non-linear effects

This section describes how continuous non-linear effects of explanatory variables can be modelled and in particular reported in graphical form.

10.1.1 Linear regression: diet data

We start out by an ordinary regression based on the `diet` data set from the `Epi` package, we plot the relationship between weight and height as well as the estimated regression line from regression of weight (outcome) on height (determinant):

```
> options( show.signif.stars=FALSE, width=100 )
> library( Epi )
> data( diet )
> names( diet )
   [1] "id"         "doe"        "dox"        "dob"        "y"          "fail"       "job"
   [8] "month"      "energy"     "height"     "weight"     "fat"        "fibre"      "energy.grp"
  [15] "chd"
> with( diet, plot(weight~height,pch=16) )
> abline( lm(weight~height,data=diet), col="red", lwd=2 )
```

We can take look at the summary of the model, and the more parsimonious output from `ci.lin`:

```
> ml <- lm( weight ~ height, data=diet )
> summary( ml )
  Call:
  lm(formula = weight ~ height, data = diet)

  Residuals:
       Min       1Q   Median       3Q      Max
  -24.7361  -7.4553   0.1608   6.9384  27.8130

  Coefficients:
              Estimate Std. Error t value Pr(>|t|)
  (Intercept) -59.91601   14.31557  -4.185 3.66e-05
  height        0.76421    0.08252   9.261  < 2e-16

  Residual standard error: 9.625 on 330 degrees of freedom
    (5 observations deleted due to missingness)
  Multiple R-squared:  0.2063,        Adjusted R-squared:  0.2039
  F-statistic: 85.76 on 1 and 330 DF,  p-value: < 2.2e-16
```

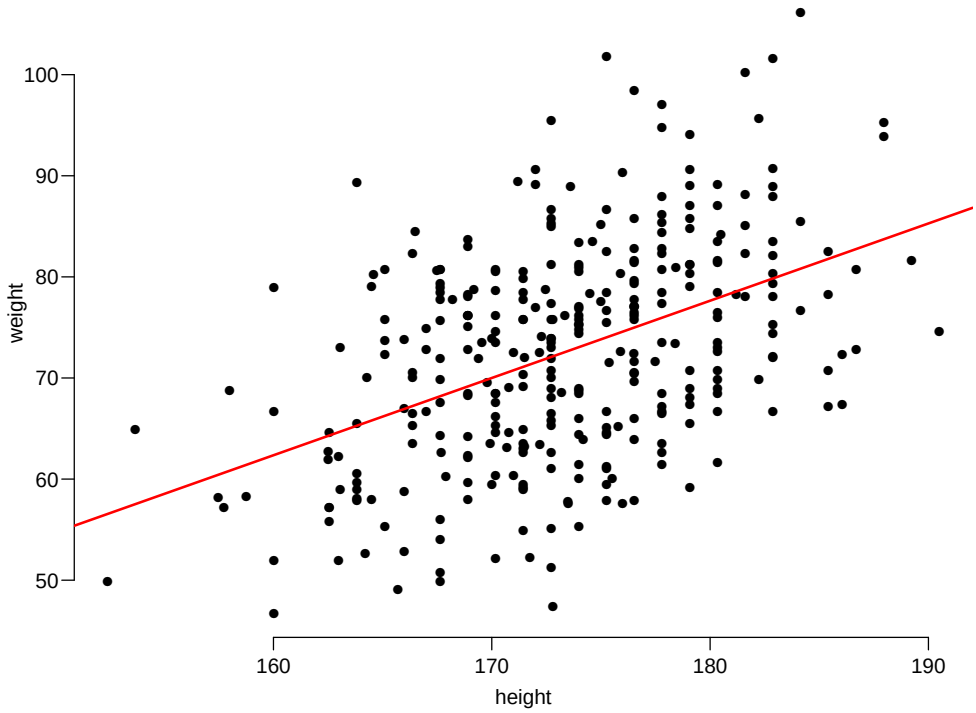


Figure 10.1: *Height and weight with regression line.*

```
> round( ci.lin( ml ), 4 )
             Estimate  StdErr       z P     2.5%   97.5%
(Intercept) -59.9160 14.3156 -4.1854 0 -87.9740 -31.858
height        0.7642  0.0825  9.2607 0   0.6025   0.926
```

Apart from the regression line it is of interest to see the *confidence limits* (for the *mean* relationship), and the *prediction limits*; intervals for prediction of weight for a given value of height. The latter is wider because it also incorporates the *residual variation*:

```
> ml <- lm( weight ~ height, data=diet )
> nd <- data.frame( height = 150:190 )
> pr.co <- predict( ml, newdata=nd, interval="conf" )
> pr.pr <- predict( ml, newdata=nd, interval="pred" )
> with( diet, plot( weight ~ height, pch=16 )  )
> matlines( nd$height, pr.co, lty=1, lwd=c(5,2,2), col="blue" )
> matlines( nd$height, pr.pr, lty=2, lwd=c(5,2,2), col="blue" )
```

We can also use a *quadratic model*, that is a model where weight is not a simple linear function of height, but is assumed to be a *quadratic* function of weight.

```
> mq <- lm( weight ~ height + I(height^2), data=diet )
> ci.lin( mq )
                  Estimate       StdErr          z         P          2.5%        97.5%
(Intercept) -1.928840e+02 2.698746e+02 -0.7147173 0.4747838 -721.82849818 336.06043842
height       2.304522e+00 3.122922e+00  0.7379378 0.4605522   -3.81629244   8.42533719
I(height^2) -4.454604e-03 9.028397e-03 -0.4933992 0.6217305   -0.02214994   0.01324073
> nd <- data.frame( height = 150:190 )
> pr.co <- predict( mq, newdata=nd, interval="conf" )
> pr.pr <- predict( mq, newdata=nd, interval="pred" )
> with( diet, plot( weight ~ height, pch=16 )  )
> matlines( nd$height, pr.co, lty=1, lwd=c(5,2,2), col="blue" )
> matlines( nd$height, pr.pr, lty=2, lwd=c(5,2,2), col="blue" )
```

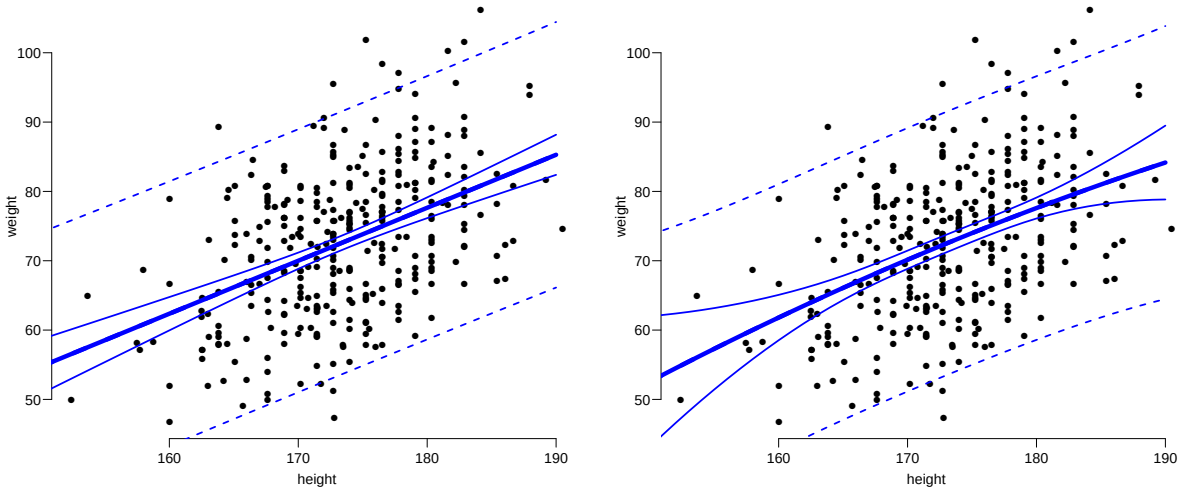


Figure 10.2: *Linear (left) and quadratic (right) relationship between height an weight, with confidence limits for the mean, and prediction limits.*

From figure 10.2 we see that there is not much curvature; the formal test is whether the coefficient to `height`^{2} is significantly different from 0; it is not; the p-value 0.622, so no evidence of non-linearity of the relationship. Interestingly, it is seen that the addition of the quadratic term increases the uncertainly of the mean estimate, but not the width of the prediction limits.

10.1.2 Poisson regression: rates and RRs

In section **??** we did some elementary calculations on fictitious rate data with very few observations.

To illustrate how we predict rates from Poisson models on real data we use a data set from the `Epi` package, with no. of cases of testis cancer and person-years in the Danish male population, classified by age and calendar time in 1-year classes:

```
> data( testisDK )
> str( testisDK )
  'data.frame':        4860 obs. of  4 variables:
   $ A: num  0 1 2 3 4 5 6 7 8 9 ...
   $ P: num  1943 1943 1943 1943 1943 ...
   $ D: num  1 1 0 1 0 0 0 0 0 0 ...
   $ Y: num  39650 36943 34588 33267 32614 ...

> head( testisDK )

    A    P D        Y
  1 0 1943 1 39649.50
  2 1 1943 1 36942.83
  3 2 1943 0 34588.33
  4 3 1943 1 33267.00
  5 4 1943 0 32614.00
  6 5 1943 0 32020.33
```

First we make a coarse table of the cases, person-years and rates by 10-year classes of age and calendar time using `stat.table`:

```
> stat.table( list(A=floor(A/10)*10,
+                  P=floor(P/10)*10),
+             list( D=sum(D),
+                   Y=sum(Y/10^5),
+                rate=ratio(D,Y,10^5) ),
+             margins=TRUE, data=testisDK )
```

	P						
A	1940	1950	1960	1970	1980	1990	Total
0	10.00	7.00	16.00	18.00	9.00	10.00	70.00
	26.05	40.37	38.85	38.21	30.71	21.66	195.84
	0.38	0.17	0.41	0.47	0.29	0.46	0.36
10	13.00	27.00	37.00	72.00	97.00	75.00	321.00
	21.36	35.05	40.04	39.06	38.47	22.61	196.59
	0.61	0.77	0.92	1.84	2.52	3.32	1.63
20	124.00	221.00	280.00	535.00	724.00	557.00	2441.00
	22.26	29.23	34.02	40.29	39.41	28.25	193.45
	5.57	7.56	8.23	13.28	18.37	19.72	12.62
30	149.00	288.00	377.00	624.00	771.00	744.00	2953.00
	21.95	30.59	28.56	34.11	39.69	27.28	182.18
	6.79	9.42	13.20	18.30	19.43	27.27	16.21
40	95.00	198.00	230.00	334.00	432.00	360.00	1649.00
	18.75	29.80	29.87	28.23	33.23	27.58	167.45
	5.07	6.64	7.70	11.83	13.00	13.05	9.85
50	40.00	79.00	140.00	151.00	193.00	155.00	758.00
	14.43	24.27	27.97	28.13	26.35	20.69	141.83
	2.77	3.26	5.01	5.37	7.32	7.49	5.34
60	29.00	43.00	54.00	83.00	82.00	44.00	335.00
	10.42	17.12	20.55	23.58	23.57	15.65	110.89
	2.78	2.51	2.63	3.52	3.48	2.81	3.02
70	18.00	26.00	35.00	41.00	40.00	32.00	192.00
	5.38	9.68	11.36	13.37	15.38	11.01	66.17
	3.35	2.69	3.08	3.07	2.60	2.91	2.90
80	7.00	9.00	13.00	19.00	18.00	21.00	87.00
	1.34	2.62	3.46	4.23	5.04	4.15	20.84
	5.24	3.44	3.75	4.49	3.57	5.06	4.18
Total	485.00	898.00	1182.00	1877.00	2366.00	1998.00	8806.00
	141.92	218.72	234.68	249.21	251.85	178.87	1275.25
	3.42	4.11	5.04	7.53	9.39	11.17	6.91

We then model the rates using a Poisson-model, because the likelihood for $(d, y) = (\texttt{D}, \texttt{Y})$ is proportional to a Poisson likelihood assuming that rates are constant in 1×1-year intervals of age and calendar time.

The first model is very coarse; we simply assume that incidence rates depend linearly on the age in in each interval and not on calendar time at all; we use `ci.lin` and `ci.exp` to show the parameters on the log-scale and on the rate-scale respectively:

```
> ml <- glm( D ~ A, offset=log(Y/1000), family=poisson, data=testisDK )
> round( ci.lin( ml ), 4 )
```

	Estimate	StdErr	z	P	2.5%	97.5%
(Intercept)	-2.8677	0.0207	-138.5578	0	-2.9083	-2.8271
A	0.0055	0.0005	11.3926	0	0.0045	0.0064

```
> round( ci.exp( ml ), 4 )
```

```
             exp(Est.)   2.5%  97.5%
(Intercept)    0.0568 0.0546 0.0592
A              1.0055 1.0046 1.0064
```

How would you interpret the estimates that are computed by `ci.exp`?

To predict the incidence rates in different ages, we must devise a data set of explanatory variables, in this case age, `A` and person-years, `Y`. Note that we put `Y` to be constant equal to 10^5, so that we get the expected number of cases during 100,000 person-years, that is the incidence rates per 100,000 PY. We then plot the estimated rates against age; note that we are using a log-scale for the rates, and hence the log-linear relationship between rates and age show up as a straight line:

```
> nd <- data.frame( A=15:60, Y=10^5 )
> pr <- ci.pred( ml, newdata=nd )
> head( cbind(nd,pr) )
```

```
    A     Y Estimate     2.5%    97.5%
1 15 1e+05 6.170105 5.991630 6.353896
2 16 1e+05 6.204034 6.028525 6.384652
3 17 1e+05 6.238149 6.065547 6.415662
4 18 1e+05 6.272452 6.102689 6.446937
5 19 1e+05 6.306943 6.139944 6.478485
6 20 1e+05 6.341624 6.177301 6.510319
```

```
> matplot( nd$A, pr,
+          type="l", lty=1, lwd=c(3,1,1), col="black", log="y" )
```

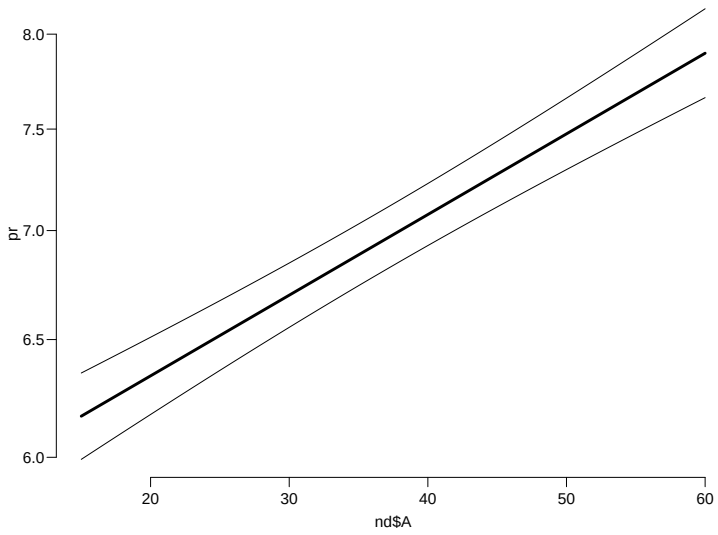


Figure 10.3: *Predicted incidence rates of testis cancer using a naive model with only a liner effect of age.*

An alternative way of deriving the rates is to make an explicit calculation from the estimated parameters of the model:

```
> round( ci.lin( ml ), 4 )
```

```
             Estimate StdErr         z P    2.5%   97.5%
(Intercept)  -2.8677 0.0207 -138.5578 0 -2.9083 -2.8271
A             0.0055 0.0005   11.3926 0  0.0045  0.0064
```

```
> Cl <- cbind( 1, nd$A )
> head( Cl )

      [,1] [,2]
 [1,]    1   15
 [2,]    1   16
 [3,]    1   17
 [4,]    1   18
 [5,]    1   19
 [6,]    1   20

> matplot( nd$A, ci.exp( ml, ctr.mat=Cl ),
+          type="l", lty=1, lwd=c(3,1,1), col="black", log="y" )
```

Verify that the two plots show the same curve.

As in the diet example we can now add a quadratic term in age, to check if there is any indication of non-linearity:

```
> mq <- glm( D ~ A + I(A^2),
+            offset=log(Y), family=poisson, data=testisDK )
> round( ci.lin( mq ), 4 )

             Estimate StdErr         z P     2.5%    97.5%
 (Intercept) -12.3656 0.0596 -207.3611 0 -12.4825 -12.2487
 A             0.1806 0.0033   54.8290 0   0.1741   0.1871
 I(A^2)       -0.0023 0.0000  -53.7006 0  -0.0024  -0.0022

> round( ci.exp( mq ), 4 )

             exp(Est.)   2.5%  97.5%
 (Intercept)    0.0000 0.0000 0.0000
 A              1.1979 1.1902 1.2057
 I(A^2)         0.9977 0.9976 0.9978
```

There is clearly a very strong indication of a non-linear relationship.

We can now plot the estimated curved relationship using a *contrast matrix* as before, but now with a column of the ages squared too:

```
> round( ci.lin( mq ), 4 )

             Estimate StdErr         z P     2.5%    97.5%
 (Intercept) -12.3656 0.0596 -207.3611 0 -12.4825 -12.2487
 A             0.1806 0.0033   54.8290 0   0.1741   0.1871
 I(A^2)       -0.0023 0.0000  -53.7006 0  -0.0024  -0.0022

> Cq <- cbind( 1, 15:60, (15:60)^2 )
> head( Cq, 4 )

      [,1] [,2] [,3]
 [1,]    1   15  225
 [2,]    1   16  256
 [3,]    1   17  289
 [4,]    1   18  324

> matplot( nd$A, ci.exp( mq, ctr.mat=Cq )*10^5,
+          type="l", lty=1, lwd=c(3,1,1), col="black", log="y" )
```

Actually we want to plot the two predictions on top of each other:

```
> matplot( nd$A, ci.exp( mq, ctr.mat=Cq )*10^5,
+          type="l", lty=1, lwd=c(3,1,1), col="black", log="y" )
> matlines( nd$A, ci.exp( ml, ctr.mat=Cl )*10^5,
+           type="l", lty=1, lwd=c(3,1,1), col="blue" )
```

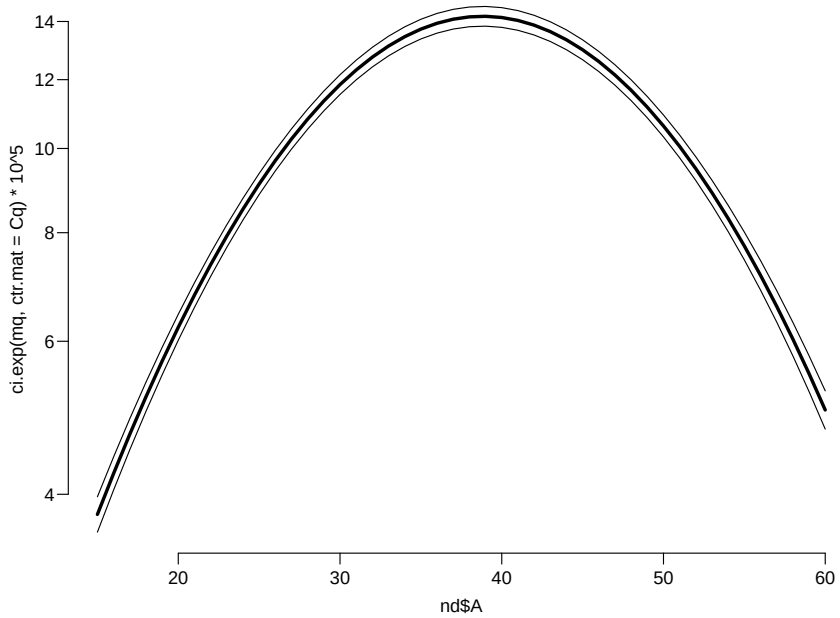


Figure 10.4: *Predicted age-specific testis cancer incidence rates using a linear (blue) and a quadratic model (black)*

It is of interest to see if the quadratic gives an adequate description of rates, so we fit a more flexible model using *natural splines* (a.k.a. *restricted cubic splines).* These are functions that are piecewise cubic functions in intervals defined by *knots*; the *natural* (*restricted*) refer to the extra restriction that the curves are linear beyond the outermost knots; in this case below age 15 and above age 65[1]. The function `Ns` from the `Epi` package is a wrapper fro the function `ns` from the `splines` package; it computes the outer knots from the supplied set of knots, but we still need to load the `splines` package:

```
> library( splines )
> ms <- glm( D ~ Ns(A,knots=seq(15,65,10)),
+                offset=log(Y), family=poisson, data=testisDK )
> round( ci.exp( ms ), 3 )
                                  exp(Est.)   2.5%  97.5%
  (Intercept)                         0.000  0.000  0.000
  Ns(A, knots = seq(15, 65, 10))1     8.548  7.650  9.551
  Ns(A, knots = seq(15, 65, 10))2     5.706  4.998  6.514
  Ns(A, knots = seq(15, 65, 10))3     1.002  0.890  1.128
  Ns(A, knots = seq(15, 65, 10))4    14.402 11.896 17.436
  Ns(A, knots = seq(15, 65, 10))5     0.466  0.429  0.505

> aa <- 15:65
> As <- Ns( aa, knots=seq(15,65,10) )
> head( As )
                 1 2           3          4           5
  [1,] 0.0000000000 0  0.00000000 0.00000000  0.00000000
  [2,] 0.0001666667 0 -0.02527011 0.07581034 -0.05054022
  [3,] 0.0013333333 0 -0.05003313 0.15009940 -0.10006626
  [4,] 0.0045000000 0 -0.07378197 0.22134590 -0.14756393
  [5,] 0.0106666667 0 -0.09600952 0.28802857 -0.19201905
  [6,] 0.0208333333 0 -0.11620871 0.34862613 -0.23241742
```

[1]This is a restriction that turns out to render the resulting curves a bit more stable — an excellent illustration of this is found on Paul Lambert's website, see http://www.le.ac.uk/hs/pl4/spline_eg.html

As before we overlay the spline fitted model with the fit from the previous (quadratic) model:

```
> matplot( aa, ci.exp( ms, ctr.mat=cbind(1,As) )*10^5,
+          log="y", xlab="Age", ylab="Testis cancer incidence rate per 100,000 PY",
+          type="l", lty=1, lwd=c(3,1,1), col="black", ylim=c(2,20) )
> matlines( nd$A, ci.exp( mq, ctr.mat=Cq )*10^5,
+           type="l", lty=1, lwd=c(3,1,1), col="blue" )
```

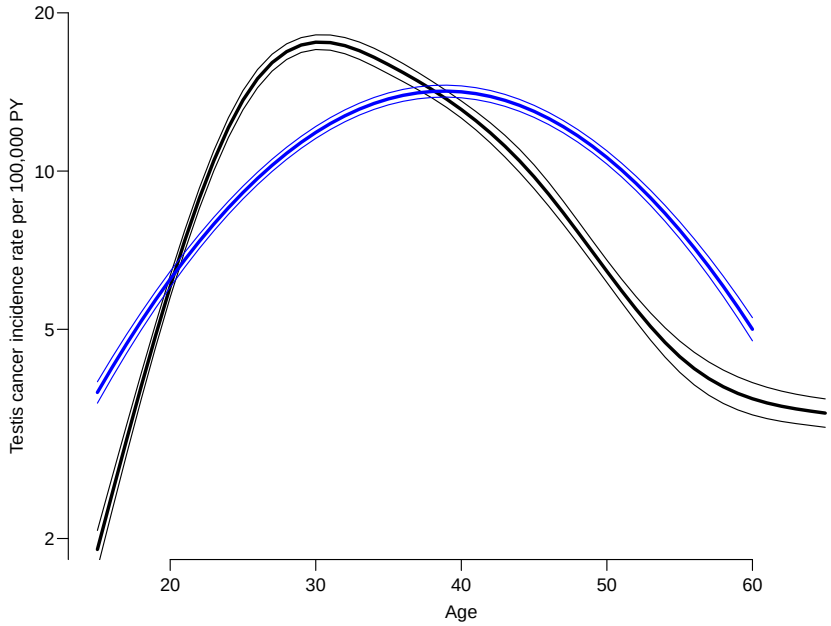


Figure 10.5: *Predicted age-specific testis cancer incidence rates using a spline (black) and at quadratic model (blue)*

10.1.3 Period effect

So far we have totally ignored the calendar time; so we insert a term for the linear trend by calendar year:

```
> msp <- glm( D ~ Ns(A,knots=seq(15,65,10)) + P,
+                 offset=log(Y), family=poisson, data=testisDK )
> round( ci.lin( msp ), 3 )
                                Estimate StdErr       z     P    2.5%   97.5%
  (Intercept)                    -58.105  1.444 -40.229 0.000 -60.935 -55.274
  Ns(A, knots = seq(15, 65, 10))1  2.120  0.057  37.444 0.000   2.009   2.231
  Ns(A, knots = seq(15, 65, 10))2  1.700  0.068  25.157 0.000   1.567   1.832
  Ns(A, knots = seq(15, 65, 10))3  0.007  0.060   0.110 0.913  -0.112   0.125
  Ns(A, knots = seq(15, 65, 10))4  2.596  0.097  26.631 0.000   2.405   2.787
  Ns(A, knots = seq(15, 65, 10))5 -0.780  0.042 -18.748 0.000  -0.861  -0.698
  P                                0.024  0.001  32.761 0.000   0.023   0.025
> round( ci.exp( msp ), 3 )
                                exp(Est.)   2.5%  97.5%
  (Intercept)                       0.000  0.000  0.000
  Ns(A, knots = seq(15, 65, 10))1   8.327  7.453  9.305
  Ns(A, knots = seq(15, 65, 10))2   5.472  4.793  6.247
  Ns(A, knots = seq(15, 65, 10))3   1.007  0.894  1.133
  Ns(A, knots = seq(15, 65, 10))4  13.405 11.074 16.226
  Ns(A, knots = seq(15, 65, 10))5   0.459  0.423  0.497
  P                                 1.024  1.023  1.026
```

We see that the coefficient is 1.024 meaning that rates increase annually by a factor of 1.024 or by 2.4%.

If we want to show predicted rates from this model, we must recognize that calendar time (P) is in the model now, so the prediction must refer to a specific point in time, here we choose 1970, and put this in as a separate column in the contrast matrix:

```
> Ca <- cbind( 1, Ns( aa, knots=seq(15,65,10) ), 1970 )
> head( Ca )
                    1 2           3          4           5
  [1,] 1 0.0000000000 0  0.00000000 0.00000000  0.00000000 1970
  [2,] 1 0.0001666667 0 -0.02527011 0.07581034 -0.05054022 1970
  [3,] 1 0.0013333333 0 -0.05003313 0.15009940 -0.10006626 1970
  [4,] 1 0.0045000000 0 -0.07378197 0.22134590 -0.14756393 1970
  [5,] 1 0.0106666667 0 -0.09600952 0.28802857 -0.19201905 1970
  [6,] 1 0.0208333333 0 -0.11620871 0.34862613 -0.23241742 1970
```

As before we overlay the fitted rates with those from the previous model:

```
> matplot( aa, ci.exp( msp, ctr.mat=Ca )*10^5,
+          log="y", xlab="Age",
+          ylab="Testis cancer incidence rate per 100,000 PY in 1970",
+          type="l", lty=1, lwd=c(3,1,1), col="black", ylim=c(2,20) )
> matlines( nd$A, ci.pred( ms, newdata=nd ),
+           type="l", lty=1, lwd=c(3,1,1), col="blue" )
```

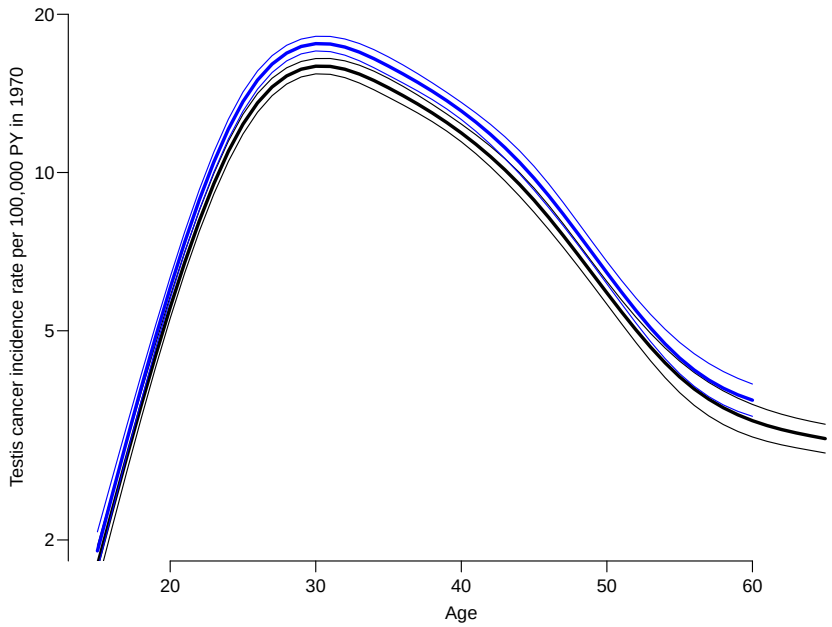


Figure 10.6: *Estimated rates from the spline model without period effect(blue) and rates in 1970 from the model with a linear period effect (black).*

The curves in figure 10.6 is in some sense both estimates of the rates in 1970; the model with no period effect namely assumes that rates do not change over time.

We would also like to see how the RR as a function of time looks; to this end we must choose a *reference* on the P-scale; and since we made the rate prediction for 1970, this is a natural choice, so we compute the RR between years 1945,. . . ,1995 and 1970:

```
> round( ci.lin( msp ), 3 )
```

```
                                Estimate StdErr       z     P    2.5%   97.5%
  (Intercept)                    -58.105  1.444 -40.229 0.000 -60.935 -55.274
  Ns(A, knots = seq(15, 65, 10))1   2.120  0.057  37.444 0.000   2.009   2.231
  Ns(A, knots = seq(15, 65, 10))2   1.700  0.068  25.157 0.000   1.567   1.832
  Ns(A, knots = seq(15, 65, 10))3   0.007  0.060   0.110 0.913  -0.112   0.125
  Ns(A, knots = seq(15, 65, 10))4   2.596  0.097  26.631 0.000   2.405   2.787
  Ns(A, knots = seq(15, 65, 10))5  -0.780  0.042 -18.748 0.000  -0.861  -0.698
  P                                 0.024  0.001  32.761 0.000   0.023   0.025
```

```
> pp <- seq(1945,1995,0.2)
> Cp <- cbind( pp - 1970 )
> head( Cp )
```

```
       [,1]
  [1,] -25.0
  [2,] -24.8
  [3,] -24.6
  [4,] -24.4
  [5,] -24.2
  [6,] -24.0
```

```
> ci.exp( msp, subset="P" )
```

```
    exp(Est.)     2.5%    97.5%
  P  1.024235 1.022769 1.025704
```

```
> matplot( pp, ci.exp( msp, subset="P", ctr.mat=Cp ),
+          log="y", ylim=c(0.5,2), xlab="Date",
+          ylab="Testis cancer incidence RR",
+          type="l", lty=1, lwd=c(3,1,1), col="black" )
> abline( h=1, v=1970 )
```

As for the initial analyses of age, we could include a quadratic effect of period; which means that the contrast matrix we shall use is the difference between the linear and quadratic functions of the years 1945 to 1995, and the same fro 1970:

```
> mspq <- glm( D ~ Ns(A,knots=seq(15,65,10)) + P + I(P^2),
+                  offset=log(Y), family=poisson, data=testisDK )
> round( ci.lin( mspq ), 3 )
```

```
                                Estimate  StdErr       z     P      2.5%    97.5%
  (Intercept)                 -807.245 205.039  -3.937 0.000 -1209.115 -405.375
  Ns(A, knots = seq(15, 65, 10))1    2.123   0.057  37.497 0.000     2.012    2.234
  Ns(A, knots = seq(15, 65, 10))2    1.707   0.068  25.249 0.000     1.575    1.840
  Ns(A, knots = seq(15, 65, 10))3    0.006   0.060   0.099 0.921    -0.113    0.125
  Ns(A, knots = seq(15, 65, 10))4    2.598   0.098  26.643 0.000     2.407    2.789
  Ns(A, knots = seq(15, 65, 10))5   -0.780   0.042 -18.767 0.000    -0.862   -0.699
  P                                  0.784   0.208   3.769 0.000     0.376    1.191
  I(P^2)                             0.000   0.000  -3.654 0.000     0.000    0.000
```

```
> Cq <- cbind( pp-1970, pp^2-1970^2 )
> head( Cq )
```

```
       [,1]      [,2]
  [1,] -25.0 -97875.00
  [2,] -24.8 -97096.96
  [3,] -24.6 -96318.84
  [4,] -24.4 -95540.64
  [5,] -24.2 -94762.36
  [6,] -24.0 -93984.00
```

```
> ci.exp( mspq, subset="P" )
```

```
         exp(Est.)      2.5%     97.5%
  P      2.1893078 1.4566021 3.2905821
  I(P^2) 0.9998075 0.9997042 0.9999107
```

```
> matplot( pp, ci.exp( mspq, subset="P", ctr.mat=Cq ),
+          log="y", ylim=c(0.5,2), xlab="Date",
+          ylab="Testis cancer incidence RR",
+          type="l", lty=1, lwd=c(3,1,1), col="black" )
> abline( h=1, v=1970 )
```

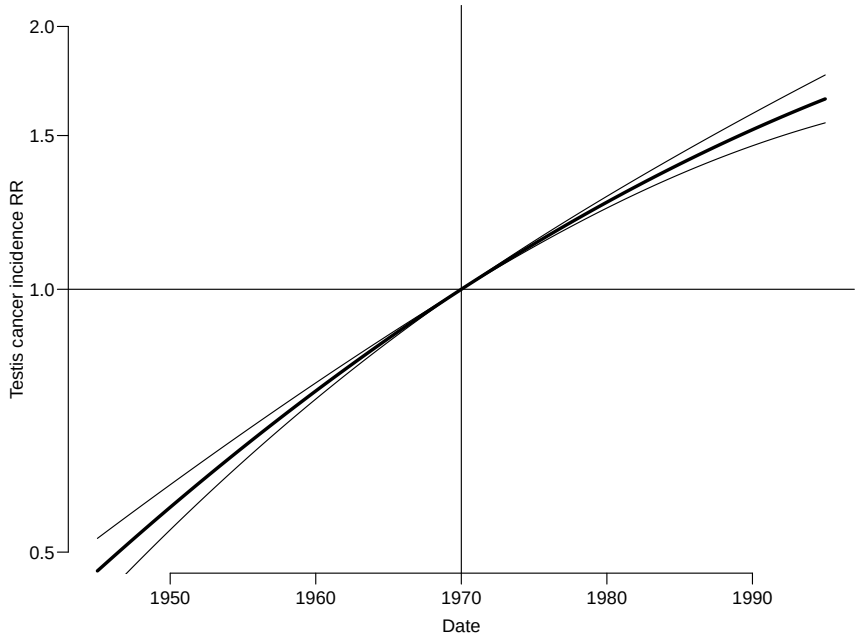


Figure 10.7: *RR relative to 1970 from a model with quadratic effect of calendar time.*

Although the coefficient to P^2 seems tiny, it is actually strongly significant, and it seems also so judging from the graph of the RR as a function of calendar time in figure 10.7.

Then we expand the model to a model where we use a spline to model the calendar time effect.

```
> msps <- glm( D ~ Ns(A,knots=seq(15,65,5)) +
+                  Ns(P,knots=seq(1950,1990,5)),
+                  offset=log(Y), family=poisson, data=testisDK )
> round( ci.exp( msps ), 3 )
                                    exp(Est.)  2.5%  97.5%
  (Intercept)                           0.000 0.000  0.000
  Ns(A, knots = seq(15, 65, 5))1        9.293 7.914 10.912
  Ns(A, knots = seq(15, 65, 5))2        8.630 7.575  9.832
  Ns(A, knots = seq(15, 65, 5))3        9.260 7.965 10.765
  Ns(A, knots = seq(15, 65, 5))4        6.592 5.652  7.688
  Ns(A, knots = seq(15, 65, 5))5        5.310 4.446  6.342
  Ns(A, knots = seq(15, 65, 5))6        3.209 2.627  3.919
  Ns(A, knots = seq(15, 65, 5))7        3.224 2.607  3.987
  Ns(A, knots = seq(15, 65, 5))8        1.076 0.918  1.260
  Ns(A, knots = seq(15, 65, 5))9        3.698 3.101  4.411
  Ns(A, knots = seq(15, 65, 5))10       0.828 0.726  0.945
  Ns(P, knots = seq(1950, 1990, 5))1    1.351 1.107  1.649
  Ns(P, knots = seq(1950, 1990, 5))2    1.451 1.222  1.723
  Ns(P, knots = seq(1950, 1990, 5))3    1.626 1.365  1.938
  Ns(P, knots = seq(1950, 1990, 5))4    2.182 1.879  2.533
  Ns(P, knots = seq(1950, 1990, 5))5    2.283 1.995  2.613
  Ns(P, knots = seq(1950, 1990, 5))6    2.354 2.122  2.613
  Ns(P, knots = seq(1950, 1990, 5))7    2.917 2.642  3.221
  Ns(P, knots = seq(1950, 1990, 5))8    2.452 2.225  2.702
```

In order to show the RR relative to 1970, we need the contrast matrix as a difference of 1) one corresponding to the splines for the years 1945–95 and 2) one where all rows are identical to the one corresponding to the 1970 value for the splines:

```
> Cs <- Ns(                pp  ,knots=seq(1950,1990,5))
> Cr <- Ns(rep(1970,length(pp)),knots=seq(1950,1990,5))
> head( cbind(Cs,Cr), 4 )
      1 2 3 4 5         6          7         8 1         2         3         4 5 6 7 8
[1,] 0 0 0 0 0 0.2535463 -0.7606388 0.5070926 0 0.1666667 0.6666667 0.1666667 0 0 0 0
[2,] 0 0 0 0 0 0.2434044 -0.7302133 0.4868089 0 0.1666667 0.6666667 0.1666667 0 0 0 0
[3,] 0 0 0 0 0 0.2332626 -0.6997877 0.4665251 0 0.1666667 0.6666667 0.1666667 0 0 0 0
[4,] 0 0 0 0 0 0.2231207 -0.6693622 0.4462414 0 0.1666667 0.6666667 0.1666667 0 0 0 0
> ci.exp( msps, subset="P" )
                                      exp(Est.)     2.5%    97.5%
Ns(P, knots = seq(1950, 1990, 5))1  1.351303 1.107224 1.649188
Ns(P, knots = seq(1950, 1990, 5))2  1.451226 1.222380 1.722914
Ns(P, knots = seq(1950, 1990, 5))3  1.626416 1.365047 1.937830
Ns(P, knots = seq(1950, 1990, 5))4  2.181622 1.879069 2.532890
Ns(P, knots = seq(1950, 1990, 5))5  2.283165 1.994875 2.613117
Ns(P, knots = seq(1950, 1990, 5))6  2.354497 2.121572 2.612993
Ns(P, knots = seq(1950, 1990, 5))7  2.917336 2.641957 3.221419
Ns(P, knots = seq(1950, 1990, 5))8  2.451882 2.224656 2.702317
> matplot( pp, ci.exp( msps, subset="P", ctr.mat=Cs-Cr ),
+          log="y", ylim=c(0.5,2), xlab="Date",
+          ylab="Testis cancer incidence RR",
+          type="l", lty=1, lwd=c(3,1,1), col="black" )
> abline( h=1, v=1970 )
```

From figure **??** it is clear that the curvature is not over the entire time-span, basically it looks as if there is a downward bend in the increase of rates around 1980.

Try to devise way to give rough estimates of the average annual increase before and after 1980.

Finally we can plot the estimated age-specific incidence rates in 1970, and the RR relative to 1970 from this model.

Consider how the contrast matrix for extraction of the age-specific rates in 1970 are constructed:

```
> par( mfrow=c(1,2) )
> Cap <- cbind( 1, Ns(                 aa  ,knots=seq(15,65,5)),
+                  Ns(rep(1970,length(aa)),knots=seq(1950,1990,5)) )
> matplot( aa, ci.exp( msps, ctr.mat=Cap )*10^5,
+          log="y", xlab="Age",
+          ylab="Testis cancer incidence rate per 100,000 PY in 1970",
+          type="l", lty=1, lwd=c(3,1,1), col="black" )
> matplot( pp, ci.exp( msps, subset="P", ctr.mat=Cs-Cr ),
+          log="y", xlab="Date", ylab="Testis cancer incidence RR",
+          type="l", lty=1, lwd=c(3,1,1), col="black" )
> abline( h=1, v=1970 )
```

In order to make the two plots more comparable we make sure that the physical size of 1 year on the x-axis of both plots is the same, and we also devise the y-axes so that a doubling of rates or RRs have the same physical extent:

```
> par( mfrow=c(1,2) )
> matplot( aa, ci.exp( msps, ctr.mat=Cap )*10^5,
+          log="y", xlab="Age",
+          ylim=c(2,20), xlim=c(15,65),
+          ylab="Testis cancer incidence rate per 100,000 PY in 1970",
+          type="l", lty=1, lwd=c(3,1,1), col="black" )
> matplot( pp, ci.exp( msps, subset="P", ctr.mat=Cs-Cr ),
+          log="y", xlab="Date",
+          ylim=c(2,20)/sqrt(2*20), xlim=c(15,65)+1930,
+          ylab="Testis cancer incidence RR",
+          type="l", lty=1, lwd=c(3,1,1), col="black" )
> abline( h=1, v=1970 )
```

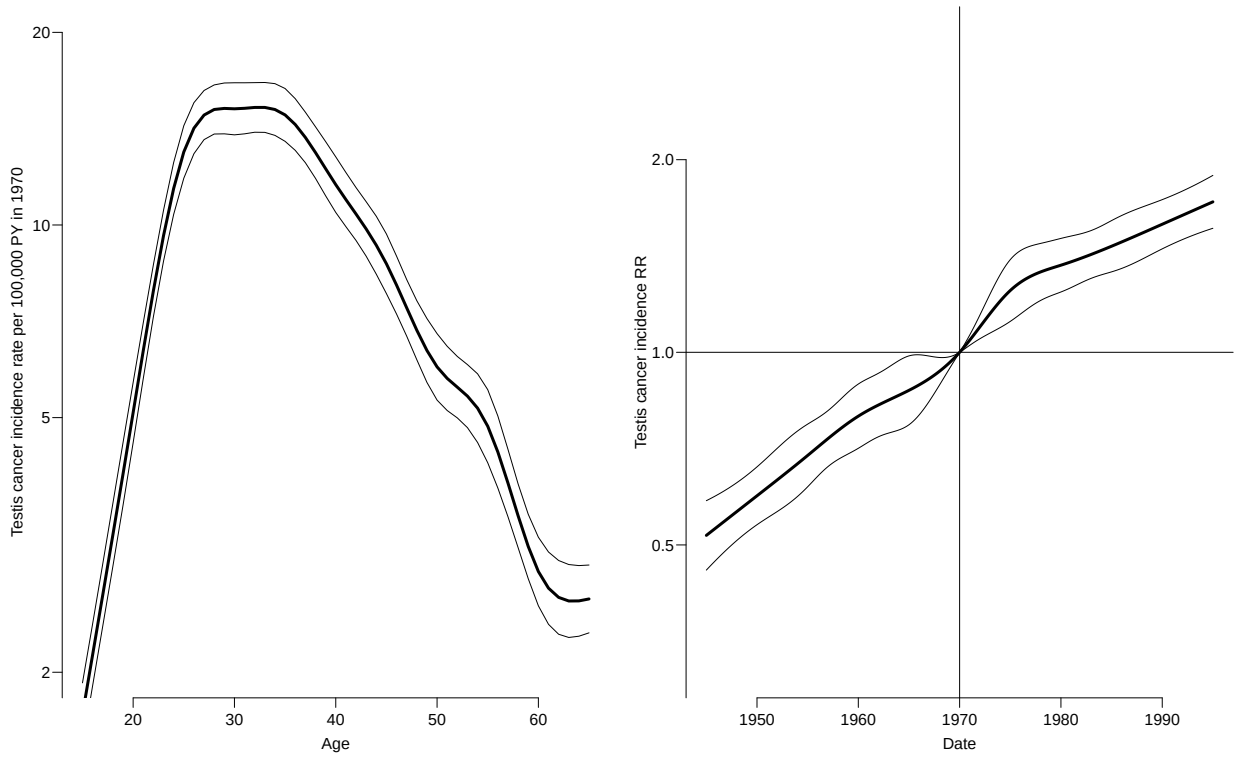


Figure 10.8: *Age-specific rates as of 1970, and RR relative to this year.*

10.1.4 Cohort effect

Instead of using calendar time we might contemplate using date of birth — the difference between calendar time and age at observation, `C` = `P` − `A`:

```
> testisDK <- transform( testisDK, C = P - A )
> head( testisDK )
  A    P D        Y    C
1 0 1943 1 39649.50 1943
2 1 1943 1 36942.83 1942
3 2 1943 0 34588.33 1941
4 3 1943 1 33267.00 1940
5 4 1943 0 32614.00 1939
6 5 1943 0 32020.33 1938
> range( testisDK$C )
[1] 1854 1996
```

We can then fit a model exactly as before, but now with date of birth, cohort, `C`, as variable, and in the prediction we will use 1950 as the reference cohort:

```
> mscs <- glm( D ~ Ns(A,knots=seq(15,65,5)) +
+                  Ns(C,knots=seq(1880,1970,5)),
+                  offset=log(Y), family=poisson, data=testisDK )
> round( ci.exp( mscs ), 3 )
                                    exp(Est.)   2.5%  97.5%
(Intercept)                             0.000  0.000  0.000
Ns(A, knots = seq(15, 65, 5))1         11.753  9.946 13.889
Ns(A, knots = seq(15, 65, 5))2         12.259 10.645 14.117
Ns(A, knots = seq(15, 65, 5))3         14.988 12.769 17.592
Ns(A, knots = seq(15, 65, 5))4         12.501 10.599 14.745
Ns(A, knots = seq(15, 65, 5))5         11.342  9.398 13.689
```

```
  Ns(A, knots = seq(15, 65, 5))6          7.771  6.294  9.595
  Ns(A, knots = seq(15, 65, 5))7          8.068  6.448 10.094
  Ns(A, knots = seq(15, 65, 5))8          3.112  2.614  3.705
  Ns(A, knots = seq(15, 65, 5))9         12.615 10.309 15.437
  Ns(A, knots = seq(15, 65, 5))10         2.465  2.120  2.868
  Ns(C, knots = seq(1880, 1970, 5))1      0.711  0.392  1.289
  Ns(C, knots = seq(1880, 1970, 5))2      1.021  0.654  1.594
  Ns(C, knots = seq(1880, 1970, 5))3      0.938  0.602  1.461
  Ns(C, knots = seq(1880, 1970, 5))4      0.936  0.646  1.357
  Ns(C, knots = seq(1880, 1970, 5))5      1.115  0.789  1.576
  Ns(C, knots = seq(1880, 1970, 5))6      1.275  0.923  1.760
  Ns(C, knots = seq(1880, 1970, 5))7      1.082  0.788  1.484
  Ns(C, knots = seq(1880, 1970, 5))8      1.717  1.267  2.328
  Ns(C, knots = seq(1880, 1970, 5))9      1.782  1.318  2.409
  Ns(C, knots = seq(1880, 1970, 5))10     2.126  1.582  2.857
  Ns(C, knots = seq(1880, 1970, 5))11     2.552  1.908  3.415
  Ns(C, knots = seq(1880, 1970, 5))12     1.697  1.273  2.262
  Ns(C, knots = seq(1880, 1970, 5))13     3.169  2.383  4.214
  Ns(C, knots = seq(1880, 1970, 5))14     3.189  2.399  4.241
  Ns(C, knots = seq(1880, 1970, 5))15     4.660  3.512  6.184
  Ns(C, knots = seq(1880, 1970, 5))16     4.274  3.206  5.699
  Ns(C, knots = seq(1880, 1970, 5))17     4.184  3.249  5.389
  Ns(C, knots = seq(1880, 1970, 5))18     4.853  3.595  6.551
```

```
> Cac <- cbind( 1, Ns(                aa  ,knots=seq(15,65,5)),
+                  Ns(rep(1950,length(aa)),knots=seq(1880,1970,5)) )
> cc <- 1870:1980
> Cc <- Ns(                cc  ,knots=seq(1880,1970,5))
> Rc <- Ns(rep(1950,length(cc)),knots=seq(1880,1970,5))
> head( cbind(cc,Cc,Rc), 4 )
```

```
        cc 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15        16        17        18 1 2 3 4 5 6 7 8 9 10 1
  [1,] 1870 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0 0.5070926 -1.521278 1.0141851 0 0 0 0 0 0 0 0 0  0
  [2,] 1871 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0 0.4563833 -1.369150 0.9127666 0 0 0 0 0 0 0 0 0  0
  [3,] 1872 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0 0.4056740 -1.217022 0.8113481 0 0 0 0 0 0 0 0 0  0
  [4,] 1873 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0 0.3549648 -1.064894 0.7099296 0 0 0 0 0 0 0 0 0  0
            12        13        14 15 16 17 18
  [1,] 0.1666667 0.6666667 0.1666667  0  0  0  0
  [2,] 0.1666667 0.6666667 0.1666667  0  0  0  0
  [3,] 0.1666667 0.6666667 0.1666667  0  0  0  0
  [4,] 0.1666667 0.6666667 0.1666667  0  0  0  0
```

```
> ci.exp( mscs, subset="C" )
```

```
                                      exp(Est.)      2.5%    97.5%
  Ns(C, knots = seq(1880, 1970, 5))1  0.7110146 0.3922167 1.288935
  Ns(C, knots = seq(1880, 1970, 5))2  1.0213110 0.6544218 1.593890
  Ns(C, knots = seq(1880, 1970, 5))3  0.9380898 0.6021630 1.461419
  Ns(C, knots = seq(1880, 1970, 5))4  0.9363640 0.6460472 1.357142
  Ns(C, knots = seq(1880, 1970, 5))5  1.1152683 0.7890542 1.576347
  Ns(C, knots = seq(1880, 1970, 5))6  1.2746243 0.9232362 1.759752
  Ns(C, knots = seq(1880, 1970, 5))7  1.0816587 0.7883781 1.484041
  Ns(C, knots = seq(1880, 1970, 5))8  1.7171832 1.2666307 2.328001
  Ns(C, knots = seq(1880, 1970, 5))9  1.7820025 1.3182664 2.408870
  Ns(C, knots = seq(1880, 1970, 5))10 2.1259480 1.5817614 2.857356
  Ns(C, knots = seq(1880, 1970, 5))11 2.5524958 1.9078097 3.415034
  Ns(C, knots = seq(1880, 1970, 5))12 1.6968365 1.2728651 2.262026
  Ns(C, knots = seq(1880, 1970, 5))13 3.1690225 2.3830575 4.214210
  Ns(C, knots = seq(1880, 1970, 5))14 3.1893629 2.3986732 4.240693
  Ns(C, knots = seq(1880, 1970, 5))15 4.6600832 3.5119014 6.183652
  Ns(C, knots = seq(1880, 1970, 5))16 4.2740883 3.2055953 5.698733
  Ns(C, knots = seq(1880, 1970, 5))17 4.1844023 3.2491442 5.388872
  Ns(C, knots = seq(1880, 1970, 5))18 4.8533026 3.5953904 6.551318
```

With the model and the contrast matrices in place we can plot estimated rates in the 1050 cohort and the RR of other birth cohorts relative to this:

```
> par( mfrow=c(1,2) )
> matplot( aa, ci.exp( mscs, ctr.mat=Cac )*10^5,
+          log="y", xlab="Age",
+          ylim=c(1.5,30), xlim=c(0,80),
+          ylab="Testis cancer incidence rate per 100,000 PY in 1950 cohort",
+          type="l", lty=1, lwd=c(3,1,1), col="black" )
> matplot( cc, ci.exp( mscs, subset="C", ctr.mat=Cc-Rc ),
+          log="y", xlab="Date",
+          ylim=c(1.5,30)/sqrt(1.5*30), xlim=c(0,80)+1890,
+          ylab="Testis cancer incidence RR",
+          type="l", lty=1, lwd=c(3,1,1), col="black" )
> abline( h=1, v=1950 )
> abline( v=c(1914,1919,1940,1946), col="green" )
```

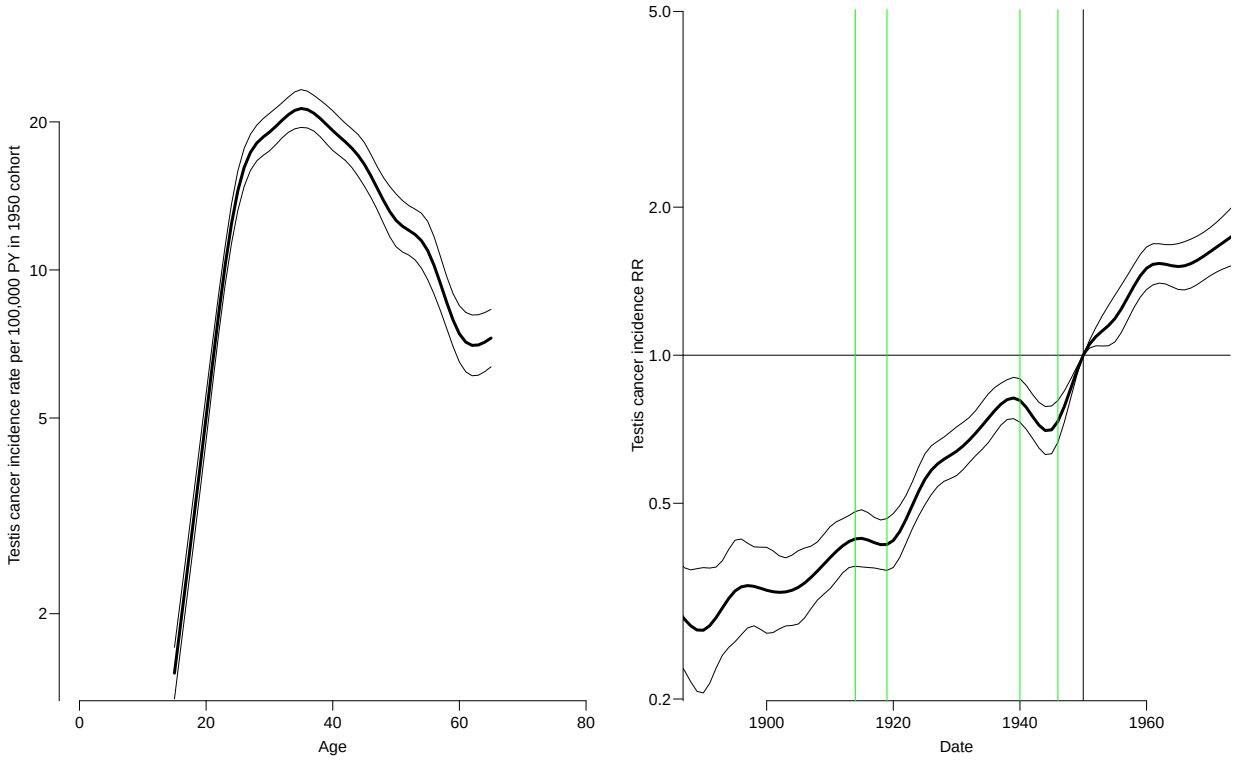


Figure 10.9: *Age-specific rates in the 1950 cohort, and RR relative to this cohort. The green vertical lines indicate the two world wars.*

Although the cohort effect is clearly over-modelled here it is clearly apparent that there is a dip in incidence rates for men born during the 2ndWW 1940–45, and there seems to be a weak tendency of the same for men born during the 1stWW 1914–18.

10.2 Diagnostic criteria in the NDR

This exercise is based on the Danish National Diabetes Register (NDR) — or to be precise a 10% bogus sample. The term "bogus" refers to the fact that the database we shall use is a 10% sample of the NDR, were all dates have been changes randomly by a quantity in the range from −7 to 7 days, so no person is identifiable, but we still have a database that behaves as the NDR in terms of incidence, mortality etc.

1. First load the Epi-package, and then get the 10% sample of the NDR from the web — the variable names are as in the original NDR, but all dates have been altered. Note that you *must* include the web-address in a `url()`-command, moreover the web-address is case-sensitive:

```
> library( Epi )
> clear()
> load( file=url("http://BendixCarstensen.com/DMreg/data/NDR2011-bogus.Rda") )
> lls()
```

2. Now take a look at the data frame, and convince yourself that the missingness-pattern is ok:

```
> str( ndr )
> summary( ndr )
```

3. Note that all the dates are factors, which is a bit impractical, so we convert them to fractional years by `cal.yr`. This is most easily done in a loop over all the date variables. R is so friendly that you can use the variable names in commands too; the date varaibels are those that start with `D_`; and `grep` is the function that seraces for a particular string in a character vector; try:

```
> names( ndr )
> grep( "D_", names(ndr) )
```

   This is a way to get the *position number* of the date variables, so conversion is dead-easy now:

```
> wh <- grep( "D_", names(ndr) )
> for( i in wh ) ndr[,i] <- cal.yr( ndr[,i] )
> head( ndr )
```

   We can use `cal.yr` without firther arguments because the dates are in standard ISO-format (`YYYY-MM-DD`), othrwise we would need the `format` argument.

4. It is a bit annoying with upper-case names and with the underscores, so we can just change the variable names by removing the first two letters and putting the rest to lower case (the 10 is just to cut the names short for convenience). And then we save the groomed register

```
> names( ndr ) <- tolower( substr(names(ndr),3,10) )
> names( ndr )
> save( ndr, file="ndr.Rda" )
```

10.2.1 Blood glucose criteria in NDR

It has been debated whether the blood glucose criteria shoudl be used at all, and we therefore create a new version of the register by computing a new date of inclusion and a new inclusion criterion, *as if* the two glucose citeria were omitted:

5. Now create two new variables, `new.aars` and `new.dto`, based only on LPR, foot therapy and medication:

```
> load( file="ndr.Rda" )
> ndr$new.dto <- with( ndr, pmin( fodt, lpr, oad, ins, na.rm=TRUE ) )
> # Assigning levels is a bit more tricky
> ndr$new.aars                          <- NA
> ndr$new.aars[ndr$new.dto==ndr$fodt]<- "fodt"
> ndr$new.aars[ndr$new.dto==ndr$oad] <- "oad"
> ndr$new.aars[ndr$new.dto==ndr$ins] <- "ins"
> ndr$new.aars[ndr$new.dto==ndr$lpr] <- "lpr"
> summary( ndr )
```

6. Now make a table of changes:

```
> addmargins( with( ndr, table( inklaars, new.aars, useNA="ifany" ) ) )
```

   Why have some non-clucose inclusions changed?

7. The question of real interest is another: How does the exclusion from the register depend on age and date. This is just a logistic regression:

```
> ndr <- transform( ndr, A = inkldto-foddto )
> mF <- glm( is.na(new.aars) ~ A + I(A^2) + I(A^3),
+            family = binomial,
+            data = subset( ndr, sex=="K" ) )
> summary( mF )
```

   We dont get much wiser from that.

8. Instead, predict the probability as a function of `A`, that we put in a *prediction data frame* — it muts contain variables with the same names as the *explanatory* variables in the model:

```
> nd <- data.frame( A=5:90 )
> pr.F <- predict( mF, newdata=nd, type="response" )
> plot( nd$A, pr.F, type="l" )
```

9. But we would like to make it a bit more flexible, so we put in a natural spline of age instaed:

```
> library( splines )
> a.kn <- c(10,20,25,30,35,4:9*10)
> mF <- glm( is.na(new.aars) ~ Ns( A, knots=a.kn ),
+            family = binomial,
+            data = subset( ndr, sex=="K" ) )
> summary( mF )
> pr.F <- predict( mF, newdata=nd, type="response" )
> plot( nd$A, pr.F, type="l" )
```

10. But we would alos like to see if it depends on time, so we include the data of diagnosis, `inkldto`, in the model too:

```
> p.kn <- seq(1996,2010,,4)
> mF <- glm( is.na(new.aars) ~ Ns( A, knots=a.kn ) +
+                              Ns( inkldto, knots=p.kn ),
+            family = binomial,
+            data = subset( ndr, sex=="K" ) )
> summary( mF )
> nd <- data.frame( A=5:90, inkldto=2000 )
> pr.F <- predict( mF, newdata=nd, type="response" )
> plot( nd$A, pr.F, type="l" )
```

11. But this only gives the prediction for the year 2000, we would like to see it for all years. In order to use `matplot`, we would like to have the predictions of the age-specific probabilities next to each oter in a matrix. We use `cbind` for that:

```
> pr.F <- NULL
> for( p in 1996:2011 )
+    {
+    nd <- data.frame( A=5:90, inkldto=p )
+    pr.F <- cbind( pr.F, predict( mF, newdata=nd, type="response" ) )
+    }
> matplot( nd$A, pr.F,
+          type="l", lty=1, lwd=1:2, col=heat.colors(25)[1:16] )
> text( c(80,80), c(55,60)/100, c(1996,2011),
+       col=heat.colors(25)[c(1,16)], font=2 )
```

12. But this is a model that assumes that the age effects is the same in all years, so we explore an inateraction model:

```
> iF <- glm( is.na(new.aars) ~ Ns( A, knots=a.kn )*Ns( inkldto, knots=p.kn ),
+            family = binomial,
+            data = subset( ndr, sex=="K" ) )
> summary( iF )
> anova( mF, iF, test="Chisq" )
```

The nice thing here is that the prediction machinery is eactly the same:

```
> pr.F <- NULL
> for( p in 1996:2011 )
+    {
+    nd <- data.frame( A=5:90, inkldto=p )
+    pr.F <- cbind( pr.F, predict( iF, newdata=nd, type="response" ) )
+    }
> matplot( nd$A, pr.F,
+          type="l", lty=1, lwd=1:2, col=heat.colors(25)[1:16] )
> text( c(80,80), c(55,60)/100, c(1996,2011),
+       col=heat.colors(25)[c(1,16)], font=2 )
```

The graph now really indicates that there is an interaction!

Bibliography

[1] Martyn Plummer and Bendix Carstensen. Lexis: An R class for epidemiological studies with long-term follow-up. *Journal of Statistical Software*, 38(5):1–12, 1 2011.

[2] Bendix Carstensen and Martyn Plummer. Using Lexis objects for multi-state models in R. *Journal of Statistical Software*, 38(6):1–18, 1 2011.