Statistical Practice in Epidemiology with 

Computer exercises

http://BendixCarstensen.com/SPE

University of Tartu, Estonia

May 2015

(compiled at Tuesday 26th May, 2015, 10:53)

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

Krista Fischer — Estonian Genome Center, University of Tartu, Estonia.
Krista.Fischer@ut.ee

Esa Läärä — Department of Mathematical Sciences, University of Oulu, Finland
Esa.Laara@oulu.fi
http://math.oulu.fi/en/personnel/esalaara.html

Martyn Plummer — International Agency for Research on Cancer, Lyon, France
plummerM@iarc.fr

Janne Pitkäniemi — Finnish Cancer Registry
janne.pitkaniemi@cancer.fi

Contents

Program

Daily timetable

9:00 – 9:30	Recap of yesterday's practicals
9:30 – 10:30	Lecture
10:30 – 10:50	Coffee break
10:50 – 12:50	Practical
12:50 – 14:00	Lunch
14:00 – 14:30	Recap of morning's practical
14:30 – 15:30	Lecture
15:30 – 16:00	Tea break
16:00 – 18:00	Practical

Friday 22 May

9:00 – 9:15	Welcome (KF)
9:15 – 10:30	Introduction to R language and commands reading data (MP)
10:30 – 10:50	Coffeee break
10:50 – 12:50	Practical: Practice with basic R Simple reading and data input
12:50 – 14:00	Lunch
14:00 – 14:30	Recap of morning practical
14:30 – 15:30	Language, indexing, `subset()`, `ifelse()`, `attach()`, `detach()`, `search`. Simple simulation. Simple graphics. (KF)
15:30 – 16:00	Tea break
16:00 – 18:00	Practical: Simple simulation Tabulation Introduction to graphs in R
18:00 – 19:00	Tour of the genome center before the
19:00 – 21:00	Welcome reception at the Estonian Genome Center (Eesti Geenivaramu), Riia 23b.

Saturday 23 May

9:00 – 9:30	Recap of yesterday's practicals.
9:30 – 10:30	Poisson regression for follow-up studies — likelihood for a constant rate Logistic regression for cc-studies (JP)
10:30 – 10:50	Coffeee break
10:50 – 12:50	Practical: Rates, rate ratio and rate difference with `glm` Logistic regression with `glm`
12:50 – 14:00	Lunch
14:00 – 14:30	Recap of morning practical
14:30 – 15:30	Linear models, fitting, predictions from models, reference category, using a quadratic and graphing it. A simple spline example. `predict`, `ci.lin`, `ci.exp`. (EL)
15:30 – 16:00	Tea break
16:00 – 18:00	Practical: Simple estimation of effects Estimation and reporting of linear and curved effects

Sunday 24 May

9:00 – 9:30	Recap of yesterday's practicals
9:30 – 10:30	More advanced graphics in R, including `ggplot2` (MP)
10:30 – 10:50	Coffee break.
10:50 – 12:50	Practical: Graphical meccano
12:50 – 14:00	Lunch

Monday 25 May

9:00 – 9:30	Recap of yesterday's practicals
9:30 – 10:30	Survival analysis: Kaplan Meier & simple Cox-model. Simple competing risks and relative survival. (JP)
10:30 – 10:50	Tea break
10:50 – 12:50	Practical: Survival and competing risks in oral cancer. Relative survival.
12:50 – 14:00	Lunch
14:00 – 14:30	Recap of morning practical
14:30 – 15:30	Dates in R; follow up representation in `Lexis` objects, time-splitting, multistate model and SMR. (BxC)
15:30 – 16:00	Coffee break.
16:00 – 18:00	Practical: Time-splitting and SMR (Danish diabetes patients)

Tuesday 26 May

9:00 – 9:30	Recap of yesterday's practicals
9:30 – 10:30	Nested and matched cc-studies & Case-cohort studies (EL)
10:30 – 10:50	Coffee break.
10:50 – 12:50	Practical: CC study: Risk factors for Coronary heart disease
12:50 – 14:00	Lunch
14:00 – 14:30	Recap of morning practical
14:30 – 15:30	Causal inference. (KF)
15:30 – 16:00	Coffee break.
16:00 – 18:00	Practical: Simulation and causal inference
19:00 –	Course dinner at Atlantis

Wednesday 27 May

9:00 – 9:30	Recap of yesterday's practicals
9:30 – 10:30	Multistate models, Poisson models for rates and simulation of `Lexis` objects (BxC)
10:30 – 10:50	Coffee break.
10:50 – 12:30	Practical: Multistate-model: Renal complications
12:30 – 13:00	Recap of morning practical
13:00 – 13:15	Wrap-up and farewell.
13:15 – 14:15	Lunch
Afternoon	Post-mortem (Faculty only).
Evening	Faculty dinner.

Further material will appear at this year's course website:
`http://bendixcarstensen.com/SPE/2015`

Chapter 1

Exercises

Datasets for the practicals in this course will be available on the local machines and on the course homepage, in http://BendixCarstensen.com/SPE/data. This is where you will also find the "household" scripts designed to save you typing.

The R-scripts used during the course for the recaps in the morning will be available in http://BendixCarstensen.com/SPE/recap.

The general convention is that when R-functions are mentioned in the text they will normally not be explained in any great detail. Hence you should get into the habit of consulting the help page for any function that you are not entirely familiar with. Either by using the help available through Help → Html help in the title bar, or by typing one of:

```
?Lexis
args( Lexis )
```

The first form brings up a help-page and the second just a listing of the function arguments with their defaults (without any explanation).

at the end of each help-page is (normally) an example showing some aspects of the use of the function. This example can be run in your R-session by typing:

```
example( Lexis )
```

This has the advantage that you can play around with the function, because the data structures used for illustration will be available in your R-session.

When running the exercises it is a good idea to have some text-editor open too, where you keep the R-commands. You can the cut and paste from this into the R-window and vice versa. Note also the File → Save History... possibility from the R command window, which allows you to dump all the commands you have written there so far into a file.

1.1 Practice with basic R

The main purpose of this session is to give participants who have not had much (or any) experience with using R a chance to practice the basics and to ask questions.

R can be installed on all major platforms. We do not assume in this exercise that you are using any particular platform or graphical user interface (GUI). Some GUIs will allow you to perform certain operations using a menu rather than typing commands (e.g. stopping R or changing your working directory). If you are using a GUI, take the time to discover the facilities it offers.

1.1.1 The working directory

A key concept in R is the *working directory* (or *folder* in the terminology of Windows). The working directory is where R looks for data files that you may want to read in and where R will write out any files you create. It is a good idea to keep separate working directories for different projects. In this course we recommend that you keep a separate working directory for each exercise.

If you are using a GUI then you will typically need to change to the correct working directory after starting R. You can do this from the menu system (Look under the "File" menu). If you are working on the command line in a terminal, then you can change to the correct working directory and then launch R by typing "R". In either case you quit R by typing

```
q()
```

at the R command prompt. You will be asked if you want to save your workspace. We recommend that you answer "no" to this question. If you answer "yes" then R will write a file named `.RData` into your workspace containing all the objects you created during your session so that they will be available next time you start R. This may seem convenient but you will soon find that your workspace becomes cluttered with old objects.

You can display the current working directory with the `getwd()` function and set it with the `setwd()` function. The command `dir()` can be used to see what files you have in the working directory.

1.1.2 Read-evaluate-print

The simplest use of R is interactively. R will read in the command you type, evaluate them, then print out the answer. This is called the *read-eval-print loop*, or REPL for people who don't like words. In this exercise, we recommend that you work interactively. As the course evolves you will find that you need to switch to *script files*. We come back to this issue at the end of the exercise.

It is important to remember that 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.

1.1.3 Using R as a calculator

Try using R as a calculator by typing different arithmetic expressions on the command line. Note that R allows you to recall previous commands using the vertical arrow key. You can edit a recalled command and then resubmit it by pressing the return key. Keeping that in mind, try the following:

```
12+16
(12+16)*5
sqrt((12+16)*5)  # square root
round(sqrt((12+16)*5),2)  # round to two decimal places
```

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

```
a <-  round(sqrt((12+16)*5),2)
```

In this case R does not print anything to the screen. You can see the results of the calculation, stored in the object `a`, by typing `a` and also use `a` for further calculations, e.g:

```
exp(a)
log10(a)   # log to the base 10
```

The left arrow expression `<-`, pronounced "gets", is called the assignment operator, and is obtained by typing `<` followed by `-` (with no space in between). It is also possible to use the equals sign `=` for assignment. Note that some R experts do not like this and recommend to use only "gets" for assignment, reserving `=` for function arguments,

You can also use a right arrow, as in

```
round(sqrt((12+16)*5),2) -> 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

```
pnorm(1.96, mean=0, sd=2)
```

will return the probability below 1.96 for a normal distribution with mean 0 and standard deviation 2. There is a number of different probability distributions implemented. For instance,

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

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

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

will return the probability above 3.84. You can also find quantiles for given probabilities:

```
qnorm(0.999)
qnorm(0.999, mean=0, sd=2)
qexp(0.999)
```

Exercise 1.

1. Find the probability above 4.3 in a chi-squared distribution on 1 degree of freedom.
2. Find the 75% quantile of the standard normal distribution

1.1.4 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.

1.1.5 Sequences

There are short-cut functions for creating vectors with a regular structure. For example, if you want a vector containing the sequence of integers from 1 to 10, you can use

```
> 1:10
```

The `seq()` function allows the creation of more general sequences. For example, the vector (15, 20, 25, ... ,85) can be created with

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

The objects created by the ":" operator and the `seq()` function are ordinary vectors, and can be combined with other vectors using the combine function:

```
> c(5,seq(from=20, to=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.

1. Create a vector `w` with components 1, -1, 2, -2
2. Display this vector
3. Obtain a description of `w` using `str()`
4. Create the vector `w+1`, and display it.
5. Create the vector `v` with components (0, 1, 5, 10, 15, ... , 75) using c() and seq().
6. Find the length of this vector.

1.1.6 Displaying and changing parts of a vector (indexing)

First try to understand the following commands:

```
>  x <- c(2, 7, 0, 9, 10, 23, 11, 4, 7, 8, 6, 0)
>  x[4]
>  x[3:5]
>  x[c(1,5,8)]
>  x[(1:6)*2]
>  x[-1]
```

You will see that you can extract elements of a vector by supplying a vector of integer indices inside square brackets. You can extract a single element by giving a scalar index (e.g. `x[4]`) and extract a sub-vector by supplying any expression that creates an integer vector. Negative subscripts mean "drop this element". So `x[-1]` returns every element of `x` except the first.

R also allows *logical subscripting*. Try the following

```
>  x > 10
>  x[x > 10]
```

The first expression creates a logical vector of the same length as `x`, where each element has the value `TRUE` or `FALSE` depending on whether or not the corresponding element of `x` is greater than 10. If you supply a logical vector as an index, R selects only those elements for which the conditions is `TRUE`

Indexing can also be used to replace parts of a vector:

```
> x[1] <- 1000
> x
```

This replaces the first element of `x`. Logical subscripting is useful for replacing parts of a vector that satisfy a certain condition:

```
>  x[x==0] <- 1
>  x
```

If you want to create an entirely new vector based on some logical condition then the `ifelse()` function

```
>  ifelse(round(x/2)==x/2,"even","odd")
```

Now try the following:

1. Display every third element in x
2. Display elements that are less than 10, but greater than 4
3. Modify the vector x, replacing by 10 all values that are greater than 10
4. Modify the vector x, multiplying by 2 all elements that are smaller than 5
5. Create a new vector y with elements 0,1,0,1, ... (12 elements) and a vector z that equals x when y=0 and 3x when y=1. (You can do it using `ifelse`, but there are other possibilities)

1.1.7 Data frames, indexing and subsets

Start with creating a simple data frame:

```
> mydata <- data.frame(name=c("Joe","Ann","Jack","Tom"),
+                      age=c(34,50,27,42),sex=c(1,2,1,1),
+                      height=c(185,170,175,182))
```

See what happens (and why):

```
>  mydata
>  names(mydata)
>  mydata[,"age"]
>  mydata[2,3]
>  mydata[,2]
>  mydata[1,]
```

Data frames are two-dimensional objects so you can use square brackets with two comma-separated arguments to take subsets. There are other ways to extract a whole column:

```
>  mydata[[2]]
>  mydata$age
```

Often it is useful to have row names defined as unique subject indicators (or names):

```
>  rownames(mydata)<-mydata$name
>  mydata["Tom",]
```

Note that only values that are unique for each individual can be row (or column) names.
Now let's create another data frame with more individuals than the in first one:

```
> weights<-data.frame(weight=c(67,81,56,90,72,79,69))
> rownames(weights)=c("Ann","Peter","Sue","Jack","Tom","Joe","Jane")
> weights[substr(rownames(weights),1,1)=="J",]
```

How to add weights to individuals in mydata, using the new data frame?

```
> mydata$weight<- weights[rownames(mydata),"weight"]
```

(There is also the function `merge()` to join datasets – see `help(merge)`).

Exercise 3. Using the same idea, add the variable `height` to the dataset `weights`. See what happens with the individuals who are not in `mydata`.

The subset function is another way of getting subsets from a data frame. To select all subjects with height less than 180 cm we use:

```
> subset(mydata, height < 180)
```

The subset function is usually clearer than the equivalent code using []:

```
> mydata[mydata$height < 180, ]
```

Another advantage of subset is that it will drop observations with missing values. In the example above, if a height is missing then subset will drop that row. But [] will do something you might not expect. It will include the row with missing `height`, but will replace every element in that row with the missing value NA.

Exercise 4. Set the height of the 2nd observation to missing with `mydata$height[2] <- NA` and compare the two subset expressions.

1.1.8 Working with "real" data frames

We shall use the births data which concern 500 mothers who had singleton births in a large London hospital. The outcome of interest is the birth weight of the baby, also dichotomised as normal or low birth weight. These data are available in the Epi package:

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

The function `objects()` shows what is in your workspace. To find out a bit more about `births` try

```
help(births)
```

```
> names(births)
> head(births)
```

Exercise 5.

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)
   ```
   Check that the object `diet` is not any more there.

1.1.9 Referencing parts of the data frame (indexing)

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. Alternatively

```
> births[1,2]
```

will list the value taken by the first subject for the second variable (which is `bweight`). 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"]
```

To list the data for the first subject on all variable except the second try

```
> births[1 , -2]
```

Exercise 6.

1. Display the data on the variable `gestwks` for row 7 in the `births` data frame.
2. Display all the data in row 7.
3. Display the first 10 rows of the data on the variable `gestwks`.

1.1.10 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)
```

Alternatively you can use

```
> with(births, summary(hyp))
```

In most datasets there will be some missing values. The summary shows the number of missing (`NA`) values for each variable.

1.1.11 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

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

Alternatively you can use

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

Note that either way `hyp` is now a factor with two levels, labelled `"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$hyp <- factor(births$hyp, labels=c("normal","hyper"))
> str(births)
```

or

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

Exercise 7.

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

1.1.12 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 simple frequency tables is `table`. The distribution of the factor `hyp` can be viewed using

```
> with(births, table(hyp))
```

or by specifying the data frame as in

```
> table(births$hyp)
```

For simple expressions the choice is a matter of taste, but `with` is preferable for more complicated expressions.

Exercise 8.

1. Find the frequency distribution of `sex`.
2. Find the two-way frequency distribution of `sex` and `hyp`.

1.1.13 Grouping the values of a numeric 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))
> with(births, table(agegrp))
```

By default the factor levels are labelled `[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.

Observations which are not inside the range specified in the `breaks()` part of the command result in missing values for the new factor. 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))
> with(births, table(agegrp))
```

shows 5 intervals of width 4.

Exercise 9.

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.

1.1.14 Generating new variables

New variables can be produced using assignment together with the usual mathematical operations and functions. For example

```
> logbw <- log(births$bweight)
```

produces the variable `logbw` in your work space (Global environment), while

```
> births$logbw <- log(births$bweight)
```

produces the variable `logbw` in the `births` data frame. Logs base 10 are obtained with `log10( )`.

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$vlow <- births$bweight<2000
> str(births)
```

creates a logical variable `vlow` (in `births` with values TRUE and FALSE, according to whether `bweight` is less than 2000 or not. One common use of logical variables is to restrict a command to a subset of the data. For example, to create a new dataframe restricted to women with babies of very low birth weight, try

```
> births.low<-subset(births, bweight<2000)
> summary(births.low)
```

Exercise 10.

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

1.1.15 Where am I?

You can save any R object to disc. For example, to save the data frame `births` try

```
> save(births, file="births2")
```

which will save the births data frame in the file `births2`. 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("births2")
```

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

1.1.16 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 `ls()` does the same as `objects()`. To see what is in the Epi package, try

```
> objects(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.

1.1.17 Attaching a data frame

The function `objects()` shows that the data frame `births` is in your workspace. To refer to variables in `births` 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 11.

1. Use `search()` to make sure you have no data frames attached.
2. Use `objects()` 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.

1.1.18 Interactive use vs scripting

You can work with R simply by typing function calls at the command prompt and reading the results as they are printed. This is OK for simple use but rapidly becomes cumbersome. If the results of one calculation are used to feed into the next calculation, it can be difficult to go back if you find you have made a mistake, or if you want to repeat the same commands with different data.

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. If you are using a GUI then you can use the built-in script editor, or you can use your favourite text editor instead if you prefer.

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.

1.2 Reading data into R

1.2.1 Introduction

It is said that Mrs Beeton, the 19th century cook and writer, began her recipe for rabbit stew with the instruction "First catch your rabbit". Sadly, the story is untrue, but it does contain an important moral. R is a language and environment for data analysis. If you want to do something interesting with it, you need data.

For teaching purposes, data sets are often embedded in R packages. The base R distribution contains a whole package dedicated to data which includes around 100 data sets. This is attached towards the end of the search path, and you can see its contents with

```
> objects("package:datasets")
```

A description of all of these objects is available using the `help()` function. For example

```
> help(Titanic)
```

gives an explanation of the `Titanic` data set, along with references giving the source of the data.

The Epi package also contains some data sets. These are not available automatically when you load the Epi package, but you can make a copy in your workspace using the `data()` function. For example

```
> library(Epi)
> data(bdendo)
```

will create a data frame called `bdendo` in your workspace containing data from a case-control study of endometrial cancer. Datasets in the Epi package also have help pages: type `help(bdendo)` for further information.

To go back to the cooking analogy, these data sets are the equivalent of microwave ready meals, carefully packaged and requiring minimal work by the consumer. Your own data will never be able in this form and you must work harder to read it in to R.

This exercise introduces you to the basics of reading external data into R. It consists of reading the same data from different formats. Although this may appear repetitive, it allows you to see the many options available to you, and should allow you to recognize when things go wrong.

You will need the following files in the sub-directory `data` of your working directory: `fem.dat`, `fem-dot.dat`, `fem.csv`, `fem.dta` (Reminder: use `setwd()` to set your working directory).

1.2.2 Data sources

Sources of data can be classified into three groups:

1. Data in human readable form, which can be inspected with a text editor.
2. Data in binary format, which can only be read by a program that understands that format (SAS, SPSS, Stata, Excel, ...).
3. Online data from a database management system (DBMS)

This exercise will deal with the first two forms of data. Epidemiological data sets are rarely large enough to justify being kept in a DBMS. If you want further details on this topic, you can consult the "R Data Import/Export" manual that comes with R.

1.2.3 Data in text files

Human-readable data files are generally kept in a rectangular format, with individual records in single rows and variables in columns. Such data can be read into a data frame in R.

Before reading in the data, you should inspect the file in a text editor and ask three questions:

1. How are columns in the table separated?
2. How are missing values represented?
3. Are variable names included in the file?

The file `fem.dat` contains data on 118 female psychiatric patients. The data set contains nine variables.

ID	Patient identifier
AGE	Age in years
IQ	Intelligence Quotient (IQ) score
ANXIETY	Anxiety (1=none, 2=mild, 3=moderate,4=severe)
DEPRESS	Depression (1=none, 2=mild, 3=moderate or severe)
SLEEP	Sleeping normally (1=yes, 2=no)
SEX	Lost interest in sex (1=yes, 2=no)
LIFE	Considered suicide (1=yes, 2=no)
WEIGHT	Weight change (kg) in previous 6 months

Inspect the file `fem.dat` with a text editor to answer the questions above.

The most general function for reading in free-format data is `read.table()`. This function reads a text file and returns a data frame. It tries to guess the correct format of each variable in the data frame (integer, double precision, or text).

Read in the table with:

```
> fem <- read.table("./data/fem.dat", header=TRUE)
```

Note that you must assign the result of `read.table()` to an object. If this is not done, the data frame will be printed to the screen and then lost.

You can see the names of the variables with

```
> names(fem)
```

The structure of the data frame can be seen with

```
> str(fem)
```

You can also inspect the top few rows with

```
> head(fem)
```

Note that the IQ of subject 9 is -99, which is an illegal value: nobody can have a negative IQ. In fact -99 has been used in this file to represent a missing value. In R the special value `NA` ("Not Available") is used to represent missing values. All R functions recognize `NA` values and will handle them appropriately, although sometimes the appropriate response is to stop the calculation with an error message.

You can recode the missing values with

```
> fem$IQ[fem$IQ == -99] <- NA
```

Of course it is much better to handle special missing value codes when reading in the data. This can be done with the `na.strings` argument of the `read.table()` function. See below.

1.2.4 Things that can go wrong

Sooner or later when reading data into R, you will make a mistake. The frustrating part of reading data into R is that most mistakes are not fatal: they simply cause the function to return a data frame that is *not what you wanted.* There are three common mistakes, which you should learn to recognize.

Forgetting the headers

The first row of the file `fem.dat` contains the variable names. The `read.table()` function does not assume this by default so you have to specify this with the argument `header=TRUE`. See what happens when you forget to include this option:

```
> fem2 <- read.table("data/fem.dat")
> str(fem2)
> head(fem2)
```

and compare the resulting data frame with `fem`. What are the names of the variables in the data frame? What is the class of the variables?

> **Explanation:** Remember that `read.table()` tries to guess the mode of the variables in the text file. Without the `header=TRUE` option it reads the first row, containing the variable names, as data, and guesses that all the variables are character, not numeric. By default, all character variables are coerced to factors by `read.table`. The result is a data frame consisting entirely of factors (You can prevent the conversion of character variables to factors with the argument `as.is=TRUE`).

If the variable names are not specified in the file, then they are given default names `V1, V2, ...`. You will soon realise this mistake if you try to access a variable in the data frame by, for example

```
> fem2$IQ
```

as the variable will not exist

There is one case where omitting the `header=TRUE` option is harmless (apart from the situation where there is no header line, obviously). When the first row of the file contains **one less** value than subsequent lines, `read.table()` infers that the first row contains the variable names, and the first column of every subsequent row contains its **row name**.

Using the wrong separator

By default, `read.table` assumes that data values are separated by any amount of white space. Other possibilities can be specified using the `sep` argument. See what happens when you assume the wrong separator, in this case a tab, which is specified using the escape sequence `"\t"`

```
> fem3 <- read.table("data/fem.dat", sep="\t")
> str(fem3)
```

How many variables are there in the data set?

> **Explanation:** If you mis-specify the separator, `read.table()` reads the whole line as a single character variable. Once again, character variables are coerced to factors, so you get a data frame with a single factor variable.

Mis-specifying the representation of missing values

The file `fem-dot.dat` contains a version of the FEM dataset in which all missing values are represented with a dot. This is a common way of representing missing values, but is not recognized by default by the `read.table()` function, which assumes that missing values are represented by "NA".

Inspect the file with a text editor, and then see what happens when you read the file in incorrectly:

```
> fem4 <- read.table("data/fem-dot.dat", header=TRUE)
> str(fem4)
```

You should have enough clues by now to work out what went wrong.

You can read the data correctly using the `na.strings` argument

```
> fem4 <- read.table("data/fem-dot.dat", header=TRUE, na.strings=".")
```

1.2.5 Spreadsheet data

Spreadsheets have become a common way of exchanging data. All spreadsheet programs can save a single sheet in *comma-separated variable* (CSV) format, which can then be read into R. There are two functions in R for reading in CSV data: `read.csv()` and `read.csv2()`.

To understand why there are two functions, inspect the contents of the function `read.csv()` by typing its name

```
> read.csv
```

```
function (file, header = TRUE, sep = ",", quote = "\"", dec = ".",
    fill = TRUE, comment.char = "", ...)
read.table(file = file, header = header, sep = sep, quote = quote,
    dec = dec, fill = fill, comment.char = comment.char, ...)
<bytecode: 0x36dd1a8>
<environment: namespace:utils>
```

The first two lines show the arguments to the `read.csv()` function and their default values (`header=TRUE`, etc) The next two lines show the *body* of the function, which shows that the default arguments are simply passed verbatim onto the `read.table()` function. Hence `read.csv()` is a *wrapper* function that chooses the correct arguments for `read.table()` for you. You only need to supply the name of the CSV file and all the other details are taken care of.

Now inspect the `read.csv2` function to find the difference between this function and `read.csv`.

> **Explanation:** The CSV format is not a single standard. The file format depends on the *locale* of your computer – the settings that determine how numbers are represented. In some countries, the decimal separator is a point "." and the variable separator in a CSV file is a comma ",". In other countries, the decimal separator is a comma "," and the variable separator is a semi-colon ";". The `read.csv()` function is used for the first format and the `read.csv2()` function is used for the second format.

The file `fem.csv` contains the FEM dataset in CSV format. Inspect the file to work out which format is used, and read it into R.

On Microsoft Windows, you can copy values directly from an open Excel spreadsheet using the clipboard. Highlight the cells you want to copy in the spread sheet and select copy from the pull-down edit menu. Then type `read.table(file="clipboard")` to read the data in. Beware, however, that the clipboard on Windows operates on the WYSIWYG principle (what-you-see-is-what-you-get). If you have a value 1.23456789 in your spreadsheet, but have formatted the cell so it is displayed to two decimal places, then the value read into R will be the truncated value 1.23.

1.2.6 Binary data

The foreign package allows you to read data in binary formats used by other statistical packages. Since R is an open source project, it can only read binary formats that are themselves "open", in the sense that the standards for reading and writing data are well-documented. For example, there is a function in the foreign package for reading SAS XPORT files, a format that has been adopted as a standard by the US Food and Drug Administration (http://www.sas.com/govedu/fda/faq.html). However, there is no function in the foreign package for reading native SAS binaries (SAS7BDAT files). Other packages are available from CRAN (http://cran.r-project.org) that offer the possibility of reading SAS binary files: see the haven and sas7bdat packages.

The file `fem.dta` contains the FEM dataset in the format used by Stata. Read it into R with

```
> library(foreign)
> fem5 <- read.dta("data/fem.dta")
> head(fem5)
```

The Stata data set contains value and variable labels. Stata variables with value labels are automatically converted to factors.

There is no equivalent of variable labels in an R data frame, but the original variable labels are not lost. They are still attached to the data frame as an invisible *attribute*, which you can see with

```
> attr(fem5, "var.labels")
```

A lot of *meta-data* is attached to the data in the form of attributes. You can see the whole list of attributes with

```
> attributes(fem5)
```

or just the attribute names with

```
> names(attributes(fem5))
```

The `read.dta()` function can only read data from Stata versions 5–12. The R Core Team has not been able to keep up with changes in the Stata format. You may wish to try the haven package and the readstata13 package, both available from CRAN.

1.2.7 Summary

In this exercise we have seen how to create a data frame in R from an external text file. We have also reviewed some common mistakes that result in garbled data.

The capabilities of the foreign package for reading binary data have also been demonstrated with a sample Stata data set.

1.3 Simple simulation

Monte Carlo methods are computational procedures dealing with simulation of artificial data from given probability distributions with the purpose of learning about the behaviour of phenomena involving random variability. These methods have a wide range of applications in statistics as well as in several branches of science and technology. By solving the following exercises you will learn to use some basic tools of statistical simulation.

1. Whenever using a *random number generator* (RNG) for a simulation study, or for producing a randomization list to be used in a clinical trial, it is a good practice to set first the *seed*. It is a number that determines the initial state of the RNG, from which it starts creating the desired sequence of pseudo-random numbers. Explicit specification of the seed enables the reproducibility of the sequence. In serious applications (like in clinical trials) it is important that the seed (and the whole randomization list) is concealed from persons with certain important roles in the study (like the physicians who recruit or treat the patients in a clinical trial). – Instead of the number 5462319 below you may use your own seed of choice.

   ```
   > set.seed(5462319)
   ```

2. Generate a random sample of size 20 from a normal distribution with mean 100 and standard deviation 10. Draw a histogram of the sampled values and compute the conventional summary statistics

   ```
   >  x <- rnorm(20, 100, 10)
   >  hist(x)
   >  c(mean(x), sd(x))
   ```

   Repeat the above lines and compare the results.

3. Now replace the sample size 20 by 1000 and run again twice the previous command lines with this size but keeping the parameter values as before. Compare the results between the two samples here as well as with those in the previous item.

4. Generate a sample of size 1000 from a Uniform(0,1) distribution (`runif(1000)`) and look at the a) histogram of the original values, b) histogram of their natural logarithms, and c) histogram of the logit-transforms of the values.

   ```
   > x <- runif(1000)
   > hist(x)
   > hist(log(x))
   > hist(log(x/(1-x)))
   ```

5. Generate 500 observations from a Bernoulli(p) distribution, or Bin$(1, p)$ distribution, taking values 1 and 0 with probabilities p and $1 - p$, respectively, when $p = 0.4$:

   ```
   > X <- rbinom(500, 1, 0.4)
   > table(X)
   ```

6. Now generate another 0/1 variable Y, being dependent on previously generated X, so that $P(Y = 1|X = 1) = 0.2$ and $P(Y = 1|X = 0) = 0.1$.

```
> Y <- rbinom(500,1,0.1*X+0.1)
> table(X,Y)
> prop.table(table(X,Y),1)
```

7. Generate data obeying a simple linear regression model $y_i = 5 + 0.5x_i + \varepsilon_i$, $i = 1, \ldots 100$, in which $\varepsilon_i \sim N(0, 10^2)$, and x_i values are integers from 1 to 100. Plot the (x_i, y_i)-values, and estimate the parameters of that model.

```
> x <- 1:100
> y <- 5 + 0.5*x + rnorm(100,0,10)
> plot(x,y)
> abline(lm(y~x))
> summary(lm(y~x))$coef
```

8. Now change the slope coefficient of x to (a) 0.05, and (b) 0.01, respectively, and perform a similar simulation run. Do you still discover association between x and y? (Look at the scatter plot and the error margin of the slope coefficient) Check also, what happens if you change the standard deviation of the error term to be a) 1, and b) 100.

1.4 Tabulation

1.4.1 Introduction

R and its add-on packages provide several different tabulation functions with different capabilities. The appropriate function to use depends on your goal. There are at least three different uses for tables.

The first use is to create simple summary statistics that will be used for further calculations in R. For example, a two-by-two table created by the `table` function can be passed to `fisher.test`, which will calculate odds ratios and confidence intervals. The appearance of these tables may, however, be quite basic, as their principal goal is to create new objects for future calculations.

A quite different use of tabulation is to make "production quality" tables for publication. You may want to generate reports from for publication in paper form, or on the World Wide Web. The package `xtable` provides this capability, but it is not covered by this course.

An intermediate use of tabulation functions is to create human-readable tables for discussion within your work-group, but not for publication. The `Epi` package provides a function `stat.table` for this purpose, and this practical is designed to introduce this function.

1.4.2 The births data

We shall use the births data which concern 500 mothers who had singleton births in a large London hospital. The outcome of interest is the birth weight of the baby, also dichotomised as normal or low birth weight. These data are available in the Epi package:

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

In order to work with this data set we need to transform some of the variables into factors. This is done with the following commands:

```
> 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 use `str(births)` to examine the modified data frame. We have transformed the binary variables `hyp` and `sex` into factors with informative labels. This will help when displaying the tables. We have also created grouped variables `agegrp` and `gest4` from the continuous variables `matage` and `gestwks` so that they can be tabulated.

1.4.3 One-way tables

The simplest table one-way table is created by

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

This creates a count of individuals, classified by levels of the factor `sex`. Compare this table with the equivalent one produced by the `table` function. Note that `stat.table` has a `data` argument that allows you to use variables in a data frame without specifying the frame.

You can display several summary statistics in the same table by giving a list of expressions to the `contents` argument:

```
> stat.table(index = sex, contents = list(count(), percent(sex)), data=births)
```

Only a limited set of expressions are allowed: see the help page for `stat.table` for details.

You can also calculate marginal tables by specifying `margin=TRUE` in your call to `stat.table`. Do this for the above table. Check that the percentages add up to 100 and the total for `count()` is the same as the number of rows of the data frame `births`.

To see how the mean birth weight changes with `sex`, try

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

Add the count to this table. Add also the margin with `margin=TRUE`.

As an alternative to `bweight` we can look at `lowbw` with

```
> stat.table(index = sex, contents = percent(lowbw), data=births)
```

All the percentages are 100! To use the `percent` function the variable `lowbw` must also be in the index, as in

```
> stat.table(index = list(sex,lowbw), contents = percent(lowbw), data=births)
```

The final column is the percentage of babies with low birth weight by different categories of gestation.

1. Obtain a table showing the frequency distribution of `gest4`.
2. Show how the mean birth weight changes with `gest4`.
3. Show how the percentage of low birth weight babies changes with `gest4`.

Another way of obtaining the percentage of low birth weight babies by gestation is to use the ratio function:

```
> stat.table(gest4,ratio(lowbw,1,100),data=births)
```

This only works because `lowbw` is coded 0/1, with 1 for low birth weight.

Tables of odds can be produced in the same way by using `ratio(lowbw, 1-lowbw)`. The `ratio` function is also very useful for making tables of rates with (say) `ratio(D,Y,1000)` where `D` is the number of failures, and `Y` is the follow-up time. We shall return to rates in a later practical.

1.4.4 Improving the Presentation of Tables

The `stat.table` function provides default column headings based on the `contents` argument, but these are not always very informative. Supply your own column headings using *tagged* lists as the value of the `contents` argument, within a `stat.table` call:

```
> stat.table(gest4,contents = list( N=count(),
+       "(%)" = percent(gest4)),data=births)
```

This improves the readability of the table. It remains to give an informative title to the index variable. You can do this in the same way: instead of giving `gest4` as the `index` argument to `stat.table`, use a named list:

```
> stat.table(index = list("Gestation time" = gest4),data=births)
```

1.4.5 Two-way Tables

The following call gives a 2×2 table showing the mean birth weight cross-classified by `sex` and `hyp`.

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

Add the count to this table and repeat the function call using `margin = TRUE` to calculate the marginal tables.

Use `stat.table` with the ratio function to obtain a 2×2 table of percent low birth weight by `sex` and `hyp`.

You can have fine-grained control over which margins to calculate by giving a logical vector to the `margin` argument. Use `margin=c(FALSE, TRUE)` to calculate margins over `sex` but not `hyp`. This might not be what you expect, but the `margin` argument indicates which of the index variables are to be **marginalized out**, not which index variables are to remain.

1.4.6 Printing

Just like every other R function, `stat.table` produces an object that can be saved and printed later, or used for further calculation. You can control the appearance of a table with an explicit call to `print()`

There are two arguments to the print method for `stat.table`. The `width` argument which specifies the minimum column width, and the `digits` argument which controls the number of digits printed after the decimal point. This table

```
> odds.tab <- stat.table(gest4, list("odds of low bw" = ratio(lowbw,1-lowbw)),
+                data=births)
> print(odds.tab)
```

shows a table of odds that the baby has low birth weight. Use `width=15` and `digits=3` and see the difference.

1.5 Graphics 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.

1.5.1 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

```
> help(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 12.

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)")
   ```

1.5.2 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 centre 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, pch=1 )
```

1.5.3 Adding to a plot

The `points()` function just used 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")
```

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 of sex.

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 13.

1. Make a three level factor for maternal age with cutpoints at 30 and 40 years using the `cut` function. (Recall that the `breaks` argument must include lower and upper limits beyond the range of the data, or you will get some missing values).
2. Use this to make the plot of bweight versus gestational weeks with three different plotting symbols. (Hint: Indexing with a factor automatically gives indexes 1,2,3 etc.).

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. The `color` function (or `colour` function if you prefer) returns a vector of the colour names that R knows about. These names can also be used to specify colours.

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 )
```

1.5.4 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.

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), ]
   ```

1.5.5 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 (This graphics device is only available on Windows).

If you want exact control of the size of your plot-file you can start a non-interactive 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:

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

This will give you a portable document file `plot1.pdf` with a graph which is 3 inches tall and 4 inches wide.

1.5.6 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

```
> help(par)
```

or better, use the the html-version through Help → Html help → Packages → graphics → 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 columnwise, 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))
```

If you want a more detailed control over the layout of multiple graphs on a single page look at `?layout`.

1.6 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.

Let λ be the true hazard rate or theoretical incidence rate, its estimator being the empirical incidence rate $\widehat{\lambda} = D/Y =$ 'no. cases/person-years'. Recall that the standard error of the empirical rate is $\mathrm{SE}(\widehat{\lambda}) = \widehat{\lambda}/\sqrt{D}$.

The simplest approximate 95% confidence interval (CI) for λ is given by

$$\widehat{\lambda} \pm 1.96 \times \mathrm{SE}(\widehat{\lambda})$$

The standard error of the empirical log-rate $\widehat{\theta} = \log(\widehat{\lambda})$ is $\mathrm{SE}(\widehat{\theta}) = 1/\sqrt{D}$. Thus a simple approximate 95% confidence interval for the log-hazard $\theta = \log(\lambda)$ is obtained from

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

When taking the exponential from the above limits, we get another approximate confidence interval for the hazard λ itself:

$$\exp\{\log(\widehat{\lambda}) \pm 1.96/\sqrt{D}\} = \widehat{\lambda} \overset{\times}{\div} \underbrace{\exp(1.96/\sqrt{D})}_{\text{error factor, EF}}$$

This approach provides a more accurate approximation with very small numbers of cases. (However, both these methods fail when $D = 0$, in which case an *exact* method or one based on *profile-likelihood* is needed.)

1. Now, suppose you have 15 events during 5532 person-years. Let's use R as a simple desk calculator to derive the rate (in 1000 person-years) and two versions of an approximate confidence interval:

```
> library( Epi )
> options(digits=4)

> D <- 15
> Y <- 5.532    # thousands of years
> rate <- D / Y
> SE.rate <- rate/sqrt(D)
> c(rate, SE.rate, rate + c(-1.96, 1.96)*SE.rate )
> SE.logr <- 1/sqrt(D)
> EF <- exp( 1.96 * SE.logr )
> c(log(rate), SE.logr)
> c( rate, EF, rate/EF, rate*EF )
```

2. *This requires some familiarity with matrix algebra.* You can explore the function `ci.mat()`, which lets you use matrix multiplication (operator `'%*%'` in R) to produce confidence interval from an estimate and its standard error (or CIs from whole columns of estimates and SEs):

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

3. Now try to achieve the estimate of λ and its CI with a Poisson model. Use the number of events as the response and the log-person-years as *offset*:

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

   What is the interpretation of the parameter in this model?

4. You can extract CIs for the model parameters directly from the fitted model on the scale determined by the *link* function with the `ci.lin()`-function. Thus, the estimate, SE, and confidence limits for the log-rate are obtained by:

```
> ci.lin( mm )
```

   However, to get the confidence limits on the original scale, the results must be exp-transformed:

```
> ci.lin( mm, Exp=T)
```

   To get just the point estimate and CI for λ from log-transformed quantities you can use function `ci.exp()`, which is actually a wrapper of `ci.lin()`:

```
> ci.exp( mm)
> ci.lin( mm, Exp=T)[, 5:7]
```

   Both functions are found in the `Epi` package. – Note that the test statistic and P-value are rarely interesting quantities for a single rate.

5. There is an alternative way of fitting a Poisson model: Use the empirical rate $\widehat{\lambda} = D/Y$ as the *scaled* Poisson response, and the person-years as *weight* instead of offset (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)
```

   Verify that this gave the same results as above.

6. The advantage of this latter approach based on weighting is that it allows sensible use of the *identity* link — the response is the same but the parameter estimated is now the rate, not the log-rate:

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

   What is the meaning of the intercept in this model?

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

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

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

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

```
> Dx <- c(3,7,5)
> Yx <- c(1.412,2.783,1.337)
> Px <- 1:3
```

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

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

9. 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" )
```

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

What is the deviance good for?

10. 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 estimated rates, that is the log(RR), is:

$$\begin{aligned} \text{var}(\log(\text{RR})) &= \text{var}\{\log(\widehat{\lambda}_1/\widehat{\lambda}_0)\} \\ &= \text{var}\{\log(\widehat{\lambda}_1)\} + \text{var}\{\log(\widehat{\lambda}_0)\} \\ &= 1/D_1 + 1/D_0 \end{aligned}$$

Based on a similar argument as before, an approximate 95% CI for the true rate ratio λ_1/λ_0 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 CI by:

```
> D0 <- 15   ; D1 <- 28
> Y0 <- 5.532 ; Y1 <- 4.783
> RR <- (D1/Y1)/(D0/Y0)
> SE.lrr <- sqrt(1/D0+1/D1)
> EF <- exp( 1.96 * SE.lrr)
> c( RR, RR/EF, RR*EF )
```

Applying matrix multiplication with `ci.mat()` should give the same result:

```
> exp( c( log(RR), SE.lrr ) %*% ci.mat() )
```

11. Now achieve this using a Poisson model:

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

What do the parameters mean in this model?

You can extract the exponentiated parameters in two ways:

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

12. 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:

$$\begin{aligned} \text{var(RD)} &= \text{var}(\widehat{\lambda}_1) + \text{var}(\widehat{\lambda}_0) \\ &= D_1/Y_1^2 + D_0/Y_0^2 \end{aligned}$$

Use this formula to compute the rate difference and a 95% confidence interval for it:

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

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

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

14. Normally one would like to get both the rates and the ratio between them. This can be achieved in one go using the *contrast matrix* argument `ctr.mat` 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
> mm <- glm( D ~ factor(expos),
+               family=poisson,  offset=log(Y) )
> ci.exp( mm, ctr.mat=CM)
```

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(expos),
+                  family=poisson(link=identity), weight=Y )
> ci.lin( ma, ctr.mat=CM )[, c(1,5,6)]
```

1.7 Logistic regression (GLM)

1.7.1 Malignant melanoma in Denmark

In the mid-80s a case-control study on risk factors for malignant melanoma was conducted in Denmark (Østerlind et al. The Danish case-control study of cutaneous malignant melanoma I: Importance of host factors. *Int J Cancer* 1988; 42: 200-206).

The cases were patients with skin melanoma (excluding lentigo melanoma), newly diagnosed from 1 Oct, 1982 to 31 March, 1985, aged 20-79, from East Denmark, and they were identified from the Danish Cancer Registry.

The controls (twice as many as cases) were drawn from the residents of East Denmark in April, 1984, as a random sample stratified by sex and age (within the same 5 year age group) to reflect the sex and age distribution of the cases. This is called group matching, and in such a study, it is necessary to control for age and sex in the statistical analysis. (Yes indeed: In spite of the fact that stratified sampling by sex and age removed the statistical association of these variables with melanoma from the final case-control data set, the analysis must control for variables which determine the probability of selecting subjects from the base population to the study sample.)

The population of East Denmark is a dynamic one. Sampling the controls only at one time point is a rough approximation of incidence density sampling, which ideally would spread out over the whole study period. Hence the exposure odds ratios calculable from the data are estimates of the corresponding hazard rate ratios between the exposure groups.

After exclusions, refusals etc., 474 cases (92% of eligible cases) and 926 controls (82%) were interviewed. This was done face-to-face with a structured questionnaire by trained interviewers, who were not informed about the subject's case-control status.

For this exercise we have selected a few host variables from the study in an ascii-file, `melanoma.dat`. The variables are listed in table 1.1.

Table 1.1: *Variables in the melanoma dataset.*

Variable	**Units or Coding**	**Type**	**Name**
Case-control status	1=case, 0=control	numeric	`cc`
Sex	1=male, 2=female	numeric	`sex`
Age at interview	age in years	numeric	`age`
Skin complexion	0=dark, 1=medium, 2=light	numeric	`skin`
Hair colour	0=dark brown/black, 1=light brown, 2=blonde, 3=red	numeric	`hair`
eye colour	0=brown, 1=grey, green, 2=blue	numeric	`eyes`
Freckles	1=many, 2=some, 3=none	numeric	`freckles`
Naevi, small	no. naevi < 5mm	numeric	`nvsmall`
Naevi, largs	no. naevi ≥ 5mm	numeric	`nvlarge`

1.7.2 Reading the data

Start R and load the `Epi` package using the function `library()`. Read the data set from the file `melanoma.dat` (this should be in the subdirectory data of your working directory) to a data frame with name `mel` using the `read.table()` function. Remember to specify that missing values are coded ".", and that variable names are in the first line of the file. View the overall structure of the data frame, and list the first 20 rows of `mel`.

1.7.3 House keeping

The structure of the data frame `mel` tells us that all the variables are numeric (integer), so first you need to do a bit of house keeping. For example the variables `sex, skin, hair, eye` need to be converted to factors, with labels, and `freckles` which is coded 4 for none down to 1 for many (not very intuitive) needs to be recoded, and relabelled.

To avoid too much typing and to leave plenty of time to think about the analysis, these house keeping commands are in a script file called `melanoma-house.r`. You should study this script carefully before running it. The coding of `freckles` can be reversed by subtracting the current codes from 4. Once recoded the variable needs to be converted to a factor with labels "none", etc. Age is currently a numeric variable recording age to the nearest year, and it will be convenient to group these values into (say) 10 year age groups, using `cut`. In this case we choose to create a new variable, rather than change the original.

Look again at the structure of the data frame `mel` and note the changes. Use the command `summary(mel)` to look at the univariate distributions.

This is enough housekeeping for now - let's turn to something a bit more interesting.

1.7.4 One variable at a time

As a first step it is a good idea to start by looking at the numbers of cases and controls by each variable separately, ignoring age and sex. Try

```
> with(mel, table(cc,skin))
```

to see the numbers of cases and controls by skin colour, and

```
> stat.table(skin, contents=ratio(cc,1-cc),data=mel)
```

Try also

```
> effx(cc,type="binary",exposure=skin,data=mel)
```

to see the effect of skin colour. Look at `hair`, `eyes` and `freckles` in the same way.

1.7.5 Generalized linear models

The function `effx` is just a wrapper for the `glm` function, and you can show this by fitting the glm directly with

```
> m <- glm(cc ~ freckles, family="binomial", data=mel)
> exp(coef(m))
```

Comparison with `effx` shows the results to be the same. An alternative way of summarizing the glm is to use

```
> ci.lin(m, Exp=TRUE)
> # or better
> round(ci.exp( m ),2)
```

Note that in `effx` the type of response is "binary" whereas in `glm` the family of probability distributions used to fit the model is "binomial". There is a 1-1 relationship between type and family:

metric	gaussian
binary	binomial
failure/count	poisson

1.7.6 Controlling for age and sex

Because the probability that a control is selected into the study depends on age and sex it is necessary to control for age and sex. For example, the effect of freckles controlled for age and sex is obtained with

```
> effx(cc, typ="binary", exposure=freckles, control=list(age.cat,sex),data=mel)
```

or

```
> m <- glm(cc ~ freckles+age.cat+sex, family="binomial", data=mel)
> round(ci.exp(m ),2)
```

1.7.7 Likelihood ratio tests

There are 2 effects for the 3 levels of `freckles`, and `glm` provides a test for each effect separately, but to test for no effect at all of `freckles` you need a likelihood ratio test. This involves fitting two models, one without `freckles` and one with, and recording the change in deviance. Because there are some missing values for freckles it is necessary to restrict the first model to those subjects who have values for freckles.

```
> m1 <- glm(cc ~ age.cat+sex, family="binomial", data=mel, subset=!is.na(freckles))
> m2 <- glm(cc ~ freckles+age.cat+sex, family="binomial", data=mel)
> anova(m1,m2,test="Chisq")
```

The change in residual deviance is $1785.9 - 1737.1 = 48.8$ on $1389 - 1387 = 2$ degrees of freedom. There are 3 effects for the 4 levels of hair colour (`hair`). Fit two glm's and use `anova` to test for no effects of hair colour.

1.7.8 Relevelling

From the above you can see that subjects at each of the 3 levels light-brown, blonde, and red, are at greater risk than subjects with dark hair, with similar odds ratios. This suggests creating a new variable `hair2` which has just two levels, dark and the other three. The `Relevel` function has been used for this in the house keeping script.

Use `effx` to compute the odds-ratio of melanoma between persons with red, blonde or light brown hair versus those with dark hair. Reproduce these results by fitting an appropriate glm. Use a likelihood ratio test to test for the effect of `hair2`.

1.7.9 Controlling for other variables

When you control the effect of an exposure for some variable you are asking a question about what would the effect be if the variable is kept constant. For example, consider the effect of `freckles` controlled for `hair2`. We first stratify by `hair2` with

```
> effx(cc,type="binary",exposure=freckles,control=list(age.cat,sex),strata=hair2,data=mel)
```

The effect of freckles is still apparent in each of the two strata for hair colour. Use `effx` to control for `hair2`.

```
> effx(cc,type="binary",exposure=freckles,control=list(age.cat,sex,hair2),data=mel)
```

It is tempting to control for variables without thinking about the question you are thereby asking. This can lead to nonsense.

1.7.10 Stratification using glm

We shall reproduce the output from

```
> effx(cc,type="binary",exposure=freckles,control=list(age.cat,sex),strata=hair2,data=mel)
```

using a glm. To do this requires a nested model formula:

```
> m <- glm(cc~hair2/freckles+age.cat+sex,family="binomial",data=mel)
> ci.exp(m )
```

In amongst all the other effects you can see the two effects of freckles for dark hair (1.61 and 2.84) and the two effects of freckles for other hair (1.42 and 3.15).

1.7.11 Naevi

The distributions of `nvsmall` and `nvlarge` are very skew to the right. You can see this with

```
> with(mel, stem(nvsmall))
> with(mel, stem(nvlarge))
```

Because of this it is wise to categorize them into a few classes

- small naevi into four: 0, 1, 2-4, and 5+;
- large naevi into three: 0, 1, and 2+.

This has been done in the house keeping script. Look at the joint frequency distribution of these new variables using `with(mel, table( ))`. Are they strongly associated?

Compute the sex- and age-adjusted OR estimates (with 90% CIs) associated with the number of small naevi first by using `effx`, and then by fitting separate glms including `sex`, `age.cat` and `nvsma4` in the model formula. Do the same with `nvlar3`. Now fit a glm containing `age.cat sex nvsma4` and `nvlar3`. What is the interpretation of the coefficients for `nvsma4` and `nvlar3`?

1.7.12 Treating freckles as a numeric exposure

The evidence for the effect of `freckles` is already convincing. However, to demonstrate how it is done, we shall perform a linear trend test by treating freckles as a numeric exposure.

```
> mel$fscore<-as.numeric(mel$freckles)
> effx(cc,type="binary",exposure=fscore,control=list(age.cat,sex),data=mel)
```

You can check for linearity of the log odds of being a case with `fscore` by comparing the model containing `freckles` as a factor with the model containing `freckles` as numeric.

```
> m1<-glm(cc~freckles+age.cat+sex, family="binomial",data=mel)
> m2<-glm(cc~fscore+age.cat+sex, family="binomial",data=mel)
> anova(m2,m1,test="Chisq")
```

There is no evidence against linearity ($p = 0.22$).

It is sometimes helpful to look at the linearity in more detail with

```
> m1<-glm(cc~C(freckles, contr.cum)+age.cat+sex, family="binomial",data=mel)
> ci.exp(m1 )
> m2<-glm(cc~fscore+age.cat+sex, family="binomial",data=mel)
> ci.exp(m2 )
```

The use of `C(freckles,contr.cum)` makes odds ratios versus the previous level not the baseline. If the logodds are linear then these odds ratios should be the same (and the same as the odds ratio for `fscore` in m2.

1.7.13 Graphical displays

The odds ratios (with CIs) can be graphically displayed using function `plotEst()` in `Epi`. It uses the value of `ci.lin()` evaluated on the fitted model object. As the intercept and the effects of age and sex are of no interest, we shall drop the corresponding rows (the 7 first ones) from the matrix produced by `ci.lin()`, and the plot is based just on the 1st, 5th and the 6th column of this matrix:

```
> m <- glm(cc~age.cat+sex+nvsma4+nvlar3, family="binomial",data=mel)
> plotEst( exp( ci.lin(m)[-(1:7), -(2:4)] ), xlog=T,vref=1 )
```

The `xlog` argument makes the OR axis logarithmic.

1.8 Estimation of effects: simple and more complex

1.8.1 Response and explanatory variables

Identifying the *response* or *outcome 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 categorical factors or numeric variables. 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.

Other measures of effect include ratios of geometric means for positive-valued metric outcomes, differences and ratios between proportions (risk difference and risk ratio), and differences between failure rates.

1.8.2 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, i.e. adjustment for confounding, without the need for familiarity with statistical modelling. It is in fact a wrapper of function `glm()` that fits generalized linear models.

We shall use the `births` data to illustrate the function.

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

Because all variables are numeric we need first to do a little housekeeping. Two of them are directly converted into factors, and categorical versions are created of two continuous variables by function `cut()`.

```
> 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 estimated effect of sex on birth weight, measured as a difference in means between girls and boys, is −197 g. The command

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

verifies this ($3032.8 - 3229.9 = -197.1$). The P-value refers to the test of the null hypothesis that there is no effect of `sex` on birth weight (quite an uninteresting hypothesis in itself!).

The same task performed by `glm()` – or by `lm()` for that matter. Note the amount of output that `summary()` method produces. The point estimate plus confidence limits can, though, be obtained by `ci.lin()`.

```
> m1 <- glm(bweight ~ sex, family=gaussian, data=births)
> summary(m1)
> ci.lin(m1)[ , c(1,5,6)]
```

– 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 estimated 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: ($16.26/11.39 = 1.427$). Also, fitting the corresponding logistic model by `glm()` agrees with those above

```
> m2 <- glm(lowbw ~ sex, family = binomial(link=logit), data = births)
> ci.exp(m2)
```

– Use `effx()` to find the effect of `hyp` on `lowbw`.

1.8.3 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 effect estimates:

```
[35,37) vs [20,35)  857
[37,39) vs [20,35) 1360
[39,45) vs [20,35) 1668
```

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.

– Estimate the effects of `gest4` on `lowbw`. Use the option `base=4` to change the baseline for `gest4` from 1 to 4. You will find quite dramatic OR values.

1.8.4 Stratified effects

As an example we shall stratify the effects of `hyp` on `bweight` by `sex` as follows:

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

The estimated effects of `hyp` in the different strata defined by `sex` are -496 and -380.' Try this with `lm()`:

```
> m3 <- lm(bweight ~ sex/hyp, data = births)
> ci.lin(m3)[ , c(1,5,6)]
```

– Use `effx()` to stratify the effect of `hyp` on `lowbw` first by `sex` and then by `gest4`. For `gest4` the estimates in the two extreme strata are highly imprecise.

1.8.5 Controlling the effect of hyp for sex

The effect of `hyp` is *controlled for* – or *adjusted for* – `sex` by first looking at the estimated effects of `hyp` in the two stata defined by `sex`, and then combining these effects if they seem sufficiently similar. In this case the estimated effects were -496 and -380 which look quite 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)
```

Same with `lm()`. When

```
> m4 <- lm(bweight ~ sex + hyp, data = births)
> ci.lin(m4)[ , c(1,5,6)]
```

The estimated effect of `hyp` on `bweight` controlled for `sex` is thus -448 g. There can be more than one control variable, e.g `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 usually it is useful to stratify first.

1.8.6 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) we can estimate the linear effect of `gestwks` with `effx()` and `lm()` as follows:

```
> effx(response=bweight, type="metric", exposure=gestwks,data=births)
> m5 <- lm(bweight ~ gestwks,data=births) ; ci.lin(m5)[ , c(1,5,6)]
```

The estimated linear effect of `gestwks` is thus 197 g per each additional week of gestation. The linear effect of `gestwks` on the log-odds of `lowbw` can be estimated similarly:

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

The linear effect of `gestwks` on the log-odds of `lowbw` is manifested as 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 the control list.

1.8.7 Checking on linearity against polynomial model

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))
> abline(m5)
```

Is the relationship reasonably linear? A common practice is to compare the fit of this model with models having higher order polynomial terms. In perinatal epidemiology a popular model for describing the relationship between gestational age and birth weight is a 3rd degree polynomial. We may update the simple linear model by adding the quadratic and cubic terms of `gestwks` using the *insulate* operator `I()`

```
> m6 <- update(m5, . ~ .  + I(gestwks^2) + I(gestwks^3))
> round(ci.lin(m6)[, c(1,5,6)], 1)
```

The intercept and linear coefficients are really spectacular – but don't make any sense! We better use a centred version of the regressor by choosing 40 weeks as the reference value. Let us also perform an F test for the null hypothesis of simple linear effect against the 3rd degree polynomial model

```
> births$gw40 <- births$gestwks-40
> mpoly <- lm(bweight ~ gw40 + I(gw40^2) + I(gw40^3), data=births)
> round(ci.lin(mpoly)[, c(1,5,6)], 1)
> anova(m5, mpoly)
```

These regression coefficients are now more interpretable. There also seems to be strong evidence against simple linear regression; both the quadratic and cubic term appear "significant".

As the next step we present graphically the fitted polynomial curve together with confidence limits for the expected responses as well as prediction intervals for individual observations with new data comprising gestational weeks from 24 to 45.

```
> nd <- data.frame(gw40 = 23:46 - 40)
> fit.poly <- predict( mpoly, newdata=nd, interval="conf" )
> pred.poly <- predict( mpoly, newdata=nd, interval="pred" )
> with( births, plot( bweight ~ gestwks, xlim = c(23, 46), cex.axis= 1.5, cex.lab = 1.5 )
> matlines( nd$gw40+40, fit.poly, lty=1, lwd=c(3,2,2), col=c('red','blue','blue') )
> matlines( nd$gw40+40, pred.poly, lty=1, lwd=c(3,2,2), col=c('red','green','green') )
```

The curve fits nicely within the range of observed values of the regressor. However, the tail behaviour in polynomial models tends to be problematic.

1.8.8 Fitting a natural spline model

One approach for flexible modelling with perhaps more reasonable tail behaviour is based on splines. By the following piece of code you can fit a natural cubic spline with 5 pre-specified knots determining the degree of smoothing.

```
> library(splines)
> mspli <- lm( bweight ~ Ns( gestwks,
+         knots = c(28,34,38,40,43)), data = births)
> round(ci.lin(mspli)[ , c(1,5,6)], 1)
```

These regression coefficients are even less interpretable than those in the uncentred polynomial model. However, graphical presentation of the fitted curve together with the confidence and prediction intervals is more informative:

```
> nds <- data.frame(gestwks = 23:46)
> fit.spli <- predict( mspli, newdata=nds, interval="conf" )
> pred.spli <- predict( mspli, newdata=nds, interval="pred" )
> with( births, plot( bweight ~ gestwks, xlim = c(23, 46), cex.axis= 1.5, cex.lab = 1.5 )
> matlines( nds$gestwks, fit.spli, lty=1, lwd=c(3,2,2), col=c('red','blue','blue') )
> matlines( nds$gestwks, pred.spli, lty=1, lwd=c(3,2,2), col=c('red','green','green') )
```

Compare this with the 3rd order curve fitted above. In natural splines the curve is constrained to be linear beyond the extreme knots.

Finally, take a look at the basic diagnostic plots from the spline model.

```
> par(mfrow=c(2,2))
> plot(mspli)
```

What can you say about the agreement with data of the assumptions of the model, like homoskedasticity and normality of the error terms?

1.8.9 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 in the graduate school at Berkeley 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?

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


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

1.9 Estimation and reporting of linear and curved effects

In this exercise we will use the `testisDK` data from the `Epi` package, which contains the number of cases of testis cancer in Denmark 1943–96:

1. First load the Danish testis cancer data, and inspect the dataset:

   ```
   library( Epi )
   sessionInfo()
   data( testisDK )
   str( testisDK )
   head( testisDK )
   ```

   Tabulate both events and person-years using `stat.table`, in say 10-year age-groups and 10-year periods of follow-up. In which ages are the age-specific testis cancer rates highest?

2. Now fit a Poisson-model for the mortality rates with a linear term for age at follow-up (current age, attained age):

   ```
   ml <- glm( D ~ A, offset=log(Y), family=poisson, data=testisDK )
   ci.exp( ml )
   ```

   What do the parameters mean?

3. Work out the the predicted log-mortality rates for ages 25 to 45, say, by doing a hand-calculation based on the coefficients:

   ```
   ( cf <- coef( ml ) )
   ```

4. However, we do not have the standard errors of these mortality rates, and hence neither the confidence intervals. This is implemented in `ci.exp`; if we provide the argument `ctr.mat=` as a matrix where each row corresponds to a prediction point and each column to a parameter from the model. Look at the help page for `ci.exp` and then try:

   ```
   ( CM <- cbind( 1, 25:45 ) )
   round( ci.exp( ml, ctr.mat=CM )*10^5, 3 )
   ```

5. Use this machinery to derive and plot the mortality rates over the range from 15 to 65 years, say:

   ```
   C1 <- cbind( 1, 15:65 )
   matplot( 15:65, ci.exp( ml, ctr.mat=C1 )*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" )
   ```

6. Now check if the mortality rates really are eksponentially increasing by age (that is linearly on the log-scale), by adding a quadratic term to the model. Note that you must use the expression `I(A^2)` in the modelleing in order to avoid that the "^" is interpreted as part of the model formula:

```
mq <- glm( D ~ A + I(A^2), offset=log(Y), family=poisson, data=testisDK )
ci.exp( mq, Exp=F )
```

Then plot the estimated rates under the quadratic model.

```
aa <- 15:65
C2 <- cbind( 1, aa, aa^2 )
matplot( aa, ci.exp( mq, ctr.mat=C2 )*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" )
```

Try to overlay the estimated rates from the model with linear efect of age — you will need the function `matlines`.

7. Repeat the same using a 3rd degree polynomial.

8. Instead of continuing with higher powers of age we could use fractions of powers, or we could use splines, piecevise polynomial curves, that fit nicely together at join points (knots). This is implemented in the `splines` package, in the function `ns`, which returns a matrix. There is a wrapper `Ns` in the `Epi`-package that automatically designate the smallest and largest knots a *boundary knots*, beyond which the resulting curve is linear:

   ```
   library( splines )
   ms <- glm( D ~ Ns(A,knots=seq(15,65,10)), offset=log(Y),
              family=poisson, data=testisDK )
   ```

   In order to extract the estimated effects, construct a contrast matrix that correspond to the parameters of the model:

   ```
   As <- Ns( aa, knots=seq(15,65,10) )
   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" )
   ```

9. Now add a linear term in calendar time `P` to the model, and make a prediction of the incidence rates in 1970, say:

   ```
   msp <- glm( D ~ Ns(A,knots=seq(15,65,10)) + P, offset=log(Y), family=poisson, data=te
   matplot( aa, ci.exp( msp, ctr.mat=cbind(1,As,1970) )*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" )
   ```

   Note that `cbind` automatically will expand the 1 and the 1970 to match the number of rows of `As`.

10. Extract the RR relative to 1970, by using the `subset` argument to `ci.exp`:

    ```
    ci.exp( msp, subset="P" )
    ```

    What is the annual relative increase in the testis cancer incidence rates? Show the RR of testis cancer by year relative to 1970 by multipling the log-RR for period with the distance form 1970, such as:

```
yy  <- 1943:1996
Cp1 <- cbind( yy - 1970 )
matplot( yy, ci.exp( msp, ctr.mat=Cp1, subset="P" ),
         log="y", xlab="Date", ylab="RR of Testis cancer",
         type="l", lty=1, lwd=c(3,1,1), col="black" )
abline( h=1 )
```

11. Try to add a quadratic term to the period effect, and plot the resulting RR relative to 1970.
    Hint: In order to extract the quadratic effects relative to 1970, you must form the matrix of linear and quadratic period, and a corresponding matrix where all rows are identical to the 1970 row:

    ```
    msp <- glm( D ~ Ns(A,knots=seq(15,65,10)) + P + I(P^2),
                    offset=log(Y), family=poisson, data=testisDK )
    Cq <- cbind( yy, yy^2 ) - cbind( rep(1970,length(yy)), 1970^2 )
    ```

    Use this matrix as arguent to `ci.exp`

12. Now investigate if there is any non-linearity in period beyond the quadratic, by fitting fit a spline for (`P`) as well, and comparing the models. Plot the resulting RR by year, relative to 1970 too. You must define a contrast matrix corresponding to the years where the prediction is made, as well as a matrix with the same number of rows, but with all rows identical to the one corresponding to the reference year. You must use the differenec of these two as the arument to `ctr.mat` in `ci.exp`.

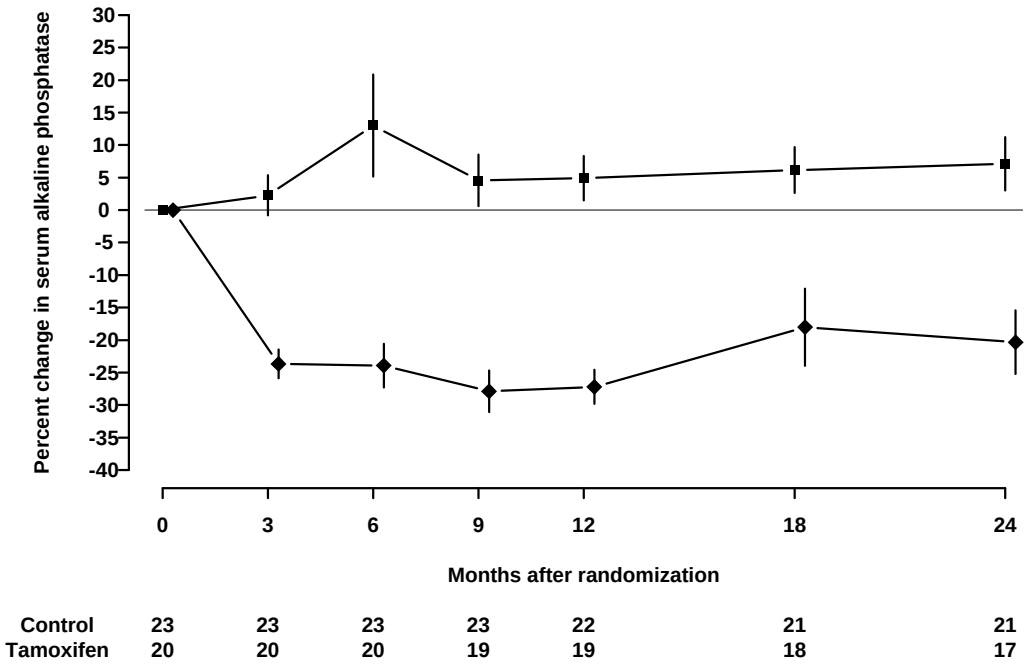
13. Plot the estimated age-specific rates in 1970 from this model. Note that you need a reference matrix for the period with all rows identical to the 1970 row, but this time with the same number of rows as the *age*-prediciton points.

14. Collect these steps in a general outline, where you first define the knots, and the points of age and period prediction, and then fit the model and do the two plots.

15. Form a new variable in the data frame, `B=P-A`, the data of birth, and repeat the last analysis with this variable instead of `P`.

1.10 Graphics meccano

The plot below is from a randomized study of the effect of Tamoxifen treatment on bone mineral metabolism, in a group of patients who were treated for breast cancer.



The data are available in the file `alkfos.csv` (using comma as separator, so `read.csv` will read it).

The purpose of this exercise is to show you how to build a similar graph using base graphics in R. This will take you through a number of fundamental techniques. The exercise will also walk you through creating the graph using `ggplot2`.

To get started, execute the following R code. You probably should not study the code in too much detail at this point. We will make a version of the code available as `alkfos-house.r` so that you just have to load it to a script window and execute it from there.

```
> alkfos <- read.csv("data/alkfos.csv") # change filename as needed
> # express the data as % change from baseline
> alkfos.pctchange <- (sweep(alkfos[-1], 1, alkfos$c0, "/") - 1)*100
> # Generate by-group statistics for each column
> (available <- aggregate(!is.na(alkfos[-1]),list(alkfos$grp), sum))
> (means <- aggregate(alkfos.pctchange, list(alkfos$grp), mean, na.rm=TRUE))
> (sds <- aggregate(alkfos.pctchange, list(alkfos$grp), sd, na.rm=TRUE))
> # aggregate() gives data frames. Convert to matrices and get rid of
> #      the 1st column (group number)
> available <- as.matrix(available[-1])
> means <- as.matrix(means[-1])
> sds <- as.matrix(sds[-1])
```

```
> sems <- sds/sqrt(available)
> # These are the examination times
> times <- c(0,3,6,9,12,18,24)
```

1.10.1 Base graphics

Now we start building the plot. It is important that you use some form of script to hold the R code since you will frequently have to modify and rerun previously entered code.

1. First, plot the means for group 1 (i.e. `means[1,]`) against `times`, using `type="b"` (look up what this does)

2. Then *add* a similar curve for group 2 to the plot using `points` or `lines`. Notice that the new points are below the y scale of the plot, so you need to revise the initial plot by setting a suitable `ylim` value.

3. It is not too important here (it was for some other variables in the study), but the S-PLUS plot has the points for the second group offset horizontally by a small amount (.25) to prevent overlap. Redo the plot with this modification.

4. Add the error bars using `segments`. (You can calculate the endpoints using `upper <- means + sems` etc.). You may have to adjust the `ylim` again.

5. Add the horizontal line at $y = 0$ using `abline`

6. Use `xlab` and `ylab` in the initial `plot` call to give better axis labels.

7. We need a nonstandard x axis. Use `xaxt="n"` to avoid plotting it at first, then add a custom axis with `axis`

8. The counts below the x axis can be added using `mtext` on lines 5 and 6 below the bottom of the plot, but you need to make room for the extra lines. Use `par(mar=.1 + c(8,4,4,2))` *before* plotting anything to achieve this.

9. Further things to fiddle with: Get rid of the bounding box. Add Control/Tamoxifen labels to the lines of counts. Perhaps use different plotting symbols. Rotate the y axis values. Modify the line widths or line styles.

10. Finally, try plotting to the `pdf()` device and view the results using Acrobat Reader. You may need to change the `pointsize` option and/or the plot dimensions for optimal appearance. You might also try saving the plot as a metafile and include it in a Word document.

1.10.2 Using ggplot2

The housekeeping script `alkfos-house.r` also creates a data frame `ggdata` containing the variables in long format. The code for generating the data frame is shown below, but you do not need to repeat it if you have run the script.

```
> ggdata <- data.frame(
+     times = rep(times, 2),
+     means = c(means[1,], means[2,]),
+     sds = c(sds[1,], sds[2,]),
+     available = c(available[1,], available[2,]),
+     treat = rep(c("placebo","tamoxifen"), each=7)
+     )
> ggdata <- transform(ggdata, sems = sds/sqrt(available))
```

To create a first approximation to the plot in `ggplot2` we use the `qplot` function (short for "quick plot"). First you must install the `ggplot2` package from CRAN and then load it:

```
> library(ggplot2)
> qplot(x=times, y=means, group=treat, geom=c("point", "line") , data=ggdata)
```

The first arguments to `qplot` are called "aesthetics" in the grammar of graphics. Here we want to plot `y=means` by `x=times` grouped by `group=treat`. The aesthetics are used by the "geometries", which are specified with the `geom` argument. Here we want to plot both points and lines. The `data` argument tells `qplot` to get the aesthetics from the data frame `ggdata`.

To add the error bars, we add a new geometry "linerange" which uses the aesthetics `ymin` and `ymax`

```
> p <- qplot(x=times, y=means, group=treat, ymin=means-sems, ymax=means+sems,
+            yintercept=0, geom=c("point", "line", "linerange"), data=ggdata)
> print(p)
```

In this case we are saving the output of `qplot` to an R object `p`. This means the plot will not appear automatically when we call `qplot`. Instead, we must explicitly print it.

Note how the y axes are automatically adjusted to include the error bars. This is because they are included in the call to `qplot` and not added later (as was the case with base graphics).

It remains to give informative axis labels and put the right tick marks on the x-axis. This is done by adding scales to the plot

```
> p <- p +
+     scale_x_continuous(name="Months after randomization", breaks=times[1:7]) +
+     scale_y_continuous(name="% change in alkaline phosphatase")
> print(p)
```

We can also change the look and feel of the plot by adding a theme (in this case the black and white theme).

```
> p + theme_bw()
```

As an alternative to `qplot`, we can use the `ggplot` function to define the data and the common aesthetics, then add the geometries with separate function calls. All the grobs (*graphical objects*) created by these function calls are combined with the `+` operator:

```
> p <- ggplot(data=ggdata,
+             aes(x=times, y=means, ymin=means-sems, ymax=means+sems, group=treat)) +
+     geom_point() +
+     geom_line() +
+     geom_linerange() +
+     geom_hline(yintercept=0, colour="darkgrey") +
+     scale_x_continuous(breaks=times[1:7])
```

This call adds another geometry "hline" which uses the aesthetic `yintercept` to add a horizontal line at 0 on the y-axis. Note that this alternate syntax allows each geometry to have its own aesthetics: here we draw the horizontal line in darkgrey instead of the default black.

1.10.3 Grid graphics

As a final, advanced topic, this subsection shows how viewports from the `grid` package may be used to display the plot and the table in the same graph. First we create a text table:

```
> tab <- ggplot(data=ggdata, aes(x=times, y=treat, label=available)) +
+     geom_text(size=3) + xlab(NULL) + ylab(NULL) +
+     scale_x_continuous(breaks=NULL)
> tab
```

Then we create a layout that will contain the graph above the table. Most of the space is taken by the graph. The `grid.show.layout` allows you to preview the layout.

```
> library(grid)
> Layout <- grid.layout(nrow = 2, ncol = 1, heights = unit(c(2, 0.25),
+                                                  c("null", "null")))
> grid.show.layout(Layout)
```

The units are relative ("null") units. You can specify exact sizes in centimetres, inches, or lines if you prefer.

We then print the graph and the table in the appropriate viewports

```
> grid.newpage() #Clear the page
> pushViewport(viewport(layout=Layout))
> print(p, vp=viewport(layout.pos.row=1, layout.pos.col=1))
> print(tab, vp=viewport(layout.pos.row=2, layout.pos.col=1))
```

Notice that the left margins do not match. You might try pad the left margin of the graph by adding a theme with the aesthetic `plot.margin`.

```
> p + theme(plot.margin = unit(c(1,1,1,1.3), "lines"))
```

1.11 Survival analysis: Oral cancer patienfts

1.11.1 Description of the data

File `oralca2.txt`, that you may access from a url address to be given in the practical, contains data from 338 patients having an oral squamous cell carcinoma diagnosed and treated in one tertiary level oncological clinic in Finland since 1985, followed-up for mortality until 31 December 2008. The dataset contains the following variables:

`sex` = sex, a factor with categories `1 = "Female", 2 = "Male"`,
`age` = age (years) at the date of diagnosing the cancer,
`stage` = TNM stage of the tumour (factor): `1 = "I", ..., 4 = "IV", 5 = "unkn"`,
`time` = follow-up time (in years) since diagnosis until death or censoring,
`event` = event ending the follow-up (numeric):
0 = censoring alive, 1 = death from oral cancer, 2 = death from other causes.

1.11.2 Loading the packages and the data

(a) Load the R packages `Epi, mstate`, and `survival` needed in this exercise.

```
> library(Epi)
> library(mstate)
> library(survival)
```

(b) Read the datafile `oralca2.txt` from a website, whose precise address will be given in the practical, into an R data frame named `orca`. Look at the head, structure and the summary of the data frame. Using function `table()` count the numbers of censorings as well as deaths from oral cancer and other causes, respectively, from the `event` variable.

```
> orca <- read.table("oralca2.txt", header=T)
> head(orca) ; str(orca) ; summary(orca)
```

1.11.3 Total mortality: Kaplan–Meier analyses

(a) We start our analysis of total mortality pooling the two causes of death into a single outcome. First, construct a *survival object* `orca$suob` from the event variable and the follow-up time using function `Surv()`. Look at the structure and summary of `orca$suob` .

```
> orca$suob <- Surv(orca$time, 1*(orca$event > 0) )
> str(orca$suob)
> summary(orca$suob)
```

(b) Create a `survfit` object `s.all`, which does the default calculations for a Kaplan–Meier analysis of the overall (marginal) survival curve.

```
> s.all <- survfit(suob ~ 1, data=orca)
```

See the structure of this object and apply `print()` method on it, too. Look at the results; what do you find?

```
> s.all
> str(s.all)
```

(c) The `summary` method for a `survfit` object would return a lengthy life table. However, the `plot` method with default arguments offers the Kaplan–Meier curve for a conventional illustration of the survival experience in the whole patient group. Alternatively, instead of graphing survival proportions, one can draw a curve describing their complements: the cumulative mortality proportions. This curve is drawn together with the survival curve as the result of the second command line below.

```
> plot(s.all)
> lines(s.all, fun = "event", mark.time=F, conf.int=F)
```

The effect of option `mark.time=F` is to omit marking the times when censorings occurred.

1.11.4 Total mortality by stage

Tumour stage is an important prognostic factor in cancer survival studies.

(a) Plot separate cumulative mortality curves for the different stage groups marking them with different colours, the order which you may define yourself. Also find the median survival time for each stage.

```
> s.stg  <- survfit(suob ~ stage, data= orca)
> col5 <- c("green", "blue", "black", "red", "gray")
> plot(s.stg, col= col5, fun="event", mark.time=F )
> s.stg
```

(b) Create now two parallel plots of which the first one describes the cumulative hazards and the second one graphs the log-cumulative hazards against log-time for the different stages. Compare the two presentations with each other and with the one in the previous item.

```
> par(mfrow=c(1,2))
> plot(s.stg, col= col5, fun="cumhaz", main="cum. hazards" )
> plot(s.stg, col= col5, fun="cloglog", main = "cloglog: log cum.haz"  )
```

(c) If the survival times were *exponentially* distributed in a given (sub)population the corresponding cloglog-curve should follow an approximately linear pattern. Could this be the case here in the different stages?

(d) Also, if the survival distributions of the different subpopulations would obey the *proportional hazards* model, the vertical distance between the cloglog-curves should be approximately constant over the time axis. Do these curves indicate serious deviation from the proportional hazards assumption?

(e) In the lecture handouts (p. 34, 37) it was observed that the crude contrast between males and females in total mortality appears unclear, but the age-adjustment in the Cox model provided a more expected hazard ratio estimate. We shall examine the confounding by age somewhat closer. First categorize the continuous age variable into, say, three categories by function `cut()` using suitable breakpoints, like 55 and 75 years, and cross-tabulate sex and age group:

```
> orca$agegr <- cut(orca$age, br=c(0,55,75, 95))
> stat.table( list( sex, agegr), list( count(), percent(agegr) ),
+              margins=T, data = orca )
```

Male patients are clearly younger than females in these data.

Now, plot Kaplan–Meier curves jointly classified by sex and age.

```
> s.agrx <- survfit(suob ~ agegr + sex, data=orca)
> par(mfrow=c(1,1))
> plot(s.agrx, fun="event", mark.time=F, xlim = c(0,15),
+              col=rep(c("red", "blue"),3), lty=c(2,2, 1,1, 5,5))
```

In each ageband the mortality curve for males is on a higher level than that for females.

1.11.5 Lexis object with multi-state set-up

Before entering to analyses of cause-specific mortality it might be instructive to apply some Lexis tools to illustrate the competing-risks set-up. More detailed explanation of these tools will be given by Bendix in this afternoon.

(a) Form a `Lexis` object from the data frame and print a summary of it. We shall name the main (and only) time axis in this object as `stime`.

```
> orca.lex <- Lexis(exit = list(stime = time),
+            exit.status = factor(event,
+    labels = c("Alive", "Oral ca. death", "Other death")),
+                    data = orca)
> summary(orca.lex)
```

(b) Draw a box diagram of the two-state set-up of competing transitions. Run first the following command line

```
boxes( orca.lex )
```

Now, move the cursor to the point in the graphics window, at which you wish to put the box for “Alive”, and click. Next, move the cursor to the point at which you wish to have the box for “Oral ca. death”, and click. Finally, do the same with the box for “Other death”. If you are not happy with the outcome, run the command line again and repeat the necessary mouse moves and clicks.

1.11.6 Event-specific cumulative mortality curves

We move on to analysing cumulative mortalities for the two causes of death separately, first overall and then by prognostic factors.

(a) Use function `Cuminc()` in package `mstate` and view the structure of the thus created object.

```
> cif1 <- Cuminc( time = "time", status= "event", data = orca)
> str(cif1)
```

(b) Function `Cuminc()` thus creates an ordinary data frame with quite self-explanatory column names. Unfortunately, no handy `plot` method is provided in the package. A simple function `plotCIF2g()` is available in the same website and folder from which you also got the data set. Pick up the source file containing the function and see, what are its arguments.

```
> source("plotCIF2g.R")
> args(plotCIF2g)
```

The main arguments are

$$\texttt{data} = \text{data frame created by function } \texttt{Cuminc()}, \tag{1.1}$$

$$\texttt{cause} = \text{indicator for the event: values 1 or 2.} \tag{1.2}$$

Other arguments are like in the ordinary `plot()` function.

(c) Draw two parallel plots describing the overall cumulative incidence curves for both causes of death

```
> par(mfrow=c(1,2))
> plotCIF2g(cif1, 1, main = "Cancer death")
> plotCIF2g(cif1, 2, main= "Other deaths")
```

Just ignore the warnings!

(d) Compute the estimated cumulative incidences by stage for both causes of death. Now you have to add argument `group` when calling `Cuminc()`. See the structure of the resulting object, in which you should observe the first column containing the grouping variable. Plot the pertinent curves in two parallel graphs. Cut the y-axis for a more efficient graphical presentation

```
> cif2 <- Cuminc( time = "time", status= "event",
+                 group = "stage", data = orca)
> str(cif2)
> par(mfrow=c(1,2))
> plotCIF2g(cif2, 1, main = "Cancer death by stage",
+    col=col5, ylim = c(0, 0.7) )
> plotCIF2g(cif2, 2, main= "Other deaths by stage",
+    col=col5, ylim = c(0, 0.7) )
```

Compare the two plots. What would you conclude about the effect of stage on the two causes of death?

1.11.7 Regression modelling of overall mortality.

(a) Fit the semiparametric proportional hazards regression model, a.k.a. the Cox model, on all deaths including sex, age and stage as covariates. Use function `coxph()` in package `survival`. It is often useful to center and scale continuous covariates like `age` here. The estimated rate ratios and their confidence intervals can also here be displayed by applying `ci.lin()` on the fitted model object.

```
> options(show.signif.stars = F)
> m1 <- coxph(suob ~ sex + I((age-65)/10) + stage, data= orca)
> summary( m1 )
> round( ci.exp(m1 ), 4 )
```

Look at the results. What are the main findings?

(b) Check whether the data are sufficiently consistent with the assumption of proportional hazards with respect to each of the variables separately as well as globally, using the `cox.zph()` function.

```
> cox.zph(m1)
```

(c) No evidence against proportionality assumption could apparently be found. Moreover, no difference can be observed between stages I and II in the estimates. On the other hand, the group with stage unknown is a complex mixture of patients from various true stages. Therefore, it may be prudent to exclude these subjects from the data and to pool the first two stage groups into one. After that fit a model in the reduced data with the new stage variable.

```
> orca2 <- subset(orca, stage != "unkn")
> orca2$st3 <- Relevel( orca2$stage, list(1:2, 3, 4:5) )
> levels(orca2$st3) = c("I-II", "III", "IV")
> m2 <- update(m1, . ~ . - stage + st3, data=orca2 )
> round( ci.exp(m2 ), 4)
```

(d) Plot the predicted cumulative mortality curves by stage, jointly stratified by sex and age, focusing only on 40 and 80 year old patients, respectively, based on the fitted model `m2`. You need to create a new artificial data frame containing the desired values for the covariates.

```
> newd <- data.frame( sex = c( rep("Male", 6), rep("Female", 6) ),
+                     age = rep( c( rep(40, 3), rep(80, 3) ), 2 ),
+                     st3 = rep( levels(orca2$st3), 4) )
> newd
> col3 <- c("green", "black", "red")
> par(mfrow=c(1,2))
> plot( survfit(m2, newdata= subset(newd, sex=="Male" & age==40)),
+      col=col3, fun="event", mark.time=F)
> lines( survfit(m2, newdata= subset(newd, sex=="Female" & age==40)),
+       col= col3, fun="event", lty = 2, mark.time=F)
> plot( survfit(m2, newdata= subset(newd, sex=="Male" & age==80)),
+      ylim = c(0,1), col= col3, fun="event", mark.time=F)
> lines( survfit(m2, newdata= subset(newd, sex=="Female" & age==80)),
```

```
+       col=col3, fun="event", lty=2, mark.time=F)
>
```

1.11.8 Modelling event-specific hazards and hazards of the subdistribution

(a) Fit the Cox model for the cause-specific hazard of cancer deaths with the same covariates as above. In this case only cancer deaths are counted as events and deaths from other causes are included into censorings.

```
> m2haz1 <- coxph( Surv( time, event==1)  ~ sex + I((age-65)/10) + st3 , data=orca2 )
> round( ci.exp(m2haz1 ), 4)
> cox.zph(m2haz1)
```

Compare the results with those of model `m2`. What are the major differences?

(b) Fit a similar model for deaths from other causes and compare the results.

```
> m2haz2 <- coxph( Surv( time, event==2)  ~ sex + I((age-65)/10) + st3 , data=orca2 )
> round( ci.exp(m2haz2 ), 4)
> cox.zph(m2haz2)
```

(c) Finally, fit the Fine–Gray model for the hazard of the subdistribution for cancer deaths with the same covariates as above. For this you have to first load package `cmprsk`, containing the necessary function `crr()`, and attach the data frame.

```
> library(cmprsk)
> attach(orca2)
> m2fg1 <- crr(time, event, cov1 = model.matrix(m2), failcode=1)
> summary(m2fg1, Exp=T)
```

Compare the results with those of model `m2` and `m2haz1`.

(d) Fit a similar model for deaths from other causes and compare the results.

```
> m2fg2 <- crr(time, event, cov1 = model.matrix(m2), failcode=2)
> summary(m2fg2, Exp=T)
```

1.11.9 Poisson regression as an alternative to Cox model

It can be shown that the Cox model with an unspecified form for the baseline hazard $\lambda_0(t)$ is mathematically equivalent to the following kind of Poisson regression model. Time is treated as a categorical factor with a dense division of the time axis into disjoint intervals or *timebands* such that only one outcome event occurs in each timeband. The model formula contains this time factor plus the desired explanatory terms.

A sufficient division of time axis is obtained by first setting the break points between adjacent timebands to be those time points at which an outcome event has been observed to occur. Then, the pertinent `lexis` object is created and after that it will be split according to those breakpoints. Finally, the Poisson regression model is fitted on the splitted `lexis` object using function `glm()` with appropriate specifications.

We shall now demonstrate the numerical equivalence of the Cox model `m2haz1` for oral cancer mortality that was fitted above, and the corresponding Poisson regression.

(a) First we form the necessary `lexis` object by just taking the relevant subset of the already available `orca.lex` object. Upon that the three-level stage factor `st3` is created as above.

```
> orca2.lex <- subset(orca.lex, stage != "unkn" )
> orca2.lex$st3 <- Relevel( orca2$stage, list(1:2, 3, 4:5) )
> levels(orca2.lex$st3) = c("I-II", "III", "IV")
```

Then, the break points of time axis are taken from the sorted event times, and the `lexis` object is split by those breakpoints. The `timeband` factor is defined according to the splitted survival times stored in variable `stime`.

```
> cuts <- sort(orca2$time[orca2$event==1])
> orca2.spl <- splitLexis( orca2.lex, br = cuts, time.scale="stime" )
> orca2.spl$timeband <- as.factor(orca2.spl$stime)
```

As a result we now have an expanded `lexis` object in which each subject has several rows; as many rows as there are such timebands during which he/she is still at risk. The outcome status `lex.Xst` has value 0 in all those timebands, over which the subject stays alive, but assumes the value 1 or 2 at his/her last interval ending at the time of death. – See now the structure of the splitted object.

```
> str(orca2.spl)
> orca2.spl[ 1:20, ]
```

(b) We are ready to fit the desired Poisson model for oral cancer death as the outcome. The splitted person-years are contained in `lex.dur`, and the explanatory variables are the same as in model `m2haz1`. – This fitting may take some time ...

```
> m2pois1 <- glm( 1*(lex.Xst=="Oral ca. death")  ~
+       -1 + timeband + sex + I((age-65)/10) + st3,
+       family=poisson, offset = log(lex.dur), data = orca2.spl)
```

We shall display the estimation results graphically for the baseline hazard (per 1000 person-years) and numerically for the rate ratios associated with the covariates. Before doing that it is useful to count the length `ntb` of the block occupied by baseline hazard in the whole vector of estimated parameters. However, owing to how the splitting to timebands was done, the last regression coefficient is necessarily zero and better be omitted when displaying the results. Also, as each timeband is quantitatively named accoding to its leftmost point, it is good to compute the midpoint values `tbmid` for the timebands

```
> tb <- as.numeric(levels(orca2.spl$timeband)) ; ntb <- length(tb)
> tbmid <- (tb[-ntb] + tb[-1])/2   # midpoints of the intervals
> round( ci.exp(m2pois1 ), 3)
> par(mfrow=c(1,1))
> plot( tbmid, 1000*exp(coef(m2pois1)[1:(ntb-1)]),
+             ylim=c(5,3000), log = "xy", type = "l")
```

Compare the regression coefficients and their error margins to those model `m2haz1`. Do you find any differences? How does the estimated baseline hazard look like?

(c) The estimated baseline looks quite ragged when based on 71 separate parameters. A smoothed estimate may be obtained by spline modelling using the tools contained in package `splines` (see the practical of Saturday 25 May afternoon). With the following code you will be able to fit a reasonable spline model for the baseline hazard and draw the estimated curve (together with a band of the 95% confidence limits about the fitted values). From the same model you should also obtain quite familiar results for the rate ratios of interest.

```
> library(splines)
> m2pspli <- update(m2pois1, . ~  ns(stime, df = 6, intercept = F) +
+         sex + I((age-65)/10) + st3)
> round( ci.exp( m2pspli ), 3)
> news <- data.frame( stime = seq(0,25, length=301), lex.dur = 1000, sex = "Female",
+                     age = 65, st3 = "I-II")
> blhaz <- predict(m2pspli, newdata = news, se.fit = T, type = "link")
> blh95 <- cbind(blhaz$fit, blhaz$se.fit) %*% ci.mat()
> par(mfrow=c(1,1))
> matplot( news$stime, exp(blh95), type = "l", lty = c(1,1,1), lwd = c(2,1,1) ,
+       col = rep("black", 3),  log = "xy", ylim = c(5,3000)  )
```

1.11.10 Analysis of relative survival

(a) Load package `bigEpi` for the estimation of relative survival. Use the (simulated) female Finnish breast cancer patients diagnosed between 1993-2012, called `sibr`.

```
> library(bigEpi)
> head(sibr)
```

(b) Prepare the data by using `lex` command in the `bigEpi` package, define follow-up time intervals, (calendar time) period that you are interested and where are the population mortality figures. Calculate 5-year observed survival (2008-2012) using period method by Ederer II (default)

```
> x <- lex(sibr,
+          fot.breaks=seq(0, 5, by = 1),
+          per.breaks = c("2008-01-01", "2013-01-01"),
+          status.var="status", pophaz = popmort)
> st <- survtab(x,surv.type = "surv.obs")
> st
```

(c) Calculate 5-year relative survival (2008-2012) using period method by Ederer II (default)

```
> x <- lex(sibr,
+          fot.breaks=seq(0, 5, by = 1),
+          per.breaks = c("2008-01-01", "2013-01-01"),
+          status.var="status", pophaz = popmort)
> st <- survtab(x,surv.type = "surv.rel")
> st
```

1.12 Time-splitting, time-scales and SMR: Diabetes in Denmark

This exercise is 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 data sets, we shall use the one with only cases of diabetes diagnosed after 1995, `DMlate`. 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.

The exercise is about assessing how mortality depends age, calendar time and duration of diabetes. And how to understand and compute SMR, and assess how it depends on these factors as well.

1. First load the data and take a look at the data:

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

   You can get a more detailed explanation of the data by referring to the help page:

```
> ?DMlate
```

2. Set up the dataset as a `Lexis` object with age, calendar time and duration of diabetes as timescales, and date of death as event. Make sure that you know what each of the arguments to `Lexis` mean:

```
> 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 )
```

   Take a look at the first few lines of the resulting dataset using `head()`.

3. Get an overall overview of the mortality by using `stat.table` to tabulate no. deaths, person-years and the crude mortality rate by sex.

4. If we want to assess how mortality depends on age, calendar time and duration, we should split the follow-up along all three time scales. In practice it is 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.

   Use `splitLexis` to split the follow-up along the age-axis:

```
> SL <- splitLexis( LL, breaks=seq(0,125,1), time.scale="A" )
> summary( SL )
```

   How many records are now in the dataset? How many person-years? Compare to the original `Lexis`-dataset.

5. Now estimate an age-specific mortality curve for men and women separately, using natural splines:

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

   Make sure you understand all the components on this modeling statement.

   With these objects we can get the estimated log-rates by using `predict`, and supplying a data frame of prediction points, so first make a data frame of prediction points, it must have variables corresponding to the predictor variables in the model, including the off-set variable.

```
> nd <-  data.frame( A = seq(10,90,0.5),
+              lex.dur = 1000)
> p.m <- predict.glm( r.m, type = "link",
+                        newdata = nd,
+                         se.fit = TRUE )
> p.f <- predict.glm( r.f, type = "link",
+                        newdata = nd,
+                         se.fit = TRUE )
> str( p.m )
```

   What is the structure of the prediction objects, when you use the `se.fit=TRUE` argument?

   Construct estimated rates with 95% c.i.s from these, and plot the two sets of estimated rates (men and women). (Hint: use `matplot`)

Graphical comparison with the population rates

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

```
> data( M.dk )
> head( M.dk )
```

   Plot the mortality rates from a particular year on top of the estimated rates, for example:

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

   Guess how to plot the mortality rates for women...

7. It would more natural to model the population mortality rates in a similar fashion as the diabetes mortality rates, try:

```
> R.m <- glm( D ~ ns( A, df=10 ),
+             offset = log( Y ),
+             family = poisson,
+               data = subset( M.dk, sex==1 & P>1994 ) )
```

Now obtain the same model for women, and construct the predicted rates as before — note that you will need a new dataset for prediction, because in this dataset the persons-years are called `Y`, while in the dataset with the patient follow-up, the person-years were in a variable called `lex.dur`.

Add the curves with predicted rates to the plot of the patient mortality rates.

Period and duration effects

8. 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. 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 time 0 on the timescale, we use 0 as the first knot.

   Therefore, find points (knots) so that the number of events is the same between each pair. Try this:

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

   Take a look at where these points are and make a similar construction for calendar time (`P`) and diabetes duration (`dur`) — remember to add 0 as a knot for the latter.

9. With these we can now model mortality rates (separately for men and women), as functions of age, calendar time and duration. To this end you will need the `splines` package, and you will need the function `ns` (look that up). Specification of natural splines is a bit clumsy, you will need expressions like in your model formula:

```
> ns( A, kn=kn.A[-c(1,length(kn.A))],Bo=kn.A[ c(1,length(kn.A))] )
```

   Therefore you can use a convenience wrapper, `Ns` that does the allocation of knots. You do not have to key it in, it is a pert of the `Epi` package. You can now specify a model very simply (remember to check the names of the vectors where you put the positions of the knots; here assumed to be `kn.A`, `kn.P` and `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" ) )
> summary( mm )
> mf <- update( mm, data = subset( SL, sex=="F" ) )
```

   What is the interpretation of the parameters (if any)?

10. How do these models fit relative to the models with only age as a descriptor of the rates?

    (Hint: Use the `anova`-function with the argument `test="Chisq"` to compare the models.

    What is the problem with this approach?

11. The models that you 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 parameters in the model.

    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.

    We will need matrices for specified sets 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 )
```

    What are the dimensions of these matrices?

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

12. Plot the three effects in three panels beside each other. Plot the estimates for men and women together.

    (Hint: Use the function `matplot` to plot both estimates and confidence limits for both sexes in one go.

    How are the estimated effects — is the chosen parametrization plausible? Would you want to reconsider the number of knots you used for any of the terms?

13. We have so far fitted models separately for men and women, we might fit a model for the entire dataset with common period and duration effects, but different age-effect for the two sexes. Try to fit this interaction model and take a look at the estimates:

```
> 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)
```

    What do these estimates refer to?

14. Now 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. Compare the total deviance for these with that of the fitted interaction model

    Hint: To find the deviance and degrees of freedom, look at the model object by saying: `names(m2)`.

    Is there any evidence of different period and duration effects between thew sexes?

15. Would you test whether there were proportional (or even identical) mortality rates between men and women? Why? And how?

16. Extract the parameters from the model, showing the separate age-effects for men and women, and the common period and duration effects. To get the age-specific rates for

men and women at the reference time and reference duration you may need the subset-argument to `ci.exp`, for example:

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

17. 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 simultaneously at 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.

    This is most easily done using the `predict` 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 )
> predict( mm, newdata=nd, se.fit=TRUE )
```

    Take a look at the result from the `predict` statement and construct prediction of mortality for men and women diagnosed in a range of ages, say 50, 60, 70, and plot these together in the same graph.

18. 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.

    When you add the effects of age and date of diagnosis you want to use a set of knots which is aligned to the variables you consider, for example:

```
> 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 ) )
```

    Now add the effects one at at time and test whether age at diagnosis or current age is the better predictor. If you just want to test whether adding a new term to the model it is convenient to use the `update` function, for example:

```
> 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" )
```

Is there any indication of whether current age or age at diagnosis, or current date or date of diagnosis is the better choice?

19. Fit the models with age at diagnosis and date of diagnosis as explanatory variables instead. To extract the effects you also need new contrast matrices, because the deaths are distributed differently along these "entry"-variables.

20. Show the effects together with the effects from the model with three time scales (that is the model with current age and current date of follow-up.

1.12.1 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, 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 to 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 rate 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.

21. 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 6 months after the left endpoint of the interval — recall we split the follow-up in 12 month intervals).

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

```
> str( SL )
> SL$Am <- floor( SL$A+0.5 )
> SL$Pm <- floor( SL$P+0.5 )
> data( M.dk )
> str( M.dk )
> M.dk <- transform( M.dk, Am = A,
+                          Pm = P,
+                         sex = factor( sex, labels=c("M","F") ) )
> str( M.dk )
```

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

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

This merge only takes rows that have information from both datasets, hence the slightly fewer rows in `SLr` than in `SL`.

22. Compute the expected number of deaths as the person-time multiplied by the corresponding population rate, and put it in a new variable. Use `stat.table` to make a table of observed, expected and the ratio (SMR) by age (suitably grouped) and sex.

23. Then model the SMR using age and date of diagnosis and diabetes duration as explanatory variables, including the log-expected-number instead of the log-person-years as offset, using separate models for men and women. Remember to exclude those units where no deaths in the population occur (that is where the rate is 0).

    Plot the estimates as you did before for the rates. What do the extracted effects represent now?

24. Is there any difference between SMR for males and females?

25. Fit the model with common SMR for the two sexes. Plot the estimated common effects for SMR.

26. Try to simplify the model to one with a simple 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.

27. What are the estimated annual change in SMR by date of diagnosis and by duration after 2 years?

Interaction models

This section is quite technical, but nonetheless important from a practical point of view, since it contains exampales of how to construct and report continuous-continuous interactions.

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

    ```
    > Six <- update( Sx, . ~. + I(A-dur):Ns(dur,knots=kn.dur) )
    > anova( Six, Sx, test="Chisq" )
    > ci.exp( Six )
    ```

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

    ```
    > Six.A <- ci.exp( Six, ctr.mat=cbind(1,AC,rf.P,dR,dR*pr.A) )
    > Six.P <- ci.exp( Six, subset="P"     , ctr.mat=cbind(pr.P-rf.P) )
    > Six.d <- ci.exp( Six, subset="kn.dur", ctr.mat=cbind(dC-dR,(dC-dR)*50) )
    > for( a in seq(55,90,5) ) Six.d <- cbind( Six.d,
    ```

```
+                 ci.exp( Six, subset="kn.dur", ctr.mat=cbind(dC-dR,(dC-dR)*a) ) )
> dim( Six.d )
> par( mfrow=c(1,3), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( pr.A, Six.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, Six.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, Six.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 )
```

29. 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, for patients diagnosed with DM at different ages.

    Note the trick with putting an `NA` at the end of `pts` and then stacking all the predictions for persons diagnosed at different ages. This means that the curve plotted will be broken between the different ages at diagnosis. Also note that we use the recycling rule when setting up the data frame.

```
> 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.si <- exp(sapply(predict( Six, newdata=nd, se.fit=TRUE )[1:2],cbind) %*% ci.mat())
> A.sm <- exp(sapply(predict( Sx , newdata=nd, se.fit=TRUE )[1:2],cbind) %*% ci.mat())


> 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.sm),
+           type="l", lty=rep(c(1,3),each=3), lwd=c(3,1,1), col="forestgreen" )
> abline( h=1 )
```

30. 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, Six, Sx, test="Chisq" )
```

And we can use the exact same code to show the interaction and plot it along the others in a similar plot:

```
> A.sX <- exp(sapply(predict( SiX, newdata=nd, se.fit=TRUE )[1:2],cbind) %*% ci.mat())
> 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.si,A.sm),
+           type="l", lty=rep(c(1,3),c(6,3)), lwd=c(3,1,1),
+           col=rep(c("magenta","forestgreen"),c(3,6)) )
> abline( h=1 )
```

1.13 Causal inference

1.13.1 Proper adjustment for confounding in regression models

The first exercise of this session will ask you to simulate some data according to pre-specified causal structure (don't take the particular example too seriously) and see how you should adjust the analysis to obtain correct estimates of the causal efeects.

Suppose one is interested in the effect of beer-drinking on body weight. Suppose in reality the following (hypothetical) causal association structure holds:

- Beer-drinking has an effect on the body weight.
- People with higher body weight tend to have higher blood pressure
- Men drink more beer than women
- Men have higher body weight than women
- Beer-drinking increases blood pressure

The task is to simulate a dataset in accordance with this model, and subsequently analyse it to see, whether the results would allow us to conclude the true association structure.

1. Sketch a causal graph (not necessarily with R) to see, how should one generate the data
2. Suppose the association parameters are as follows:
   - People who drink beer weigh on average $2kg$ more than those who don't.
   - One kg difference in body weight corresponds in average to $0.5mmHg$ difference in (systolic) blood pressures
   - Men drink more beer than women — the probability of beer-drinking is 0.2 for females and 0.7 for males
   - Men weigh on average $10kg$ more than women
   - Beer-drinking increases blood pressure by $20mmHg$ in average .

   The R commands to generate the data are:

```
> sex <- c(rep(0,500),rep(1,500))  # 500 females, 500 males
> beer <- rbinom(1000,1,0.2+0.5*sex)
> weight <- 60 + 10*sex + 2*beer + rnorm(1000,0,7)
> bp <- 110 + 0.5*weight + 20*beer + rnorm(1000,0,10)
```

3. Now fit the following models for body weight as dependent variable and beer-drinking as independent variable. Look, what is the estimated effect size:
   (a) Unadjusted (just simple linear regression)
   (b) Adjusted for sex

(c) Adjusted for sex and blood pressure

4. Which (if any) of the models gives an unbiased estimate of the actual causal effect of interest?

5. How can the answer be seen from the graph?

6. Now change the data-generation algorithm so, that in fact beer does not have any effect on the body weight, but has a negative effect on the blood pressure (change +20 to -20). Look, what are the conclusions in the above models now. Which of the models is a correct one and how do you see it from the causal graph?

7. Now suppose beer-drinking does only affect the body weight of males, but not the weight of females. Change the algorithm to incorporate that *interaction*. Compare the results obtained from correctly adjusted models with and without interaction.

1.13.2 Instrumental variables estimation, Mendelian -randomization and assumptions

In the lecture slides it was shown that in a model for diabetes, both BMI and FTO genotype were significant (if a logistic regression model is fitted instead, the p-value becomes even smaller). Seeing such result in a real dataset may misleadingly be interpretes as an evidence of a direct effect of FTO genotype on diabetes. Conduct a simulation study to verify that one may see a significant genotype effect on outcome in such model if in fact the assumptions for Instrumental Variables estimation (Mendelian Randomization) are valid – genotype has a direct effect on the exposure only, whereas exposure-outcome association is confounded.

1. Start by generating the genotype variable as *Binomial(2,p)*, with $p = 0.2$:

```
> n <- 10000
> G <- rbinom(n,2,0.2)
> table(G)
```

2. Also generate the confounder variable U

```
> U <- rnorm(n)
```

3. Generate a continuous (normally distributed) exposure variable BMI so that it depends on G and U. Check with linear regression, whether there is enough power to get significant parameter estimates. For instance:

```
> BMI <- 10 + 0.5*G + 2*U + rnorm(n)
```

(You may change the parameters 10, 0.5 and 2 to something else, also you may replace `rnorm(n)` by `rnorm(n,0,3)`, for instance, to add more error variability.) Check whether you get the right regression coefficients back, regardless of whether you adjust for U or not:

```
> summary(lm(BMI~U+G))
> summary(lm(BMI~G))
```

4. Finally generate Y ("Blood glycose level") so that it depends on BMI and U (but not on G).

```
> Y <- 3 + 0.1*BMI + 1.5*U + rnorm(n,0,0.5)
```

5. Verify, that simple regression model for Y, with X as a covariate, results in a biased estimate of the causal effect (parameter estimate is different from what was generated)

```
> mxy<-lm(Y ~ BMI)
> summary(mxy)
```

(how different is the estimate from 0.1?) In the equation to generate Y, replace the coefficient of U by -3 – see whether the direction of bias has changed.

6. Try to obtain an IV (instrumental variables) estimate, using G as an instrument, by following the algorithm in the lecture notes (use two linear models and obtain a ratio of the parameter estimates). Does the estimate get closer to the generated effect size?

```
> mgy<-lm(Y ~ G)
> bgy<-mgy$coef[2]
> mgx<-lm(BMI ~ G)
> bgx<-mgx$coef[2]
> bgy/bgx
```

7. Estimate a regression model for Y with two covariates, G and BMI. Do you see a significant effect of G?

8. Could you explain analytically, why may one see a significant parameter estimate for G there?

9. (*optional*) Using library `sem` and function `tsls`, obtain a two-stage least squares estimate for the causal effect. Do you get the same estimate as before?

```
> library(sem)
> summary(tsls(Y ~ BMI, ~G))
```

10. (*optional*) A proper simulation study would require the analysis to be run several times, to see the extent of variability in the parameter estimates. A simple way to do it here would be using a `for`-loop. Try the following code (you may also modify some parameters and try again).

```
> n <- 10000
> nsim<-10       # 10 simulations (change it, if you want more)
> mr<-rep(NA,nsim)   # empty vector for the outcome parameters
> for (i in 1:nsim) { # start the loop
+ G <- rbinom(n,2,0.2)
+ U <- rnorm(n)
+ BMI <- 25 + 0.7*G + 2*U + rnorm(n)
+ Y <- 3 + 0.1*BMI - 1.5*U + rnorm(n,0,0.5)
+ mgy<-lm(Y ~ G)
+ bgy<-mgy$coef[2]
```

```
+ mgx<-lm(BMI ~ G)
+ bgx<-mgx$coef[2]
+ mr[i]<-bgy/bgx
+ }   # end the loop
```

Now look at the distribution of the parameter estimate:

```
> summary(mr)
```

Why are simulation exercises useful for causal inference?

If we simulate the data, we know the data-generating mechanism and the "true" causal effects. So this is a way to check, whether an analysis approach will lead to estimates that correspond to what is generated. One could expect to see similar phenomena in real data analysis, if the data-generation mechanism is similar to what was used in simulations.

1.14 Nested case-control study and case-cohort study: Risk factors of coronary heart disease

In this exercise we shall apply both the nested case-control (NCC) design and the case-cohort (CC) design in sampling control subjects from a defined cohort or closed study population. The case group comprises those cohort members who die from coronary heart disease (CHD) during a > 20 years follow-up of the cohort. The risk factors of interest are cigarette smoking, systolic blood pressure, and total cholesterol level.

Our study population is an occupational cohort comprising 1501 men working in blue-collar jobs in one Nordic country. Eligible subjects had no history of coronary heart disease when recruited to the study in the early 1990s. Smoking habits and many other items were inquired at baseline by a questionnaire, and blood pressure was measured by a research nurse, the values being written down on the questionnaire. Serum samples were also taken from the cohort members at the same time and were stored in a freezer. For some reason, the data in the questionnaires were not entered to any computer file, but the questionnaires were kept in a safe storehouse for further purposes. Also, no biochemical analyses were initially performed for the sera collected from the participants. However, dates of birth and dates of entry to the study were recorded in an electronic file.

In 2010 the study was suddenly reactivated by those investigators of the original team who were still alive then. As the first step mortality follow-up of the cohort members was executed by record linkage to the national population register, from which the dates of death and emigration were obtained. Another linkage was performed with the national register of causes of death in order to get the deaths from coronary heard disease identified. As a result a data file `occoh.txt` was completed containing the following variables:

`id`	= identification number,
`birth`	= date of birth,
`entry`	= date of recruitment and baseline measurements,
`exit`	= date of exit from mortality follow-up,
`death`	= indicator for vital status at the end of follow-up,
	= 1, if dead from any cause, and = 0, if alive,
`chdeath`	= indicator for death from coronary heart disease,
	= 1, if "yes", and 0, if "no".

This exercise is divided into five main parts:

(1) Description of the study base or the follow-up experience of the whole cohort, identification of the cases and illustrating the risk sets.

(2) Nested case-control study within the cohort: (i) selection of controls by risk set or time-matched sampling using function `ccwc()` in package `Epi`, (ii) collection of exposure data for cases and controls from the pertinent data base of the whole cohort to the case-control data set using function `merge()`, and (iii) analysis of case-control data using function `clogit()` in package `survival()`,

(3) Case-cohort study within the cohort: (i) selection of a subcohort by simple random sampling from the cohort, (ii) fitting the Cox model to the data by weighted partial

likelihood using function `coxph()` in package `survival()` with appropriate weighting and correction of estimated covariance matrix for the model coefficients; also using function `cch()` in package `survival()` for the same task.

(4) Comparison of results from all previous analyses, also with those from a full cohort design.

(5) Further tasks and homework.

1.14.1 Reading the cohort data, illustrating the study base and risk sets

1. Load the packages `Epi` and `survival`. Read in the cohort data file and name the resulting data frame as `oc`. See its structure and print the univariate summaries.

```
> library(Epi)
> library(survival)
> oc <- read.table("./data/occoh.txt",header=T)
> str(oc)
> summary(oc)
```

2. It is convenient to change all the dates into fractional calendar years

```
> oc$ybirth <- cal.yr(oc$birth)
> oc$yentry <- cal.yr(oc$entry)
> oc$yexit <- cal.yr(oc$exit)
```

We shall also compute the age at entry and at exit, respectively, as age will be the main time scale in our analyses.

```
> oc$agentry <- oc$yentry - oc$ybirth
> oc$agexit <- oc$yexit - oc$ybirth
```

3. As the next step we shall create a `lexis` object from the data frame along the calendar period and age axes, and as the outcome event we specify the coronary death.

```
> oc.lex <- Lexis( entry = list( per = yentry,
+                                age = yentry - ybirth ),
+                  exit = list( per = yexit),
+            exit.status = chdeath,
+                     id = id, data = oc)
> str(oc.lex)
> summary(oc.lex)
```

Now we plot the follow-up lines and outcome cases in a conventional Lexis diagram

```
> plot(oc.lex,
+   grid = list( seq(1990, 2010, 5), seq( 35, 85, 5)),
+          xlim = c(1990, 2010), ylim = c(35, 85),
+          lty.grid = 1, xaxs='i', yaxs = 'i',
+          xlab = "Calendar year", ylab = "Age (years)")
> points( oc.lex,  pch = c(NA, 16)[oc.lex$lex.Xst+1], cex=0.5)
```

4. As age is here the main time axis, we shall illustrate the *study base* or the follow-up lines and outcome events along the age scale, being ordered by age at exit. Vertical lines at those ages when new coronary deaths occur are drawn to identify the pertinent *risk sets*. For that purpose it is useful first to sort the data frame and the `lexis` object jointly by age at exit & age at entry, and to give a new ID number according to that order.

```
> oc.ord <- cbind(ID = 1:1501, oc[ order( oc$agexit, oc$agentry), ] )
> oc.lexord <- Lexis( entry = list( age = agentry ),
+                        exit = list( age = agexit),
+                exit.status = chdeath,
+                          id = ID, data = oc.ord)
> plot(oc.lexord,
+    time.scale = "age",
+          xlim = c(40, 80),
+          xlab = "Age (years)" )
> points(oc.lexord, time.scale = "age",
+     pch = c(NA, 16)[oc.lexord$lex.Xst+1], cex=0.5)
> with( subset(oc.lexord, lex.Xst==1),
+    abline( v = agexit, lty = 3, lwd = 0.5 ))
```

Now we zoom the graphical illustration of the risk sets into event times occurring between 50 to 58 years.

```
> plot(oc.lexord,
+    time.scale = "age", ylim = c(5, 65),
+          xlim = c(50, 58),
+          xlab = "Age (years)" )
> points(oc.lexord, time.scale = "age",
+     pch = c(NA, 16)[oc.lexord$lex.Xst+1], cex=0.5)
> with( subset(oc.lexord, lex.Xst==1),
+    abline( v = agexit, lty = 3, lwd = 0.5 ) )
```

1.14.2 Nested case-control study

We shall now employ the strategy of *risk-set sampling* or *time-matched* sampling of controls, *i.e.* we are conducting a *nested case-control study* within the cohort.

1. The risk sets are defined according to the age at diagnosis of the case. Further matching is applied for age at entry by 1-year agebands. For this purpose we first generate a categorical variable `agen2` for age at entry

```
> oc.lex$agen2 <- cut(oc.lex$agentry, br = seq(40, 62, 1) )
```

Matched sampling from risk sets may be carried out using function `ccwc()` found in the `Epi` package. Its main arguments are the times of `entry` and `exit` which specify the time at risk along the main time scale (here age), and the outcome variable to be given in the `fail` argument. The number of controls per case is set to be two, and the additional matching factor is given. – After setting the RNG seed (with your own number), make a call of this function and see the structure of the resulting data frame `cactrl` containing the cases and the chosen individual controls.

```
> set.seed(9863157)
> cactrl <-
+    ccwc(entry=agentry, exit=agexit, fail=chdeath,
+         controls = 2, match= agen2,
+         include = list(id, agentry),
+         data=oc.lex, silent=F)
> str(cactrl)
```

Check the meaning of the four first columns of the case-control data frame from the help page of function `ccwc()`.

2. Now we shall start collecting data on the risk factors for the cases and their matched controls, including determination of the total cholesterol levels from the frozen sera! The storehouse of the risk factor measurements for the whole cohort is file `occoh-Xdata.txt`. It contains values of the following variables.

`id` =	identification number, the same as in `occoh.txt`,
`smok` =	cigarette smoking with categories,
	1: "never", 2: "former", 3: "1-14/d", 4: "15+/d",
`sbp` =	systolic blood pressure (mmHg),
`tchol` =	total cholesterol level (mmol/l).

```
> ocX <- read.table("./data/occoh-Xdata.txt", header=T)
> str(ocX)
```

3. In the next step we collect the values of the risk factors for our cases and controls by merging the case-control data frame and the storehouse file. In this operation we use the `Map` variable of `cactrl` and `id` variable in `ocX` as the keys to link each individual case and control with his own data on risk factors.

```
> oc.ncc <- merge(cactrl, ocX[, c("id", "smok", "tchol", "sbp")],
+    by.x = "Map", by.y = "id")
> str(oc.ncc)
```

4. We shall treat smoking as categorical and total cholesterol and systolic blood pressure as quantitative risk factors, but the values of the latter will be divided by 10 to get more interpretable effect estimates.

   Convert the smoking variable into a factor.

```
> oc.ncc$smok <- factor(oc.ncc$smok,
+     labels = c("never", "ex", "1-14/d", ">14/d"))
```

5. It is useful to start the analysis of case-control data by simple tabulations by the categorized risk factors. Crude estimates of the rate ratios associated with them, in which matching is ignored, can be obtained as instructed in Janne's lecture on Poisson and logistic models on Saturday 23 May. We shall focus on smoking

```
> stat.table( index = list( smok, Fail ),
+           contents = list( count(), percent(smok) ),
+            margins = T, data = oc.ncc )
> smok.crncc <- glm( Fail ~ smok, family=binomial, data = oc.ncc)
> round(ci.lin(smok.crncc, Exp=T)[, 5:7], 3)
```

6. A proper analysis takes into account matching that was employed in the selection of controls for each case from the pertinent risk set further restricted to subjects who were about the same age at entry as the case was. Also, adjustment for the other risk factors is desirable. In this analysis function `clogit()` in `survival` package is utilized. It is in fact a wrapper of function `coxph()`.

```
> m.clogit <- clogit( Fail ~ smok + I(sbp/10) + tchol +
+        strata(Set), data = oc.ncc )
> summary(m.clogit)
> round(ci.exp(m.clogit), 3)
```

Compare these with the crude estimates obtained above.

1.14.3 Case-cohort study

Now we start applying the second major outcome-selective sampling strategy for collecting exposure data from a big study population

1. The subcohort is selected as a simple random sample ($n = 260$) from the whole cohort. The `id`-numbers of the individuals that are selected will be stored in vector `subcids`, and `subcind` is an indicator for inclusion to the subcohort.

```
> N <- 1501; n <- 260
> set.seed(1579863)
> subcids <- sample(N, n )
> oc.lex$subcind <- 1*(oc.lex$id %in% subcids)
```

2. We form the data frame `oc.cc` to be used in the subsequent analysis selecting the union of the subcohort members and the case group from the data frame of the full cohort. After that we collect the data of the risk factors from the data storehouse for the subjects in the case-cohort data

```
> oc.cc <- subset( oc.lex, subcind==1 | chdeath ==1)
> oc.cc <- merge( oc.cc, ocX[, c("id", "smok", "tchol", "sbp")],
+    by.x = "id", by.y = "id")
> str(oc.cc)
```

3. The next five lines of R script provide a graphical illustration of the lifelines contained in the case-cohort data. Lines for the subcohort non-cases are grey without bullet at exit, those for subcohort cases are blue with blue bullet at exit, and for cases outside the subcohort the lines are black and dotted with black bullets at exit.

```
> plot(subset(oc.lexord, chdeath==0 & id %in% subcids),
+    time.scale = "age",
+          xlim = c(40, 80), # ylim = c(18,210),
+          xlab = "Age (years)" )
> lines(subset(oc.lexord, chdeath==1 & id %in% subcids ),
+    time.scale = "age", lwd=0.5, col = 'blue' )
> points(subset(oc.lexord, chdeath==1 & id %in% subcids ),
+     time.scale = "age", pch = 16, col='blue',  cex=0.5)
> lines(subset(oc.lexord, chdeath==1 & !(id %in% subcids) ),
+    time.scale = "age", lty = 3, lwd=0.5, col = 'black' )
> points(subset(oc.lexord, chdeath==1 & !(id %in% subcids) ),
+     time.scale = "age", pch = 16, col='black',  cex=0.5)
```

4. Define the categorical smoking variable again.

```
> oc.cc$smok <- factor(oc.cc$smok,
+     labels = c("never", "ex", "1-14/d", ">14/d"))
```

A crude estimate of the hazard ratio for the various smoking categories k vs. non-smokers ($k = 1$) can be obtained by tabulating cases (D_k) and person-years (y_k) in the subcohort by smoking and then computing the relevant exposure odds ratio for each category:

$$\mathrm{HR}_k^{\mathrm{crude}} = \frac{D_k/D_1}{y_k/y_1}$$

```
> sm.cc <- stat.table( index = smok,
+    contents = list( Cases = sum(lex.Xst), Pyrs = sum(lex.dur) ),
+          margins = T, data = oc.cc)
> print(sm.cc, digits = c(sum=0, ratio=1))
> HRcc <- (sm.cc[ 1, -5]/sm.cc[ 1, 1])/(sm.cc[ 2, -5]/sm.cc[2, 1])
> round(HRcc, 3)
```

5. To estimate jointly the rate ratios associated with the categorized risk factors we now fit the pertinent Cox model applying the method of *weighted partial likelihood* as presented by Ling & Ying (1993) and Barlow (1994). The weights for all cases and non-cases in the subcohort are first computed and added to the data frame.

```
> N.nonc <- N-sum(oc.lex$chdeath)  # non-cases in whole cohort
> n.nonc <- sum(oc.cc$subcind * (1-oc.cc$chdeath)) # non-cases in subcohort
> wn <- N.nonc/n.nonc           # weight for non-cases in subcohort
> c(N.nonc, n.nonc, wn)
> oc.cc$w <- ifelse(oc.cc$subcind==1 & oc.cc$chdeath==0, wn, 1)
```

Next, the Cox model is fitted by the method of weighted partial likelihood using `coxph()`, such that the robust covariance matrix will be used as the source of standard errors for the coefficients.

```
> oc.cc$surob <- with(oc.cc, Surv(agentry, agexit, chdeath) )
> cc.we <- coxph( surob ~  smok + I(sbp/10)  + tchol, robust = T,
+        weight = w, data = oc.cc)
> summary(cc.we)
> round( ci.exp(cc.we), 3)
```

The covariance matrix for the coefficients may also be computed by the dfbeta-method. After that a comparison is made between standard errors from the naive, robust and dfbeta covariance matrix, respectively. You will see that the naive SEs are essentially smaller than those obtained by the robust and the dfbeta method, respectively.

```
> dfbw <- resid(cc.we, type='dfbeta')
> covdfb.we <- cc.we$naive.var +
+    (n.nonc*(N.nonc-n.nonc)/N.nonc)*var(dfbw[ oc.cc$chdeath==0, ] )
> cbind( sqrt(diag(cc.we$naive.var)), sqrt(diag(cc.we$var)),
+     sqrt(diag(covdfb.we))  )
```

6. The same analysis can also be done using function `cch()` in package `survival` with `method = "LinYing"` as follows:

```
> cch.LY <- cch( surob ~  smok + I(sbp/10)  + tchol, stratum=NULL,
+    subcoh = ~subcind, id = ~id,  cohort.size = N, data = oc.cc,
+     method ="LinYing" )
> summary(cch.LY)
```

7. The `summary()` method for the `cch()` object does not print the standard errors for the coefficients. The following comparison demonstrates numerically that the method of Lin & Ying is the same as weighted partial likelihood coupled with dfbeta covariance matrix.

```
> cbind( coef( cc.we), coef(cch.LY) )
> round( cbind( sqrt(diag(cc.we$naive.var)), sqrt(diag(cc.we$var)),
+     sqrt(diag(covdfb.we)), sqrt(diag(cch.LY$var))  ), 3)
```

1.14.4 Full cohort analysis and comparisons

Finally, suppose the investigators could afford to collect the data of risk factors from the storehouse for the whole cohort.

1. Let us form the data frame corresponding to the full cohort design and convert again smoking to be categorical.

```
> oc.full <- merge( oc.lex, ocX[, c("id", "smok", "tchol", "sbp")],
+    by.x = "id", by.y = "id")
> oc.full$smok <- factor(oc.full$smok,
+     labels = c("never", "ex", "1-14/d", ">14/d"))
```

Juts for comparison with the corresponding analysis in case-cohort data perform a similar crude estimation of hazard ratios associated with smoking.

```
> sm.coh <- stat.table( index = smok,
+    contents = list( Cases = sum(lex.Xst), Pyrs = sum(lex.dur) ),
+          margins = T, data = oc.full)
> print(sm.coh, digits = c(sum=0, ratio=1))
> HRcoh <- (sm.coh[ 1, -5]/sm.coh[ 1, 1])/(sm.coh[ 2, -5]/sm.coh[2, 1])
> round(HRcoh, 3)
```

2. Fit now the Cox model to the full cohort, and there is no need to employ extra tricks upon the ordinary `coxph()` fit.

```
> cox.coh <- coxph( Surv(agentry, agexit, chdeath) ~
+         smok + I(sbp/10)  + tchol, data = oc.full)
> summary(cox.coh)
```

3. Lastly, a comparison of the point estimates and standard errors between the different designs, including variants of analysis for the case-cohort design, can be performed.

```
> betas <- round(cbind( coef(cox.coh),
+      coef(m.clogit),
+      coef(cc.we), coef(cch.LY) ), 3)
> colnames(betas) <-  c("coh", "ncc", "cc.we", "cch.LY")
> betas
> SEs <- round(cbind( sqrt(diag(cox.coh$var)),
+     sqrt(diag(m.clogit$var)), sqrt(diag(cc.we$naive.var)),
+     sqrt(diag(cc.we$var)),  sqrt(diag(covdfb.we)),
+     sqrt(diag(cch.LY$var)) ), 3)
> colnames(SEs) <- c("coh", "ncc", "ccwe-nai",
+        "ccwe-rob", "ccwe-dfb", "cch-LY")
> SEs
```

You will notice that the point estimates of the coefficients obtained from the full cohort, nested case-control, and case-cohort analyses, respectively, are somewhat variable.

However, the standard errors across the NCC and different proper CC analyses are relatively similar. Those from a naive covariance matrix of a CC analysis, though, are practically equal to the SEs from the full cohort analysis, reflecting the fact that the naive analysis implicitly assumes there being as much information available as there is with full cohort data.

1.14.5 Further exercises and homework

1. If you have time, you could run both the NCC study and CC study again but now with a larger control group or subcohort; for example 4 controls per case in NCC and $n = 520$ as the subcohort size in CC. Remember resetting the seed first. Pay attention in the results to how much closer will be the point estimates and the proper SEs to those obtained from the full cohort design.

2. Instead of simple linear terms for `sbp` and `tchol` you could try to fit spline models to describe their effects.

3. A popular alternative to weighted partial likelihood in the analysis of case-cohort data is the *pseudo-likelihood method* (Prentice 1986), which is based on "late entry" to follow-up of the case subjects not belonging to the subcohort. A longer way of applying this approach, which you could try at home after the course, would first require manipulation of the `oc.cc` data frame, as outlined on slide 34. Then `coxph()` would be called like in model object `cc.we` above but now with weights = 1. Similar

corrections on the covariance matrix are needed, too. However, a shorter way is provided by function `cch()` which you can apply directly to the case-cohort data `oc.cc` as before but now with `method = "Prentice"`. – Try this and compare the results with those obtained by weighted partial likelihood in models `cc.we` and `cch.LY`.

4. Yet another computational solution for maximizing weighted partial likelihood is provided by a combination of functions `twophase()` and `svycoxph()` of the `survey` package. The approach is illustrated with an example in a vignette "Two-phase designs in epidemiology" by Thomas Lumley (see `http://cran.r-project.org/web/packages/survey/vignettes/epi.pdf`, p. 4–7). – You can try this at home and check that you would obtain similar results as with models `cc.we` and `cch.LY`.

1.15 Renal complications: Time-dependent variables and multiple states

The following practical exercise is based on the data from paper:

P Hovind, L Tarnow, P Rossing, B Carstensen, and HH Parving: Improved survival in patients obtaining remission of nephrotic range albuminuria in diabetic nephropathy. *Kidney Int*, **66**(3):1180–1186, Sept 2004.

You can find a `.pdf`-version of the paper here:
http://BendixCarstensen.com/~bxc/AdvCoh/papers/Hovind.2004.pdf

1.15.1 The renal failure dataset

The dataset `renal.dta` contains data on follow up of 125 patients from Steno Diabetes Center. They enter the study when they are diagnosed with nephrotic range albuminuria (NRA). This is a condition where the levels of albumin in the urine is exceeds a certain level as a sign of kidney disease. The levels may however drop as a consequence of treatment, this is called remission. Patients exit the study at death or kidney failure (dialysis or transplant).

Table 1.2: *Variables in* `renal.dta`.

`id`	Patient id
`sex`	1=male, 2=female
`dob`	Date of birth
`doe`	Date of entry into the study (2.5 years after NRA)
`dor`	Date of remission. Missing if no remission has occurred
`dox`	Date of exit from study
`event`	Exit status: 1,2,3=event (end stage renal disease, ESRD), 0=censored

1. The dataset is in Stata-format. Read the dataset using `read.dta` from the `foreign` package, and take a look at the first 20 records:

```
> library( foreign )
> renal <- read.dta( "http://BendixCarstensen.com/SPE/data/renal.dta" )
> head( renal )
```

2. First do a simple survival curve for the cohort. The event we are interested in is given by codes 1,2 or 3 in the variable `event`

   The function `Surv()` defines a survival object, used as the response variable in a survival analysis. If called with two arguments the first is the survival time and the second the event indicator. If called with three arguments, the first is the time of entry, the second is the time of exit and the third the event indicator.

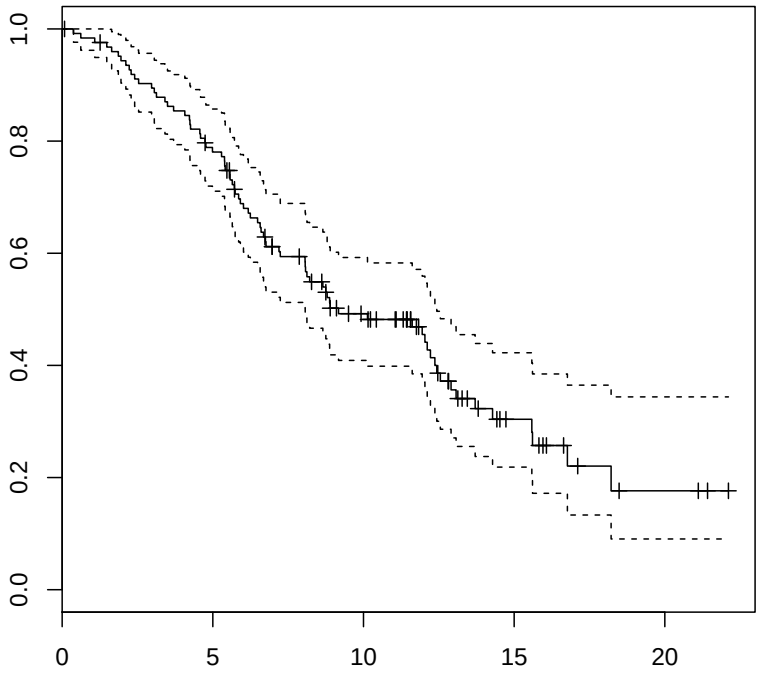


Figure 1.1: *Kaplan-Meier estimator of the event probability from the renal data set.*

```
> library( survival )
> sf <- survfit( Surv( dox-doe, event>0 ) ~ 1, data=renal )
> plot( sf )
```

3. Now use `Lexis` to declare the data as survival data with age, calendar time and time since entry into the study as timescales. Note that any coding of event > 0 will be labeled "ESRD", i.e. renal death (death of kidney (transplant or dialysis), or person).

```
> library( Epi )
> Lr <- Lexis( entry = list( per=doe,
+                            age=doe-dob,
+                            tfi=0 ),
+              exit = list( per=dox ),
+       exit.status = factor( event>0, labels=c("NRA","ESRD") ),
+              data = renal )
> str( Lr )
> summary( Lr, scale=1000 )
```

4. Visualize the data in a Lexis-diagram, using the `plot` method for `Lexis` objects. What do you see?

```
> plot( Lr, col="black", lwd=3 )
> subset( Lr, age<0 )
```

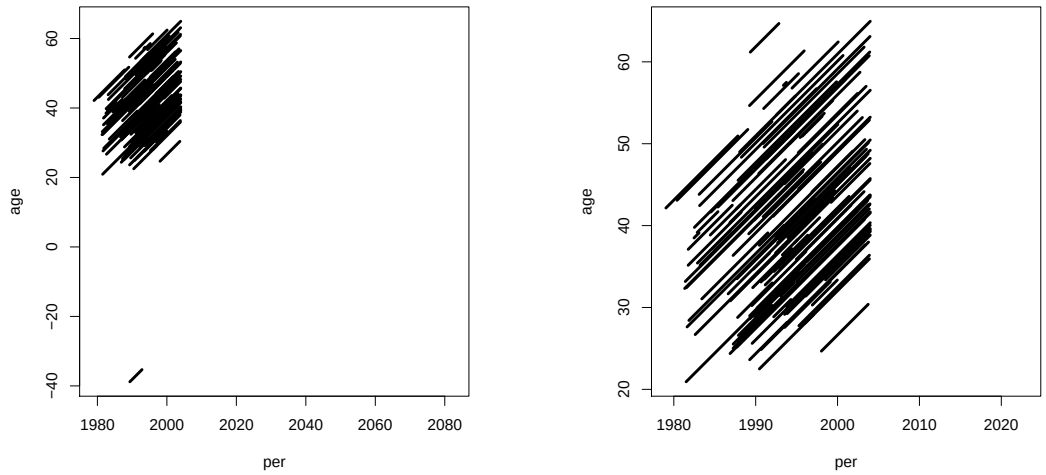


Figure 1.2: *Default Lexis diagram before and after correction of the obvious data outlier.*

5. Correct the data, e.g. by:

```
> Lr <- transform( Lr, age=age+100*(dob>2000), dob=dob-100*(dob>2000) )
> str( Lr )
> subset( Lr, id==586 )


> plot( Lr, col="black", lwd=3 )
```

6. Now try to produce a slightly more fancy Lexis diagram. Note that we have a x-axis of 40 years, and a y-axis of 80 years, so when specifying the output file adjust the *total* width of the plot so that the use `mai` to specifiy the margins of the plot leaves a plotting area twice as high as wide:

```
> # pdf( "lexis-fancy.pdf", height=80/5+1, width=40/5+1 )
> # x11( height=80/5+1, width=40/5+1 )
> par( mai=c(3,3,1,1)/4, mgp=c(3,1,0)/1.6 )
> plot( Lr, 1:2, col=c("blue","red")[Lr$sex], lwd=3, grid=0:20*5,
+       xlab="Calendar time", ylab="Age",
+       xlim=c(1970,2010), ylim=c(0,80), xaxs="i", yaxs="i", las=1 )
> # dev.off()
```

7. Make a Cox-regression analysis with the variables sex and age at entry into the study. Give the hazard ratio between males and females and between two persons who differ 10 years in age at entry. Give the 95% confidence intervals for this as well.

```
> library( survival )
> mc <- coxph( Surv( tfi, tfi+lex.dur, lex.Xst=="ESRD" ) ~
+              I(age/10) + sex, data=Lr )
> summary( mc )
```

8. Show the estimated survival function for a male aged 50.

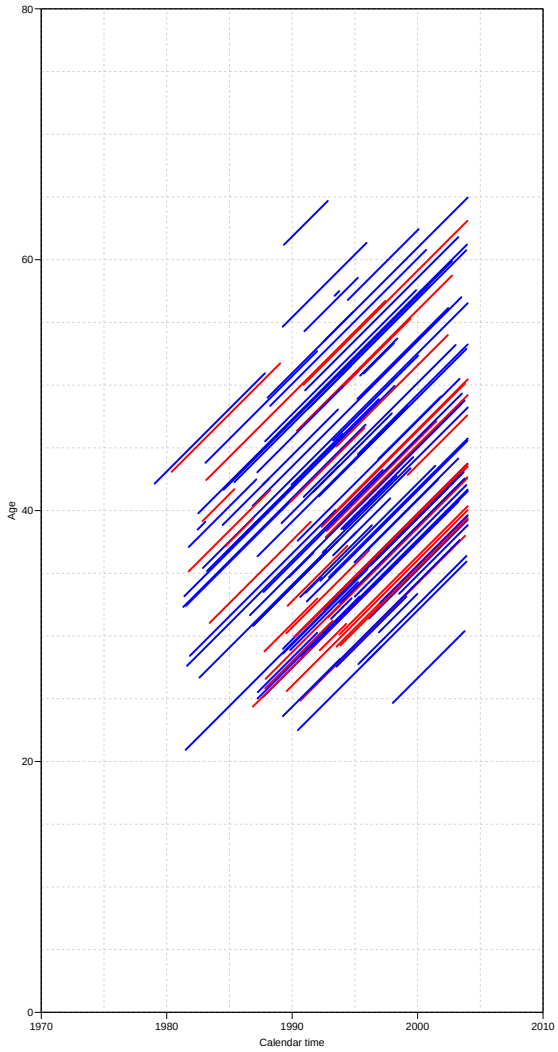


Figure 1.3: *The more fancy version of the Lexis diagram for the renal data.*

```
> plot( survfit( mc, newdata=data.frame(age=50,sex=1) ), col="black", lwd=3 )
```

9. The main focus of the paper is however to assess whether occurrence of remission (return to a lower level of albumin excretion, an indication of kidney recovery) influences mortality.

   "Remission" is a time-dependent variable which is initially 0, but takes the value when remission occurs. In order to handle this, each person who gets remission must have two records:

   - One record for the time before remission, where entry is `doe`, exit is `dor`, remission is 0, and event is 0.
   - One record for the time after remission, where entry is `dor`, exit is `dox`, remission is 1, and event is 0 or 1 according to whether the person had an event at `dox`.

   This is accomplished using the `cutLexis` function on the just defined `Lexis` object. Remember to declare the "NRA" state as a precursor state, i.e. a state that is *less*

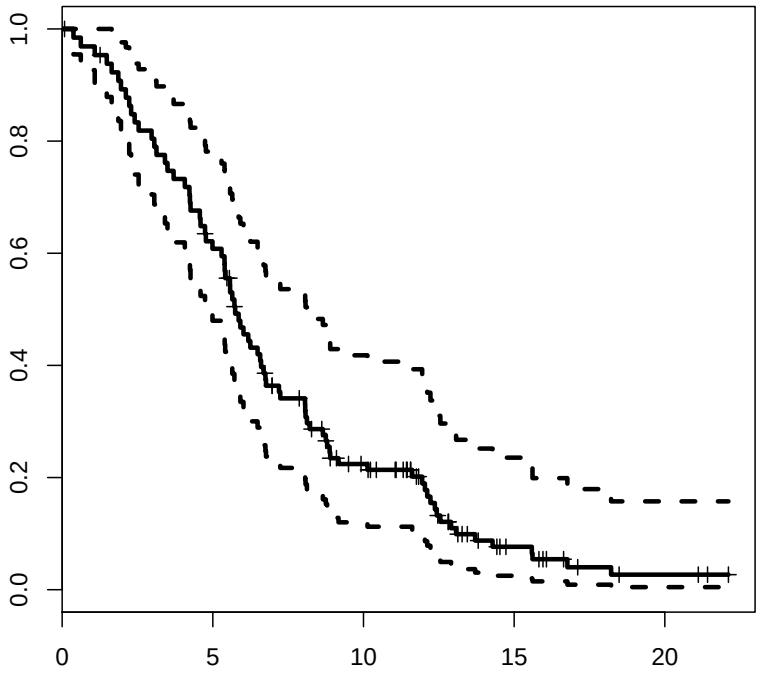


Figure 1.4: *Estimated survival for a 50 year old male.*

severe than "Remission" in the sense that a person will stay in the "Remission" state unless he goes to the "ESRD" state.

```
> Lc <- cutLexis( Lr, cut=Lr$dor, timescale="per",
+                 new.state="Rem", precursor.states="NRA" )
> summary( Lc )
```

10. Show how the states are connected and the number of transitions between them by using `boxes`. This is an interactive command that requires you to click in the graph window:

    ```
    > boxes( Lc )
    ```

    Alternatively you can let R try to place the boxes for you, and even compute rates (in this case per 100 PY):

    ```
    > boxes( Lc, boxpos=TRUE, scale.R=100 )
    ```

    How many transitions are there from remission to death?

11. Now make a Lexis diagram where different colouring is used for different segments of the follow-up:

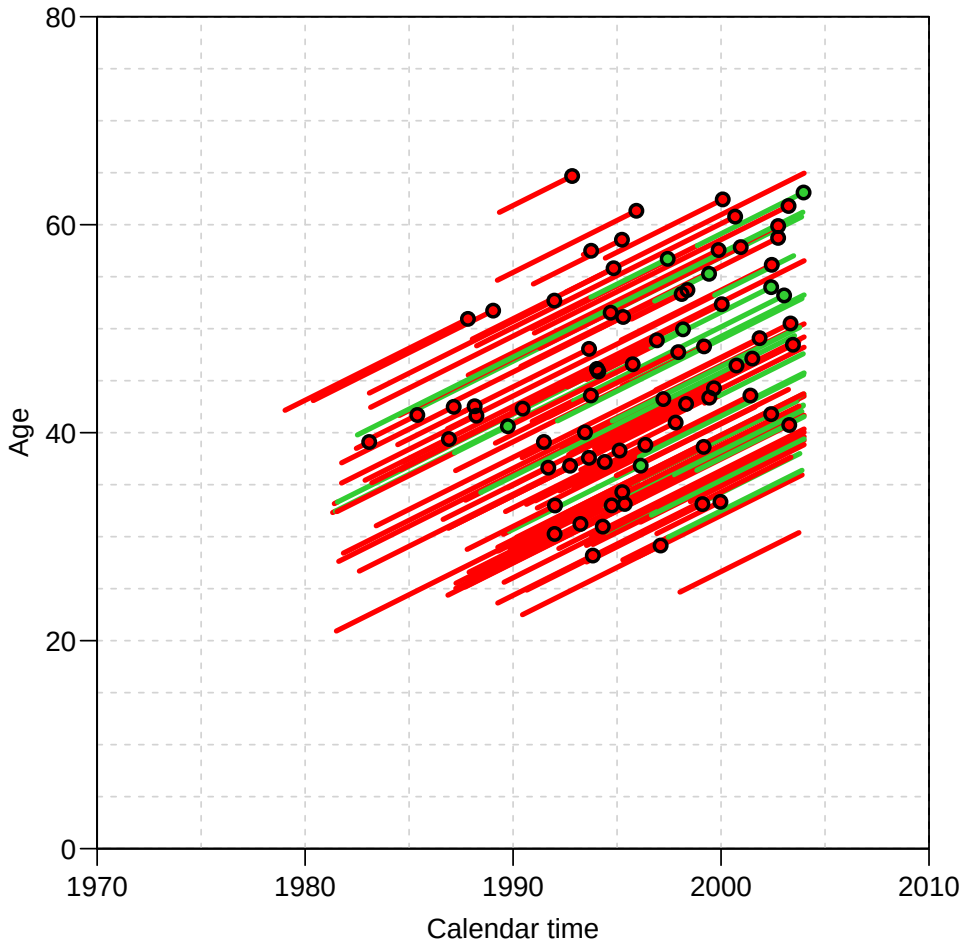


Figure 1.5: *Lexis diagram for the split data, where time after remission is shown in green.*

```
> par( mai=c(3,3,1,1)/4, mgp=c(3,1,0)/1.6 )
> plot( Lc, col=c("red","limegreen")[(Lc$lex.Cst=="Rem")+1],
+       xlab="Calendar time", ylab="Age",
+       lwd=3, grid=0:20*5, xlim=c(1970,2010), ylim=c(0,80), xaxs="i", yaxs="i", las=1
> points( Lc, pch=c(NA,16)[(Lc$lex.Xst=="ESRD")+1],
+             col=c("red","limegreen")[(Lc$lex.Cst=="Rem")+1])
> points( Lc, pch=c(NA,1)[(Lc$lex.Xst=="ESRD")+1],
+             col="black", lwd=2 )
```

12. List the first records of the dataframe `Lc` (using `head()`) and make sure that you understand how persons and follow-up time is represented in this dataset.

13. Make a Cox-regression of mortality (i.e. endpoint "ESRD") with sex, age at entry and remission as explanatory variables, and using time since entry as timescale. Note how the variable `lex.Cst` is used as a time-dependent variable:

```
> m1 <- coxph( Surv( tfi, tfi+lex.dur, lex.Xst=="ESRD" ) ~
+              sex + I((doe-dob-50)/10) + (lex.Cst=="Rem"), data=Lc )
> summary( m1 )
```

14. What is the assumptions about the rate of ESRD/death between persons in remission and persons not?

15. What is the assumption about the incidence rates of remission? (cf. figure 1.6).

1.15.2 Splitting follow-up time

In order to explore the effect of remission on the ESRD/death rate, we will split the data further into small pieces of follow-up. To this end we use the function `Lexis`. The rates can then be modeled using a multiplicative Poisson-model, and the shape of the *rates* be explored. Furthermore, we can allow effects of both time since NRA and current age. To allow this we will use splines, so we need the splines package, too.

16. Split the follow-up time every 0.5 years after entry, and make sure that the number of events and risk time is the same as before:

```
> sLc <- splitLexis( Lc, "tfi", breaks=seq(0,30,0.5) )
> summary( Lc, scale=1000 )
> summary(sLc, scale=1000 )
```

17. Now try to fit the Poisson-model corresponding to the Cox-model we fitted previously. The function `ns()` produces a model matrix corresponding to a piecewise cubic function.

```
> library( splines )
> mp <- glm( as.numeric(lex.Xst=="ESRD") ~
+                   ns( tfi, df=4 ) +
+                   factor(sex) + I((doe-dob-40)/10) +  (lex.Cst=="Rem") +
+                   offset( log(lex.dur) ),
+            family = poisson, data=sLc )
> summary( mp )
```

Another possibility, which is formally more correct is to use the midpoint of the intervals in which the time is split. These are accessed by the function `timeBand`:

```
> mx <- glm( as.numeric(lex.Xst=="ESRD") ~
+                   ns( timeBand(sLc,"tfi","mid"), df=4 ) +
+                   factor(sex) + I((doe-dob-40)/10) +  (lex.Cst=="Rem") +
+                   offset( log(lex.dur) ),
+            family = poisson, data=sLc )
> summary( mx )
```

Note the similarity of the results.

18. What kind of assumptions are made in this model about the transition rates in the figure 1.6?

19. You can extract the parameters from the models using `ci.lin`, try:

```
> ci.lin( mp )
> ci.lin( mp, subset=6:8 )
> ci.lin( mp, subset=6:8, Exp=TRUE )
> ci.lin( mp, subset=c("sex","dob","Cst"), Exp=TRUE )
```

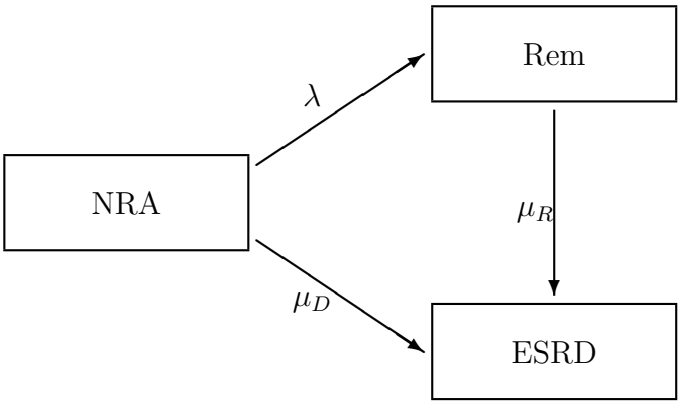


Figure 1.6: *Illness-death model for the renal data.*

The latter is wrapped up in the function `ci.exp()`:

```
> ci.exp( mp, subset=c("sex","dob","Cst") )
```

Compare with the estimates from the Cox-model. Use:

```
> ci.exp( m1 )
> ci.exp( mp, subset=c("sex","dob","Cst") ) /
+ ci.exp( m1 )
```

1.15.3 Splines and predictions

In the model we have used splines to model the effect of the covariate `tfi`, time since NRA. The model assumes that the rates are constant in 6-month intervals (the units we split into). The spline models how the rates in each of these intervals relate to each other. Spline functions are functions that are cubic between a set of *knots* and constrained to have the same 0th, 1st and 2nd derivative at the knots. The natural splines generated by `ns()` further have the property that they are linear beyond the outermost, so called boundary, knots. Splines are linear combinations of polynomials, and so the models are just usual linear models. The only requirement to generate the columns of the model matrix is fixing the knots. Even if we have 40+ intervals on this time-scale, we only used 8 parameters to model the effect.

To see the effect of time, the estimated coefficients for the time effect must be multiplied with a matrix where each row represents a particular time. This can be done explicitly for chosen values of time by using the `ns` function.

However, the function `ns` chooses the knots based on the data. So if we want to have the same parametrization for a new set of points, we must first extract the knots and boundary knots used in the model. They are stored in the attributes of the `ns` object, and can be found using the function `attr()`. These are then used to generate a model matrix with rows corresponding to a prespecified set of time-points. This matrix is then multiplied with the estimated parameters to get the estimated effects. This is simplest achieved by using `ci.exp`, which also has the facility to select subsets of parameters from a function and to extract standard errors too.

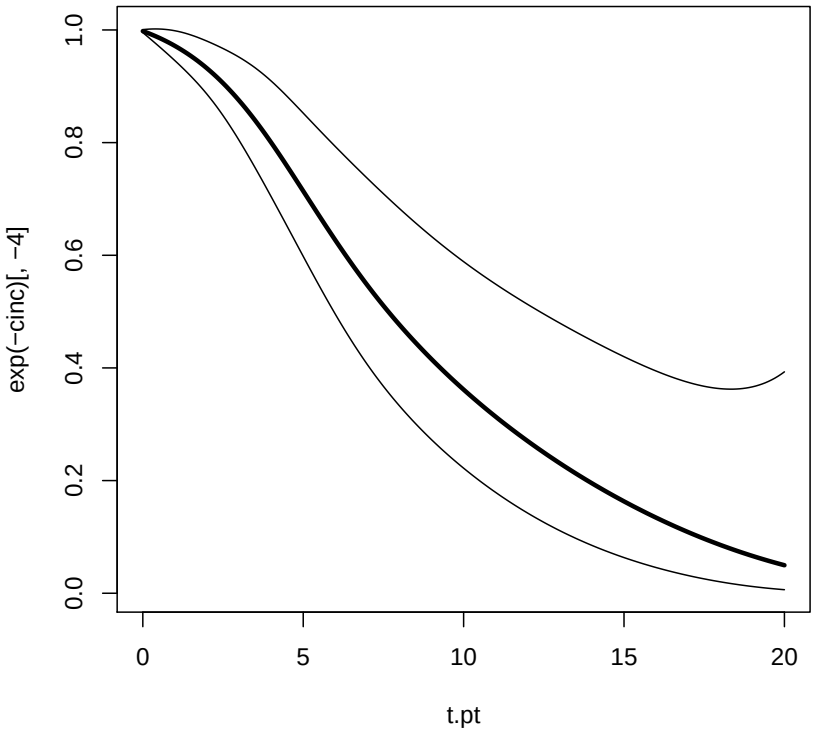


Figure 1.7: *Estimated survival curve for a man entering the study at 40, assuming the the rate of remission is 0.*

20. Plot the survival function for a male aged 40 at entry (NRA), based on the model `mp`, using the function `ci.cum`.

```
> # Get the knots
> t.kn <- attr( ns( sLc$tfi, df=4 ), "knots" )
> t.Bo <- attr( ns( sLc$tfi, df=4 ), "Boundary.knots" )
> # Then decide the points of prediction
> t.pt <- seq( 0,20,0.1 )
> nt <- length( t.pt )
> # Then make the contrast matrix to generate log-rates, and use ci.cum
> # to compute the cumulative rates
> MT <- ns( t.pt, knots=t.kn, Bo=t.Bo )
> cmat <- cbind( 1, MT, 1, 0, 0 )
> cinc <- ci.cum( mx, ctr.mat=cmat, intl=0.1 )
> # Note that there is no guarantee that the lower limit of the c.i. for
> # the cumulative incidence is positive:
> cinc[1:20,]
> matplot( t.pt, exp( -cinc )[,-4], type="l", lwd=c(3,1,1), lty=1, col="black" )
```

21. What is the problem with the previous question?

    When there is a time dependent variable involved, in this case the occurrence of remission, the survival function is only meaningful if a model for the occurrence of

this is assumed too. So the survival function is quite artificially conditional on a remission rate of 0.

This is a general concern in *multistate* models; in order to make sensible statements about the probability of being in any one state, it is necessary to know *all* transitions rates, *i.e.* to have a statistical model fro them.

22. Use the extracted knots to generate the relevant model matrix, and use this for plotting effect of time since NRA:

```
> MT <- ns(              t.pt   , knots=t.kn, Bo=t.Bo ) -
+       ns( rep(0,length(t.pt)), knots=t.kn, Bo=t.Bo )
> # The extract, multiply and exponentiate:
> t.eff <- ci.exp( mp, subset="ns", ctr.mat=MT )
> # Then plot the effects
> matplot( t.pt, t.eff, log="y", type="l", lty=1, col="black", lwd=c(3,1,1),
+          xlab="Time since NRA", ylab="Rate ratio" )
> abline( h=1 )
```

What is shown in this plot?
How relevant is that?

23. The next step is to include a spline effect of current age in the model too. Try:

```
> ma <- glm( as.numeric(lex.Xst=="ESRD") ~
+                   ns( tfi, df=4 ) +
+                   ns( age, df=4 ) +
+                   sex + I((doe-dob)/10) + as.numeric( lex.Cst=="Rem" ) +
+                   offset( log(lex.dur) ),
+            family = poisson, data=sLc )
> summary( ma )
```

Why is the effect of age at entry (`I((doe-dob)/10)`) set to `NA` in the output from the model?

24. Is there a non-linear effect of current age on mortality rates? You can compare models using `anova`:

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

25. *(Esoteric, hard, but quite important)* Try to replace the `ns( tfi, df=4 )` with the term `ns( timeBand(sLc,"tfi","mid"), df=4 )` in the model. Why is the age at entry effect not `NA` in this case?

26. In this model (i.e. `ma`) we can show the effect of time since NRA as before, as well as the effect of current age. Try:

```
> # First get the knots in order to generate the splines
> a.kn <- attr( ns( sLc$age, df=4 ), "knots")
> a.Bo <- attr( ns( sLc$age, df=4 ), "Boundary.knots")
> # Now we can make the matrix to multiply with the coefficients:
> a.pt <- seq( 40,70,0.2)
> Ma <- ns(               a.pt   , knots=a.kn, Bo=a.Bo ) -
+       ns( rep(60,length(a.pt)), knots=a.kn, Bo=a.Bo )
```

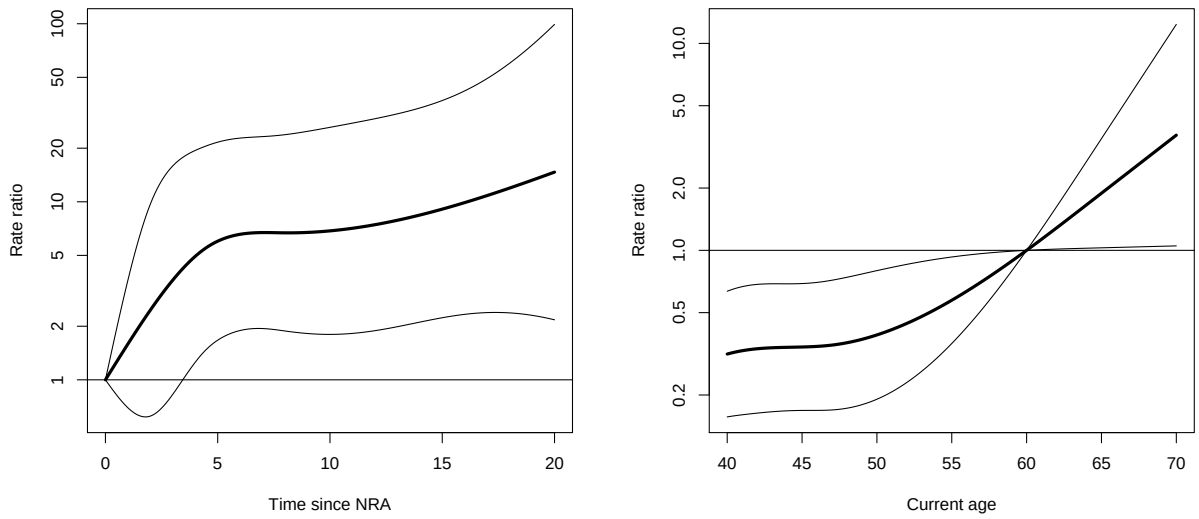


Figure 1.8: *Rate-ratio of ESRD by time since entry, relative to time at entry and by current age, relative to age 60.*

```
> a.eff <- ci.exp( ma, subset="age", ctr.mat=Ma )
> matplot( a.pt, a.eff, log="y", type="l", lty=1, col="black", lwd=c(3,1,1),
+          xlab="Current age", ylab="Rate ratio" )
> abline( h=1 )
```

How is the rate (as a function of age) behaving relative to what you would expect for the general population?

27. We are however also interested in knowing the absolute magnitude of the rates of ESRD/death for patients without remission. This would include the intercept as well as the rate-ratio function.

    The times where we want the rates should be the same as before, but now we must specify an intercept (just a column of 1s), the sex (for males just a column of 1s) and the age. The latter is a matrix of identical rows, each corresponding to the age-effect at 60 years, say. Try:

```
> MTa <- ns( rep(60,length(t.pt)), knots=a.kn, Bo=a.Bo )
> T.inc <- ci.exp( ma, subset=c("Int","tfi","age","sex"),
+                  ctr.mat=cbind(1,MT,MTa,1) )
> matplot( t.pt, T.inc, log="y", type="l", lty=1, col="black", lwd=c(5,2,2), las=1,
+          xlab="Time since NRA", ylab="Rate per year", ylim=c(0.01,5) )
```

    What is the current age at each time point of this curve?

28. Now try to make the same prediction of rates, but this time for a person that is 50 years *at entry*. Then the rows of the age-effects matrix should correspond to age at entry *plus* time since entry (i.e. current age):

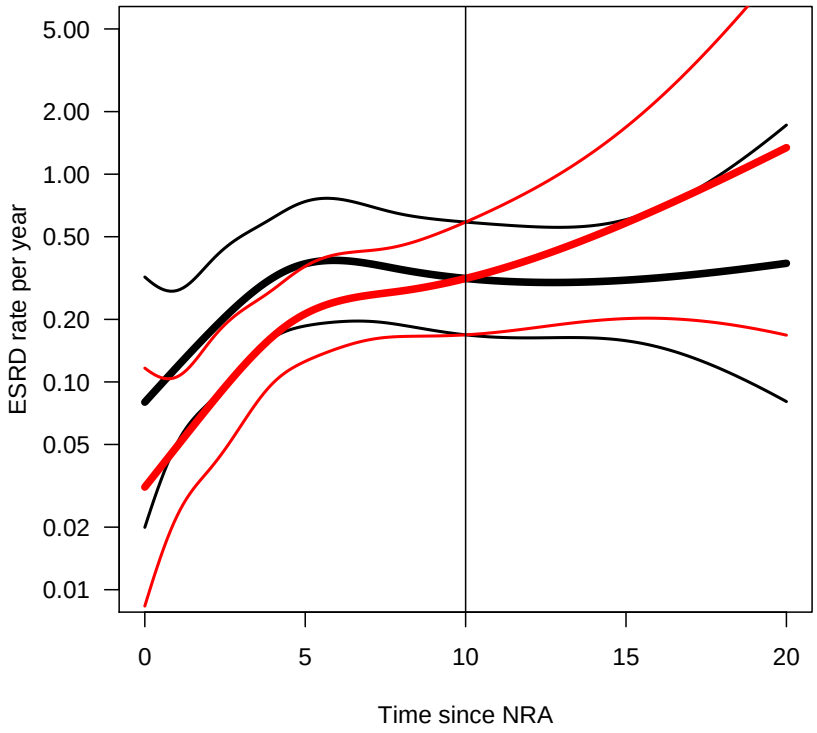


Figure 1.9: *Rate of ESRD by time since entry, for a current age of 60 (black) and an entry age of 50 (red).*

```
> MTa <- ns( 50+t.pt, knots=a.kn, Bo=a.Bo )
> t.inc <- ci.exp( ma, subset=c("Int","tfi","age","sex"),
+                      ctr.mat=cbind(1,MT,MTa,1) )
```

Now plot these estimated rates on top of the other:

```
> matplot( t.pt, T.inc, log="y", type="l", lty=1, col="black", lwd=c(5,2,2), las=1,
+          xlab="Time since NRA", ylab="ESRD rate per year", ylim=c(0.01,5) )
> matlines( t.pt, t.inc, type="l", lty=1, col="red", lwd=c(5,2,2) )
> abline( v=10 )
```

What is the meaning of the latter set of rates as opposed to the former?

29. How would you accomplish this analysis with a Cox-model?

30. Apart from the two timescales, time since NRA and current age, a third timescale may be of interest, namely time since remission. However this is only relevant for persons who actually have a remission, so start by checking how many events there are in this group:

```
> summary( sLc )
```

How many go in remission, and how many deaths are in this group?

31. With this rather limited number of events we can certainly not expect to be able to model anything more complicated than a linear trend with time since remission.

    The variable we want to have in the model is current date (`per`) minus date of remission (`dor`): `per-dor)`, but *only* positive values of it. This can be fixed by using `pmax()`, but we must also deal with all those who have missing values, so we use the construct:

    ```
    > pmax( per-dor, 0, na.rm=TRUE )
    ```

    Make sure that you understand what goes on here.

32. We can now expand the model with this variable. We need not write the entire model statement again, we can just say:

    ```
    > mx <- update( ma, . ~ . + pmax( (per-dor)/10, 0, na.rm=TRUE ) )
    > summary( mx )
    ```

    Is the effect significant? Can a substantial effect of time since remission be ruled out?

33. What is your overall conclusion — is it consonant with the conclusion in the paper?

1.15.4 Prediction in a multistate model

In this exercise we expand the previous one so that we can actually make proper statements about the survival probabilities. But in order to do this we must know not only how the occurrence of remission influences the rate of death/ESRD, but we must also model the occurrence of remission itself.

The following exercise will be quite similar to the example in the help file for `simLexis` (which you should read now!).

34. The rates of ESRD were modelled by a Poisson model with effects of age and time since NRA. But in the modelling of the remission rates, the number of events is rather small, so we restrict this model to only time since NRA and sex:

    ```
    > mr <- glm( as.numeric(lex.Xst=="Rem") ~
    +                   ns( tfi, df=4 ) + sex +
    +                   offset( log(lex.dur) ),
    +            family = poisson,
    +            data=subset( sLc, lex.Cst=="NRA" ) )
    > summary( mr )
    ```

    Is there an effect of sex?

35. How is the effect of time since NRA?

    In order to answer this, try to set up the relevant contrast matrices as previously. Note that this needs to be done afresh, since the knots automatically chosen by `ns` are different from those used in the model for the ESRD outcome. But we must construct the knots afresh using `ns`:

```
> # Get the knots
> ( r.kn <- attr( ns( subset(sLc,lex.Cst=="NRA")$tfi, df=4 ), "knots" ) )
> ( r.Bo <- attr( ns( subset(sLc,lex.Cst=="NRA")$tfi, df=4 ), "Boundary.knots" ) )
> # Construct the matrix to use
> MR <- ns(              t.pt   , knots=r.kn, Bo=r.Bo ) -
+       ns( rep(0,length(t.pt)), knots=r.kn, Bo=r.Bo )
> r.eff <- ci.exp( mr, subset="ns", ctr.mat=MR )
> # Then plot the effects
> matplot( t.pt, r.eff, log="y", type="l", lty=1, col="black", lwd=c(3,1,1),
+          ylim=c(0.1,10), xlab="Time since NRA", ylab="Remission rate ratio" )
> abline( h=1 )
```

Is there any effect of time since NRA?

What happens if you use, say, 5 years since NRA as reference?

You may want to try:

```
> MR <- ns(              t.pt   , knots=r.kn, Bo=r.Bo ) -
+       ns( rep(5,length(t.pt)), knots=r.kn, Bo=r.Bo )
> r.eff <- ci.exp( mr, subset="ns", ctr.mat=MR )
> # Then plot the effects
> matplot( t.pt, r.eff, log="y", type="l", lty=1, col="black", lwd=c(3,1,1),
+          ylim=c(0.1,10), xlab="Time since NRA", ylab="Remission rate ratio" )
> abline( h=1 )
```

36. If we want to predict the probability of being in each of the three states using these estimated rates, we can either do analytical calculations of the probabilities from the estimated rates, or we can *simulate* the life course through a model using the estimated rates. That will give a simulated cohort (in the form of a Lexis object), and we can then just count the number of persons in each state at each of a set of time points.

    This is accomplished using the function `simLexis`. The input to this is the initial status of the persons whose life-course we shall simulate, and the transition rates in suitable form:

    - The persons we simulate: We set up the persons as men aged 50 at NRA. The input is in the form of a `Lexis` object (where `lex.dur` and `lex.Xst` will be ignored). Note that in order to carry over the `time.scales` and the `time.since` attributes, we construct the input object using `subset` to select columns, and `NULL` to select rows (see the example in the help file for `simLexis`):

      ```
      > inL <- subset( sLc, select=1:11 )[NULL,]
      > str( inL )
      > timeScales(inL)
      > inL[1,"lex.id"] <- 1
      > inL[1,"per"] <- 2000
      > inL[1,"age"] <- 50
      > inL[1,"tfi"] <- 0
      > inL[1,"lex.Cst"] <- "NRA"
      > inL[1,"lex.Xst"] <- NA
      > inL[1,"lex.dur"] <- NA
      > inL[1,"sex"] <- 1
      > inL[1,"doe"] <- 2000
      ```

```
> inL[1,"dob"] <- 1950
> inL
```

- The other input for the simulation is the transitions, which is a list with an element for each transient state (that is "NRA" and "Rem"), each of which is again a list with names equal to the states that can be reached from the transient state. The content of the list will be `glm` objects, in this case the models we just fitted, describing the transition rates:

```
> Tr <- list( "NRA" = list( "Rem"  = mr,
+                           "ESRD" = ma ),
+             "Rem" = list( "ESRD" = ma ) )
```

With this as input we can now generate a cohort, using `N=10` to simulate life course of 10 persons (with identical starting values):

```
> options( width=100 )
> ( iL <- simLexis( Tr, inL, N=10 ) )
> str( iL )
```

37. Now generate the life course of 10,000 persons, and look at the summary:

```
> system.time(
+ sM <- simLexis( Tr, inL, N=10000 ) )
> summary( sM )
```

Why are there so many ESRD-events in the resulting data set?

38. Now we want to count how many persons are present in each state at each time for the first 10 years after entry (at age 50). This can be done by using `nState`:

```
> nSt <- nState( sM, at=seq(0,10,0.1), from=50, time.scale="age" )
> head( nSt )
```

39. Once we have the counts of persons in each state at the designated time points, we compute the cumulative fraction over the states, arranged in order given by `perm`:

```
> pp <- pState( nSt, perm=1:3 )
> head( pp )
```

40. Try to plot the cumulative probabilities using the `plot` method for `pState` objects:

```
> plot( pp )
```

41. A quantity of particular interest would be how many patients actually get a remission. This is not deductible from the plot just shown, because those who get ESRD after remission are counted in that box.

The simplest way to doctor that is to modify the simulated object (`sM` in the above notation), so that those exiting to "ESRD" from "Rem" are counted in a separate state. Note that we must also change the formal set of levels of `lex.Cst`:

```
> xM <- transform( sM, lex.Xst = factor( ifelse( lex.Xst=="ESRD" & lex.Cst=="Rem",
+                                                "ESRD(Rem)",
+                                                as.character(lex.Xst) ),
+                                        levels=c("NRA","Rem","ESRD(Rem)","ESRD") ),
+                      lex.Cst = factor( as.character(lex.Cst),
+                                        levels=c("NRA","Rem","ESRD(Rem)","ESRD") ) )
> summary( sM )
> summary( xM )
> boxes( xM, boxpos=TRUE )
```

42. Having done this, try to compute the number of persons in each of the 4 states, and the cumulative proportions to be plotted:

```
> xSt <- nState( xM, at=seq(0,10,0.1), from=50, time.scale="age" )
> xp <- pState( xSt, perm=1:4 )
> head( xp )
> plot( xp, col=c("red","limegreen","forestgreen","red") )
> lines( as.numeric(rownames(xp)), xp[,"Rem"], lwd=2 )
```

43. The proper solution would however be to realize up front that this would be of interest, and thus define the state following "Rem" as "ESRD(Rem)", when using `cutLexis`. This is done by adding the argument `split.states`:

```
> Lc <- cutLexis( Lr, cut=Lr$dor, timescale="per",
+                 new.state="Rem", precursor.states="NRA",
+                 split.states=TRUE )
> summary( Lc )
```

Now you can define a new version of the input dataset (with 1 person that you simulate 10,000 times, say) and the transition object.

Chapter 2

Solutions

There is a chapter for each of the exercises that has been used at the course. This is either a printout of the R-program that performs the analyses, as well as the graphs produced by the programs, or output from an R-weave solution.

The code and the output from these programs are also available from the course homepage in http://BendixCarstensen/SPE/R.

2.1 Practice with basic R

```
 R 3.1.0
 ------------------------------------------
 Program:  basic.R
  Folder:  C:\Bendix\undervis\SPE\Repos\pracs
 Started:  onsdag 28. maj 2014, 18:39:39
 ------------------------------------------
> ### R code from vignette source basic.rnw
>
> ###################################################
> ### code chunk number 1: basic.rnw:86-87
> ###################################################
> pchisq(3.84,1)
[1] 0.9499565
>
>
> ###################################################
> ### code chunk number 2: basic.rnw:91-92
> ###################################################
> pchisq(3.84,1,lower.tail=FALSE)
[1] 0.05004352
>
>
> ###################################################
> ### code chunk number 3: basic.rnw:124-125
> ###################################################
> v <- c(4, 6, 1, 2.2)
>
>
> ###################################################
> ### code chunk number 4: basic.rnw:134-135
> ###################################################
> m <- list(4, 6, "name of company")
>
>
> ###################################################
> ### code chunk number 5: basic.rnw:142-145
> ###################################################
> v
[1] 4.0 6.0 1.0 2.2
> 3+v
[1] 7.0 9.0 4.0 5.2
> 3*v
[1] 12.0 18.0  3.0  6.6
>
>
> ###################################################
> ### code chunk number 6: basic.rnw:159-160
> ###################################################
> seq(15, 85, by=5)
 [1] 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85
>
>
> ###################################################
> ### code chunk number 7: basic.rnw:163-164
> ###################################################
> c(5,seq(20, 85, by=5))
 [1]  5 20 25 30 35 40 45 50 55 60 65 70 75 80 85
>
>
> ###################################################
> ### code chunk number 8: Create w
> ###################################################
> w<-c(1,-1,2,-2)
>
>
> ###################################################
> ### code chunk number 9: Display w
> ###################################################
> w
[1]  1 -1  2 -2
>
>
> ###################################################
> ### code chunk number 10: Structure of w
> ###################################################
> str(w)
 num [1:4] 1 -1 2 -2
>
>
> ###################################################
> ### code chunk number 11: Create and display w+1
> ###################################################
> w+1
[1]  2  0  3 -1
```

```
>
>
> ###################################################
> ### code chunk number 12: Create v
> ###################################################
> v<-c(0,1,seq(5,75,5))
> v
 [1]  0  1  5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
>
>
> ###################################################
> ### code chunk number 13: Find length of v
> ###################################################
> length(v)
[1] 17
>
>
> ###################################################
> ### code chunk number 14: basic.rnw:204-213
> ###################################################
>  x <- c(2,7,0,9,10,23,11,4,7,8,6,0)
>  x[4]
[1] 9
>  x[3:5]
[1]  0  9 10
>  x[c(1,5,8)]
[1]  2 10  4
>  x[x>10]
[1] 23 11
>  x[(1:6)*2]
[1]  7  9 23  4  8  0
>  x[x==0]<-1
>  x
 [1]  2  7  1  9 10 23 11  4  7  8  6  1
>  ifelse(round(x/2)==x/2,"even","odd")
 [1] "even" "odd"  "odd"  "odd"  "even" "odd"  "odd"  "even" "odd"  "even"
[11] "even" "odd"
>
>
> ###################################################
> ### code chunk number 15: basic.rnw:229-232
> ###################################################
> mydata <- data.frame(name=c("Joe","Ann","Jack","Tom"),
+                      age=c(34,50,27,42),sex=c(1,2,1,1),
+                      height=c(185,170,175,182))
>
>
> ###################################################
> ### code chunk number 16: basic.rnw:236-244
> ###################################################
>  mydata
  name age sex height
1  Joe  34   1    185
2  Ann  50   2    170
3 Jack  27   1    175
4  Tom  42   1    182
>  mydata[[2]]
[1] 34 50 27 42
>  names(mydata)
[1] "name"   "age"    "sex"    "height"
>  mydata[,"age"]
[1] 34 50 27 42
>  mydata$age
[1] 34 50 27 42
>  mydata[2,3]
[1] 2
>  mydata[,2]
[1] 34 50 27 42
>  mydata[1,]
  name age sex height
1  Joe  34   1    185
>
>
> ###################################################
> ### code chunk number 17: basic.rnw:248-250
> ###################################################
>  rownames(mydata)<-mydata$name
>  mydata["Tom",]
    name age sex height
Tom  Tom  42   1    182
>
>
> ###################################################
> ### code chunk number 18: basic.rnw:255-258
> ###################################################
> weights<-data.frame(weight=c(67,81,56,90,72,79,69))
> rownames(weights)=c("Ann","Peter","Sue","Jack","Tom","Joe","Jane")
```

```
> weights[substr(rownames(weights),1,1)=="J",]
[1] 90 79 69
>
>
> ##################################################
> ### code chunk number 19: basic.rnw:263-264
> ##################################################
> mydata$weight<- weights[rownames(mydata),"weight"]
>
>
> ##################################################
> ### code chunk number 20: basic.rnw:275-276
> ##################################################
> subset(mydata, height < 180)
     name age sex height weight
Ann   Ann  50   2    170     67
Jack Jack  27   1    175     90
>
>
> ##################################################
> ### code chunk number 21: basic.rnw:279-280
> ##################################################
> mydata[mydata$height < 180, ]
     name age sex height weight
Ann   Ann  50   2    170     67
Jack Jack  27   1    175     90
>
>
> ##################################################
> ### code chunk number 22: Looking at births data
> ##################################################
> library(Epi)

Attaching package: 'Epi'

The following object is masked from 'package:base':

    merge.data.frame

> data(births)
> objects()
[1] "births"  "m"       "mydata"  "v"       "w"       "weights" "x"
>
>
> ##################################################
> ### code chunk number 23: more about births
> ##################################################
> names(births)
[1] "id"      "bweight" "lowbw"   "gestwks" "preterm" "matage"  "hyp"
[8] "sex"
> head(births)
  id bweight lowbw gestwks preterm matage hyp sex
1  1    2974     0   38.52       0     34   0   2
2  2    3270     0      NA      NA     30   0   1
3  3    2620     0   38.15       0     35   0   2
4  4    3751     0   39.80       0     31   0   1
5  5    3200     0   38.89       0     33   1   1
6  6    3673     0   40.97       0     33   0   2
>
>
> ##################################################
> ### code chunk number 24: basic.rnw:313-314
> ##################################################
> data(diet)
>
>
> ##################################################
> ### code chunk number 25: basic.rnw:323-324
> ##################################################
> rm(diet)
>
>
> ##################################################
> ### code chunk number 26: basic.rnw:335-336
> ##################################################
> births[1,"bweight"]
[1] 2974
>
>
> ##################################################
> ### code chunk number 27: basic.rnw:340-341
> ##################################################
> births[1,2]
[1] 2974
>
>
> ##################################################
```

```
> ### code chunk number 28: basic.rnw:344-345
> ###################################################
> births[2,"bweight"]
[1] 3270
>
>
> ###################################################
> ### code chunk number 29: basic.rnw:349-350
> ###################################################
> births[1:10, "bweight"]
 [1] 2974 3270 2620 3751 3200 3673 3628 3773 3960 3405
>
>
> ###################################################
> ### code chunk number 30: basic.rnw:353-354
> ###################################################
> births[, "bweight"]
  [1] 2974 3270 2620 3751 3200 3673 3628 3773 3960 3405 4020 2724 3001 3039 3662
 [16] 3035 3351 3804 3573 3283 2894 1203 3306 3753 2844 3585 3798 3164 3739 1780
 [31] 4022 3942 2887 2391 3911 3509 3566 3652 3279 3007 3053 3503 3120 3743 3592
 [46] 3184 3234 2581 3305 3678 3542 2148 3774 3079 3465 2887 4501 3375 3886 2849
 [61] 2002 2213 2797 3303 3296 2921 2929 3385 1946 3354 3189 3392 2696 2597 3409
 [76] 3190 3571 4512 2215 3315 3341 3451 3338 2507 3316 4141 4071 2893 3064 3603
 [91] 3554 4027 2418 3092 2671 3430 3244 3259 3942 3576 3784 2796 3226 3138 3715
[106] 3773 2623 2830 3332 2911 2894 3593 2950 3605 3198 3183 2487 2092 3932 3995
[121] 3592 4092 1663 1546 3149 3062 3467 2610 3261 2926 3727 3545 3423 2959 3879
[136] 2935 3552 2764 4069 3267 3178  628 3288 3233 3985 3676 2579 3461 2990 2995
[151] 2922 3449 3292 3621 3636 3471 3360 2338 3567 3007 2740 2428 3027 3261 1999
[166] 3117 2751 3824 2762 4057 3501 3503 3001 1791 3582 3575 3117 3260 2704 2545
[181] 1019 2609 2127 3126 3606 2257 3051 3146 3696 1402 2539 3550 2252 2842 2679
[196] 3526 3184 3180 3561 2698 4131 1880 3807 3005 3448 4340 3518 3296 2505 3188
[211] 3109 3134 2992  708 3571 3463 1741 4122 3141 2328 2404 3079 2837 3216 2951
[226]  981 3882 3385 2545 3096 3138 3341 2768 3367 3768 3486 2913 4436 3943 3009
[241] 3041 3099 3647 2698 3078 3976 2729 3254 3163 3919 2718 3269  693 2879 2590
[256] 3311 2831 2991 2699 3695 3250 1570 3554 4205 3694 3505 3023 3723 2482 2622
[271] 3558 2090 3446 2938  864 1764 3096 3003 3197 3664 1618 3059 2360 2961 3323
[286] 2982 2974 4423 3105 2446 3590 3450 3291 2866 2802 2953 3379 3386 3770 2188
[301] 3813 3576 3290 3502 3591 3734 2989 3546 3783 3754 2297 1500 1595 3376 3349
[316] 3727 2719 3365 4553 3007 1801 3451 3249 3092 2646 3739 3767  924 3428 3004
[331] 3625 2804 2274 3075 2699 2605 1325 2736 3192 3184 2855 3392 3040 3249 3156
[346] 2606 2859 2558 2905 3331 3730 3683 3398 3435 4319 4179 1431 3154 2914 3646
[361] 3980 3399 3621 2659 2890 2425 2694 3804 2784 1541 3282 3695 2769 3286 2744
[376] 3193 2666 3102 3333 2924 1824 3045 2943 3425 2863 3024 4287 3207 3090 3257
[391] 3370 3200 4226 3160 2732 2977 2968 3562 3316 3482 2969 4035 3704 3082 3404
[406] 3041 3519 3432 3733 2622 2497 2797 3337 2407 3383 3134 3559 3516 3497 3350
[421] 2969 3851 2697 2801 2362 2978 3082 3545 1874 3140 3457 3093 2353 3184 3222
[436] 2944 3179 3463 1324 3064 3581 2399 3830 3376 2098 3601 3386 3122 3551 2944
[451] 4224 2719 4133 2885 3042 4197 3419 3294 4304 3306 3102 4041 4182 2949 3757
[466] 3486 3770 3087 4300 3122 3337 3565 3611 4516 3718 1938 2878 2639 2417 2963
[481] 2843 2595 3455 3061 3095 2552 3409 3152 3172 3237 3948 3542 3322 3349 2968
[496] 2852 3187 3054 3178 2918
>
>
> ###################################################
> ### code chunk number 31: basic.rnw:357-358
> ###################################################
> births[1 , -2]
  id lowbw gestwks preterm matage hyp sex
1  1     0   38.52       0     34   0   2
>
>
> ###################################################
> ### code chunk number 32: Display row 7 of gestwks
> ###################################################
> births[7,"gestwks"]
[1] 42.14
>
>
> ###################################################
> ### code chunk number 33: Display all the data in row 7
> ###################################################
> births[7,]
  id bweight lowbw gestwks preterm matage hyp sex
7  7    3628     0   42.14       0     29   0   2
>
>
> ###################################################
> ### code chunk number 34: Display first 10 rows of gestwks
> ###################################################
> births[1:10,"gestwks"]
 [1] 38.52    NA 38.15 39.80 38.89 40.97 42.14 40.21 42.03 39.33
>
>
> ###################################################
> ### code chunk number 35: basic.rnw:381-382
> ###################################################
> summary(births)
```

```
       id            bweight          lowbw          gestwks          preterm
 Min.   :  1.0   Min.   : 628   Min.   :0.00   Min.   :24.69   Min.   :0.0000
 1st Qu.:125.8   1st Qu.:2862   1st Qu.:0.00   1st Qu.:37.94   1st Qu.:0.0000
 Median :250.5   Median :3188   Median :0.00   Median :39.12   Median :0.0000
 Mean   :250.5   Mean   :3137   Mean   :0.12   Mean   :38.72   Mean   :0.1286
 3rd Qu.:375.2   3rd Qu.:3551   3rd Qu.:0.00   3rd Qu.:40.09   3rd Qu.:0.0000
 Max.   :500.0   Max.   :4553   Max.   :1.00   Max.   :43.16   Max.   :1.0000
                                               NA's   :10      NA's   :10
     matage           hyp             sex
 Min.   :23.00   Min.   :0.000   Min.   :1.000
 1st Qu.:31.00   1st Qu.:0.000   1st Qu.:1.000
 Median :34.00   Median :0.000   Median :1.000
 Mean   :34.03   Mean   :0.144   Mean   :1.472
 3rd Qu.:37.00   3rd Qu.:0.000   3rd Qu.:2.000
 Max.   :43.00   Max.   :1.000   Max.   :2.000

>
>
> ###################################################
> ### code chunk number 36: basic.rnw:385-386
> ###################################################
> names(births)
[1] "id"      "bweight" "lowbw"   "gestwks" "preterm" "matage"  "hyp"
[8] "sex"
>
>
> ###################################################
> ### code chunk number 37: basic.rnw:395-396
> ###################################################
> summary(births$hyp)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   0.000   0.144   0.000   1.000
>
>
> ###################################################
> ### code chunk number 38: basic.rnw:400-401
> ###################################################
> with(births, summary(hyp))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   0.000   0.144   0.000   1.000
>
>
> ###################################################
> ### code chunk number 39: Convert hyp to be a factor
> ###################################################
> births$hyp <- factor(births$hyp)
> str(births)
'data.frame': 500 obs. of  8 variables:
 $ id     : num  1 2 3 4 5 6 7 8 9 10 ...
 $ bweight: num  2974 3270 2620 3751 3200 ...
 $ lowbw  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ gestwks: num  38.5 NA 38.2 39.8 38.9 ...
 $ preterm: num  0 NA 0 0 0 0 0 0 0 0 ...
 $ matage : num  34 30 35 31 33 33 29 37 36 39 ...
 $ hyp    : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
 $ sex    : num  2 1 2 1 1 2 2 1 2 1 ...
>
>
> ###################################################
> ### code chunk number 40: basic.rnw:424-426
> ###################################################
> births <- transform(births, hyp=factor(hyp))
> str(births)
'data.frame': 500 obs. of  8 variables:
 $ id     : num  1 2 3 4 5 6 7 8 9 10 ...
 $ bweight: num  2974 3270 2620 3751 3200 ...
 $ lowbw  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ gestwks: num  38.5 NA 38.2 39.8 38.9 ...
 $ preterm: num  0 NA 0 0 0 0 0 0 0 0 ...
 $ matage : num  34 30 35 31 33 33 29 37 36 39 ...
 $ hyp    : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
 $ sex    : num  2 1 2 1 1 2 2 1 2 1 ...
>
>
> ###################################################
> ### code chunk number 41: Label the levels
> ###################################################
> births$hyp <- factor(births$hyp,labels=c("normal","hyper"))
> str(births)
'data.frame': 500 obs. of  8 variables:
 $ id     : num  1 2 3 4 5 6 7 8 9 10 ...
 $ bweight: num  2974 3270 2620 3751 3200 ...
 $ lowbw  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ gestwks: num  38.5 NA 38.2 39.8 38.9 ...
 $ preterm: num  0 NA 0 0 0 0 0 0 0 0 ...
 $ matage : num  34 30 35 31 33 33 29 37 36 39 ...
 $ hyp    : Factor w/ 2 levels "normal","hyper": 1 1 1 1 2 1 1 1 1 1 ...
```

```
 $ sex    : num  2 1 2 1 1 2 2 1 2 1 ...
>
>
> ####################################################
> ### code chunk number 42: Label the levels
> ####################################################
> births <- transform(births,hyp=factor(hyp,labels=c("normal","hyper")))
> str(births)
'data.frame': 500 obs. of  8 variables:
 $ id     : num  1 2 3 4 5 6 7 8 9 10 ...
 $ bweight: num  2974 3270 2620 3751 3200 ...
 $ lowbw  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ gestwks: num  38.5 NA 38.2 39.8 38.9 ...
 $ preterm: num  0 NA 0 0 0 0 0 0 0 0 ...
 $ matage : num  34 30 35 31 33 33 29 37 36 39 ...
 $ hyp    : Factor w/ 2 levels "normal","hyper": 1 1 1 1 2 1 1 1 1 1 ...
 $ sex    : num  2 1 2 1 1 2 2 1 2 1 ...
>
>
> ####################################################
> ### code chunk number 43: Convert sex to a factor
> ####################################################
> births$sex<-factor(births$sex)
>
>
> ####################################################
> ### code chunk number 44: :abel levels of sex
> ####################################################
> births$sex<-factor(births$sex,labels=c("M","F"))
>
>
> ####################################################
> ### code chunk number 45: Distribution of hyp
> ####################################################
> with(births, table(hyp))
hyp
normal  hyper
   428     72
>
>
> ####################################################
> ### code chunk number 46: Distribution of hyp
> ####################################################
> table(births$hyp)

normal  hyper
   428     72
>
>
> ####################################################
> ### code chunk number 47: Distribution of sex
> ####################################################
> table(births$sex)

  M   F
264 236
> with(births,table(sex))
sex
  M   F
264 236
>
>
> ####################################################
> ### code chunk number 48: Joint distribution of sex and hyp
> ####################################################
> with(births,table(sex,hyp))
   hyp
sex normal hyper
  M    221    43
  F    207    29
>
>
> ####################################################
> ### code chunk number 49: Cut matage into four groups
> ####################################################
> births <- transform(births,agegrp=cut(matage, breaks=c(20,30,35,40,45),right=FALSE))
> with(births, table(agegrp))
agegrp
[20,30) [30,35) [35,40) [40,45)
     70     200     194      36
>
>
> ####################################################
> ### code chunk number 50: Cut matage into 5 groups
> ####################################################
> births <- transform(births,agegrp=cut(matage,breaks=5,right=FALSE))
> with(births, table(agegrp))
```

```
agegrp
[23,27) [27,31) [31,35) [35,39) [39,43)
     16      83     171     166      64
>
>
> ###################################################
> ### code chunk number 51: Summarize gestwks
> ###################################################
> with(births, summary(gestwks))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  24.69   37.94   39.12   38.72   40.09   43.16      10
>
>
> ###################################################
> ### code chunk number 52: Cut gestwks
> ###################################################
> births <- transform(births,gest4=cut(gestwks, breaks=c(20,35,37,39,45)))
>
>
> ###################################################
> ### code chunk number 53: Cut gestwks into 5 groups
> ###################################################
> births <- transform(births, gest5=cut(gestwks, breaks=5))
> with(births,table(gest5))
gest5
(24.7,28.4] (28.4,32.1] (32.1,35.8] (35.8,39.5] (39.5,43.2]
          5           7          27         236         215
>
>
> ###################################################
> ### code chunk number 54: Create a new variable in the global environment
> ###################################################
> logbw <- log(births$bweight)
>
>
> ###################################################
> ### code chunk number 55: Create a new variable in the data frame
> ###################################################
> births$logbw <- log(births$bweight)
>
>
> ###################################################
> ### code chunk number 56: Create a logical variable
> ###################################################
> births$vlow <- births$bweight<2000
> str(births)
'data.frame': 500 obs. of  13 variables:
 $ id     : num  1 2 3 4 5 6 7 8 9 10 ...
 $ bweight: num  2974 3270 2620 3751 3200 ...
 $ lowbw  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ gestwks: num  38.5 NA 38.2 39.8 38.9 ...
 $ preterm: num  0 NA 0 0 0 0 0 0 0 0 ...
 $ matage : num  34 30 35 31 33 33 29 37 36 39 ...
 $ hyp    : Factor w/ 2 levels "normal","hyper": 1 1 1 1 2 1 1 1 1 1 ...
 $ sex    : Factor w/ 2 levels "M","F": 2 1 2 1 1 2 2 1 2 1 ...
 $ agegrp : Factor w/ 5 levels "[23,27)","[27,31)",..: 3 2 4 3 3 3 2 4 4 5 ...
 $ gest4  : Factor w/ 4 levels "(20,35]","(35,37]",..: 3 NA 3 4 3 4 4 4 4 4 ...
 $ gest5  : Factor w/ 5 levels "(24.7,28.4]",..: 4 NA 4 5 4 5 5 5 5 4 ...
 $ logbw  : num  8 8.09 7.87 8.23 8.07 ...
 $ vlow   : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
>
>
> ###################################################
> ### code chunk number 57: Ceate new data frame as a subset of births
> ###################################################
> births.low<-subset(births, bweight<2000)
> summary(births.low)
       id           bweight         lowbw      gestwks         preterm
 Min.   : 22.0   Min.   : 628   Min.   :1   Min.   :24.69   Min.   :0.0000
 1st Qu.:175.8   1st Qu.:1233   1st Qu.:1   1st Qu.:30.71   1st Qu.:1.0000
 Median :257.5   Median :1558   Median :1   Median :32.64   Median :1.0000
 Mean   :249.6   Mean   :1462   Mean   :1   Mean   :32.75   Mean   :0.8621
 3rd Qu.:326.2   3rd Qu.:1788   3rd Qu.:1   3rd Qu.:35.14   3rd Qu.:1.0000
 Max.   :476.0   Max.   :1999   Max.   :1   Max.   :40.45   Max.   :1.0000
                                            NA's   :1       NA's   :1
     matage          hyp     sex        agegrp       gest4          gest5
 Min.   :25.00   normal:17   M:12   [23,27): 1   (20,35]:21   (24.7,28.4]: 5
 1st Qu.:31.00   hyper :13   F:18   [27,31): 6   (35,37]: 4   (28.4,32.1]: 6
 Median :34.00                      [31,35):10   (37,39]: 3   (32.1,35.8]:13
 Mean   :33.57                      [35,39): 9   (39,45]: 1   (35.8,39.5]: 4
 3rd Qu.:36.75                      [39,43): 4   NA's   : 1   (39.5,43.2]: 1
 Max.   :41.00                                                NA's       : 1

     logbw         vlow
 Min.   :6.443   Mode:logical
 1st Qu.:7.117   TRUE:30
 Median :7.351   NA's:0
```

```
 Mean   :7.240
 3rd Qu.:7.489
 Max.   :7.600

>
>
> ###################################################
> ### code chunk number 58: Create logical variable early
> ###################################################
> early<-births$gestwks<30
> table(early)
early
FALSE  TRUE
  485     5
>
>
> ###################################################
> ### code chunk number 59: Display id numbers for a subset of births
> ###################################################
> tmp <- subset(births, early)
> tmp$id
[1] 142 181 214 226 275
>
>
> ###################################################
> ### code chunk number 60: Saving births as births2
> ###################################################
> save(births, file="births2")
>
>
> ###################################################
> ### code chunk number 61: Loading births2
> ###################################################
> load("births2")
>
>
> ###################################################
> ### code chunk number 62: basic.rnw:622-623
> ###################################################
> search()
 [1] ".GlobalEnv"        "package:Epi"       "package:utils"
 [4] "package:datasets"  "package:foreign"   "package:graphics"
 [7] "package:grDevices" "package:stats"     "package:methods"
[10] "Autoloads"         "package:base"
>
>
> ###################################################
> ### code chunk number 63: basic.rnw:629-630
> ###################################################
> objects()
 [1] "births"     "births.low" "early"      "logbw"      "m"
 [6] "mydata"     "tmp"        "v"          "w"          "weights"
[11] "x"
>
>
> ###################################################
> ### code chunk number 64: basic.rnw:635-636
> ###################################################
> objects(2)
  [1] "apc.fit"                "apc.frame"
  [3] "apc.lines"              "apc.plot"
  [5] "Aplot"                  "as.Date.cal.yr"
  [7] "boxarr"                 "boxes"
  [9] "boxes.Lexis"            "boxes.matrix"
 [11] "boxes.MS"               "breaks"
 [13] "cal.yr"                 "ccwc"
 [15] "ci.cum"                 "ci.exp"
 [17] "ci.lin"                 "ci.mat"
 [19] "ci.pd"                  "clear"
 [21] "clogistic"              "contr.2nd"
 [23] "contr.cum"              "contr.diff"
 [25] "contr.orth"             "countLexis"
 [27] "Cplot"                  "cutLexis"
 [29] "dbox"                   "detrend"
 [31] "dur"                    "effx"
 [33] "effx.match"             "entry"
 [35] "etm"                    "etm.Lexis"
 [37] "exit"                   "factorize"
 [39] "factorize.Lexis"        "fillarr"
 [41] "fit.add"                "fit.mult"
 [43] "float"                  "ftrend"
 [45] "gen.exp"                "Icens"
 [47] "Lexis"                  "Lexis.diagram"
 [49] "Lexis.lines"            "Life.lines"
 [51] "lines.apc"              "lines.Lexis"
 [53] "lines.pState"           "linesEst"
 [55] "lls"                    "merge.data.frame"
```

```
 [57] "merge.Lexis"             "mh"
 [59] "msdata"                  "msdata.Lexis"
 [61] "N2Y"                     "NArray"
 [63] "ncut"                    "nice"
 [65] "Ns"                      "nState"
 [67] "pc.lines"                "pc.matlines"
 [69] "pc.matpoints"            "pc.points"
 [71] "pctab"                   "plot.apc"
 [73] "plot.Lexis"              "plot.pState"
 [75] "plotEst"                 "plotevent"
 [77] "points.Lexis"            "pointsEst"
 [79] "Pplot"                   "print.floated"
 [81] "print.Icens"             "print.summary.Lexis"
 [83] "projection.ip"           "pState"
 [85] "PY.ann"                  "PY.ann.Lexis"
 [87] "rateplot"                "Relevel"
 [89] "Relevel.Lexis"           "ROC"
 [91] "simLexis"                "splitLexis"
 [93] "stack.Lexis"             "stat.table"
 [95] "status"                  "subset.Lexis"
 [97] "subset.stacked.Lexis"    "summary.Lexis"
 [99] "tbox"                    "timeBand"
[101] "timeScales"              "tmat"
[103] "tmat.Lexis"              "transform.Lexis"
[105] "transform.stacked.Lexis" "twoby2"
[107] "Wald"                    "ZArray"
>
>
> ###################################################
> ### code chunk number 65: basic.rnw:659-660
> ###################################################
> attach(births)
The following object is masked _by_ .GlobalEnv:

    logbw
>
>
> ###################################################
> ### code chunk number 66: basic.rnw:664-665
> ###################################################
> objects(2)
 [1] "agegrp"  "bweight" "gest4"   "gest5"   "gestwks" "hyp"     "id"
 [8] "logbw"   "lowbw"   "matage"  "preterm" "sex"     "vlow"
>
>
> ###################################################
> ### code chunk number 67: basic.rnw:672-673
> ###################################################
> hyp
  [1] normal normal normal normal hyper  normal normal normal normal normal
 [11] normal normal normal normal normal normal normal normal normal normal
 [21] normal normal normal normal normal normal normal hyper  normal normal
 [31] normal normal normal normal normal normal normal normal normal normal
 [41] normal normal hyper  hyper  normal normal hyper  normal normal normal
 [51] hyper  normal normal hyper  normal normal normal normal normal hyper
 [61] hyper  hyper  normal normal normal normal normal normal hyper  normal
 [71] normal normal normal normal normal normal normal normal normal normal
 [81] normal normal normal hyper  normal normal normal normal normal normal
 [91] normal normal hyper  normal normal normal normal normal normal normal
[101] normal normal normal hyper  normal hyper  normal normal normal normal
[111] normal normal normal normal normal normal normal hyper  normal normal
[121] normal normal hyper  normal hyper  normal normal normal normal normal
[131] normal normal hyper  hyper  normal normal normal normal normal normal
[141] hyper  normal normal normal hyper  normal normal normal normal normal
[151] normal normal normal normal normal normal normal normal normal normal
[161] normal hyper  normal hyper  hyper  normal normal normal normal normal
[171] normal normal normal normal normal normal normal normal normal normal
[181] hyper  normal normal hyper  normal normal normal normal normal normal
[191] normal normal normal hyper  normal normal normal normal normal normal
[201] normal normal normal normal normal normal normal normal normal normal
[211] normal normal hyper  hyper  normal normal normal normal normal normal
[221] hyper  normal normal hyper  normal hyper  normal normal normal hyper
[231] normal normal normal hyper  normal normal normal hyper  normal normal
[241] normal normal normal normal normal normal normal normal normal normal
[251] hyper  normal hyper  normal normal normal normal normal normal normal
[261] normal hyper  normal normal normal normal normal normal normal normal
[271] normal hyper  normal normal normal normal normal normal normal normal
[281] normal normal normal normal normal normal normal normal normal normal
[291] normal normal hyper  normal normal normal normal hyper  normal normal
[301] hyper  hyper  normal normal normal normal normal normal normal normal
[311] normal normal hyper  normal hyper  normal normal normal normal normal
[321] normal normal normal hyper  hyper  normal normal normal normal normal
[331] normal normal normal normal normal normal normal hyper  normal normal
[341] normal hyper  hyper  normal normal hyper  normal hyper  normal normal
[351] normal normal normal normal normal normal normal normal normal normal
[361] normal normal normal normal normal normal normal normal normal hyper
[371] normal normal hyper  normal normal normal normal normal normal hyper
```

```
[381] hyper  normal normal normal normal normal normal normal normal normal
[391] normal normal normal hyper  hyper  normal normal hyper  normal normal
[401] normal normal normal normal normal normal normal normal normal hyper
[411] normal normal normal normal normal normal normal normal normal normal
[421] hyper  normal hyper  normal normal normal hyper  normal hyper  normal
[431] normal hyper  normal normal normal normal normal hyper  normal
[441] normal normal normal normal normal normal hyper  normal normal normal
[451] normal normal normal hyper  normal normal normal normal normal normal
[461] normal normal hyper  normal normal normal normal normal normal normal
[471] normal normal normal normal normal normal normal normal normal normal
[481] normal normal normal normal hyper  normal normal normal normal normal
[491] normal normal normal normal normal normal hyper  normal normal normal
Levels: normal hyper
>
>
> ###################################################
> ### code chunk number 68: basic.rnw:682-683
> ###################################################
> subgrp <- bweight[hyp==1]
>
>
> ###################################################
> ### code chunk number 69: basic.rnw:687-689
> ###################################################
> objects(1)
 [1] "births"     "births.low" "early"      "logbw"      "m"
 [6] "mydata"     "subgrp"     "tmp"        "v"          "w"
[11] "weights"    "x"
> objects(2)
 [1] "agegrp"  "bweight" "gest4"   "gest5"   "gestwks" "hyp"     "id"
 [8] "logbw"   "lowbw"   "matage"  "preterm" "sex"     "vlow"
>
>
> ###################################################
> ### code chunk number 70: basic.rnw:699-701
> ###################################################
> attach(births)
The following object is masked _by_ .GlobalEnv:

    logbw
The following objects are masked from births (position 3):

    agegrp, bweight, gest4, gest5, gestwks, hyp, id, logbw, lowbw,
    matage, preterm, sex, vlow
> search()
 [1] ".GlobalEnv"        "births"            "births"
 [4] "package:Epi"       "package:utils"     "package:datasets"
 [7] "package:foreign"   "package:graphics"  "package:grDevices"
[10] "package:stats"     "package:methods"   "Autoloads"
[13] "package:base"
>
>
> ###################################################
> ### code chunk number 71: basic.rnw:706-707
> ###################################################
> detach(births)
>
>
>

 --------------------------------------------
 Program: basic.R
  Folder: C:\Bendix\undervis\SPE\Repos\pracs
   Ended: onsdag 28. maj 2014, 18:39:40
 Elapsed: 00:00:00
 --------------------------------------------
> proc.time()
   user  system elapsed
   0.99    0.20    1.37
```

2.2 Reading data into R

2.2.1 Introduction

It is said that Mrs Beeton, the 19th century cook and writer, began her recipe for rabbit stew with the instruction "First catch your rabbit". Sadly, the story is untrue, but it does contain an important moral. R is a language and environment for data analysis. If you want to do something interesting with it, you need data.

For teaching purposes, data sets are often embedded in R packages. The base R distribution contains a whole package dedicated to data which includes around 100 data sets. This is attached towards the end of the search path, and you can see its contents with

```
> objects("package:datasets")
```

A description of all of these objects is available using the `help()` function. For example

```
> help(Titanic)
```

gives an explanation of the `Titanic` data set, along with references giving the source of the data.

The Epi package also contains some data sets. These are not available automatically when you load the Epi package, but you can make a copy in your workspace using the `data()` function. For example

```
> library(Epi)
> data(bdendo)
```

will create a data frame called `bdendo` in your workspace containing data from a case-control study of endometrial cancer. Datasets in the Epi package also have help pages: type `help(bdendo)` for further information.

To go back to the cooking analogy, these data sets are the equivalent of microwave ready meals, carefully packaged and requiring minimal work by the consumer. Your own data will never be able in this form and you must work harder to read it in to R.

This exercise introduces you to the basics of reading external data into R. It consists of reading the same data from different formats. Although this may appear repetitive, it allows you to see the many options available to you, and should allow you to recognize when things go wrong.

You will need the following files in the sub-directory `data` of your working directory: `fem.dat`, `fem-dot.dat`, `fem.csv`, `fem.dta` (Reminder: use `setwd()` to set your working directory).

2.2.2 Data sources

Sources of data can be classified into three groups:

1. Data in human readable form, which can be inspected with a text editor.
2. Data in binary format, which can only be read by a program that understands that format (SAS, SPSS, Stata, Excel, ...).
3. Online data from a database management system (DBMS)

This exercise will deal with the first two forms of data. Epidemiological data sets are rarely large enough to justify being kept in a DBMS. If you want further details on this topic, you can consult the "R Data Import/Export" manual that comes with R.

2.2.3 Data in text files

Human-readable data files are generally kept in a rectangular format, with individual records in single rows and variables in columns. Such data can be read into a data frame in R.

Before reading in the data, you should inspect the file in a text editor and ask three questions:

1. How are columns in the table separated?
2. How are missing values represented?
3. Are variable names included in the file?

The file `fem.dat` contains data on 118 female psychiatric patients. The data set contains nine variables.

ID	Patient identifier
AGE	Age in years
IQ	Intelligence Quotient (IQ) score
ANXIETY	Anxiety (1=none, 2=mild, 3=moderate,4=severe)
DEPRESS	Depression (1=none, 2=mild, 3=moderate or severe)
SLEEP	Sleeping normally (1=yes, 2=no)
SEX	Lost interest in sex (1=yes, 2=no)
LIFE	Considered suicide (1=yes, 2=no)
WEIGHT	Weight change (kg) in previous 6 months

Inspect the file `fem.dat` with a text editor to answer the questions above.

The most general function for reading in free-format data is `read.table()`. This function reads a text file and returns a data frame. It tries to guess the correct format of each variable in the data frame (integer, double precision, or text).

Read in the table with:

```
> fem <- read.table("./data/fem.dat", header=TRUE)
```

Note that you must assign the result of `read.table()` to an object. If this is not done, the data frame will be printed to the screen and then lost.

You can see the names of the variables with

```
> names(fem)
```

The structure of the data frame can be seen with

```
> str(fem)
```

You can also inspect the top few rows with

```
> head(fem)
```

Note that the IQ of subject 9 is -99, which is an illegal value: nobody can have a negative IQ. In fact -99 has been used in this file to represent a missing value. In R the special value `NA` ("Not Available") is used to represent missing values. All R functions recognize `NA` values and will handle them appropriately, although sometimes the appropriate response is to stop the calculation with an error message.

You can recode the missing values with

```
> fem$IQ[fem$IQ == -99] <- NA
```

Of course it is much better to handle special missing value codes when reading in the data. This can be done with the `na.strings` argument of the `read.table()` function. See below.

2.2.4 Things that can go wrong

Sooner or later when reading data into R, you will make a mistake. The frustrating part of reading data into R is that most mistakes are not fatal: they simply cause the function to return a data frame that is *not what you wanted.* There are three common mistakes, which you should learn to recognize.

Forgetting the headers

The first row of the file `fem.dat` contains the variable names. The `read.table()` function does not assume this by default so you have to specify this with the argument `header=TRUE`. See what happens when you forget to include this option:

```
> fem2 <- read.table("data/fem.dat")
> str(fem2)
> head(fem2)
```

and compare the resulting data frame with `fem`. What are the names of the variables in the data frame? What is the class of the variables?

> **Explanation:** Remember that `read.table()` tries to guess the mode of the variables in the text file. Without the `header=TRUE` option it reads the first row, containing the variable names, as data, and guesses that all the variables are character, not numeric. By default, all character variables are coerced to factors by `read.table`. The result is a data frame consisting entirely of factors (You can prevent the conversion of character variables to factors with the argument `as.is=TRUE`).

If the variable names are not specified in the file, then they are given default names `V1, V2, ...`. You will soon realise this mistake if you try to access a variable in the data frame by, for example

```
> fem2$IQ
```

as the variable will not exist

There is one case where omitting the `header=TRUE` option is harmless (apart from the situation where there is no header line, obviously). When the first row of the file contains **one less** value than subsequent lines, `read.table()` infers that the first row contains the variable names, and the first column of every subsequent row contains its **row name**.

Using the wrong separator

By default, `read.table` assumes that data values are separated by any amount of white space. Other possibilities can be specified using the `sep` argument. See what happens when you assume the wrong separator, in this case a tab, which is specified using the escape sequence `"\t"`

```
> fem3 <- read.table("data/fem.dat", sep="\t")
> str(fem3)
```

How many variables are there in the data set?

> **Explanation:** If you mis-specify the separator, `read.table()` reads the whole line as a single character variable. Once again, character variables are coerced to factors, so you get a data frame with a single factor variable.

Mis-specifying the representation of missing values

The file `fem-dot.dat` contains a version of the FEM dataset in which all missing values are represented with a dot. This is a common way of representing missing values, but is not recognized by default by the `read.table()` function, which assumes that missing values are represented by "NA".

Inspect the file with a text editor, and then see what happens when you read the file in incorrectly:

```
> fem4 <- read.table("data/fem-dot.dat", header=TRUE)
> str(fem4)
```

You should have enough clues by now to work out what went wrong.

You can read the data correctly using the `na.strings` argument

```
> fem4 <- read.table("data/fem-dot.dat", header=TRUE, na.strings=".")
```

2.2.5 Spreadsheet data

Spreadsheets have become a common way of exchanging data. All spreadsheet programs can save a single sheet in *comma-separated variable* (CSV) format, which can then be read into R. There are two functions in R for reading in CSV data: `read.csv()` and `read.csv2()`.

To understand why there are two functions, inspect the contents of the function `read.csv()` by typing its name

```
> read.csv
```

```
function (file, header = TRUE, sep = ",", quote = "\"", dec = ".",
    fill = TRUE, comment.char = "", ...)
read.table(file = file, header = header, sep = sep, quote = quote,
    dec = dec, fill = fill, comment.char = comment.char, ...)
<bytecode: 0x36dd1a8>
<environment: namespace:utils>
```

The first two lines show the arguments to the `read.csv()` function and their default values (`header=TRUE`, etc) The next two lines show the *body* of the function, which shows that the default arguments are simply passed verbatim onto the `read.table()` function. Hence `read.csv()` is a *wrapper* function that chooses the correct arguments for `read.table()` for you. You only need to supply the name of the CSV file and all the other details are taken care of.

Now inspect the `read.csv2` function to find the difference between this function and `read.csv`.

> **Explanation:** The CSV format is not a single standard. The file format depends on the *locale* of your computer – the settings that determine how numbers are represented. In some countries, the decimal separator is a point "." and the variable separator in a CSV file is a comma ",". In other countries, the decimal separator is a comma "," and the variable separator is a semi-colon ";". The `read.csv()` function is used for the first format and the `read.csv2()` function is used for the second format.

The file `fem.csv` contains the FEM dataset in CSV format. Inspect the file to work out which format is used, and read it into R.

On Microsoft Windows, you can copy values directly from an open Excel spreadsheet using the clipboard. Highlight the cells you want to copy in the spread sheet and select copy from the pull-down edit menu. Then type `read.table(file="clipboard")` to read the data in. Beware, however, that the clipboard on Windows operates on the WYSIWYG principle (what-you-see-is-what-you-get). If you have a value 1.23456789 in your spreadsheet, but have formatted the cell so it is displayed to two decimal places, then the value read into R will be the truncated value 1.23.

2.2.6 Binary data

The foreign package allows you to read data in binary formats used by other statistical packages. Since R is an open source project, it can only read binary formats that are themselves "open", in the sense that the standards for reading and writing data are well-documented. For example, there is a function in the foreign package for reading SAS XPORT files, a format that has been adopted as a standard by the US Food and Drug Administration (http://www.sas.com/govedu/fda/faq.html). However, there is no function in the foreign package for reading native SAS binaries (SAS7BDAT files). Other packages are available from CRAN (http://cran.r-project.org) that offer the possibility of reading SAS binary files: see the haven and sas7bdat packages.

The file `fem.dta` contains the FEM dataset in the format used by Stata. Read it into R with

```
> library(foreign)
> fem5 <- read.dta("data/fem.dta")
> head(fem5)
```

The Stata data set contains value and variable labels. Stata variables with value labels are automatically converted to factors.

There is no equivalent of variable labels in an R data frame, but the original variable labels are not lost. They are still attached to the data frame as an invisible *attribute*, which you can see with

```
> attr(fem5, "var.labels")
```

A lot of *meta-data* is attached to the data in the form of attributes. You can see the whole list of attributes with

```
> attributes(fem5)
```

or just the attribute names with

```
> names(attributes(fem5))
```

The `read.dta()` function can only read data from Stata versions 5–12. The R Core Team has not been able to keep up with changes in the Stata format. You may wish to try the haven package and the readstata13 package, both available from CRAN.

2.2.7 Summary

In this exercise we have seen how to create a data frame in R from an external text file. We have also reviewed some common mistakes that result in garbled data.

The capabilities of the foreign package for reading binary data have also been demonstrated with a sample Stata data set.

2.3 Simple simulation

Monte Carlo methods are computational procedures dealing with simulation of artificial data from given probability distributions with the purpose of learning about the behaviour of phenomena involving random variability. These methods have a wide range of applications in statistics as well as in several branches of science and technology. By solving the following exercises you will learn to use some basic tools of statistical simulation.

1. Whenever using a *random number generator* (RNG) for a simulation study, or for producing a randomization list to be used in a clinical trial, it is a good practice to set first the *seed*. It is a number that determines the initial state of the RNG, from which it starts creating the desired sequence of pseudo-random numbers. Explicit specification of the seed enables the reproducibility of the sequence. In serious applications (like in clinical trials) it is important that the seed (and the whole randomization list) is concealed from persons with certain important roles in the study (like the physicians who recruit or treat the patients in a clinical trial). – Instead of the number 5462319 below you may use your own seed of choice.

   ```
   > set.seed(5462319)
   ```

2. Generate a random sample of size 20 from a normal distribution with mean 100 and standard deviation 10. Draw a histogram of the sampled values and compute the conventional summary statistics

   ```
   >  x <- rnorm(20, 100, 10)
   >  hist(x)
   >  c(mean(x), sd(x))

   [1] 98.36953 10.22482
   ```

   Repeat the above lines and compare the results.

   ```
   [1] 97.87647 10.17289
   ```

3. Now replace the sample size 20 by 1000 and run again twice the previous command lines with this size but keeping the parameter values as before. Compare the results between the two samples here as well as with those in the previous item.

   ```
   >  x <- rnorm(1000, 100, 10)
   >  hist(x)
   >  c(mean(x), sd(x))

   [1] 99.94878  9.85793

   [1] 100.09159  10.13483
   ```

4. Generate a sample of size 1000 from a Uniform(0,1) distribution (`runif(1000)`) and look at the a) histogram of the original values, b) histogram of their natural logarithms, and c) histogram of the logit-transforms of the values.

```
> x <- runif(1000)
> hist(x)
> hist(log(x))
> hist(log(x/(1-x)))
```

5. Generate 500 observations from a Bernoulli(p) distribution, or Bin$(1, p)$ distribution, taking values 1 and 0 with probabilities p and $1 - p$, respectively, when $p = 0.4$:

```
> X <- rbinom(500, 1, 0.4)
> table(X)
```

```
X
  0   1
281 219
```

6. Now generate another 0/1 variable Y, being dependent on previously generated X, so that $P(Y = 1|X = 1) = 0.2$ and $P(Y = 1|X = 0) = 0.1$.

```
> Y <- rbinom(500,1,0.1*X+0.1)
> table(X,Y)
```

```
   Y
X     0   1
  0 261  20
  1 175  44
```

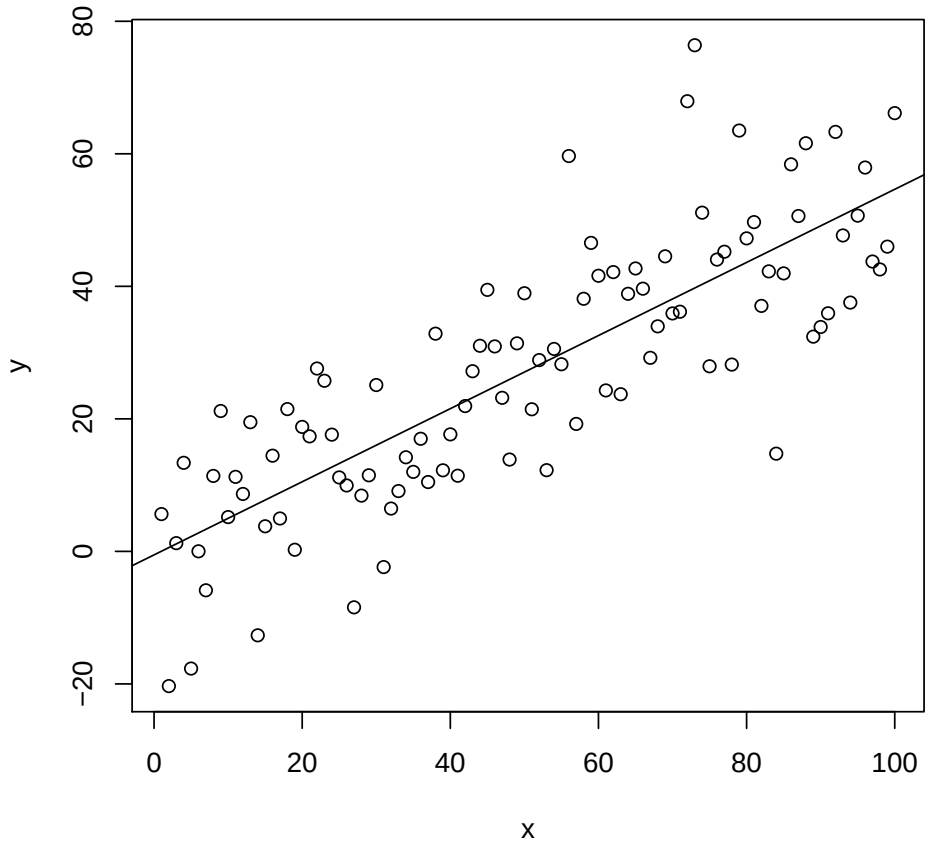
```
> prop.table(table(X,Y),1)
```

```
   Y
X            0          1
  0 0.92882562 0.07117438
  1 0.79908676 0.20091324
```

7. Generate data obeying a simple linear regression model $y_i = 5 + 0.5x_i + \varepsilon_i$, $i = 1, \ldots 100$, in which $\varepsilon_i \sim N(0, 10^2)$, and x_i values are integers from 1 to 100. Plot the (x_i, y_i)-values, and estimate the parameters of that model.

```
> x <- 1:100
> y <- 5 + 0.5*x + rnorm(100,0,10)
> plot(x,y)
> abline(lm(y~x))
> summary(lm(y~x))$coef
```

```
             Estimate Std. Error    t value     Pr(>|t|)
(Intercept) -0.5298900 2.35827861 -0.2246935 8.226855e-01
x            0.5517185 0.04054264 13.6083512 2.592089e-24
```



8. Now change the slope coefficient of x to (a) 0.05, and (b) 0.01, respectively, and perform a similar simulation run. Do you still discover association between x and y? (Look at the scatter plot and the error margin of the slope coefficient) Check also, what happens if you change the standard deviation of the error term to be a) 1, and b) 100.

2.4 Tabulation

2.4.1 Introduction

R and its add-on packages provide several different tabulation functions with different capabilities. The appropriate function to use depends on your goal. There are at least three different uses for tables.

The first use is to create simple summary statistics that will be used for further calculations in R. For example, a two-by-two table created by the `table` function can be passed to `fisher.test`, which will calculate odds ratios and confidence intervals. The appearance of these tables may, however, be quite basic, as their principal goal is to create new objects for future calculations.

A quite different use of tabulation is to make "production quality" tables for publication. You may want to generate reports from for publication in paper form, or on the World Wide Web. The package `xtable` provides this capability, but it is not covered by this course.

An intermediate use of tabulation functions is to create human-readable tables for discussion within your work-group, but not for publication. The `Epi` package provides a function `stat.table` for this purpose, and this practical is designed to introduce this function.

2.4.2 The births data

We shall use the births data which concern 500 mothers who had singleton births in a large London hospital. The outcome of interest is the birth weight of the baby, also dichotomised as normal or low birth weight. These data are available in the Epi package:

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

[1] "id"      "bweight" "lowbw"   "gestwks" "preterm" "matage"  "hyp"
[8] "sex"

> head(births)

  id bweight lowbw gestwks preterm matage hyp sex
1  1    2974     0   38.52       0     34   0   2
2  2    3270     0      NA      NA     30   0   1
3  3    2620     0   38.15       0     35   0   2
4  4    3751     0   39.80       0     31   0   1
5  5    3200     0   38.89       0     33   1   1
6  6    3673     0   40.97       0     33   0   2
```

The housekeeping file for these data is `births-house.r`, which you can download from a location to be specified in the practical. Assuming that you have copied the housekeeping file into the subdirectory `/data` of your working directory, you should now start the session by running this file with the command:

```
> source("data/births-house.r")
```

Make sure you know what the script file has done using `str(births)`.

2.4.3 One-way tables

The simplest table one-way table is created by

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

```
--------------
sex   count()
--------------
M           264
F           236
--------------
```

This creates a count of individuals, classified by levels of the factor `sex`. Compare this table with the equivalent one produced by the `table` function. Note that `stat.table` has a `data` argument that allows you to use variables in a data frame without specifying the frame.

You can display several summary statistics in the same table by giving a list of expressions to the `contents` argument:

```
> stat.table(index = sex, contents = list(count(), percent(sex)), data=births)
```

```
---------------------------
sex   count() percent(sex)
---------------------------
M           264         52.8
F           236         47.2
---------------------------
```

Only a limited set of expressions are allowed: see the help page for `stat.table` for details.

You can also calculate marginal tables by specifying `margin=TRUE` in your call to `stat.table`. Do this for the above table. Check that the percentages add up to 100 and the total for `count()` is the same as the number of rows of the data frame `births`.

```
> stat.table(index = sex, contents = list(count(), percent(sex)),
+   margin=TRUE, data=births)
```

```
-----------------------------
sex     count() percent(sex)
-----------------------------
M           264         52.8
F           236         47.2

Total       500        100.0
-----------------------------
```

To see how the mean birth weight changes with `sex`, try

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

```
--------------------
sex   mean(bweight)
--------------------
M           3229.90
F           3032.83
--------------------
```

Add the count to this table. Add the margin with `margin=TRUE`.

```
> stat.table(index = sex, contents = list(count(), mean(bweight)),
+    margin=T, data=births)
```

```
------------------------------
sex     count() mean(bweight)
------------------------------
M           264       3229.90
F           236       3032.83

Total       500       3136.88
------------------------------
```

As an alternative to `bweight` we can look at `lowbw` with

```
> stat.table(index = sex, contents = percent(lowbw), data=births)
```

```
---------------------
sex   percent(lowbw)
---------------------
M               100.0
F               100.0
---------------------
```

All the percentages are 100! To use the `percent` function the variable `lowbw` must also be in the index, as in

```
> stat.table(index = list(sex,lowbw), contents = percent(lowbw), data=births)
```

```
----------------------
     ------lowbw------
sex         0       1
----------------------
M        89.8    10.2
F        86.0    14.0
----------------------
```

The final column is the percentage of babies with low birth weight by different categories of gestation.

1. Obtain a table showing the frequency distribution of `gest4`.
2. Show how the mean birth weight changes with `gest4`.
3. Show how the percentage of low birth weight babies changes with `gest4`.

```
> stat.table(index = gest4, contents = count(), data=births)

 ------------------
 gest4     count()
 ------------------
 [20,35)        31
 [35,37)        32
 [37,39)       167
 [39,45)       260
 ------------------

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

 ------------------------
 gest4     mean(bweight)
 ------------------------
 [20,35)         1733.74
 [35,37)         2590.31
 [37,39)         3093.77
 [39,45)         3401.26
 ------------------------

> stat.table(index = list(lowbw,gest4), contents = percent(lowbw), data=births)

 ---------------------------------------
        -------------gest4--------------
 lowbw    [20,35) [35,37) [37,39) [39,45)
 ---------------------------------------
 0           19.4    59.4    89.2    98.8
 1           80.6    40.6    10.8     1.2
 ---------------------------------------
```

Another way of obtaining the percentage of low birth weight babies by gestation is to use the ratio function:

```
> stat.table(gest4,ratio(lowbw,1,100),data=births)

 ----------------------
 gest4     ratio(lowbw,
                1, 100)
 ----------------------
 [20,35)          80.65
 [35,37)          40.62
 [37,39)          10.78
 [39,45)           1.15
 ----------------------
```

This only works because `lowbw` is coded 0/1, with 1 for low birth weight.

Tables of odds can be produced in the same way by using `ratio(lowbw, 1-lowbw)`. The `ratio` function is also very useful for making tables of rates with (say) `ratio(D,Y,1000)` where `D` is the number of failures, and `Y` is the follow-up time. We shall return to rates in a later practical.

2.4.4 Improving the Presentation of Tables

The `stat.table` function provides default column headings based on the `contents` argument, but these are not always very informative. Supply your own column headings using *tagged* lists as the value of the `contents` argument, within a `stat.table` call:

```
> stat.table(gest4,contents = list( N=count(),
+      "(%)" = percent(gest4)),data=births)
```

```
-------------------------
gest4           N     (%)
-------------------------
[20,35)        31     6.3
[35,37)        32     6.5
[37,39)       167    34.1
[39,45)       260    53.1
-------------------------
```

This improves the readability of the table. It remains to give an informative title to the index variable. You can do this in the same way: instead of giving `gest4` as the `index` argument to `stat.table`, use a named list:

```
> stat.table(index = list("Gestation time" = gest4),data=births)
```

```
--------------------
Gestation   count()
time
--------------------
[20,35)          31
[35,37)          32
[37,39)         167
[39,45)         260
--------------------
```

2.4.5 Two-way Tables

The following call gives a 2×2 table showing the mean birth weight cross-classified by `sex` and `hyp`.

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

```
----------------------
     -------hyp-------
sex    normal   hyper
----------------------
M     3310.75 2814.40
F     3079.50 2699.72
----------------------
```

Add the count to this table and repeat the function call using `margin = TRUE` to calculate the marginal tables.

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

```
 --------------------------------
        -----------hyp-----------
 sex       normal   hyper    Total
 --------------------------------
 M            221      43      264
         3310.75 2814.40 3229.90

 F            207      29      236
         3079.50 2699.72 3032.83


 Total        428      72      500
         3198.90 2768.21 3136.88
 --------------------------------
```

Use `stat.table` with the ratio function to obtain a 2×2 table of percent low birth weight by `sex` and `hyp`.

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

```
 --------------------------------
        -----------hyp-----------
 sex       normal   hyper    Total
 --------------------------------
 M            221      43      264
         3310.75 2814.40 3229.90

 F            207      29      236
         3079.50 2699.72 3032.83


 Total        428      72      500
         3198.90 2768.21 3136.88
 --------------------------------
```

```
> stat.table(list(sex,hyp), contents=list(count(),ratio(lowbw,1,100)),margin=T, data=births
```

```
 --------------------------------
        -----------hyp-----------
 sex       normal   hyper    Total
 --------------------------------
 M            221      43      264
            6.79   27.91   10.23

 F            207      29      236
           12.08   27.59   13.98


 Total        428      72      500
            9.35   27.78   12.00
 --------------------------------
```

You can have fine-grained control over which margins to calculate by giving a logical vector to the `margin` argument. Use `margin=c(FALSE, TRUE)` to calculate margins over `sex` but not `hyp`. This might not be what you expect, but the `margin` argument indicates which of the index variables are to be **marginalized out**, not which index variables are to remain.

2.4.6 Printing

Just like every other R function, `stat.table` produces an object that can be saved and printed later, or used for further calculation. You can control the appearance of a table with an explicit call to `print()`

There are two arguments to the print method for `stat.table`. The `width` argument which specifies the minimum column width, and the `digits` argument which controls the number of digits printed after the decimal point. This table

```
> odds.tab <- stat.table(gest4, list("odds of low bw" = ratio(lowbw,1-lowbw)),
+                data=births)
> print(odds.tab)
```

```
------------------
gest4     odds of
           low bw
------------------
[20,35)      4.17
[35,37)      0.68
[37,39)      0.12
[39,45)      0.01
------------------
```

shows a table of odds that the baby has low birth weight. Use `width=15` and `digits=3` and see the difference.

```
> print(odds.tab, width=15, digits=3)
```

```
--------------------------
gest4      odds of low bw
--------------------------
[20,35)             4.167
[35,37)             0.684
[37,39)             0.121
[39,45)             0.012
--------------------------
```

2.5 Graphics in R

2.6 Calculation of rates, RR and RD

```
R version 3.2.0 (2015-04-16) -- "Full of Ingredients"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

 R 3.2.0
 ------------------------------------------------
 Program:
  Folder:  /home/bendix/teach/SPE/repos/pracs
 Started:  Tuesday 26. May 2015, 10:51:11
 ------------------------------------------------
> ### R code from vignette source '/home/bendix/teach/SPE/repos/pracs/rates-rrrd.Rnw'
>
> ###################################################
> ### code chunk number 1: rates-rrrd.Rnw:33-35
> ###################################################
> library( Epi )

Attaching package: 'Epi'

The following object is masked from 'package:base':

    merge.data.frame

> options(digits=4)
>
>
> ###################################################
> ### code chunk number 2: rates-rrrd.Rnw:37-46
> ###################################################
> D <- 15
> Y <- 5.532    # thousands of years
> rate <- D / Y
> SE.rate <- rate/sqrt(D)
> c(rate, SE.rate, rate + c(-1.96, 1.96)*SE.rate )
[1] 2.7115 0.7001 1.3393 4.0837
> SE.logr <- 1/sqrt(D)
> EF <- exp( 1.96 * SE.logr )
> c(log(rate), SE.logr)
[1] 0.9975 0.2582
> c( rate, EF, rate/EF, rate*EF )
[1] 2.711 1.659 1.635 4.498
>
>
> ###################################################
> ### code chunk number 3: rates-rrrd.Rnw:54-58
> ###################################################
> ci.mat
function (alpha = 0.05, df = Inf)
{
    ciM <- rbind(c(1, 1, 1), qt(1 - alpha/2, df) * c(0, -1, 1))
    colnames(ciM) <- c("Estimate", paste(formatC(100 * alpha/2,
        format = "f", digits = 1), "%", sep = ""), paste(formatC(100 *
        (1 - alpha/2), format = "f", digits = 1), "%", sep = ""))
    ciM
}
<environment: namespace:Epi>
> ci.mat()
     Estimate  2.5% 97.5%
[1,]        1  1.00  1.00
[2,]        0 -1.96  1.96
> c( rate, SE.rate ) %*% ci.mat()
     Estimate  2.5% 97.5%
[1,]    2.711 1.339 4.084
> exp( c( log(D/Y), 1/sqrt(D) ) %*% ci.mat() )
     Estimate  2.5% 97.5%
[1,]    2.711 1.635 4.498
>
>
> ###################################################
```

```
> ### code chunk number 4: rates-rrrd.Rnw:63-65
> ###################################################
> mm <- glm( D ~ 1, family=poisson(link=log), offset=log(Y) )
> summary( mm )

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

Deviance Residuals:
[1]  0

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    0.998      0.258    3.86  0.00011

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 0  on 0  degrees of freedom
Residual deviance: 0  on 0  degrees of freedom
AIC: 6.557

Number of Fisher Scoring iterations: 3

>
>
> ###################################################
> ### code chunk number 5: rates-rrrd.Rnw:74-75
> ###################################################
> ci.lin( mm )
            Estimate StdErr     z         P   2.5% 97.5%
(Intercept)   0.9975 0.2582 3.863 0.0001119 0.4914 1.504
>
>
> ###################################################
> ### code chunk number 6: rates-rrrd.Rnw:79-80
> ###################################################
> ci.lin( mm, Exp=T)
            Estimate StdErr     z         P exp(Est.)  2.5% 97.5%
(Intercept)   0.9975 0.2582 3.863 0.0001119     2.711 1.635 4.498
>
>
> ###################################################
> ### code chunk number 7: rates-rrrd.Rnw:84-86
> ###################################################
> ci.exp( mm)
            exp(Est.)  2.5% 97.5%
(Intercept)     2.711 1.635 4.498
> ci.lin( mm, Exp=T)[, 5:7]
exp(Est.)      2.5%     97.5%
    2.711     1.635     4.498
>
>
> ###################################################
> ### code chunk number 8: rates-rrrd.Rnw:95-97
> ###################################################
> mmx <- glm( D/Y ~ 1, weight=Y, family=poisson )
Warning message:
In dpois(y, mu, log = TRUE) : non-integer x = 2.711497
> ci.exp( mmx)
            exp(Est.)  2.5% 97.5%
(Intercept)     2.711 1.635 4.498
>
>
> ###################################################
> ### code chunk number 9: rates-rrrd.Rnw:104-105
> ###################################################
> ma <- glm( D/Y ~ 1, family=poisson(link=identity), weight=Y )
Warning message:
In dpois(y, mu, log = TRUE) : non-integer x = 2.711497
>
>
> ###################################################
> ### code chunk number 10: rates-rrrd.Rnw:113-115
> ###################################################
> ci.lin( ma )
            Estimate StdErr     z         P  2.5% 97.5%
(Intercept)    2.711 0.7001 3.873 0.0001075 1.339 4.084
> ci.lin( ma )[,c(1,5,6)]
Estimate     2.5%    97.5%
   2.711    1.339    4.084
>
>
> ###################################################
> ### code chunk number 11: rates-rrrd.Rnw:123-126
> ###################################################
> Dx <- c(3,7,5)
> Yx <- c(1.412,2.783,1.337)
```

```
> Px <- 1:3
>
>
> ###################################################
> ### code chunk number 12: rates-rrrd.Rnw:129-131
> ###################################################
> m1 <- glm( Dx ~ 1, family=poisson, offset=log(Yx)  )
> ci.exp(m1)
            exp(Est.)  2.5% 97.5%
(Intercept)     2.711 1.635 4.498
>
>
> ###################################################
> ### code chunk number 13: rates-rrrd.Rnw:136-137
> ###################################################
> mp <- glm( Dx ~ factor(Px), offset=log(Yx), family=poisson )
>
>
> ###################################################
> ### code chunk number 14: rates-rrrd.Rnw:141-142
> ###################################################
> 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        0.7
2         0        0.0  2      0.7      0.7
>
>
> ###################################################
> ### code chunk number 15: rates-rrrd.Rnw:167-173
> ###################################################
> D0 <- 15   ; D1 <- 28
> Y0 <- 5.532 ; Y1 <- 4.783
> RR <- (D1/Y1)/(D0/Y0)
> SE.lrr <- sqrt(1/D0+1/D1)
> EF <- exp( 1.96 * SE.lrr)
> c( RR, RR/EF, RR*EF )
[1] 2.159 1.153 4.042
>
>
> ###################################################
> ### code chunk number 16: rates-rrrd.Rnw:177-178
> ###################################################
> exp( c( log(RR), SE.lrr ) %*% ci.mat() )
     Estimate  2.5% 97.5%
[1,]    2.159 1.153 4.042
>
>
> ###################################################
> ### code chunk number 17: rates-rrrd.Rnw:182-184
> ###################################################
> D <- c(D0,D1) ; Y <- c(Y0,Y1); expos <- 0:1
> mm <- glm( D ~ factor(expos), family=poisson, offset=log(Y) )
>
>
> ###################################################
> ### code chunk number 18: rates-rrrd.Rnw:189-191
> ###################################################
> ci.exp( mm)
                exp(Est.)  2.5% 97.5%
(Intercept)         2.711 1.635 4.498
factor(expos)1      2.159 1.153 4.042
> ci.lin( mm, E=T)[,5:7]
                exp(Est.)  2.5% 97.5%
(Intercept)         2.711 1.635 4.498
factor(expos)1      2.159 1.153 4.042
>
>
> ###################################################
> ### code chunk number 19: rates-rrrd.Rnw:203-206
> ###################################################
> rd <- diff( D/Y )
> sd <- sqrt( sum( D/Y^2 ) )
> c( rd, sd ) %*% ci.mat()
     Estimate   2.5% 97.5%
[1,]    3.143 0.5765 5.709
>
>
> ###################################################
> ### code chunk number 20: rates-rrrd.Rnw:211-214
> ###################################################
> ma <- glm( D/Y ~ factor(expos),
+           family=poisson(link=identity), weight=Y )
Warning messages:
```

```
1: In dpois(y, mu, log = TRUE) : non-integer x = 2.711497
2: In dpois(y, mu, log = TRUE) : non-integer x = 5.854066
> ci.lin( ma )[, c(1,5,6)]
                Estimate   2.5% 97.5%
(Intercept)        2.711 1.3393 4.084
factor(expos)1     3.143 0.5765 5.709
>
>
> ##################################################
> ### code chunk number 21: rates-rrrd.Rnw:221-227
> ##################################################
> 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(expos),
+              family=poisson,  offset=log(Y) )
> ci.exp( mm, ctr.mat=CM)
           exp(Est.)  2.5% 97.5%
rate 0         2.711 1.635 4.498
rate 1         5.854 4.042 8.479
RR 1 vs. 0     2.159 1.153 4.042
>
>
> ##################################################
> ### code chunk number 22: rates-rrrd.Rnw:233-237
> ##################################################
> rownames( CM ) <- c("rate 0","rate 1","RD 1 vs. 0")
> ma <- glm( D/Y ~ factor(expos),
+                family=poisson(link=identity), weight=Y )
Warning messages:
1: In dpois(y, mu, log = TRUE) : non-integer x = 2.711497
2: In dpois(y, mu, log = TRUE) : non-integer x = 5.854066
> ci.lin( ma, ctr.mat=CM )[, c(1,5,6)]
           Estimate   2.5% 97.5%
rate 0        2.711 1.3393 4.084
rate 1        5.854 3.6857 8.022
RD 1 vs. 0    3.143 0.5765 5.709
>
>
>

 ------------------------------------------
 Program:
  Folder: /home/bendix/teach/SPE/repos/pracs
    Ended: Tuesday 26. May 2015, 10:51:11
 Elapsed: 00:00:00
 ------------------------------------------
> proc.time()
   user  system elapsed
  1.081   0.035   1.108
```

2.7 Logistic regression (GLM)

2.8 Simple estimation of effects

2.9 Estimation and reporting of linear and curved effects

In this exercise we will use the `testisDK` data from the `Epi` package, which contains the number of cases of testis cancer in Denmark 1943–96:

1. First we load the Danish testis cancer data, and inspect the dataset:

```
library( Epi )
sessionInfo()
```

```
R version 3.1.0 (2014-04-10)
Platform: i386-w64-mingw32/i386 (32-bit)

locale:
[1] LC_COLLATE=Danish_Denmark.1252  LC_CTYPE=Danish_Denmark.1252
[3] LC_MONETARY=Danish_Denmark.1252 LC_NUMERIC=C
[5] LC_TIME=Danish_Denmark.1252

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

other attached packages:
[1] Epi_1.1.65     foreign_0.8-61

loaded via a namespace (and not attached):
[1] tools_3.1.0
```

```
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
```

We can tabulate both events (testis cancer diagnoses) and person-years using either `xtabs` or `stat.table`, the latter is a bit more versatile, because we can get rates too:

```
round( ftable( xtabs( cbind(D,PY=Y/1000) ~ I(floor(A/10)*10) +
                                           I(floor(P/10)*10),
                      data=testisDK ),
               row.vars=c(3,1) ), 1 )
```

```
                      I(floor(P/10) * 10)   1940   1950   1960   1970   1980   1990
   I(floor(A/10) * 10)
D  0                                        10.0    7.0   16.0   18.0    9.0   10.0
   10                                       13.0   27.0   37.0   72.0   97.0   75.0
   20                                      124.0  221.0  280.0  535.0  724.0  557.0
   30                                      149.0  288.0  377.0  624.0  771.0  744.0
   40                                       95.0  198.0  230.0  334.0  432.0  360.0
   50                                       40.0   79.0  140.0  151.0  193.0  155.0
   60                                       29.0   43.0   54.0   83.0   82.0   44.0
   70                                       18.0   26.0   35.0   41.0   40.0   32.0
   80                                        7.0    9.0   13.0   19.0   18.0   21.0
PY 0                                     2604.7 4037.3 3885.0 3820.9 3070.9 2165.5
   10                                    2135.7 3505.2 4004.1 3906.1 3847.4 2261.0
   20                                    2225.5 2923.2 3401.6 4028.6 3941.2 2824.6
   30                                    2195.2 3058.8 2856.2 3410.6 3968.8 2728.4
   40                                    1874.9 2980.1 2986.8 2823.1 3322.6 2757.7
   50                                    1442.8 2426.5 2796.6 2813.3 2635.0 2069.2
   60                                    1041.9 1711.8 2055.1 2358.1 2357.3 1565.0
   70                                     537.6  967.9 1136.1 1336.9 1538.0 1100.9
   80                                     133.6  261.6  346.3  423.5  504.2  414.6
```

```
stat.table( list(A=floor(A/10)*10,
                 P=floor(P/10)*10),
            list( D=sum(D),
                  Y=sum(Y/1000),
               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
          2604.66  4037.31  3884.97  3820.88  3070.87  2165.54   19584.22
             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
          2135.73  3505.19  4004.13  3906.08  3847.40  2260.97   19659.48
             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
          2225.55  2923.22  3401.65  4028.57  3941.18  2824.58   19344.74
             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
          2195.23  3058.81  2856.20  3410.58  3968.81  2728.35   18217.97
             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
          1874.92  2980.15  2986.83  2823.11  3322.59  2757.72  16745.30
             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
          1442.85  2426.54  2796.60  2813.32  2635.00  2069.18  14183.49
             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
          1041.94  1711.79  2055.08  2358.05  2357.28  1564.98  11089.13
             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
           537.62   967.88  1136.06  1336.95  1538.02  1100.86   6617.39
             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
           133.57   261.61   346.26   423.50   504.20   414.61   2083.75
             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
         14192.04 21872.50 23467.78 24921.03 25185.34 17886.80 127525.49
             3.42     4.11     5.04     7.53     9.39    11.17      6.91
-------------------------------------------------------------------------
```

Note that for this type of cancer the peak age-specific rates are in the 30es.

2. We then fit a Poisson-model for the mortality rates with a linear term for age:

```
ml <- glm( D ~ A, offset=log(Y), family=poisson, data=testisDK )
ci.exp( ml )
```

```
               exp(Est.)         2.5%        97.5%
(Intercept) 5.682883e-05 0.0000545697 0.0000591815
A           1.005499e+00 1.0045507062 1.0064479370
```

The parameter labeled `A` gives the annual increase in mortality by age (0.55%/year), but the intercept parameter is meaningless; it is the predicted mortality per 1 person-year (because we used `Y` in the offset, and this is in units of person-years), but for a 0 year old male.

3. We can work out the predicted log-mortality rates for ages 25 to 45, say, by doing a hand-calculation based on the coefficients:

```
( cf <- coef( ml ) )
```

```
 (Intercept)            A
-9.775466746  0.005483811
```

We now have the intercept (the log-rate) and the slopes for age and calendar time, so to get the age-specific rates in ages 50 to 60 we just take the intercept and add the slope mulitlied by the vector of ages.

```
round( cbind( 25:45, exp( cf[1] + cf[2]*(25:45) )*10^5 ), 3 )
```

```
      [,1]  [,2]
 [1,]   25 6.518
 [2,]   26 6.554
 [3,]   27 6.590
 [4,]   28 6.626
 [5,]   29 6.662
 [6,]   30 6.699
 [7,]   31 6.736
 [8,]   32 6.773
 [9,]   33 6.810
[10,]   34 6.848
[11,]   35 6.885
[12,]   36 6.923
[13,]   37 6.961
[14,]   38 7.000
[15,]   39 7.038
[16,]   40 7.077
[17,]   41 7.116
[18,]   42 7.155
[19,]   43 7.194
[20,]   44 7.234
[21,]   45 7.273
```

Note that we also multiplied by 10^5 in order to get the rates in units of cases per 100,000 person-years.

4. But we do not have the standard errors of these mortality rates, and hence neither the confidence intervals. This is implemented in `ci.exp`; if we provide the argument `ctr.mat=` as a matrix where each row corresponds to a prediction point and each column to a parameter from the model.

   Thus for each age we need the corresponding multipliers for the coefficients:

   ```
   ( CM <- cbind( 1, 25:45 ) )
   ```

   ```
        [,1] [,2]
    [1,]    1   25
    [2,]    1   26
    [3,]    1   27
    [4,]    1   28
    [5,]    1   29
    [6,]    1   30
    [7,]    1   31
    [8,]    1   32
    [9,]    1   33
   [10,]    1   34
   [11,]    1   35
   [12,]    1   36
   [13,]    1   37
   [14,]    1   38
   [15,]    1   39
   [16,]    1   40
   ```

```
[17,]    1   41
[18,]    1   42
[19,]    1   43
[20,]    1   44
[21,]    1   45
```

```
round( ci.exp( ml, ctr.mat=CM )*10^5, 3 )
```

```
     exp(Est.)  2.5% 97.5%
 [1,]     6.518 6.365 6.674
 [2,]     6.554 6.403 6.708
 [3,]     6.590 6.441 6.742
 [4,]     6.626 6.479 6.777
 [5,]     6.662 6.516 6.812
 [6,]     6.699 6.554 6.847
 [7,]     6.736 6.592 6.883
 [8,]     6.773 6.630 6.919
 [9,]     6.810 6.667 6.956
[10,]     6.848 6.705 6.993
[11,]     6.885 6.743 7.031
[12,]     6.923 6.780 7.069
[13,]     6.961 6.817 7.108
[14,]     7.000 6.855 7.147
[15,]     7.038 6.892 7.187
[16,]     7.077 6.929 7.228
[17,]     7.116 6.966 7.268
[18,]     7.155 7.003 7.310
[19,]     7.194 7.040 7.352
[20,]     7.234 7.077 7.394
[21,]     7.273 7.113 7.437
```

5. We can now use this machinery to plot the mortality rates over the range from 15 to 65 years:

```
C1 <- cbind( 1, 15:65 )
matplot( 15:65, ci.exp( ml, ctr.mat=C1 )*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" )
```

6. Now suppose we want to see if the mortality rates really are exponentially increasing by age (that is linearly on the log-scale), we could add a quadratic term to the model:

```
mq <- glm( D ~ A + I(A^2), offset=log(Y), family=poisson, data=testisDK )
ci.exp( mq, Exp=F )
```

```
                Estimate          2.5%         97.5%
(Intercept) -12.365625166 -12.482504296 -12.248746037
A             0.180595889   0.174140158   0.187051619
I(A^2)       -0.002325937  -0.002410829  -0.002241045
```

Note that we must use the function `I()` to prevent the "^" to be interpreted as part of the model formula.

We can then plot the estimated rates using the same machinery, but now with 3 columns in the matrix:

```
aa <- 15:65
C2 <- cbind( 1, aa, aa^2 )
matplot( aa, ci.exp( mq, ctr.mat=C2 )*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" )
matlines( aa, ci.exp( ml, ctr.mat=C1 )*10^5,
          type="l", lty=1, lwd=c(3,1,1), col="blue" )
```

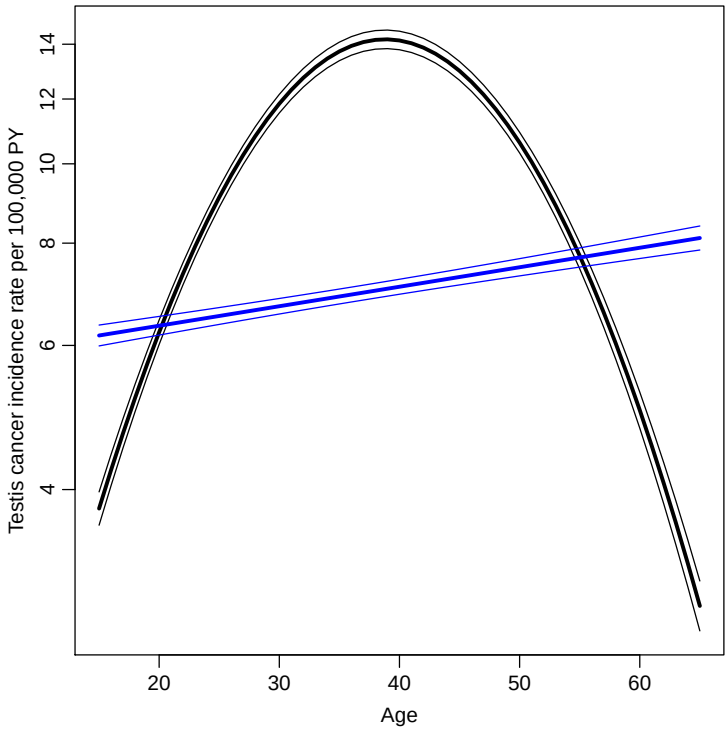


Figure 2.1: *Testis cancer incidence rates overall, modelled by 2nd degree polynomial, overlaid by the estimated linear estimate.*

Which indeed is dramatically different — we see that the model with quadratic effect gives a much better fit; a deviance of 4800 on 1 d.f.:

```
anova( mq, ml, test="Chisq" )
```

```
Analysis of Deviance Table

Model 1: D ~ A + I(A^2)
Model 2: D ~ A
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1      4857     7815.8
2      4858    12648.0 -1  -4832.1 < 2.2e-16
```

7. We could do the same using a 3rd degree polynomial:

```
mc <- glm( D ~ A + I(A^2) + I(A^3), offset=log(Y), family=poisson, data=testisDK )
C3 <- cbind( 1, aa, aa^2, aa^3 )
matplot( aa, ci.exp( mc, ctr.mat=C3 )*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" )
```

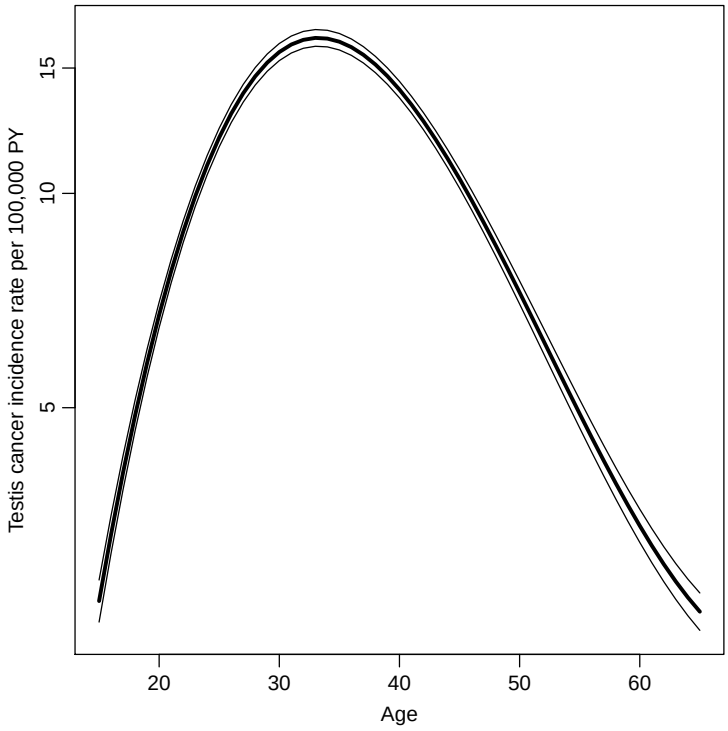


Figure 2.2: *Testis cancer incidence rates overall, modelled by 3rd degree polynomial.*

8. Instead of continuing with higher powers of age we could use fractions of powers ("fractional polynomials"), or we could use splines, which are piecewise polynomial curves, that fit nicely together at join points (knots). This is implemented in the `splines` package, in the function `ns`, which returns a matrix. There is a wrapper `Ns` in the `Epi`-package that automatically designate the smallest and largest knots as *boundary knots*, beyond which the resulting curve is linear:

```
library( splines )
ms <- glm( D ~ Ns(A,knots=seq(15,65,10)), offset=log(Y), family=poisson, data=testisD
As <- Ns( aa, knots=seq(15,65,10) )
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" )
```

9. Now in addition to this we would like to see how the dependence on calendar was, so we add a linear term to the model, and make a prediction for 1970, say:

```
msp <- glm( D ~ Ns(A,knots=seq(15,65,10)) + P, offset=log(Y), family=poisson, data=te
CM <- cbind( 1, Ns( aa, knots=seq(15,65,10) ), 1970 )
```

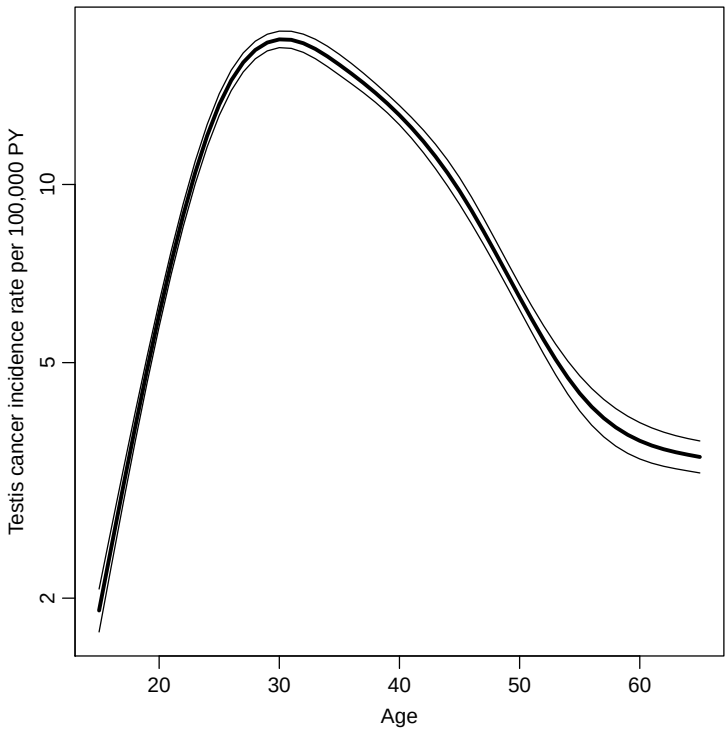


Figure 2.3: *Testis cancer incidence rates overall, modelled by splines.*

```
matplot( aa, ci.exp( msp, ctr.mat=CM )*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" )
```

10. We would also like to see how the RR relative to 1970 is, so we select only the period parameter, using the `subset` argument:

```
ci.exp( msp, subset="P" )
```

```
  exp(Est.)     2.5%    97.5%
P  1.024235 1.022769 1.025704
```

So we have an increase of 2.4% per year.

To get the RR relative to 1970 for the years 1943 to 1996 we must multiply the log-RR for period with the distance form 1970, such as:

```
yy  <- 1943:1996
Cp1 <- cbind( yy - 1970 )
matplot( yy, ci.exp( msp, ctr.mat=Cp1, subset="P" ),
         log="y", xlab="Date", ylab="RR of Testis cancer",
         type="l", lty=1, lwd=c(3,1,1), col="black" )
abline( h=1 )
```

11. As above we might like to see how it looks if we add a quadratic to the period effect:

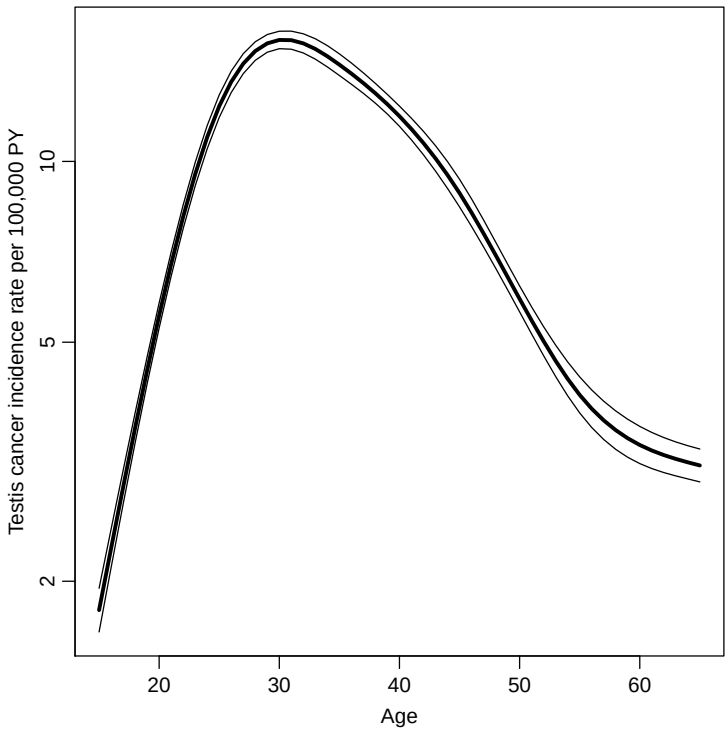


Figure 2.4: *Testis cancer incidence rate in 1970. Note the different level of the rates relative to the overall plot above.*

```
msp <- glm( D ~ Ns(A,knots=seq(15,65,10)) + P + I(P^2),
                offset=log(Y), family=poisson, data=testisDK )
Cq <- cbind( yy, yy^2 ) - cbind( rep(1970,length(yy)), 1970^2 )
matplot( yy, ci.exp( msp, ctr.mat=Cq, subset="P" ),
         log="y", xlab="Age", ylab="Testis cancer incidence RR",
         type="l", lty=1, lwd=c(3,1,1), col="black" )
abline( h=1, v=1970 )
```

12. But we would like also to see if there were some non-linearity beyond the quadratic, with period as well, so we fit a spline for period (`P`) as well

```
mssp <- glm( D ~ Ns(A,knots=seq(15,65,10)) +
                 Ns(P,knots=seq(1950,1990,10)),
                 offset=log(Y), family=poisson, data=testisDK )
anova( mssp, msp, test="Chisq" )
```

```
Analysis of Deviance Table

Model 1: D ~ Ns(A, knots = seq(15, 65, 10)) + Ns(P, knots = seq(1950,
    1990, 10))
Model 2: D ~ Ns(A, knots = seq(15, 65, 10)) + P + I(P^2)
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1      4850     4787.9
2      4852     4792.3 -2   -4.488    0.106
```

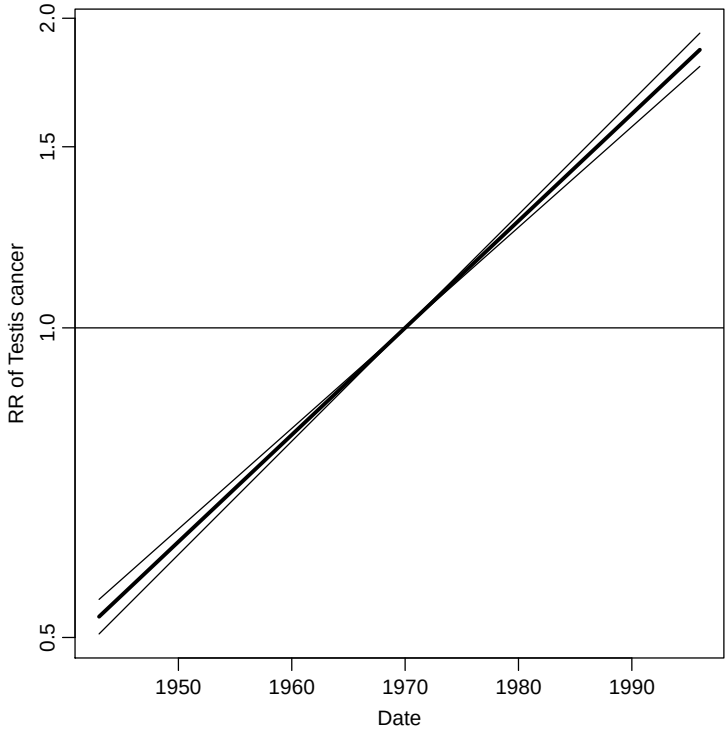


Figure 2.5: *Testis cancer incidence rate-ratio relative to 1970.*

But as above we must compute the *difference* in the contribution from period in year y and in the reference year, here 1970. So every row of the contrast matrix must have the corresponding contribution from the reference year subtracted

```
Ps <- Ns(                yy  , knots=seq(1950,1990,10) )
Pr <- Ns( rep(1970,length(yy)), knots=seq(1950,1990,10) )
matplot( yy, ci.exp( mssp, ctr.mat=Ps-Pr, subset="P" ),
         log="y", xlab="Age", ylab="Testis cancer incidence RR",
         type="l", lty=1, lwd=c(3,1,1), col="black" )
```

13. But for this model we would also like to see the estimated age-specific rates in say 1970.

    To this end we need a reference matrix for the year with a number of rows equal to the number of age-prediction points:

```
Ar <- Ns( rep(1970,length(aa)), knots=seq(1950,1990,10) )
matplot( aa, ci.exp( mssp, ctr.mat=cbind(1,As,Ar) )*10^5,
         log="y", xlab="Age", ylab="Testis cancer incidence RR",
         type="l", lty=1, lwd=c(3,1,1), col="black" )
```

14. In order to all do this in one go where we have overview of what we do, what is needed is:

    - the knots for age and period
    - the prediction points for age and period

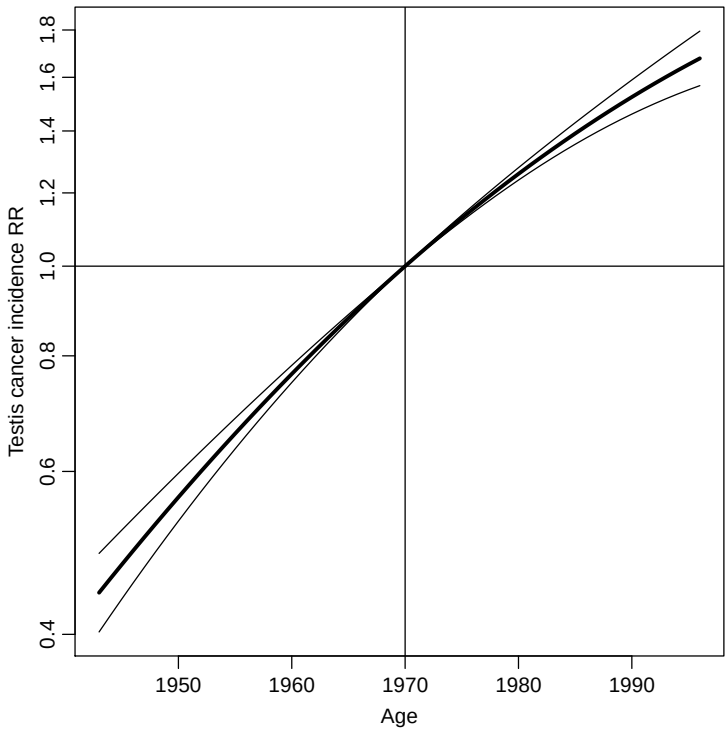


Figure 2.6: *Testis cancer incidence rate-ratio relative to 1970.*

- the reference point for period

— then we can derive everything else from this:

```
a.kn <- seq(15,65,10)
p.kn <- seq(1950,1990,10)
a.pt <- 10:65
p.pt <- 1945:1993
p.ref <- 1970
na <- length(a.pt)
np <- length(p.pt)
As <- Ns( a.pt, knots=a.kn )
Ps <- Ns( p.pt, knots=p.kn )
Prp <- Ns( rep(p.ref,np), knots=p.kn )
Pra <- Ns( rep(p.ref,na), knots=p.kn )
mAP <- glm( D ~ Ns(A,knots=a.kn) + Ns(P,knots=p.kn),
                offset=log(Y), family=poisson, data=testisDK )
par( mfrow=c(1,2) )
matplot( a.pt, ci.exp( mAP, ctr.mat=cbind(1,As,Pra) )*10^5,
         log="y", xlab="Age",
         ylab=paste( "Testis cancer incidence per 100,000 PY, in", p.ref ),
         type="l", lty=1, lwd=c(3,1,1), col="black",
         ylim=c(1,20) )
matplot( p.pt, ci.exp( mAP, ctr.mat=Ps-Prp, subset="P" ),
         log="y", xlab="Age", ylab="Testis cancer incidence RR",
         type="l", lty=1, lwd=c(3,1,1), col="black",
         ylim=c(1,20)/4 )
abline( h=1, v=p.ref )
```

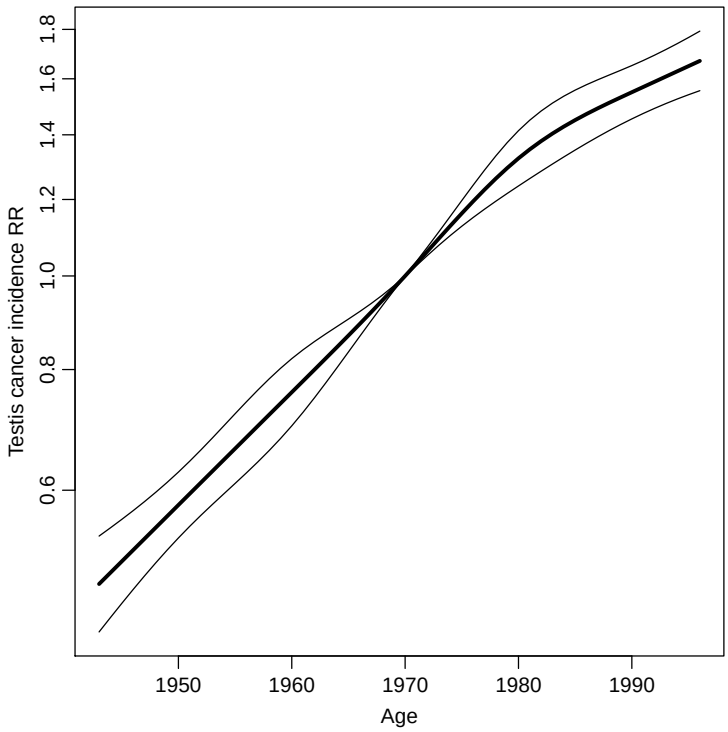


Figure 2.7: *Incidence rates of testis cancer in 1950 per 100,000 PY.*

15. Finally with this in place we could do the same for a model where we had replaced `P`, the data of follow-up by the the date of birth, `B=P-A`:

```
testisDK <- transform( testisDK, B = P-A )
# with( testisDK, hist( rep(B,D), breaks=100, col="black" ) )
a.kn <- seq(15,65,5)
b.kn <- seq(1900,1970,5)
a.pt <- 10:65
b.pt <- 1890:1970
b.ref <- 1940
na <- length(a.pt)
nb <- length(b.pt)
As <- Ns( a.pt, knots=a.kn )
Bs <- Ns( b.pt, knots=b.kn )
Brb <- Ns( rep(b.ref,nb), knots=b.kn )
Bra <- Ns( rep(b.ref,na), knots=b.kn )
mAB <- glm( D ~ Ns(A,knots=a.kn) + Ns(B,knots=b.kn),
                offset=log(Y), family=poisson, data=testisDK )
par( mfrow=c(1,2) )
matplot( a.pt, ci.exp( mAB, ctr.mat=cbind(1,As,Bra) )*10^5,
         log="y", xlab="Age",
         ylab=paste( "Testis cancer incidence per 100,000 PY, in", b.ref, "birth coho
         type="l", lty=1, lwd=c(3,1,1), col="black",
         ylim=c(1,20) )
matplot( b.pt, ci.exp( mAB, ctr.mat=Bs-Brb, subset="B" ),
         log="y", xlab="Age", ylab="Testis cancer incidence RR",
         type="l", lty=1, lwd=c(3,1,1), col="black",
         ylim=c(1,20)/4 )
```

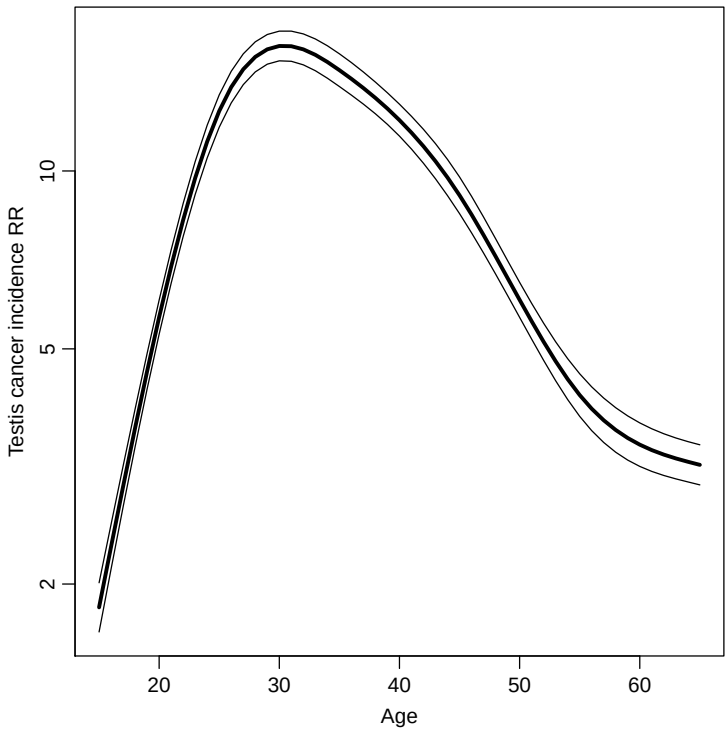


Figure 2.8: *Relative risk by date of follow-up.*

```
abline( h=1, v=b.ref )
```

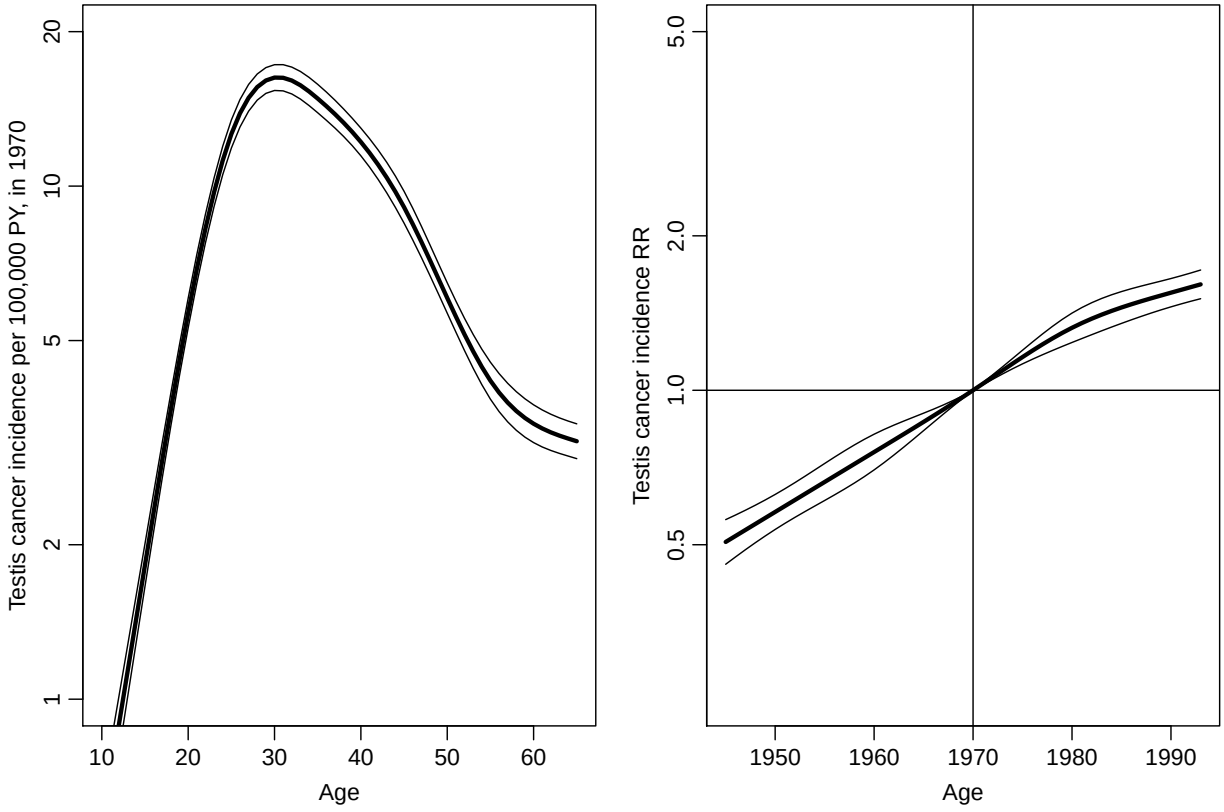


Figure 2.9: *Incidence rates of testis cancer in 1970, and RR relative to this.*

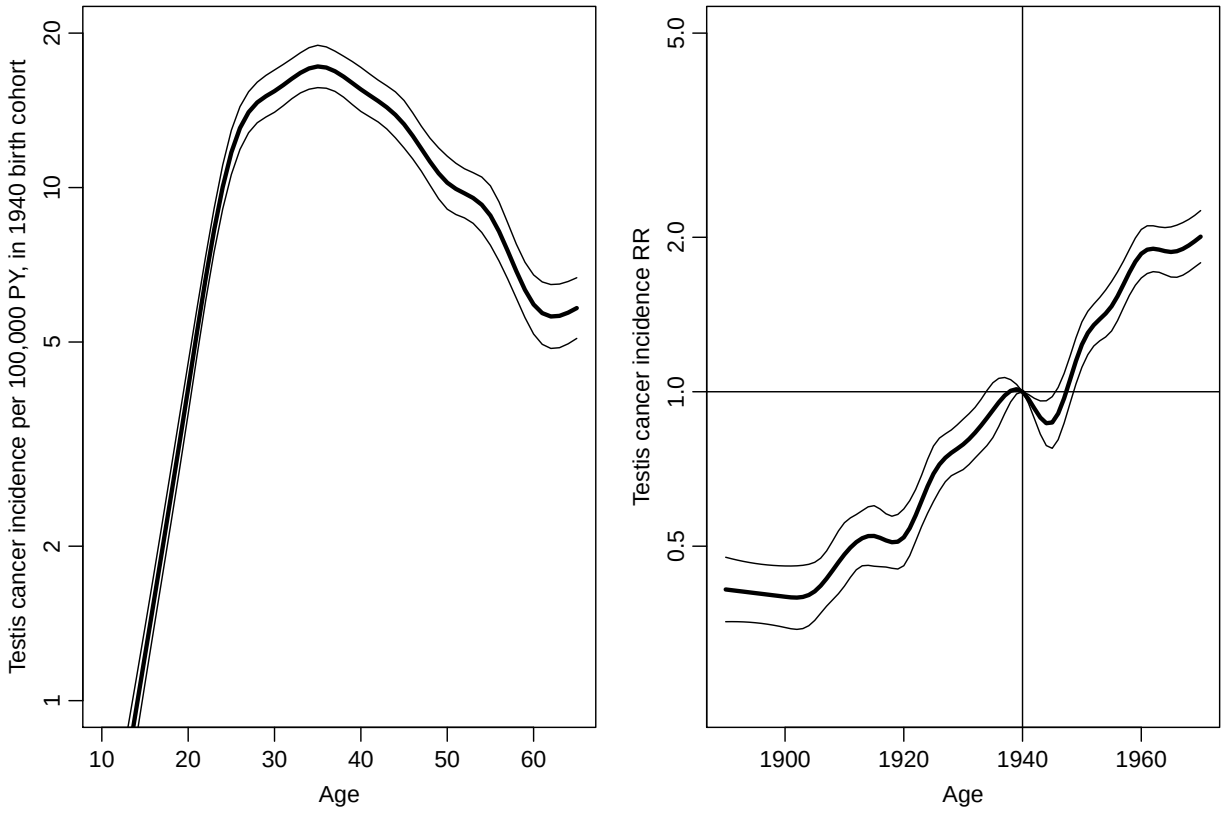


Figure 2.10: *Incidence rates of testis cancer in the 1940 birth cohort, and RR relative to this. We see that there is a considerable effect of birth cohort — there seems to be an effect og being born during the 1st or 2nd world war.*

2.10 Graphics meccano

2.10.1 Details of the classical part of the exercise

First we read in the basic dat from the experiemnet:

```
> ( alkfos <- read.csv("./data/alkfos.csv") )
```

```
   grp  c0  c3  c6  c9 c12 c18 c24
1    1 142 140 159 162 152 175 148
2    1 120 126 120 146 134 119 116
3    1 175 161 168 164 213 194 221
4    1 234 203 174 197 289 174 189
5    1  94 107 146 124 128  98 114
6    1 128  97 113 203  NA  NA  NA
7    1 202 189 208 203 209 200 218
8    1 190 277 270 171 141 192 190
9    1 104 117 135 122 112 133 123
10   1 112  95 114 122 118 119 138
11   1 160 169 178 208 220 215 232
12   1 214 211 215 240 227 288 260
13   1 113 138 112 114 109 106 111
14   1 237 245 219 213 215 225 228
15   1 205 213 248 222 225 207 172
16   1 202 231 236 185 204 226 147
17   1 137 128 136 146 152 132 150
18   1 175 163 167 144 168  NA  NA
19   1 174 151 150 133 134 149 146
20   1  81  81  83  74  82  84 108
21   1 113 131 298 124 126 140 129
22   1 104 114 124 102  94 122 125
23   1 178 172 159 155 157 153 164
24   2 150 122 103 109 103  87 109
25   2 173 127 117 124 143 123 144
26   2 191 174 165 160 177 184  NA
27   2 191 159 157 161 150 187 215
28   2 230 150 144 153 125 124 152
29   2 145 134 167 141 112 212 194
30   2 128  92  89  78  83  78  80
31   2 102  86  80  76  82  79  68
32   2 180 124 116 117 124  NA  NA
33   2 153  96  97  96  93 156 110
34   2 115  79  79  79  73  69  72
35   2 150 113 124 102 100 109 101
36   2 182 147 156  79 135  NA 162
37   2 175 146 157 140 143 158 162
38   2 146  86  81  80  87  89  95
39   2  92  80  95  95  86 119  NA
40   2 228 177 185 181 190 182 192
41   2 178 119 107  NA 102 110  94
42   2 213 185 152 142 158 178 194
43   2 161 107 104 107  NA 118 129
```

Then we find out how many observations that are available for each varaibel in each group (that is how many that are *not* missing (`!is.na`):

```
> (available <- aggregate( !is.na(alkfos), list(alkfos$grp), sum))
```

```
  Group.1 grp c0 c3 c6 c9 c12 c18 c24
1       1  23 23 23 23 23  22  21  21
2       2  20 20 20 20 19  19  18  17
```

Then we compute the percentage change since the first visit; we divide all mesurements by the first measurement `alkfos$c0`. For these we compute the mean and the standard errors at each point for each group:

```
> alkfos.pctchange <- (sweep(alkfos[-1], 1, alkfos$c0, "/") - 1)*100
> (means <- aggregate(alkfos.pctchange, list(alkfos$grp), mean, na.rm=TRUE))
```

```
  Group.1 c0         c3        c6         c9       c12        c18        c24
1       1  0   2.266971  13.01274   4.574702   4.89126   6.165274   7.116543
2       2  0 -23.678490 -23.93942 -27.893094 -27.21339 -18.025015 -20.339544
```

```
> (sds   <- aggregate(alkfos.pctchange, list(alkfos$grp),   sd, na.rm=TRUE))
```

```
  Group.1 c0        c3       c6       c9      c12      c18      c24
1       1  0 14.815678 37.66257 19.00972 16.07197 16.18549 18.77437
2       2  0  9.869802 15.04181 13.97600 11.41099 25.13837 20.25576
```

```
> available <- as.matrix(available[-(1:2)])
```

Put the means and sds into matrices which we will use for plotting of points and error bars. The quantities needed ultimately are `means`, `upr` and `lwr`

```
> means <- as.matrix(means[-1])
>   sds <- as.matrix(sds[-1])
>  sems <- sds/sqrt(available)
>   upr <- means + sems
>   lwr <- means - sems
```

Now we need a vector of times and also some idea of the required extent of the y-axis.

```
> time <- c(0,3,6,9,12,18,24)
> (ylim <- range(means+sems,means-sems))
```

```
[1] -31.09941  20.86593
```

For the graph we first set the size of the margins (you should make a print of the help page for "`par`", which is the functin that sets all relevant graphics parameters).

```
> par(mar=.1 + c(8,4,4,2))
> plot( time, means[1,],
+       type="b",   # boths dots and lines
+       xaxt="n",   # no x-axis
+       ylim=ylim,
+       ylab="alkaline phosphatase" )
> # Add the points for the second gropu
> points(time,means[2,], type="b")
> # The the vertial error-bars
> segments( time, upr[1,], time, lwr[1,] )
> segments( time, upr[2,], time, lwr[2,])
> # Draw the x-axis at the times
> axis( 1, at=time )
> # Plot the availabe numbers below the x-axis
> mtext( available[1,], side=1, line=5, at=time)
> mtext( available[2,], side=1, line=6, at=time)


> par(mar=.1 + c(8,5,4,2))
> plot(time,means[1,], type="b", ylim=ylim, xaxt="n",
+                      ylab="% change in alkaline phosphatase",
+                      xlab="Months after randomization",
+                      las=1, # All axis labels horizontal
+                      pch=16, # Dot as plotting symbol
+                      bty="n", lwd=2 ) # No box around the plot
> segments(time ,  upr[1,], time , lwr[1,], lwd=2 )
> time2 <- time + 0.25 # Plot the points and bars a little bit offset
> points  (time2,means[2,], type="b", pch=18, lwd=2, cex=1.3)
> segments(time2,  upr[2,], time2, lwr[2,], lwd=2 )
> axis(1,at=time)
> mtext(available[1,],side=1, line=5, at=time)
> mtext(available[2,],side=1, line=6, at=time)
> # par("usr") reaturns the actual x- and y coordinates of the axis ends.
> mtext("Placebo"  , side=1, line=5, adj=1, at=par("usr")[1] )
> mtext("Tamoxifen", side=1, line=6, adj=1, at=par("usr")[1] )
> abline(h=0) # Horizontal line
```

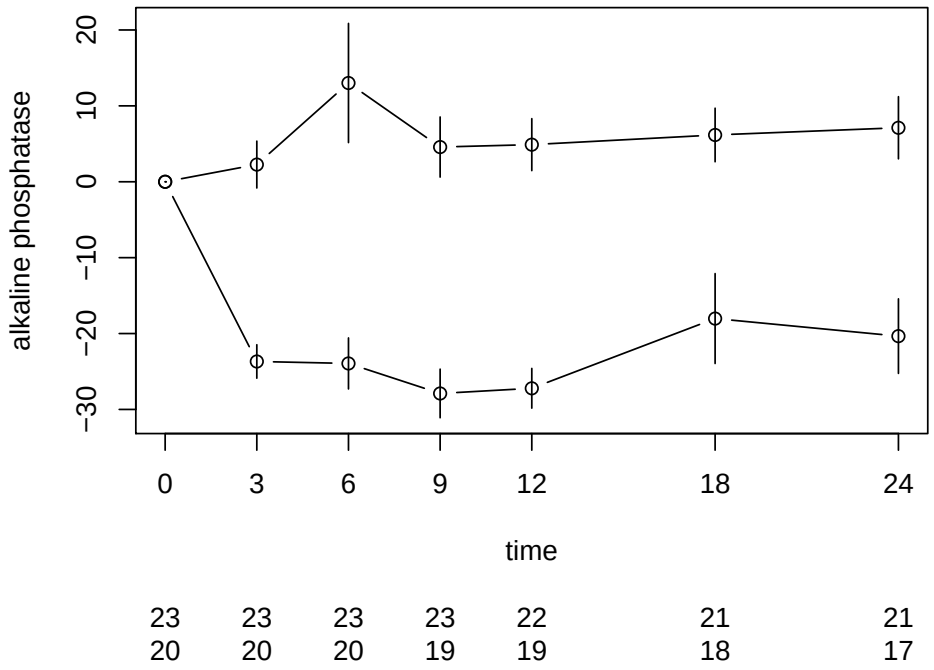


Figure 2.11: *The first shot at the graph.*

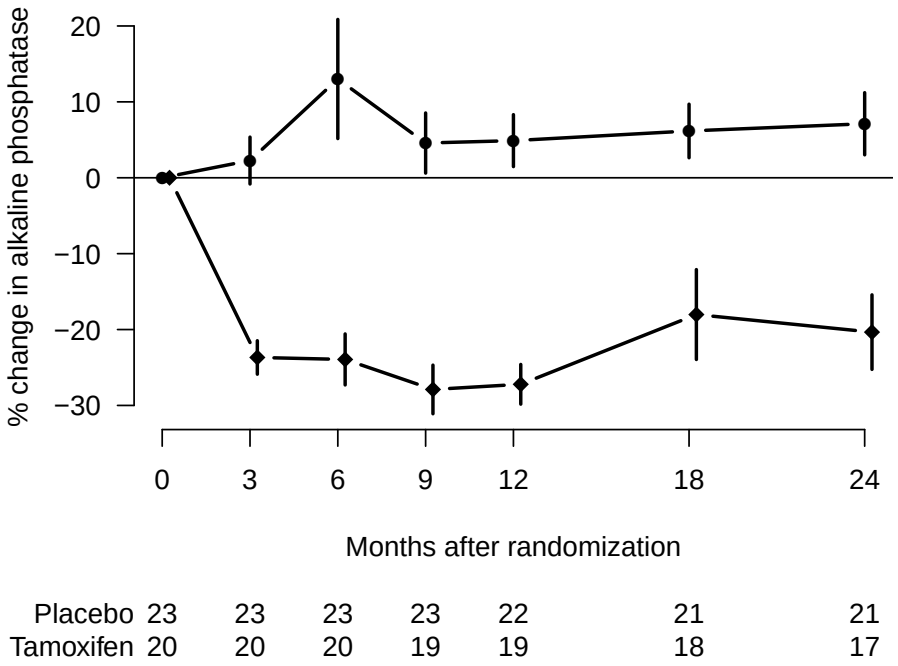


Figure 2.12: *The more elaborate version of the graph.*

2.11 Survival analysis: Oral cancer patients

2.11.1 Description of the data

File `oralca2.txt`, that you may access from a url address to be given in the practical, contains data from 338 patients having an oral squamous cell carcinoma diagnosed and treated in one tertiary level oncological clinic in Finland since 1985, followed-up for mortality until 31 December 2008. The dataset contains the following variables:

`sex` = sex, a factor with categories `1 = "Female", 2 = "Male"`,
`age` = age (years) at the date of diagnosing the cancer,
`stage` = TNM stage of the tumour (factor): `1 = "I", ..., 4 = "IV", 5 = "unkn"`,
`time` = follow-up time (in years) since diagnosis until death or censoring,
`event` = event ending the follow-up (numeric):
0 = censoring alive, 1 = death from oral cancer, 2 = death from other causes.

2.11.2 Loading the packages and the data

(a) Load the R packages `Epi, mstate`, and `survival` needed in this exercise.

(b) Read the datafile `oralca2.txt` from a website, whose precise address will be given in the practical, into an R data frame named `orca`. Look at the head, structure and the summary of the data frame. Using function `table()` count the numbers of censorings as well as deaths from oral cancer and other causes, respectively, from the `event` variable.

```
> orca <- read.table("./data/oralca2.txt", header=T)
> head(orca) ; str(orca) ; summary(orca)
```

```
     sex      age stage  time event
1   Male 65.42274  unkn 5.081     0
2 Female 83.08783   III 0.419     1
3   Male 52.59008    II 7.915     2
4   Male 77.08630     I 2.480     2
5   Male 80.33622    IV 2.500     1
6 Female 82.58132    IV 0.167     2
```

```
'data.frame':        338 obs. of  5 variables:
 $ sex  : Factor w/ 2 levels "Female","Male": 2 1 2 2 2 1 2 2 1 2 ...
 $ age  : num  65.4 83.1 52.6 77.1 80.3 ...
 $ stage: Factor w/ 5 levels "I","II","III",..: 5 3 2 1 4 4 2 5 4 2 ...
 $ time : num  5.081 0.419 7.915 2.48 2.5 ...
 $ event: int  0 1 2 2 1 2 0 1 1 0 ...
```

```
    sex           age         stage         time           event
 Female:152   Min.   :15.15   I   :50   Min.   : 0.085   Min.   :0.0000
 Male  :186   1st Qu.:53.24   II  :77   1st Qu.: 1.333   1st Qu.:0.0000
              Median :64.86   III :72   Median : 3.869   Median :1.0000
              Mean   :63.51   IV  :68   Mean   : 5.662   Mean   :0.9941
              3rd Qu.:74.29   unkn:71   3rd Qu.: 8.417   3rd Qu.:2.0000
              Max.   :92.24             Max.   :23.258   Max.   :2.0000
```

2.11.3 Total mortality: Kaplan–Meier analyses

(a) We start our analysis of total mortality pooling the two causes of death into a single outcome. First, construct a *survival object* `orca$suob` from the event variable and the follow-up time using function `Surv()`. Look at the structure and summary of `orca$suob` .

```
> orca$suob <- Surv(orca$time, 1*(orca$event > 0) )
> str(orca$suob)
```

```
 Surv [1:338, 1:2]  5.081+  0.419   7.915   2.480   2.500   0.167   5.925+  1.503  13.
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:2] "time" "status"
 - attr(*, "type")= chr "right"
```

```
> summary(orca$suob)
```

```
      time            status
 Min.   : 0.085   Min.   :0.0000
 1st Qu.: 1.333   1st Qu.:0.0000
 Median : 3.869   Median :1.0000
 Mean   : 5.662   Mean   :0.6775
 3rd Qu.: 8.417   3rd Qu.:1.0000
 Max.   :23.258   Max.   :1.0000
```

(b) Create a `survfit` object `s.all`, which does the default calculations for a Kaplan–Meier analysis of the overall (marginal) survival curve.

```
> s.all <- survfit(suob ~ 1, data=orca)
```

See the structure of this object and apply `print()` method on it, too. Look at the results; what do you find?

```
> s.all
```

```
Call: survfit(formula = suob ~ 1, data = orca)

records   n.max n.start  events  median 0.95LCL 0.95UCL
 338.00  338.00  338.00  229.00    5.42    4.33    6.92
```

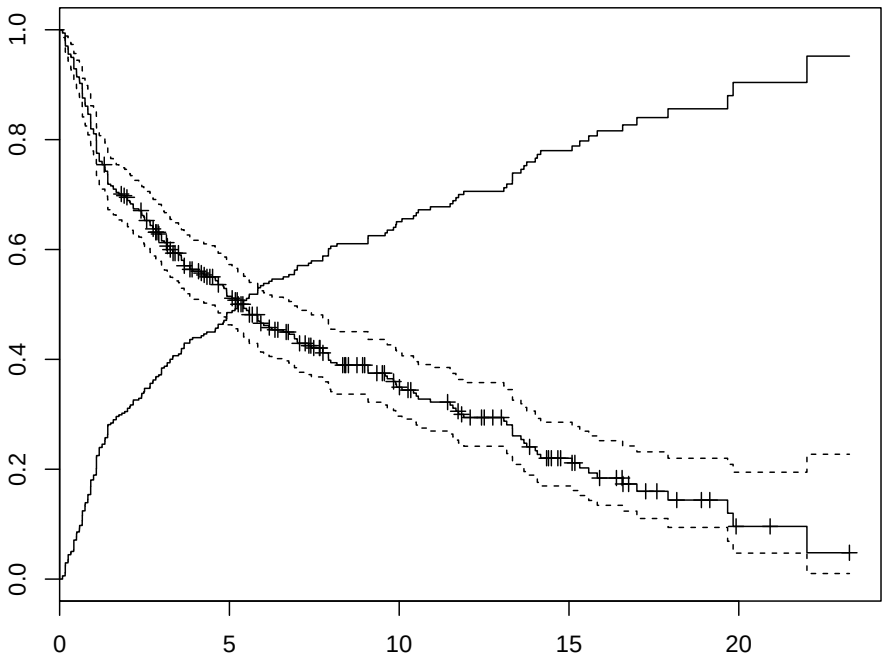
```
> str(s.all)
```

```
List of 13
 $ n        : int 338
 $ time     : num [1:251] 0.085 0.162 0.167 0.17 0.246 0.249 0.252 0.329 0.334 0.413 .
 $ n.risk   : num [1:251] 338 336 334 330 328 327 326 323 322 321 ...
 $ n.event  : num [1:251] 2 2 4 2 1 1 3 1 1 1 ...
 $ n.censor : num [1:251] 0 0 0 0 0 0 0 0 0 0 ...
 $ surv     : num [1:251] 0.994 0.988 0.976 0.97 0.967 ...
 $ type     : chr "right"
```

```
$ std.err  : num [1:251] 0.0042 0.00595 0.00847 0.0095 0.00998 ...
$ upper    : num [1:251] 1 1 0.993 0.989 0.987 ...
$ lower    : num [1:251] 0.986 0.977 0.96 0.953 0.949 ...
$ conf.type: chr "log"
$ conf.int : num 0.95
$ call     : language survfit(formula = suob ~ 1, data = orca)
- attr(*, "class")= chr "survfit"
```

(c) The `summary` method for a `survfit` object would return a lengthy life table. However, the `plot` method with default arguments offers the Kaplan–Meier curve for a conventional illustration of the survival experience in the whole patient group. Alternatively, instead of graphing survival proportions, one can draw a curve describing their complements: the cumulative mortality proportions. This curve is drawn together with the survival curve as the result of the second command line below.

```
> plot(s.all)
> lines(s.all, fun = "event", mark.time=F, conf.int=F)
```



The effect of option `mark.time=F` is to omit marking the times when censorings occurred.

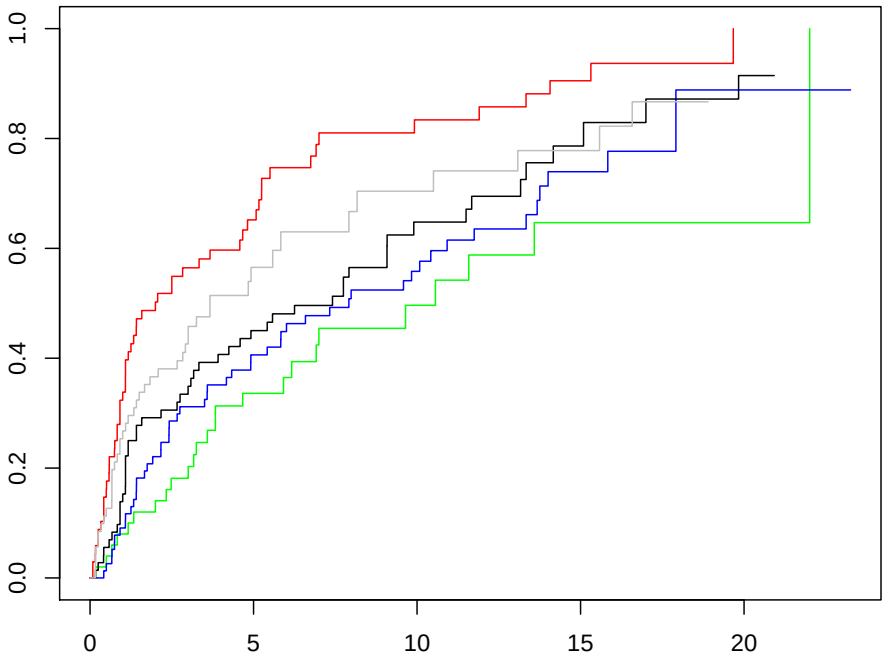
2.11.4 Total mortality by stage

Tumour stage is an important prognostic factor in cancer survival studies.

(a) Plot separate cumulative mortality curves for the different stage groups marking them with different colours, the order which you may define yourself. Also find the median survival time for each stage.

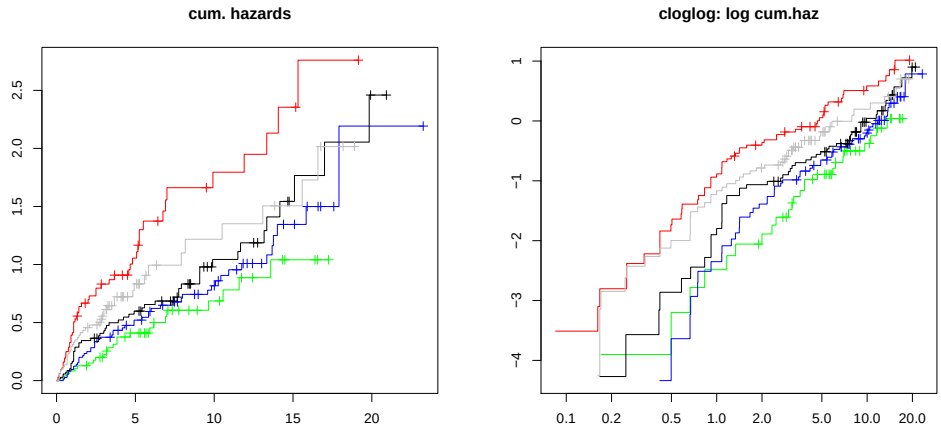
```
> s.stg  <- survfit(suob ~ stage, data= orca)
> col5 <- c("green", "blue", "black", "red", "gray")
> plot(s.stg, col= col5, fun="event", mark.time=F )
> s.stg
```

```
Call: survfit(formula = suob ~ stage, data = orca)

            records n.max n.start events median 0.95LCL 0.95UCL
stage=I          50    50      50     25  10.56    6.17      NA
stage=II         77    77      77     51   7.92    4.92   13.34
stage=III        72    72      72     51   7.41    3.92    9.90
stage=IV         68    68      68     57   2.00    1.08    4.82
stage=unkn       71    71      71     45   3.67    2.83    8.17
```



(b) Create now two parallel plots of which the first one describes the cumulative hazards and the second one graphs the log-cumulative hazards against log-time for the different stages. Compare the two presentations with each other and with the one in the previous item.

```
> par(mfrow=c(1,2))
> plot(s.stg, col= col5, fun="cumhaz", main="cum. hazards" )
> plot(s.stg, col= col5, fun="cloglog", main = "cloglog: log cum.haz"  )
```



(c) If the survival times were *exponentially* distributed in a given (sub)population the corresponding cloglog-curve should follow an approximately linear pattern. Could this be the case here in the different stages?

(d) Also, if the survival distributions of the different subpopulations would obey the *proportional hazards* model, the vertical distance between the cloglog-curves should be approximately constant over the time axis. Do these curves indicate serious deviation from the proportional hazards assumption?

(e) In the lecture handouts (p. 34, 37) it was observed that the crude contrast between males and females in total mortality appears unclear, but the age-adjustment in the Cox model provided a more expected hazard ratio estimate. We shall examine the confounding by age somewhat closer. First categorize the continuous age variable into, say, three categories by function `cut()` using suitable breakpoints, like 55 and 75 years, and cross-tabulate sex and age group:

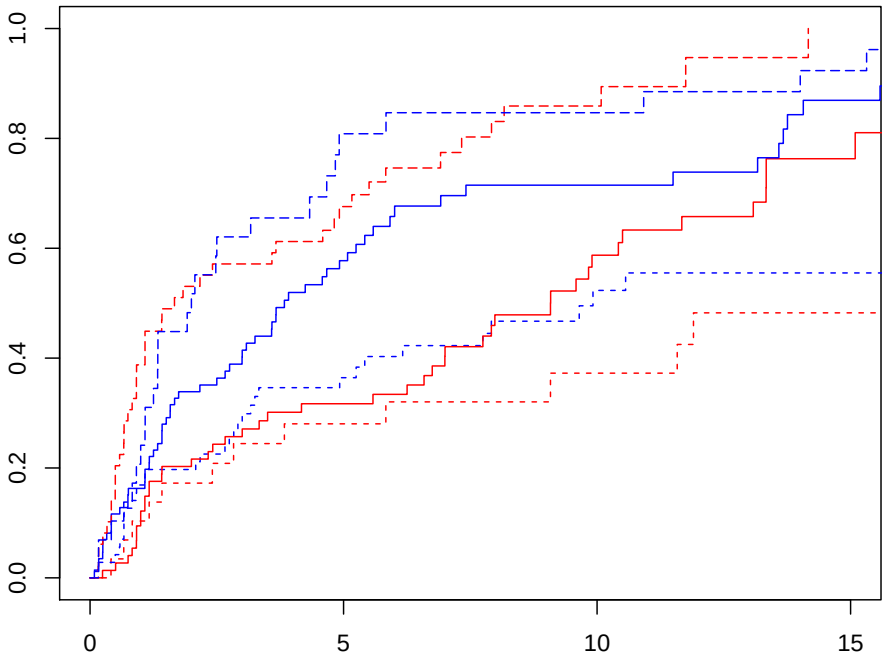
```
> orca$agegr <- cut(orca$age, br=c(0,55,75, 95))
> stat.table( list( sex, agegr), list( count(), percent(agegr) ),
+              margins=T, data = orca )
```

```
----------------------------------------
        -------------agegr--------------
sex       (0,55] (55,75] (75,95]   Total
----------------------------------------
Female        29      74      49     152
            19.1    48.7    32.2   100.0

Male          71      86      29     186
            38.2    46.2    15.6   100.0


Total        100     160      78     338
            29.6    47.3    23.1   100.0
----------------------------------------
```

Male patients are clearly younger than females in these data.

Now, plot Kaplan–Meier curves jointly classified by sex and age.

```
> s.agrx <- survfit(suob ~ agegr + sex, data=orca)
> par(mfrow=c(1,1))
> plot(s.agrx, fun="event", mark.time=F, xlim = c(0,15),
+              col=rep(c("red", "blue"),3), lty=c(2,2, 1,1, 5,5))
```



In each ageband the mortality curve for males is on a higher level than that for females.

2.11.5 Lexis object with multi-state set-up

Before entering to analyses of cause-specific mortality it might be instructive to apply some Lexis tools to illustrate the competing-risks set-up. More detailed explanation of these tools will be given by Bendix in this afternoon.

(a) Form a `Lexis` object from the data frame and print a summary of it. We shall name the main (and only) time axis in this object as `stime`.

```
> orca.lex <- Lexis(exit = list(stime = time),
+            exit.status = factor(event,
+    labels = c("Alive", "Oral ca. death", "Other death")),
+                  data = orca)

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

```
> summary(orca.lex)


Transitions:
     To
From    Alive Oral ca. death Other death  Records:  Events: Risk time:  Persons:
  Alive   109            122         107       338      229    1913.67       338
```

(b) Draw a box diagram of the two-state set-up of competing transitions. Run first the following command line

```
boxes( orca.lex )
```

Now, move the cursor to the point in the graphics window, at which you wish to put the box for "Alive", and click. Next, move the cursor to the point at which you wish to have the box for "Oral ca. death", and click. Finally, do the same with the box for "Other death". If you are not happy with the outcome, run the command line again and repeat the necessary mouse moves and clicks.

2.11.6 Event-specific cumulative mortality curves

We move on to analysing cumulative mortalities for the two causes of death separately, first overall and then by prognostic factors.

(a) Use function `Cuminc()` in package `mstate` and view the structure of the thus created object.

```
> cif1 <- Cuminc( time = "time", status= "event", data = orca)
> str(cif1)


'data.frame':        162 obs. of  7 variables:
 $ time  : num  0 0.085 0.162 0.167 0.17 0.246 0.249 0.252 0.329 0.334 ...
 $ Surv  : num  1 0.994 0.988 0.976 0.97 ...
 $ CI.1  : num  0 0.00592 0.01183 0.01775 0.02071 ...
 $ CI.2  : num  0 0 0 0.00592 0.00888 ...
 $ seSurv: num  0 0.00417 0.00588 0.00827 0.00922 ...
 $ seCI.1: num  0 0.00417 0.00588 0.00718 0.00775 ...
 $ seCI.2: num  0 0 0 0.00417 0.0051 ...
```

(b) Function `Cuminc()` thus creates an ordinary data frame with quite self-explanatory column names. Unfortunately, no handy `plot` method is provided in the package. A simple function `plotCIF2g()` is available in the same website and folder from which you also got the data set. Pick up the source file containing the function and see, what are its arguments.

```
> source("data/plotCIF2g.R")
> args(plotCIF2g)
```

```
function (data, cause = 1, ylab = "Cumulative incidence", xlab = "Time",
    ylim = c(0, 1), cex = 1, cex.lab = 1, cex.main = 1, cex.axis = 1,
    col = palette("default"), lty = rep(1, 8), lwd = 1, xlim = NULL,
    main = "CIF for event 1")
NULL
```

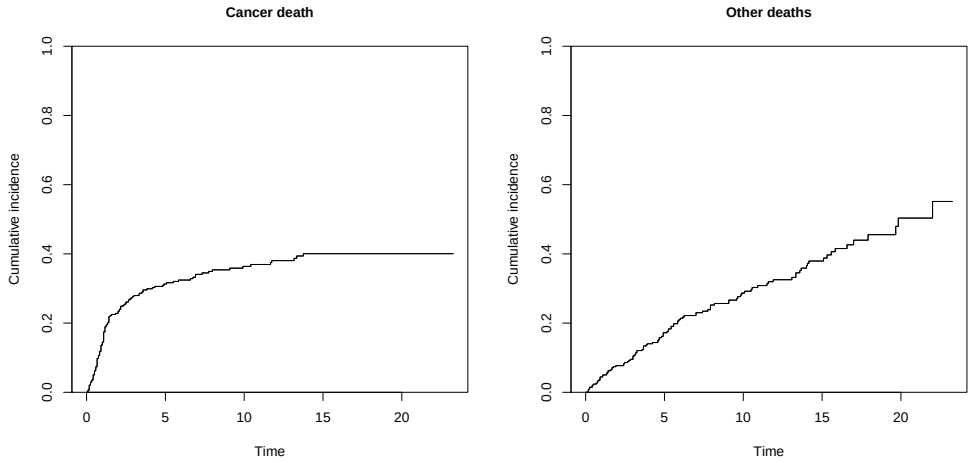
The main arguments are

`data`	=	data frame created by function `Cuminc()`,
`cause`	=	indicator for the event: values 1 or 2.

Other arguments are like in the ordinary `plot()` function.

(c) Draw two parallel plots describing the overall cumulative incidence curves for both causes of death

```
> par(mfrow=c(1,2))
> plotCIF2g(cif1, 1, main = "Cancer death")
> plotCIF2g(cif1, 2, main= "Other deaths")
```



Just ignore the warnings!

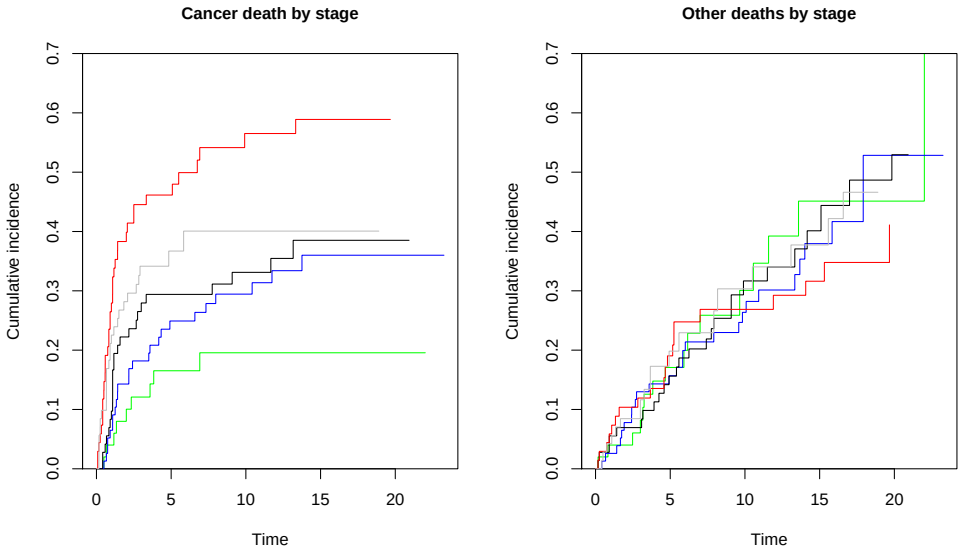
(d) Compute the estimated cumulative incidences by stage for both causes of death. Now you have to add argument `group` when calling `Cuminc()`. See the structure of the resulting object, in which you should observe the first column containing the grouping variable. Plot the pertinent curves in two parallel graphs. Cut the y-axis for a more efficient graphical presentation

```
> cif2 <- Cuminc( time = "time", status= "event",
+                 group = "stage", data = orca)
> str(cif2)
```

```
'data.frame':        211 obs. of  8 variables:
 $ stage : Factor w/ 5 levels "I","II","III",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ time  : num  0 0.17 0.498 0.665 0.832 ...
 $ Surv  : num  1 0.98 0.96 0.94 0.92 ...
 $ CI.1  : num  0 0 0.02 0.04 0.04 ...
```

```
 $ CI.2  : num  0 0.02 0.02 0.02 0.04 ...
 $ seSurv: num  0 0.0198 0.0277 0.0336 0.0384 ...
 $ seCI.1: num  0 0 0.0198 0.0277 0.0277 ...
 $ seCI.2: num  0 0.0198 0.0198 0.0198 0.0277 ...
```

```
> par(mfrow=c(1,2))
> plotCIF2g(cif2, 1, main = "Cancer death by stage",
+    col=col5, ylim = c(0, 0.7) )
> plotCIF2g(cif2, 2, main= "Other deaths by stage",
+    col=col5, ylim = c(0, 0.7) )
```



Compare the two plots. What would you conclude about the effect of stage on the two causes of death?

2.11.7 Regression modelling of overall mortality.

(a) Fit the semiparametric proportional hazards regression model, a.k.a. the Cox model, on all deaths including sex, age and stage as covariates. Use function `coxph()` in package `survival`. It is often useful to center and scale continuous covariates like `age` here. The estimated rate ratios and their confidence intervals can also here be displayed by applying `ci.lin()` on the fitted model object.

```
> options(show.signif.stars = F)
> m1 <- coxph(suob ~ sex + I((age-65)/10) + stage, data= orca)
> summary( m1 )
```

```
Call:
coxph(formula = suob ~ sex + I((age - 65)/10) + stage, data = orca)

  n= 338, number of events= 229

                 coef exp(coef) se(coef)     z Pr(>|z|)
sexMale       0.35139   1.42104  0.14139 2.485 0.012947
```

```
I((age - 65)/10) 0.41603   1.51593  0.05641 7.375 1.65e-13
stageII          0.03492   1.03554  0.24667 0.142 0.887421
stageIII         0.34545   1.41262  0.24568 1.406 0.159708
stageIV          0.88542   2.42399  0.24273 3.648 0.000265
stageunkn        0.58441   1.79393  0.25125 2.326 0.020016

                 exp(coef) exp(-coef) lower .95 upper .95
sexMale              1.421     0.7037    1.0771     1.875
I((age - 65)/10)     1.516     0.6597    1.3573     1.693
stageII              1.036     0.9657    0.6386     1.679
stageIII             1.413     0.7079    0.8728     2.286
stageIV              2.424     0.4125    1.5063     3.901
stageunkn            1.794     0.5574    1.0963     2.935

Concordance= 0.674  (se = 0.021 )
Rsquare= 0.226   (max possible= 0.999 )
Likelihood ratio test= 86.76  on 6 df,   p=1.11e-16
Wald test            = 80.5  on 6 df,   p=2.776e-15
Score (logrank) test = 82.86  on 6 df,   p=8.882e-16
```

```
> round( ci.exp(m1 ), 4 )
```

```
                 exp(Est.)   2.5%  97.5%
sexMale             1.4210 1.0771 1.8748
I((age - 65)/10)    1.5159 1.3573 1.6932
stageII             1.0355 0.6386 1.6793
stageIII            1.4126 0.8728 2.2864
stageIV             2.4240 1.5063 3.9007
stageunkn           1.7939 1.0963 2.9354
```

Look at the results. What are the main findings?

(b) Check whether the data are sufficiently consistent with the assumption of proportional hazards with respect to each of the variables separately as well as globally, using the `cox.zph()` function.

```
> cox.zph(m1)
```

```
                     rho    chisq     p
sexMale          -0.00137 0.000439 0.983
I((age - 65)/10)  0.07539 1.393597 0.238
stageII          -0.04208 0.411652 0.521
stageIII         -0.06915 1.083755 0.298
stageIV          -0.10044 2.301780 0.129
stageunkn        -0.09663 2.082042 0.149
GLOBAL                NA 4.895492 0.557
```

(c) No evidence against proportionality assumption could apparently be found. Moreover, no difference can be observed between stages I and II in the estimates. On the other hand, the group with stage unknown is a complex mixture of patients from various true stages. Therefore, it may be prudent to exclude these subjects from the data and to pool the first two stage groups into one. After that fit a model in the reduced data with the new stage variable.

```
> orca2 <- subset(orca, stage != "unkn")
> orca2$st3 <- Relevel( orca2$stage, list(1:2, 3, 4:5) )
> levels(orca2$st3) = c("I-II", "III", "IV")
> m2 <- update(m1, . ~ . - stage + st3, data=orca2 )
> round( ci.exp(m2 ), 4)
```

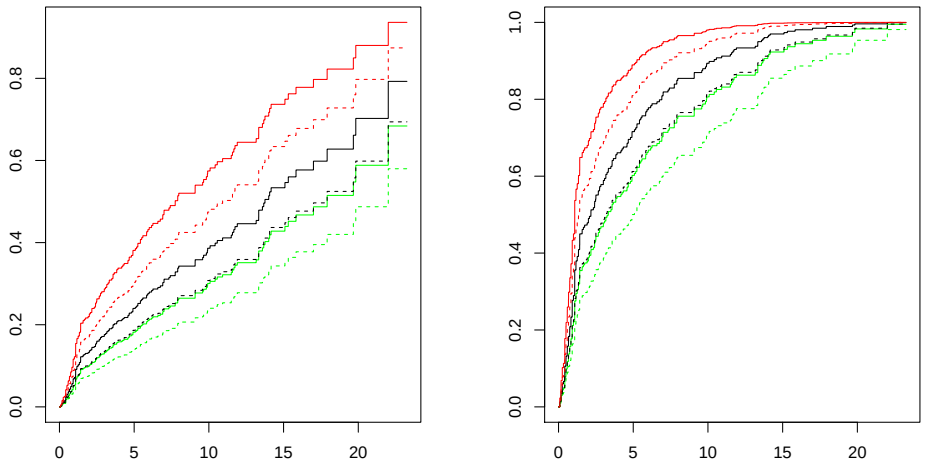
```
                 exp(Est.)   2.5%  97.5%
sexMale             1.3284 0.9763 1.8074
I((age - 65)/10)    1.4637 1.2959 1.6534
st3III              1.3657 0.9547 1.9536
st3IV               2.3900 1.6841 3.3919
```

(d) Plot the predicted cumulative mortality curves by stage, jointly stratified by sex and age, focusing only on 40 and 80 year old patients, respectively, based on the fitted model `m2`. You need to create a new artificial data frame containing the desired values for the covariates.

```
> newd <- data.frame( sex = c( rep("Male", 6), rep("Female", 6) ),
+                     age = rep( c( rep(40, 3), rep(80, 3) ), 2 ),
+                     st3 = rep( levels(orca2$st3), 4) )
> newd
```

```
      sex age  st3
1    Male  40 I-II
2    Male  40  III
3    Male  40   IV
4    Male  80 I-II
5    Male  80  III
6    Male  80   IV
7  Female  40 I-II
8  Female  40  III
9  Female  40   IV
10 Female  80 I-II
11 Female  80  III
12 Female  80   IV
```

```
> col3 <- c("green", "black", "red")
> par(mfrow=c(1,2))
> plot( survfit(m2, newdata= subset(newd, sex=="Male" & age==40)),
+      col=col3, fun="event", mark.time=F)
> lines( survfit(m2, newdata= subset(newd, sex=="Female" & age==40)),
+       col= col3, fun="event", lty = 2, mark.time=F)
> plot( survfit(m2, newdata= subset(newd, sex=="Male" & age==80)),
+      ylim = c(0,1), col= col3, fun="event", mark.time=F)
> lines( survfit(m2, newdata= subset(newd, sex=="Female" & age==80)),
+       col=col3, fun="event", lty=2, mark.time=F)
>
```



2.11.8 Modelling event-specific hazards and hazards of the subdistribution

(a) Fit the Cox model for the cause-specific hazard of cancer deaths with the same covariates as above. In this case only cancer deaths are counted as events and deaths from other causes are included into censorings.

```
> m2haz1 <- coxph( Surv( time, event==1)  ~ sex + I((age-65)/10) + st3 , data=orca2 )
> round( ci.exp(m2haz1 ), 4)
```

```
                 exp(Est.)   2.5%  97.5%
sexMale             1.0171 0.6644 1.5569
I((age - 65)/10)    1.4261 1.2038 1.6893
st3III              1.5140 0.9012 2.5434
st3IV               3.1813 1.9853 5.0978
```

```
> cox.zph(m2haz1)
```

```
                     rho chisq      p
sexMale           0.0651 0.405 0.5246
I((age - 65)/10)  0.2355 6.001 0.0143
st3III           -0.1120 1.174 0.2785
st3IV            -0.1858 3.163 0.0753
GLOBAL                NA 9.352 0.0529
```

Compare the results with those of model `m2`. What are the major differences?

(b) Fit a similar model for deaths from other causes and compare the results.

```
                 exp(Est.)   2.5%  97.5%
sexMale             1.8103 1.1528 2.8431
I((age - 65)/10)    1.4876 1.2491 1.7715
st3III              1.2300 0.7488 2.0206
st3IV               1.6407 0.9522 2.8270
```

```
                      rho   chisq     p
sexMale          -0.04954 0.22019 0.639
I((age - 65)/10)  0.03484 0.10107 0.751
st3III            0.01369 0.01639 0.898
st3IV            -0.00411 0.00149 0.969
GLOBAL                NA 0.45596 0.978
```

(c) Finally, fit the Fine–Gray model for the hazard of the subdistribution for cancer deaths with the same covariates as above. For this you have to first load package `cmprsk`, containing the necessary function `crr()`, and attach the data frame.

```
> library(cmprsk)
> attach(orca2)
> m2fg1 <- crr(time, event, cov1 = model.matrix(m2), failcode=1)
> summary(m2fg1, Exp=T)

Competing Risks Regression

Call:
crr(ftime = time, fstatus = event, cov1 = model.matrix(m2), failcode = 1)

                    coef exp(coef) se(coef)      z p-value
sexMale          -0.0808     0.922   0.2118 -0.381 7.0e-01
I((age - 65)/10)  0.2791     1.322   0.0918  3.039 2.4e-03
st3III            0.3739     1.453   0.2588  1.445 1.5e-01
st3IV             1.0346     2.814   0.2327  4.446 8.8e-06

                 exp(coef) exp(-coef)  2.5% 97.5%
sexMale              0.922      1.084 0.609  1.40
I((age - 65)/10)     1.322      0.756 1.104  1.58
st3III               1.453      0.688 0.875  2.41
st3IV                2.814      0.355 1.783  4.44

Num. cases = 267
Pseudo Log-likelihood = -493
Pseudo likelihood ratio test = 32.1  on 4 df,
```

Compare the results with those of model `m2` and `m2haz1`.

(d) Fit a similar model for deaths from other causes and compare the results.

```
> m2fg2 <- crr(time, event, cov1 = model.matrix(m2), failcode=2)
> summary(m2fg2, Exp=T)

Competing Risks Regression

Call:
crr(ftime = time, fstatus = event, cov1 = model.matrix(m2), failcode = 2)

                    coef exp(coef) se(coef)      z p-value
sexMale            0.558     1.748   0.2264  2.467   0.014
I((age - 65)/10)   0.187     1.205   0.0775  2.412   0.016
st3III             0.086     1.090   0.2428  0.354   0.720
st3IV             -0.225     0.799   0.2795 -0.803   0.420
```

```
                 exp(coef) exp(-coef)  2.5% 97.5%
sexMale              1.748      0.572 1.122  2.72
I((age - 65)/10)     1.205      0.830 1.036  1.40
st3III               1.090      0.918 0.677  1.75
st3IV                0.799      1.252 0.462  1.38

Num. cases = 267
Pseudo Log-likelihood = -438
Pseudo likelihood ratio test = 9.43  on 4 df,
```

2.11.9 Poisson regression as an alternative to Cox model

It can be shown that the Cox model with an unspecified form for the baseline hazard $\lambda_0(t)$ is mathematically equivalent to the following kind of Poisson regression model. Time is treated as a categorical factor with a dense division of the time axis into disjoint intervals or *timebands* such that only one outcome event occurs in each timeband. The model formula contains this time factor plus the desired explanatory terms.

A sufficient division of time axis is obtained by first setting the break points between adjacent timebands to be those time points at which an outcome event has been observed to occur. Then, the pertinent `lexis` object is created and after that it will be split according to those breakpoints. Finally, the Poisson regression model is fitted on the splitted `lexis` object using function `glm()` with appropriate specifications.

We shall now demonstrate the numerical equivalence of the Cox model `m2haz1` for oral cancer mortality that was fitted above, and the corresponding Poisson regression.

(a) First we form the necessary `lexis` object by just taking the relevant subset of the already available `orca.lex` object. Upon that the three-level stage factor `st3` is created as above.

```
> orca2.lex <- subset(orca.lex, stage != 'unkn' )
> orca2.lex$st3 <- Relevel( orca2$stage, list(1:2, 3, 4:5) )
> levels(orca2.lex$st3) = c("I-II", "III", "IV")
```

Then, the break points of time axis are taken from the sorted event times, and the `lexis` object is split by those breakpoints. The `timeband` factor is defined according to the splitted survival times stored in variable `stime`.

```
> cuts <- sort(orca2$time[orca2$event==1])
> orca2.spl <- splitLexis( orca2.lex, br = cuts, time.scale="stime" )
> orca2.spl$timeband <- as.factor(orca2.spl$stime)
```

As a result we now have an expanded `lexis` object in which each subject has several rows; as many rows as there are such timebands during which he/she is still at risk. The outcome status `lex.Xst` has value 0 in all those timebands, over which the subject stays alive, but assumes the value 1 or 2 at his/her last interval ending at the time of death. – See now the structure of the splitted object.

```
> str(orca2.spl)
```

```
Classes 'Lexis' and 'data.frame':        12637 obs. of  14 variables:
 $ lex.id  : int  2 2 2 2 2 2 3 3 3 3 ...
 $ stime   : num  0 0.085 0.162 0.252 0.329 0.413 0 0.085 0.162 0.252 ...
 $ lex.dur : num  0.085 0.077 0.09 0.077 0.084 0.006 0.085 0.077 0.09 0.077 ...
 $ lex.Cst : Factor w/ 3 levels "Alive","Oral ca. death",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ lex.Xst : Factor w/ 3 levels "Alive","Oral ca. death",..: 1 1 1 1 1 2 1 1 1 1 ...
 $ sex     : Factor w/ 2 levels "Female","Male": 1 1 1 1 1 1 2 2 2 2 ...
 $ age     : num  83.1 83.1 83.1 83.1 83.1 ...
 $ stage   : Factor w/ 5 levels "I","II","III",..: 3 3 3 3 3 3 2 2 2 2 ...
 $ time    : num  0.419 0.419 0.419 0.419 0.419 ...
 $ event   : int  1 1 1 1 1 1 2 2 2 2 ...
 $ suob    : Surv [1:12637, 1:2]  0.419   0.419   0.419   0.419   0.419   0.419   7.91
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr  "time" "status"
  ..- attr(*, "type")= chr "right"
 $ agegr   : Factor w/ 3 levels "(0,55]","(55,75]",..: 3 3 3 3 3 3 1 1 1 1 ...
 $ st3     : Factor w/ 3 levels "I-II","III","IV": 2 2 2 2 2 2 1 1 1 1 ...
 $ timeband: Factor w/ 72 levels "0","0.085","0.162",..: 1 2 3 4 5 6 1 2 3 4 ...
 - attr(*, "breaks")=List of 1
  ..$ stime: num  0.085 0.162 0.252 0.329 0.413 0.419 0.496 0.498 0.504 0.58 ...
 - attr(*, "time.scales")= chr "stime"
 - attr(*, "time.since")= chr ""
```

```
> orca2.spl[ 1:20, ]
```

```
   lex.id stime lex.dur lex.Cst        lex.Xst    sex      age stage  time event   suo
1       2 0.000   0.085   Alive          Alive Female 83.08783   III 0.419     1 0.419
2       2 0.085   0.077   Alive          Alive Female 83.08783   III 0.419     1 0.419
3       2 0.162   0.090   Alive          Alive Female 83.08783   III 0.419     1 0.419
4       2 0.252   0.077   Alive          Alive Female 83.08783   III 0.419     1 0.419
5       2 0.329   0.084   Alive          Alive Female 83.08783   III 0.419     1 0.419
6       2 0.413   0.006   Alive Oral ca. death Female 83.08783   III 0.419     1 0.419
7       3 0.000   0.085   Alive          Alive   Male 52.59008    II 7.915     2 7.915
8       3 0.085   0.077   Alive          Alive   Male 52.59008    II 7.915     2 7.915
9       3 0.162   0.090   Alive          Alive   Male 52.59008    II 7.915     2 7.915
10      3 0.252   0.077   Alive          Alive   Male 52.59008    II 7.915     2 7.915
11      3 0.329   0.084   Alive          Alive   Male 52.59008    II 7.915     2 7.915
12      3 0.413   0.006   Alive          Alive   Male 52.59008    II 7.915     2 7.915
13      3 0.419   0.077   Alive          Alive   Male 52.59008    II 7.915     2 7.915
14      3 0.496   0.002   Alive          Alive   Male 52.59008    II 7.915     2 7.915
15      3 0.498   0.006   Alive          Alive   Male 52.59008    II 7.915     2 7.915
16      3 0.504   0.076   Alive          Alive   Male 52.59008    II 7.915     2 7.915
17      3 0.580   0.003   Alive          Alive   Male 52.59008    II 7.915     2 7.915
18      3 0.583   0.003   Alive          Alive   Male 52.59008    II 7.915     2 7.915
19      3 0.586   0.003   Alive          Alive   Male 52.59008    II 7.915     2 7.915
20      3 0.589   0.076   Alive          Alive   Male 52.59008    II 7.915     2 7.915
    st3 timeband
1   III        0
2   III    0.085
3   III    0.162
4   III    0.252
5   III    0.329
6   III    0.413
```

```
7  I-II       0
8  I-II   0.085
9  I-II   0.162
10 I-II   0.252
11 I-II   0.329
12 I-II   0.413
13 I-II   0.419
14 I-II   0.496
15 I-II   0.498
16 I-II   0.504
17 I-II    0.58
18 I-II   0.583
19 I-II   0.586
20 I-II   0.589
```

(b) We are ready to fit the desired Poisson model for oral cancer death as the outcome. The splitted person-years are contained in `lex.dur`, and the explanatory variables are the same as in model `m2haz1`. – This fitting may take some time ...

```
> m2pois1 <- glm( 1*(lex.Xst=="Oral ca. death")  ~
+       -1 + timeband + sex + I((age-65)/10) + st3,
+       family=poisson, offset = log(lex.dur), data = orca2.spl)
```

We shall display the estimation results graphically for the baseline hazard (per 1000 person-years) and numerically for the rate ratios associated with the covariates. Before doing that it is useful to count the length `ntb` of the block occupied by baseline hazard in the whole vector of estimated parameters. However, owing to how the splitting to timebands was done, the last regression coefficient is necessarily zero and better be omitted when displaying the results. Also, as each timeband is quantitatively named accoding to its leftmost point, it is good to compute the midpoint values `tbmid` for the timebands

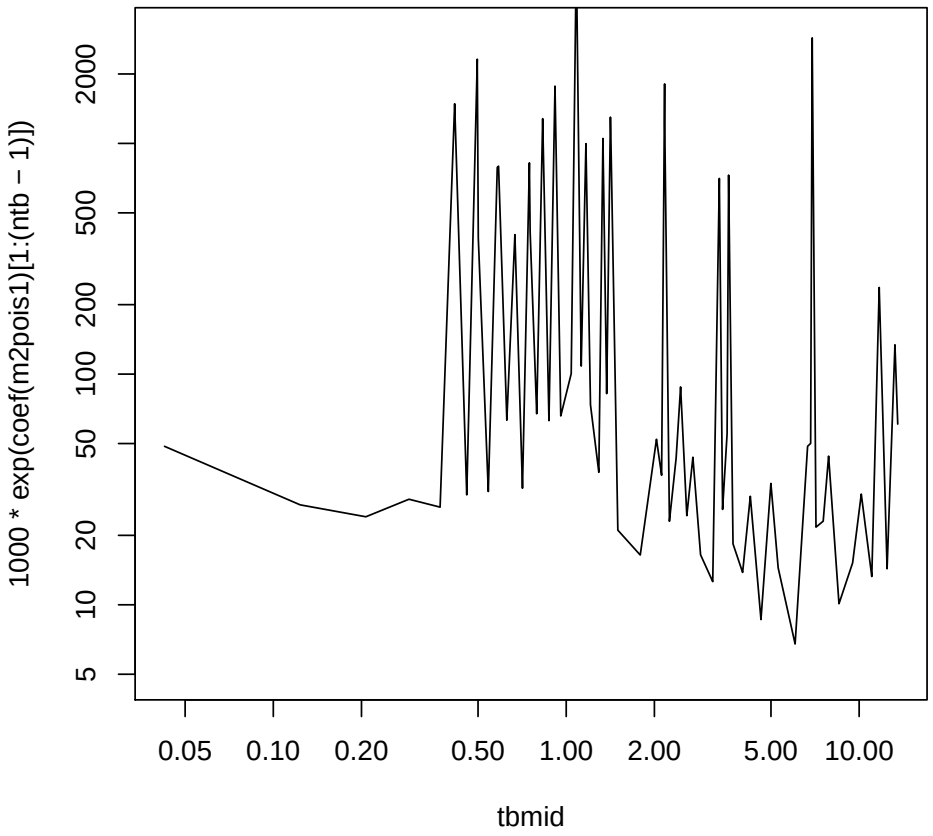
```
> tb <- as.numeric(levels(orca2.spl$timeband)) ; ntb <- length(tb)
> tbmid <- (tb[-ntb] + tb[-1])/2   # midpoints of the intervals
> round( ci.exp(m2pois1 ), 3)
```

```
                exp(Est.)  2.5%  97.5%
timeband0           0.049 0.012  0.205
timeband0.085       0.027 0.004  0.200
timeband0.162       0.024 0.003  0.177
timeband0.252       0.029 0.004  0.211
timeband0.329       0.027 0.004  0.195
timeband0.413       1.486 0.521  4.239
timeband0.419       0.030 0.004  0.220
timeband0.496       2.317 0.552  9.724
timeband0.498       0.390 0.053  2.865
timeband0.504       0.031 0.004  0.228
timeband0.58        0.787 0.107  5.785
timeband0.583       0.792 0.108  5.821
timeband0.586       0.797 0.108  5.856
timeband0.589       0.063 0.015  0.265
timeband0.665       0.402 0.055  2.957
timeband0.671       0.032 0.004  0.235
```

```
timeband0.747        0.824 0.112  6.052
timeband0.75         0.413 0.056  3.033
timeband0.756        0.067 0.016  0.283
timeband0.83         1.281 0.174  9.408
timeband0.832        0.063 0.015  0.264
timeband0.914        1.772 0.423  7.425
timeband0.917        0.066 0.016  0.276
timeband0.999        0.100 0.031  0.329
timeband1.081        6.554 2.874 14.946
timeband1.084        0.108 0.033  0.355
timeband1.166        0.998 0.136  7.311
timeband1.169        0.074 0.018  0.308
timeband1.251        0.038 0.005  0.275
timeband1.333        1.051 0.144  7.687
timeband1.336        0.082 0.020  0.343
timeband1.413        1.300 0.312  5.421
timeband1.418        1.113 0.152  8.146
timeband1.421        0.021 0.003  0.154
timeband1.58         0.016 0.004  0.069
timeband1.999        0.052 0.007  0.382
timeband2.067        0.036 0.005  0.266
timeband2.166        1.811 0.248 13.237
timeband2.168        1.216 0.166  8.891
timeband2.171        0.023 0.003  0.168
timeband2.33         0.043 0.006  0.317
timeband2.415        0.088 0.021  0.367
timeband2.5          0.024 0.003  0.178
timeband2.661        0.044 0.006  0.318
timeband2.752        0.016 0.002  0.120
timeband2.998        0.013 0.002  0.092
timeband3.329        0.705 0.097  5.148
timeband3.335        0.026 0.004  0.189
timeband3.502        0.055 0.008  0.402
timeband3.581        0.729 0.100  5.323
timeband3.587        0.018 0.003  0.134
timeband3.833        0.014 0.002  0.101
timeband4.17         0.030 0.004  0.215
timeband4.331        0.009 0.001  0.063
timeband4.914        0.034 0.005  0.245
timeband5.079        0.014 0.002  0.105
timeband5.503        0.007 0.001  0.049
timeband6.587        0.049 0.007  0.354
timeband6.749        0.050 0.007  0.364
timeband6.913        2.867 0.394 20.860
timeband6.916        0.022 0.003  0.158
timeband7.329        0.023 0.003  0.168
timeband7.748        0.044 0.006  0.321
timeband7.984        0.010 0.001  0.074
timeband9.084        0.015 0.002  0.111
timeband9.919        0.030 0.004  0.220
timeband10.42        0.013 0.002  0.097
timeband11.671       0.237 0.033  1.734
timeband11.748       0.014 0.002  0.105
timeband13.166       0.134 0.018  0.979
timeband13.333       0.061 0.008  0.445
```

```
timeband13.755       0.000 0.000    Inf
sexMale              1.015 0.663  1.554
I((age - 65)/10)     1.423 1.201  1.685
st3III               1.509 0.898  2.535
st3IV                3.178 1.983  5.093
```

```
> par(mfrow=c(1,1))
> plot( tbmid, 1000*exp(coef(m2pois1)[1:(ntb-1)]),
+              ylim=c(5,3000), log = 'xy', type = 'l')
```



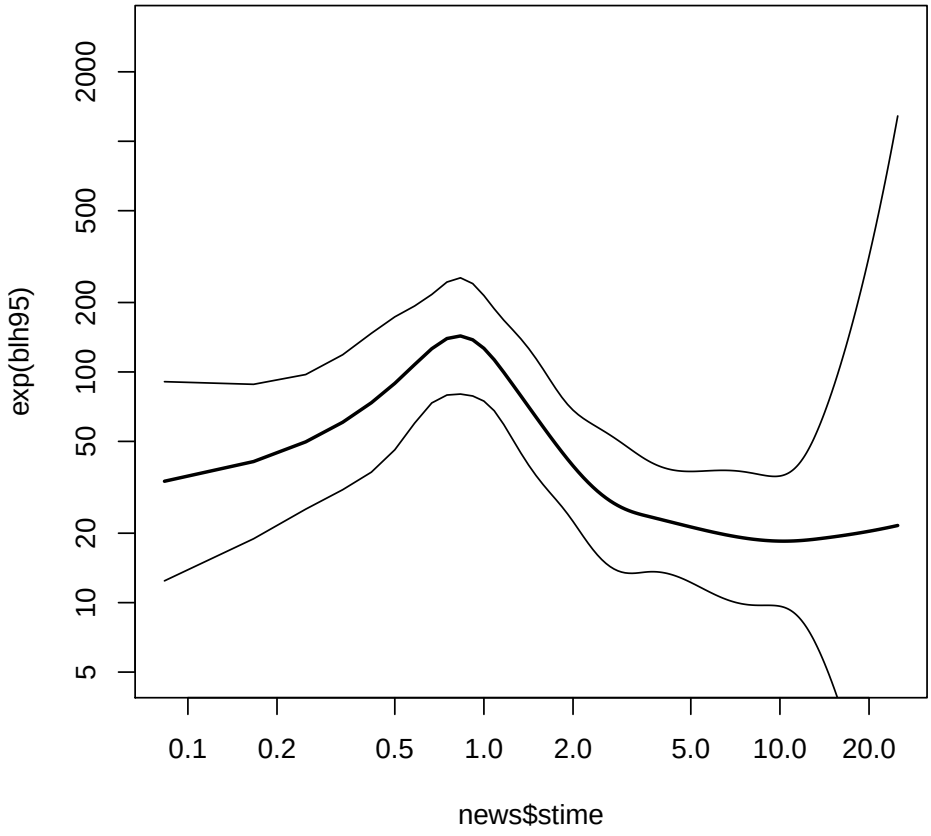
Compare the regression coefficients and their error margins to those model `m2haz1`. Do you find any differences? How does the estimated baseline hazard look like?

(c) The estimated baseline looks quite ragged when based on 71 separate parameters. A smoothed estimate may be obtained by spline modelling using the tools contained in package `splines` (see the practical of Saturday 25 May afternoon). With the following code you will be able to fit a reasonable spline model for the baseline hazard and draw the estimated curve (together with a band of the 95% confidence limits about the fitted values). From the same model you should also obtain quite familiar results for the rate ratios of interest.

```
> library(splines)
> m2pspli <- update(m2pois1, . ~  ns(stime, df = 6, intercept = F) +
+         sex + I((age-65)/10) + st3)
> round( ci.exp( m2pspli ), 3)
```

```
                                  exp(Est.)  2.5%  97.5%
(Intercept)                           0.028 0.008  0.101
ns(stime, df = 6, intercept = F)1     6.505 1.776 23.823
ns(stime, df = 6, intercept = F)2     2.678 0.560 12.803
ns(stime, df = 6, intercept = F)3     0.976 0.227  4.187
ns(stime, df = 6, intercept = F)4     0.423 0.105  1.699
ns(stime, df = 6, intercept = F)5     1.567 0.082 29.939
ns(stime, df = 6, intercept = F)6     0.434 0.121  1.558
sexMale                               1.021 0.667  1.563
I((age - 65)/10)                      1.431 1.208  1.696
st3III                                1.514 0.901  2.543
st3IV                                 3.185 1.988  5.104
```

```
> news <- data.frame( stime = seq(0,25, length=301), lex.dur = 1000, sex = 'Female',
+                     age = 65, st3 = 'I-II')
> blhaz <- predict(m2pspli, newdata = news, se.fit = T, type = 'link')
> blh95 <- cbind(blhaz$fit, blhaz$se.fit) %*% ci.mat()
> par(mfrow=c(1,1))
> matplot( news$stime, exp(blh95), type = 'l', lty = c(1,1,1), lwd = c(2,1,1) ,
+       col = rep('black', 3),  log = 'xy', ylim = c(5,3000)  )
```



2.11.10 Analysis of relative survival

(a) Load package `bigEpi` for the estimation of relative survival. Use the (simulated) female Finnish breast cancer patients diagnosed between 1993-2012, called `sibr`.

```
> library(bigEpi)

Changes in version 0.1.9:

    o   ltable bugfix

    o   survtab: lifetable survival now works when all persons in an interval are cens

> head(sibr)

   sex    bi_date    dg_date    ex_date status   dg_age
1:   1 1932-08-20 2009-06-12 2012-12-31      0 76.80996
2:   1 1950-02-19 2002-03-08 2012-12-31      0 52.04658
3:   1 1915-06-11 2002-05-26 2003-01-30      1 86.95616
4:   1 1936-02-11 2012-10-12 2012-12-31      0 76.66667
5:   1 1934-12-05 1993-06-21 2012-12-31      0 58.54247
6:   1 1956-01-09 2002-08-15 2012-12-31      0 46.59732
```

(b) Prepare the data by using `lex` command in the `bigEpi` package, define follow-up time intervals, (calendar time) period that you are interested and where are the population mortality figures. Calculate 5-year observed survival (2008-2012) using period method by Ederer II (default)

```
> x <- lex(sibr,
+          fot.breaks=seq(0, 5, by = 1),
+          per.breaks = c("2008-01-01", "2013-01-01"),
+          status.var="status", pophaz = popmort)
```

```
WARNING: dropped 1 rows where dg_date == ex_date
NOTE: 36  rows in expanded data had age values >= 101 ; assumed for these the same exp
```

```
> st <- survtab(x,surv.type = "surv.obs")
> st
```

```
   surv.int delta pyrs n.start   d n.cens surv.obs.lo surv.obs surv.obs.hi SE.surv.obs
1:   [0, 1[     1 2869    2941 125    568      0.9494   0.9574      0.9641    0.003730
2:   [1, 2[     1 2705    2806 102    567      0.9116   0.9219      0.9311    0.004976
3:   [2, 3[     1 2611    2658  88    522      0.8795   0.8914      0.9022    0.005779
4:   [3, 4[     1 2486    2548  77    512      0.8512   0.8642      0.8762    0.006380
5:   [4, 5[     1 2301    2417  81    516      0.8201   0.8343      0.8475    0.006970
```

(c) Calculate 5-year relative survival (2008-2012) using period method by Ederer II (default)

```
> x <- lex(sibr,
+          fot.breaks=seq(0, 5, by = 1),
+          per.breaks = c("2008-01-01", "2013-01-01"),
+          status.var="status", pophaz = popmort)
```

```
WARNING: dropped 1 rows where dg_date == ex_date
NOTE: 36  rows in expanded data had age values >= 101 ; assumed for these the same exp
```

```
> st <- survtab(x,surv.type = "surv.rel")
> st
```

```
   surv.int delta pyrs n.start   d n.cens d.exp surv.obs.lo surv.obs surv.obs.hi SE.su
1:   [0, 1[     1 2869    2941 125    568 50.06      0.9494   0.9574      0.9641    0.
2:   [1, 2[     1 2705    2806 102    567 45.58      0.9116   0.9219      0.9311    0.
3:   [2, 3[     1 2611    2658  88    522 44.91      0.8795   0.8914      0.9022    0.
4:   [3, 4[     1 2486    2548  77    512 43.09      0.8512   0.8642      0.8762    0.
5:   [4, 5[     1 2301    2417  81    516 40.00      0.8201   0.8343      0.8475    0.
   r.e2.lo   r.e2 r.e2.hi  SE.r.e2
1:  0.9656 0.9742  0.9807 0.003796
2:  0.9429 0.9541  0.9632 0.005149
3:  0.9254 0.9385  0.9494 0.006085
4:  0.9112 0.9258  0.9381 0.006834
5:  0.8933 0.9094  0.9232 0.007597
```

2.12 Time-splitting, time-scales and SMR: Diabetes in Denmark

This exercise is 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.

The exercise is about assessing how mortality depends age, calendar time and duration of diabetes. And how to understand and compute SMR, and assess how it depends on these factors as well.

1. 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          M
 M:5185   Min.   :1898   Min.   :1995   Min.   :1995   Min.   :1995   Min.   :1995   M
 F:4815   1st Qu.:1930   1st Qu.:2000   1st Qu.:2002   1st Qu.:2001   1st Qu.:2001   1
          Median :1941   Median :2004   Median :2005   Median :2004   Median :2005   M
          Mean   :1942   Mean   :2003   Mean   :2005   Mean   :2004   Mean   :2004   M
          3rd Qu.:1951   3rd Qu.:2007   3rd Qu.:2008   3rd Qu.:2007   3rd Qu.:2007   3
          Max.   :2008   Max.   :2010   Max.   :2010   Max.   :2010   Max.   :2010   M
                                        NA's   :7497   NA's   :4503   NA's   :8209
```

2. 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.

   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
```

3. 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 surprising, we see that men have a higher mortality than women.

4. 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), time.scale="A" )
> summary( SL )

Transitions:
     To
From    Alive Dead  Records:  Events: Risk time:  Persons:
  Alive 61627 2499     64126     2499   54273.27      9996
```

5. 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

```
> nd <-  data.frame( A = seq(10,90,0.5),
+              lex.dur = 1000)
> p.m <- predict.glm( r.m, type = "link",
+                        newdata = nd,
+                         se.fit = TRUE )
> p.f <- predict.glm( r.f, type = "link",
+                        newdata = nd,
+                         se.fit = TRUE )
> str( p.m )

List of 3
 $ fit           : Named num [1:161] -0.149 -0.1233 -0.0975 -0.0716 -0.0455 ...
  ..- attr(*, "names")= chr [1:161] "1" "2" "3" "4" ...
 $ se.fit        : Named num [1:161] 1.45 1.41 1.37 1.33 1.29 ...
  ..- attr(*, "names")= chr [1:161] "1" "2" "3" "4" ...
 $ residual.scale: num 1
```

   From the structure of the predicted rates (`p.m`) we can construct the predicted rates with confidence intervals by using matrix multiplication and the `ci.mat` function:

```
> ci.mat()

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

```
> lr.m <- cbind(p.m$fit,p.m$se.fit) %*% ci.mat()
> lr.f <- cbind(p.f$fit,p.f$se.fit) %*% ci.mat()
```

. . . and finally we can plot the two sets of estimated rates:

```
> matplot( seq(10,90,0.5), exp( cbind(lr.m,lr.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" )
```

Graphical comparison with the population rates

6. 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
```

7. 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 <- predict.glm( R.m, type = "link", newdata = nd )
> P.f <- predict.glm( R.f, type = "link", 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),
+          exp( cbind(lr.m,lr.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), exp(cbind( P.m,P.f )), lty="12",
+           col=c("blue","red"), lwd=3 )
```

Period and duration effects

8. 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.

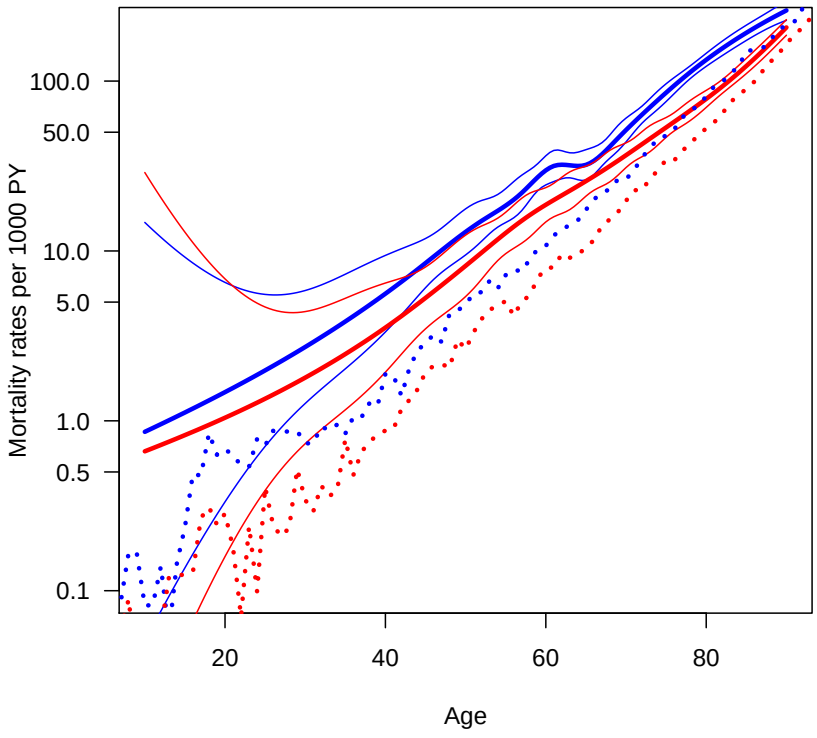


Figure 2.13: *Age-specific mortality rates for Danish diabetes patients as estimated from a model with only age. Broken lines are empirical rates from 2005. Blue: men, red: women.*

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 ) )
> kn.P <- with( subset( SL, lex.Xst=="Dead" ),
+               quantile( P+lex.dur, probs=seq(5,95,30)/100 ) )
> kn.dur <- c(0,with( subset( SL, lex.Xst=="Dead" ),
+               quantile( dur+lex.dur, probs=seq(5,95,10)/100 ) ))
```

9. 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[-c(1,length(kn.A))],
+                                    Bo=kn.A[ c(1,length(kn.A))] ) +
+                                ns( P, kn=kn.P[-c(1,length(kn.P))],
+                                    Bo=kn.P[ c(1,length(kn.P))] ) +
+                              ns( dur, kn=kn.dur[-c(1,length(kn.dur))],
+                                    Bo=kn.dur[ c(1,length(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[-c(1, length(kn.A))],
    Bo = kn.A[c(1, length(kn.A))]) + ns(P, kn = kn.P[-c(1, length(kn.P))],
    Bo = kn.P[c(1, length(kn.P))]) + ns(dur, kn = kn.dur[-c(1,
    length(kn.dur))], Bo = kn.dur[c(1, length(kn.dur))]), family = poisson,
```

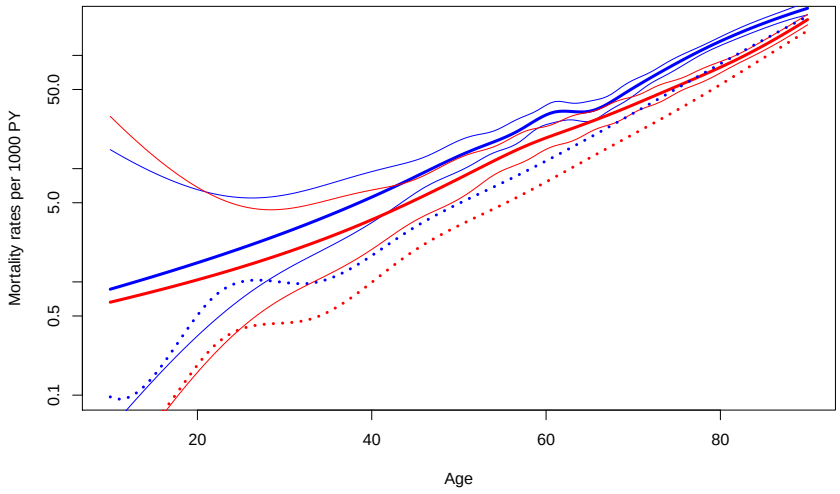


Figure 2.14: *Age-specific mortality rates for Danish diabetes patients as estimated from a model with only age. Broken lines are modeled population rates 1995–2010. Blue: men, red: women.*

```
    data = subset(SL, sex == "M"), offset = log(lex.dur))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.0444  -0.3011  -0.2191  -0.1395   4.0606

Coefficients:
                                                                                  Estim
(Intercept)                                                                       -3.21
ns(A, kn = kn.A[-c(1, length(kn.A))], Bo = kn.A[c(1, length(kn.A))])1             0.68
ns(A, kn = kn.A[-c(1, length(kn.A))], Bo = kn.A[c(1, length(kn.A))])2             1.21
ns(A, kn = kn.A[-c(1, length(kn.A))], Bo = kn.A[c(1, length(kn.A))])3             1.50
ns(A, kn = kn.A[-c(1, length(kn.A))], Bo = kn.A[c(1, length(kn.A))])4             1.92
ns(A, kn = kn.A[-c(1, length(kn.A))], Bo = kn.A[c(1, length(kn.A))])5             2.12
ns(A, kn = kn.A[-c(1, length(kn.A))], Bo = kn.A[c(1, length(kn.A))])6             1.88
ns(A, kn = kn.A[-c(1, length(kn.A))], Bo = kn.A[c(1, length(kn.A))])7             2.42
ns(A, kn = kn.A[-c(1, length(kn.A))], Bo = kn.A[c(1, length(kn.A))])8             3.16
ns(A, kn = kn.A[-c(1, length(kn.A))], Bo = kn.A[c(1, length(kn.A))])9             2.47
ns(P, kn = kn.P[-c(1, length(kn.P))], Bo = kn.P[c(1, length(kn.P))])1            -0.27
ns(P, kn = kn.P[-c(1, length(kn.P))], Bo = kn.P[c(1, length(kn.P))])2            -0.49
ns(P, kn = kn.P[-c(1, length(kn.P))], Bo = kn.P[c(1, length(kn.P))])3            -0.29
ns(dur, kn = kn.dur[-c(1, length(kn.dur))], Bo = kn.dur[c(1, length(kn.dur))])1  -0.30
ns(dur, kn = kn.dur[-c(1, length(kn.dur))], Bo = kn.dur[c(1, length(kn.dur))])2  -0.81
ns(dur, kn = kn.dur[-c(1, length(kn.dur))], Bo = kn.dur[c(1, length(kn.dur))])3  -0.39
ns(dur, kn = kn.dur[-c(1, length(kn.dur))], Bo = kn.dur[c(1, length(kn.dur))])4  -0.79
ns(dur, kn = kn.dur[-c(1, length(kn.dur))], Bo = kn.dur[c(1, length(kn.dur))])5  -0.59
ns(dur, kn = kn.dur[-c(1, length(kn.dur))], Bo = kn.dur[c(1, length(kn.dur))])6  -0.17
ns(dur, kn = kn.dur[-c(1, length(kn.dur))], Bo = kn.dur[c(1, length(kn.dur))])7  -0.92
ns(dur, kn = kn.dur[-c(1, length(kn.dur))], Bo = kn.dur[c(1, length(kn.dur))])8  -0.10
ns(dur, kn = kn.dur[-c(1, length(kn.dur))], Bo = kn.dur[c(1, length(kn.dur))])9  -0.85
ns(dur, kn = kn.dur[-c(1, length(kn.dur))], Bo = kn.dur[c(1, length(kn.dur))])10  0.06

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 11288  on 32697  degrees of freedom
Residual deviance: 10010  on 32675  degrees of freedom
AIC: 12742

Number of Fisher Scoring iterations: 7
```

As a small aside; the specification of the natural splines is cumbersome with specification of both knots and boundary knots and the output from `summary()` is very large, so we use a convenience wrapper that does this on the fly:

```
> source( "http://BendixCarstensen.com/SPE/R/Ns.r" )
> Ns

function (x, df = NULL, knots = NULL, intercept = FALSE, Boundary.knots = NULL)
{
    if (is.null(Boundary.knots)) {
        if (!is.null(knots)) {
            knots <- sort(unique(knots))
            ok <- c(1, length(knots))
            Boundary.knots <- knots[ok]
            knots <- knots[-ok]
```

```
        }
    }
    ns(x, df = df, knots = knots, intercept = intercept, Boundary.knots = Boundary.kno
}
```

Having defined this, the model specification and summary is much simpler:

```
> 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
-1.0444  -0.3011  -0.2191  -0.1395   4.0606

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)          -3.21711    0.10190 -31.571  < 2e-16
Ns(A, kn = kn.A)1     0.68906    0.18264   3.773 0.000161
Ns(A, kn = kn.A)2     1.21939    0.16510   7.386 1.52e-13
Ns(A, kn = kn.A)3     1.50702    0.18593   8.105 5.27e-16
Ns(A, kn = kn.A)4     1.92383    0.17609  10.925  < 2e-16
Ns(A, kn = kn.A)5     2.12200    0.18983  11.178  < 2e-16
Ns(A, kn = kn.A)6     1.88170    0.20204   9.314  < 2e-16
Ns(A, kn = kn.A)7     2.42353    0.17150  14.131  < 2e-16
Ns(A, kn = kn.A)8     3.16568    0.13276  23.844  < 2e-16
Ns(A, kn = kn.A)9     2.47621    0.12664  19.554  < 2e-16
Ns(P, kn = kn.P)1    -0.27240    0.11656  -2.337 0.019439
Ns(P, kn = kn.P)2    -0.49119    0.16991  -2.891 0.003842
Ns(P, kn = kn.P)3    -0.29024    0.10322  -2.812 0.004927
Ns(dur, kn = kn.dur)1 -0.30091    0.23451  -1.283 0.199437
Ns(dur, kn = kn.dur)2 -0.81243    0.22716  -3.576 0.000348
Ns(dur, kn = kn.dur)3 -0.39040    0.21414  -1.823 0.068284
Ns(dur, kn = kn.dur)4 -0.79923    0.21901  -3.649 0.000263
Ns(dur, kn = kn.dur)5 -0.59604    0.20574  -2.897 0.003767
Ns(dur, kn = kn.dur)6 -0.17280    0.19660  -0.879 0.379436
Ns(dur, kn = kn.dur)7 -0.92102    0.20048  -4.594 4.35e-06
Ns(dur, kn = kn.dur)8 -0.10567    0.16499  -0.640 0.521896
Ns(dur, kn = kn.dur)9 -0.85446    0.24909  -3.430 0.000603
Ns(dur, kn = kn.dur)10  0.06945    0.14170   0.490 0.624043

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 11288  on 32697  degrees of freedom
Residual deviance: 10010  on 32675  degrees of freedom
AIC: 12742
```

```
Number of Fisher Scoring iterations: 7
```

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

10. 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     32675      10010
2     32688      10097 -13    -86.6 6.222e-13
```

```
> 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     31405     8744.4
2     31418     8808.1 -13  -63.653 1.157e-08
```

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.

11. 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 spline basis as used in the model.

    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.

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

12. 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 2.15 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.

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" ) )
```

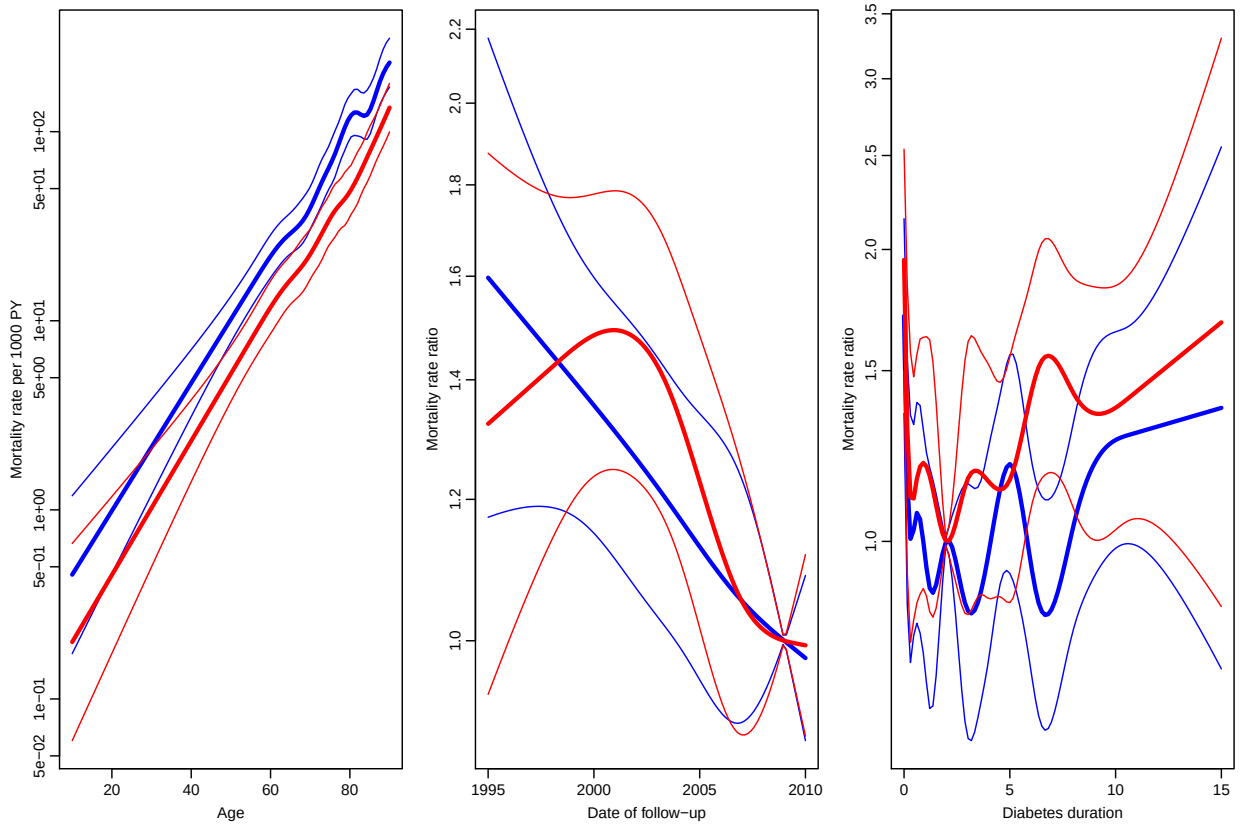


Figure 2.15: *Estimates from model for mortality of Danish diabetes patients. The duration is modeled with 10 parameters, which is clearly way too much.*

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

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

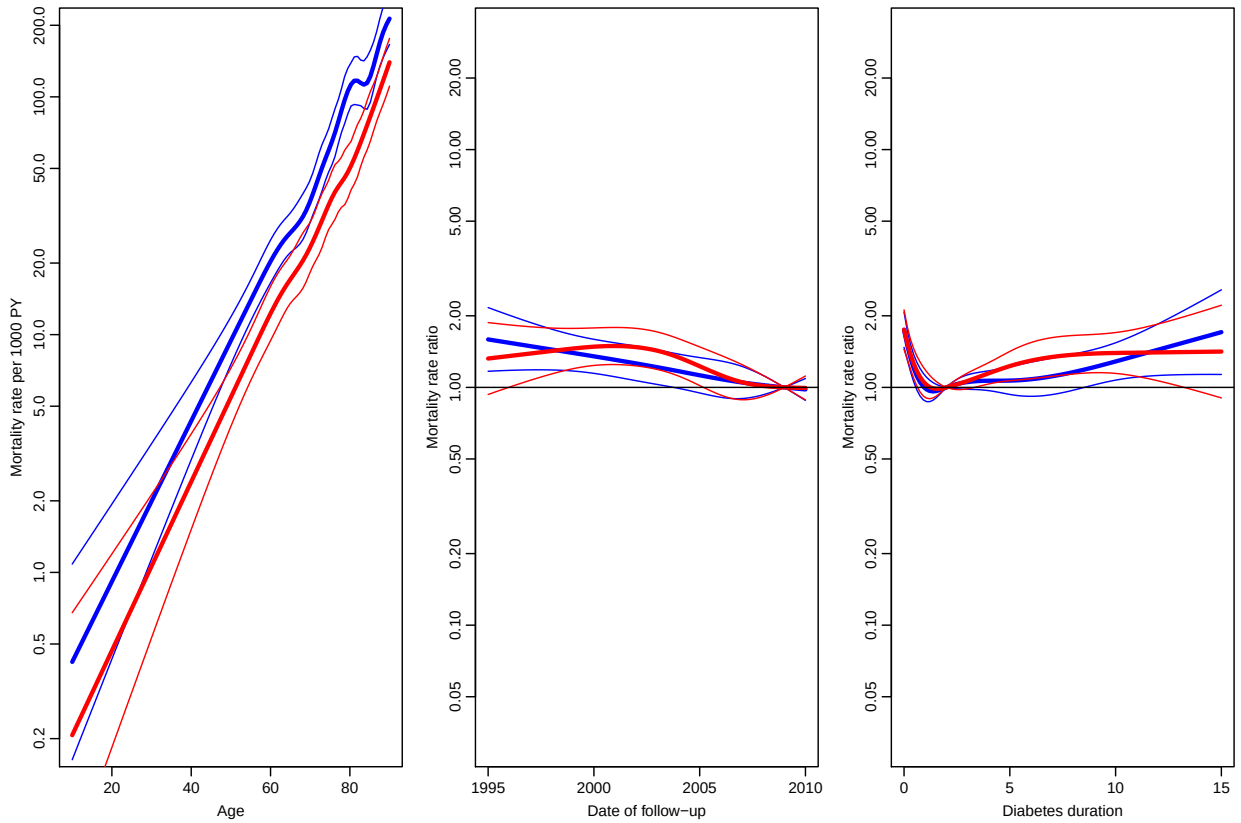


Figure 2.16: *Estimates from the model for mortality of Danish diabetes patients with only 5 knots (corresponding to 4 parameters) for duration.*

```
> 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 )
```

13. We have so far fitted models separately for men and women, but judging from the display of the parameters in figure 2.17, 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 ),
```

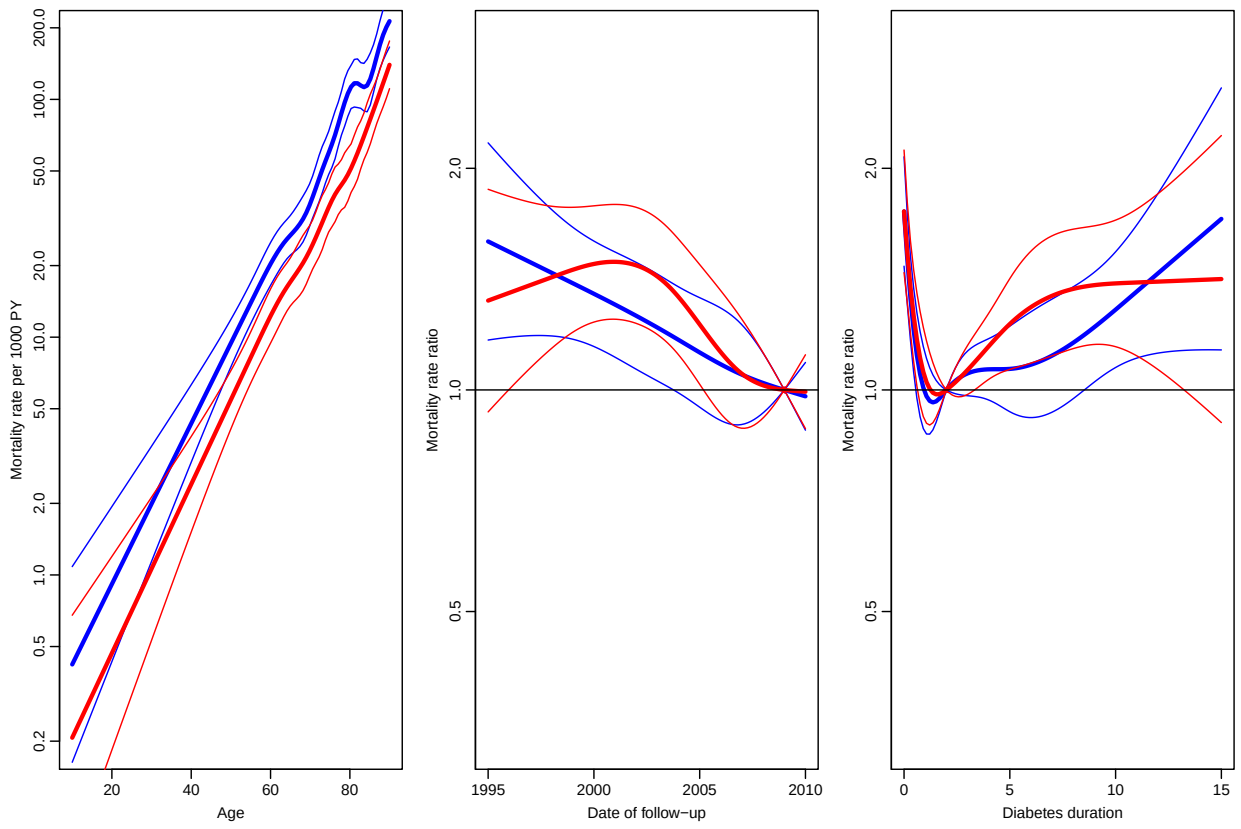


Figure 2.17: *Estimates from model for mortality of Danish diabetes patients.*

```
+             offset = log( lex.dur ),
+             family = poisson,
+               data = SL )
> ci.exp(m2)
```

```
                          exp(Est.)       2.5%       97.5%
(Intercept)             0.03579237  0.0302410  0.04236282
sexF                    0.66657884  0.5344046  0.83144370
Ns(P, kn = kn.P)1       0.72547140  0.6136877  0.85761656
Ns(P, kn = kn.P)2       0.68310826  0.5324727  0.87635829
Ns(P, kn = kn.P)3       0.70425590  0.6066292  0.81759391
Ns(dur, kn = kn.dur)1   0.59963846  0.4994206  0.71996685
Ns(dur, kn = kn.dur)2   0.83011043  0.7026141  0.98074222
Ns(dur, kn = kn.dur)3   0.43036334  0.3246658  0.57047162
Ns(dur, kn = kn.dur)4   1.07957947  0.9188562  1.26841594
sexM:Ns(A, kn = kn.A)1  1.99255665  1.3930634  2.85003686
sexF:Ns(A, kn = kn.A)1  2.29270232  1.4385172  3.65409872
sexM:Ns(A, kn = kn.A)2  3.37498942  2.4423063  4.66385131
sexF:Ns(A, kn = kn.A)2  3.34775743  2.2559188  4.96803339
sexM:Ns(A, kn = kn.A)3  4.50026933  3.1262878  6.47810609
sexF:Ns(A, kn = kn.A)3  4.57342778  2.9485292  7.09378832
sexM:Ns(A, kn = kn.A)4  6.86263799  4.8597269  9.69103852
sexF:Ns(A, kn = kn.A)4  5.07588761  3.3572466  7.67433495
sexM:Ns(A, kn = kn.A)5  8.26769328  5.6990906 11.99397541
sexF:Ns(A, kn = kn.A)5  6.38110530  4.2594149  9.55964737
sexM:Ns(A, kn = kn.A)6  6.64708490  4.4773994  9.86816990
sexF:Ns(A, kn = kn.A)6  8.63141522  6.0007807 12.41527270
sexM:Ns(A, kn = kn.A)7 11.12936225  7.9570794 15.56635267
sexF:Ns(A, kn = kn.A)7 10.60687235  7.8238629 14.37982012
sexM:Ns(A, kn = kn.A)8 23.40325077 18.0491615 30.34557292
sexF:Ns(A, kn = kn.A)8 27.66533384 21.1941508 36.11235494
sexM:Ns(A, kn = kn.A)9 11.76296454  9.1806331 15.07165505
sexF:Ns(A, kn = kn.A)9 14.83464848 11.5157684 19.11004014
```

14. 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:

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

```
[1] 0.473991
```

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

15. 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, 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.

16. We can now extract the parameters from the model. Note that the sequence (and hence meaning) naming 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)              0.03579237  0.0302410  0.04236282
sexM:Ns(A, kn = kn.A)1   1.99255665  1.3930634  2.85003686
sexM:Ns(A, kn = kn.A)2   3.37498942  2.4423063  4.66385131
sexM:Ns(A, kn = kn.A)3   4.50026933  3.1262878  6.47810609
sexM:Ns(A, kn = kn.A)4   6.86263799  4.8597269  9.69103852
sexM:Ns(A, kn = kn.A)5   8.26769328  5.6990906 11.99397541
sexM:Ns(A, kn = kn.A)6   6.64708490  4.4773994  9.86816990
sexM:Ns(A, kn = kn.A)7  11.12936225  7.9570794 15.56635267
sexM:Ns(A, kn = kn.A)8  23.40325077 18.0491615 30.34557292
sexM:Ns(A, kn = kn.A)9  11.76296454  9.1806331 15.07165505
Ns(P, kn = kn.P)1        0.72547140  0.6136877  0.85761656
Ns(P, kn = kn.P)2        0.68310826  0.5324727  0.87635829
Ns(P, kn = kn.P)3        0.70425590  0.6066292  0.81759391
Ns(dur, kn = kn.dur)1    0.59963846  0.4994206  0.71996685
Ns(dur, kn = kn.dur)2    0.83011043  0.7026141  0.98074222
Ns(dur, kn = kn.dur)3    0.43036334  0.3246658  0.57047162
Ns(dur, kn = kn.dur)4    1.07957947  0.9188562  1.26841594
```

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

```
                          exp(Est.)       2.5%       97.5%
(Intercept)              0.03579237  0.0302410  0.04236282
sexF                     0.66657884  0.5344046  0.83144370
sexF:Ns(A, kn = kn.A)1   2.29270232  1.4385172  3.65409872
sexF:Ns(A, kn = kn.A)2   3.34775743  2.2559188  4.96803339
sexF:Ns(A, kn = kn.A)3   4.57342778  2.9485292  7.09378832
sexF:Ns(A, kn = kn.A)4   5.07588761  3.3572466  7.67433495
sexF:Ns(A, kn = kn.A)5   6.38110530  4.2594149  9.55964737
sexF:Ns(A, kn = kn.A)6   8.63141522  6.0007807 12.41527270
sexF:Ns(A, kn = kn.A)7  10.60687235  7.8238629 14.37982012
sexF:Ns(A, kn = kn.A)8  27.66533384 21.1941508 36.11235494
sexF:Ns(A, kn = kn.A)9  14.83464848 11.5157684 19.11004014
Ns(P, kn = kn.P)1        0.72547140  0.6136877  0.85761656
Ns(P, kn = kn.P)2        0.68310826  0.5324727  0.87635829
Ns(P, kn = kn.P)3        0.70425590  0.6066292  0.81759391
Ns(dur, kn = kn.dur)1    0.59963846  0.4994206  0.71996685
Ns(dur, kn = kn.dur)2    0.83011043  0.7026141  0.98074222
Ns(dur, kn = kn.dur)3    0.43036334  0.3246658  0.57047162
Ns(dur, kn = kn.dur)4    1.07957947  0.9188562  1.26841594
```

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) )
> fi.A <- ci.exp( m2, subset=c("Int","sexF","P","dur"), ctr.mat=cbind(1,1,AC,PR,dR) )
> 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"),eac
+          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","bla
+          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","bla
+          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.

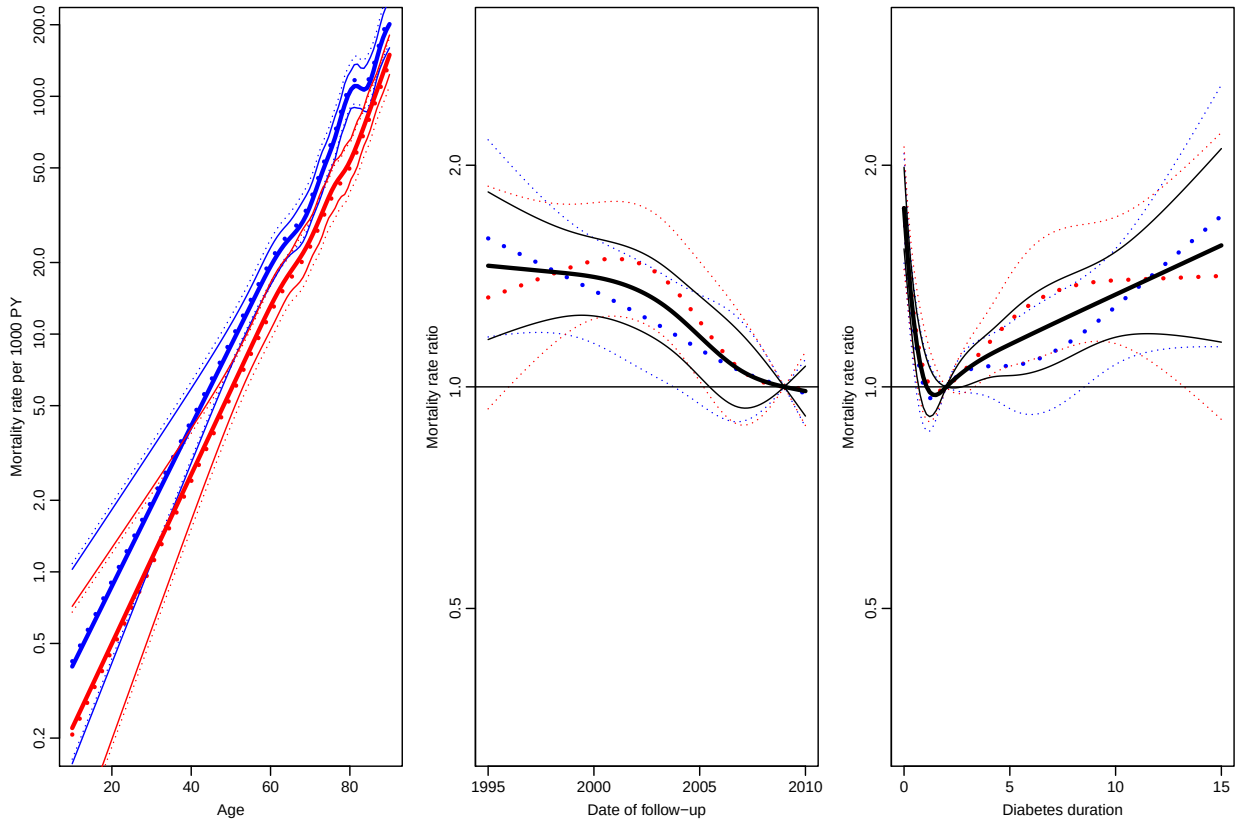


Figure 2.18: *Estimates from models for mortality of Danish diabetes patients. The broken lines are from the full interaction model, full lines with common effects of date and duration. Men:blue, women:red, both (i.e. common): black.*

17. 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 `predict` 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 )
> predict( mm, newdata=nd, se.fit=TRUE )

$fit
       1        2        3        4        5        6        7        8        9
3.267129 2.743749 2.799280 2.896099 2.953923 2.999912 3.057107 3.128086 3.208354 3.293
      14       15       16       17       18       19       20       21
3.640437 3.724499 3.804419 3.883603 3.967258 4.060710 4.169289 4.296557

$se.fit
         1          2          3          4          5          6          7
0.15795936 0.14233345 0.11950284 0.10897940 0.09461150 0.09037873 0.09600808 0.0996765
        11         12         13         14         15         16         17          1
0.10552798 0.12224274 0.13906924 0.15579972 0.18262347 0.22314661 0.27243810 0.3257126
        21
0.48844361

$residual.scale
[1] 1
```

    We can wrap this so that we get the predicted rates with confidence intervals:

```
> sapply(predict( mm, newdata=nd, se.fit=TRUE )[1:2],cbind) %*% ci.mat()

      Estimate     2.5%    97.5%
 [1,] 3.267129 2.957534 3.576724
 [2,] 2.743749 2.464781 3.022717
 [3,] 2.799280 2.565059 3.033502
 [4,] 2.896099 2.682503 3.109695
 [5,] 2.953923 2.768488 3.139358
 [6,] 2.999912 2.822773 3.177051
 [7,] 3.057107 2.868934 3.245279
 [8,] 3.128086 2.932723 3.323448
 [9,] 3.208354 3.014682 3.402027
[10,] 3.293304 3.101715 3.484893
[11,] 3.379910 3.173079 3.586741
[12,] 3.466715 3.227123 3.706306
```

```
[13,] 3.553489 3.280918 3.826059
[14,] 3.640437 3.335075 3.945799
[15,] 3.724499 3.366563 4.082434
[16,] 3.804419 3.367060 4.241778
[17,] 3.883603 3.349634 4.417572
[18,] 3.967258 3.328872 4.605643
[19,] 4.060710 3.316015 4.805405
[20,] 4.169289 3.318858 5.019719
[21,] 4.296557 3.339225 5.253889
```

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(0,15,0.1) )
+ {
+ nd <- data.frame( A=A+pts,
+                   P=P+pts,
+                 dur=  pts,
+             lex.dur=1000 )
+ cbind( nd$A,
+ exp(sapply(predict( mm, newdata=nd, se.fit=TRUE )[1:2],cbind) %*% ci.mat()),
+ exp(sapply(predict( mf, newdata=nd, se.fit=TRUE )[1:2],cbind) %*% ci.mat())
+       )
+ }
> DMm( 50, 1996, pts=0:10 )
```

```
         Estimate     2.5%    97.5%  Estimate      2.5%    97.5%
 [1,] 50 25.39307 19.28622 33.43360 12.923295  9.259298 18.03717
 [2,] 51 15.04576 11.73788 19.28584  8.520286  6.305111 11.51372
 [3,] 52 15.90491 12.94488 19.54178  9.139234  7.081519 11.79487
 [4,] 53 17.52097 14.44036 21.25879 10.808554  8.506719 13.73324
 [5,] 54 18.55717 15.58285 22.09921 12.862220 10.332394 16.01146
 [6,] 55 19.41537 16.30862 23.11395 15.069673 12.140844 18.70505
 [7,] 56 20.53363 17.03991 24.74367 17.052232 13.589632 21.39709
 [8,] 57 22.00858 18.21885 26.58662 18.514317 14.692504 23.33026
 [9,] 58 23.80149 19.82767 28.57175 19.241654 15.303932 24.19256
[10,] 59 25.89341 21.49848 31.18678 19.346241 15.278189 24.49747
[11,] 60 28.29472 23.00241 34.80467 19.263952 14.879354 24.94059
```

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) )
```

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

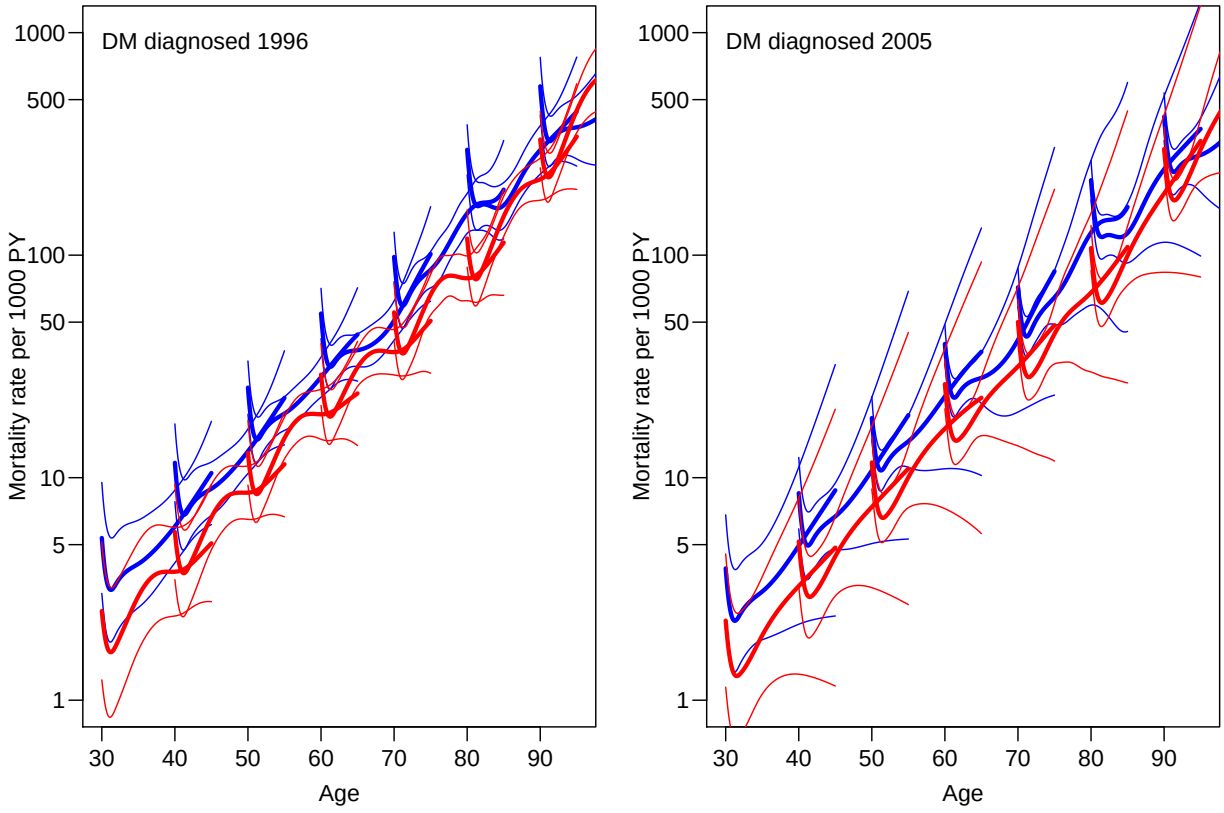


Figure 2.19: *Estimates of mortality of Danish diabetes patients. Note that unlike what is detectable from the plots of the separate effects it seems that for men mortality is higher the younger age at diagnosis, but not for women.*

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 ) )
> 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     32681      10024
2     32673      10014  8   9.6200   0.2927
3     32681      10024 -8  -9.5799   0.2958


> 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     32681      10024
2     32678      10020  3   4.1180   0.2490
3     32680      10020 -2  -0.3256   0.8498
```

```
> 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     31411     8752.3
2     31403     8742.4  8   9.9473   0.2687
3     31411     8747.4 -8  -5.0449   0.7528


> 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     31411     8752.3
2     31408     8750.5  3   1.8693  0.59998
3     31410     8757.3 -2  -6.8135  0.03315
```

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 0.03).

19. So we 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
> M.P <- ci.exp( Mm, subset="P"  , ctr.mat=PC-PR )
> M.d <- ci.exp( Mm, subset="kn.dur", ctr.mat=dC-dR )
> F.A <- ci.exp( Mf, ctr.mat=cbind(1,AC,PR,dR) ) * 1000
> F.P <- ci.exp( Mf, subset="P"  , ctr.mat=PC-PR )
> 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 2.20

```
> 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
> 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 )
```

20. 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(1,3),each=6), lwd=c(3,1,1), col=rep(c("blue","red"),eac
+          log="y", ylim=c(0.2,180),
+          xlab="Age at diagnosis/follow-up", ylab="Mortality rate at 2 years duration
> matplot( pr.P,  cbind(M.P,F.P,m.P,f.P),
+          type="l", lty=rep(c(1,3),each=6), lwd=c(3,1,1), col=rep(c("blue","red"),eac
+          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(1,3),each=6), lwd=c(3,1,1), col=rep(c("blue","red"),eac
+          log="y", ylim=c(1/3,3),
+          xlab="Diabetes duration", ylab="Mortality rate ratio" )
> abline( h=1 )
```

From figure 2.21 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.

2.12.1 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, 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.

21. 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

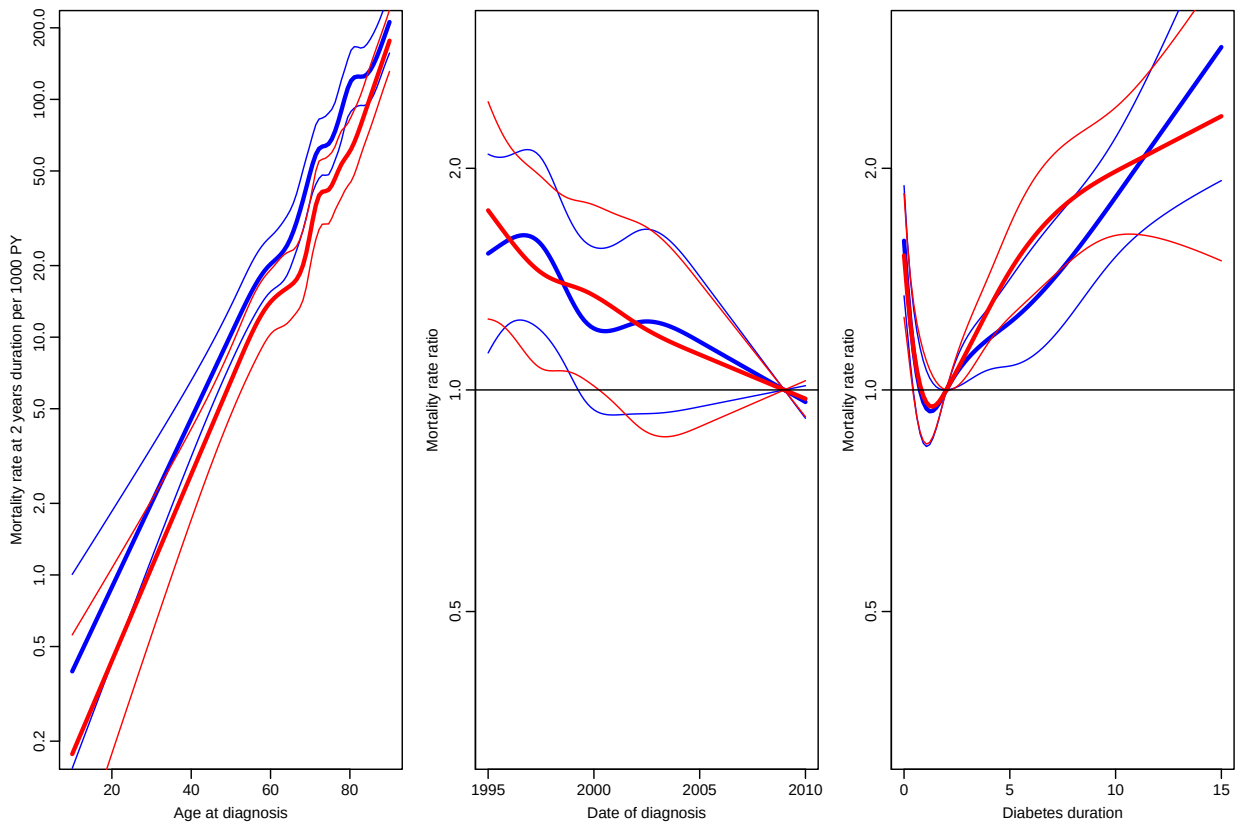


Figure 2.20: *Model for diabetes patient mortality using age and date at diagnosis.*

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 'Lexis' and 'data.frame':       64126 obs. of  14 variables:
 $ lex.id : int  1 1 1 1 1 1 1 1 1 1 ...
 $ A      : num  58.7 59 60 61 62 ...
 $ P      : num  1999 1999 2000 2001 2002 ...
 $ dur    : num  0 0.339 1.339 2.339 3.339 ...
 $ lex.dur: num  0.339 1 1 1 1 ...
 $ 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 ...
```

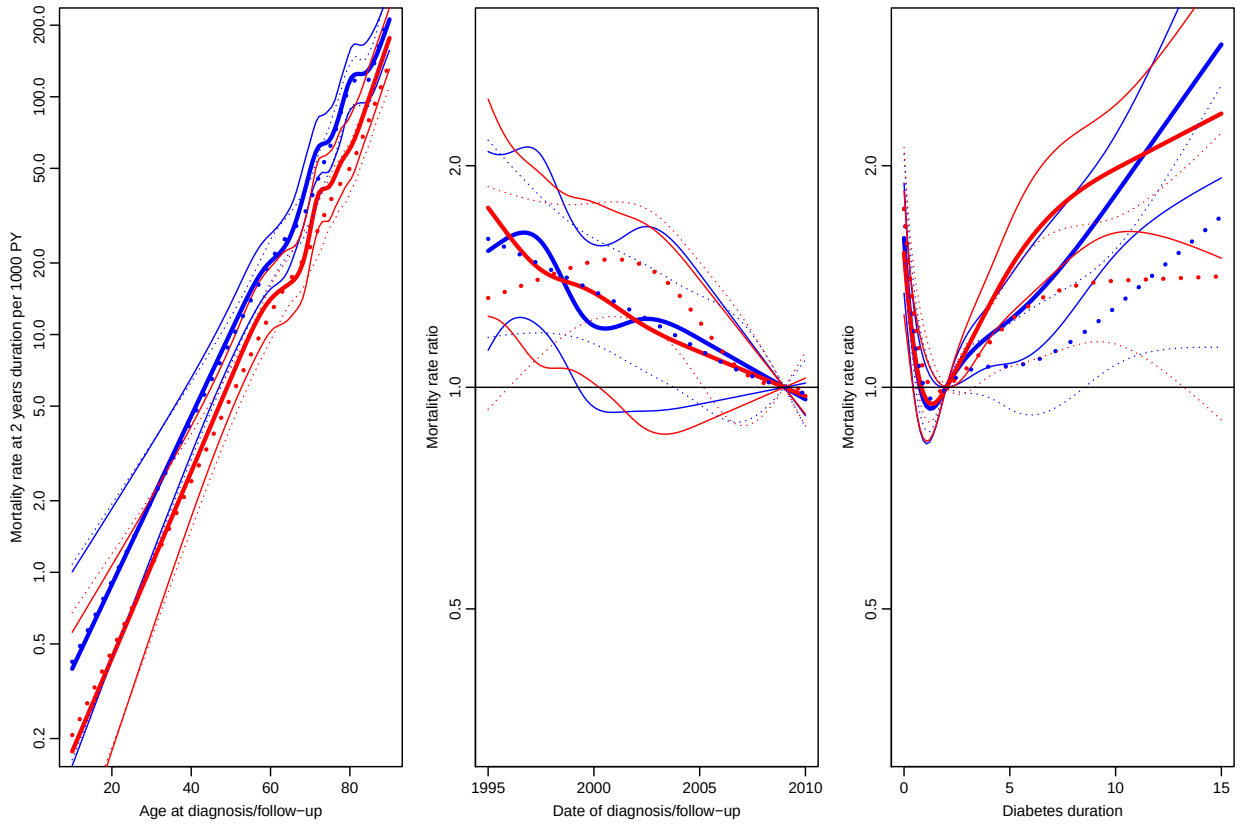


Figure 2.21: *Comparison of estimates from two different models; the full lines give the estimates from the model where age and date are included as fixed variables with the value at diabetes diagnosis, whereas the dotted lines are estimates from the model where age and calendar time are included as time scales.*

```
 $ 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 1 2 3 4 5 6 7 8 9 ...
  ..$ 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] 64126    16
```

```
> dim( SLr )
```

```
[1] 64114    17
```

This merge only takes rows that have information from both datasets, hence the slightly fewer rows in `SLr` than in SL.

22. 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 )
```

Age	Sex: M	Sex: F	Total
0	0.00	0.00	0.00
	0.02	0.01	0.03
	0.00	0.00	0.00
10	1.00	1.00	2.00
	0.13	0.04	0.17
	7.75	24.84	11.82
20	0.00	0.00	0.00
	0.35	0.18	0.53
	0.00	0.00	0.00
30	5.00	4.00	9.00
	1.43	1.02	2.45
	3.49	3.92	3.67
40	32.00	15.00	47.00
	9.48	5.03	14.51
	3.38	2.98	3.24
50	119.00	62.00	181.00
	48.55	22.03	70.58
	2.45	2.81	2.56
60	275.00	157.00	432.00
	142.55	74.16	216.71
	1.93	2.12	1.99
70	486.00	331.00	817.00
	276.03	204.69	480.71
	1.76	1.62	1.70
80	348.00	423.00	771.00
	255.07	319.26	574.33
	1.36	1.32	1.34
90	76.00	160.00	236.00
	63.41	122.30	185.71
	1.20	1.31	1.27

```
Total   1342.00 1153.00 2495.00
         797.03  748.71 1545.74
           1.68    1.54    1.61
--------------------------------
```

We see that the 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.

23. 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) )
> sM.P <- ci.exp( Sm, subset="P"     , ctr.mat=PC-PR )
> sM.d <- ci.exp( Sm, subset="kn.dur", ctr.mat=dC-dR )
> sF.A <- ci.exp( Sf, ctr.mat=cbind(1,AC,PR,dR) )
> sF.P <- ci.exp( Sf, subset="P"     , ctr.mat=PC-PR )
> 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 )
```

24. It seems reasonably from figure **??** 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     64083      18764
```

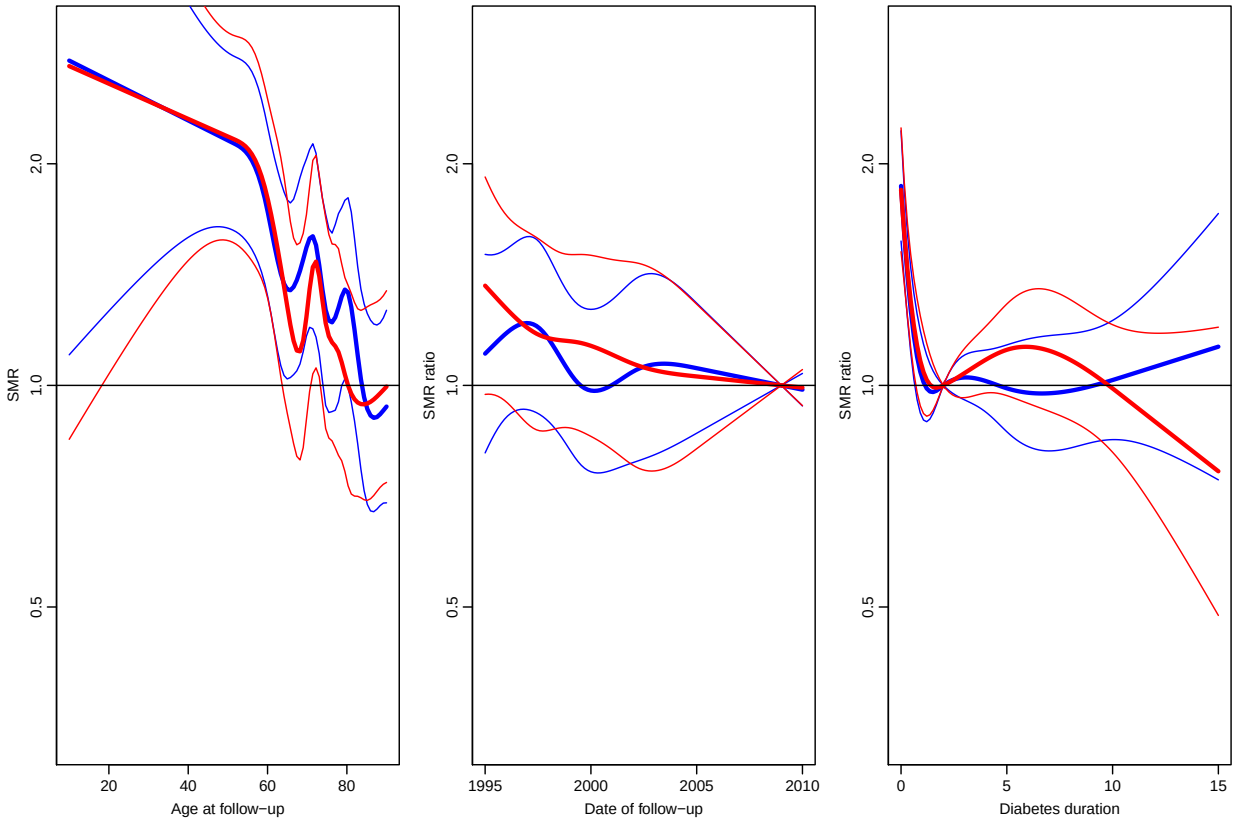


Figure 2.22: *SMR in the diabetic population relative to the (entire) Danish population. Clearly the effect of age is over-modeled.*

```
2     64082      18764  1   0.0004   0.9834
3     64066      18752 16  12.1924   0.7306
```

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

25. 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 )
```

26. 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,
+          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),
```

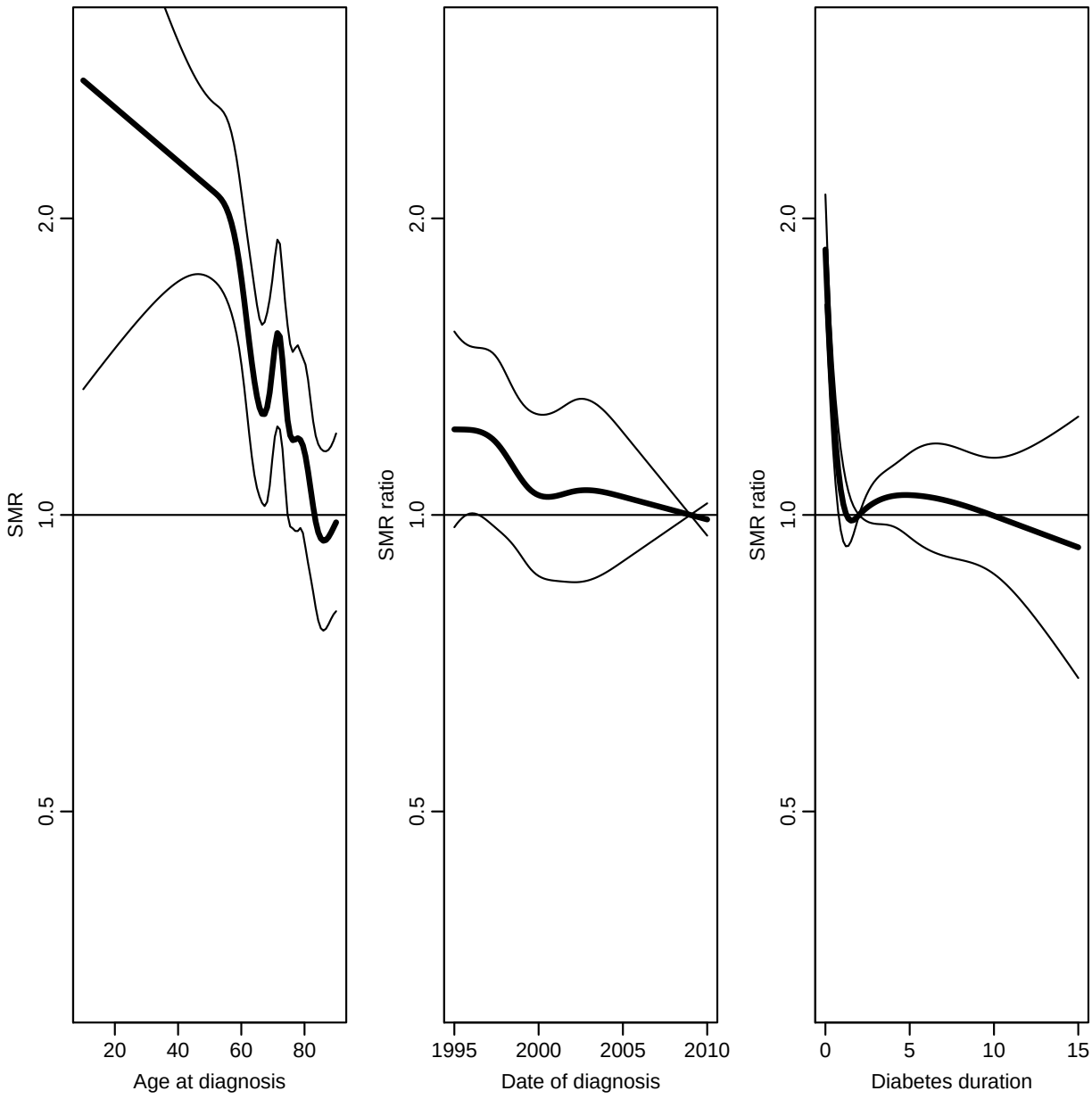


Figure 2.23: *SMR in the diabetic population for both sexes, relative to the (entire) Danish population.*

```
+          xlab="Diabetes duration", ylab="SMR ratio" )
> abline( h=1,v=2 )
```

27. 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)  1.539058 2.713708 0.3502251
```

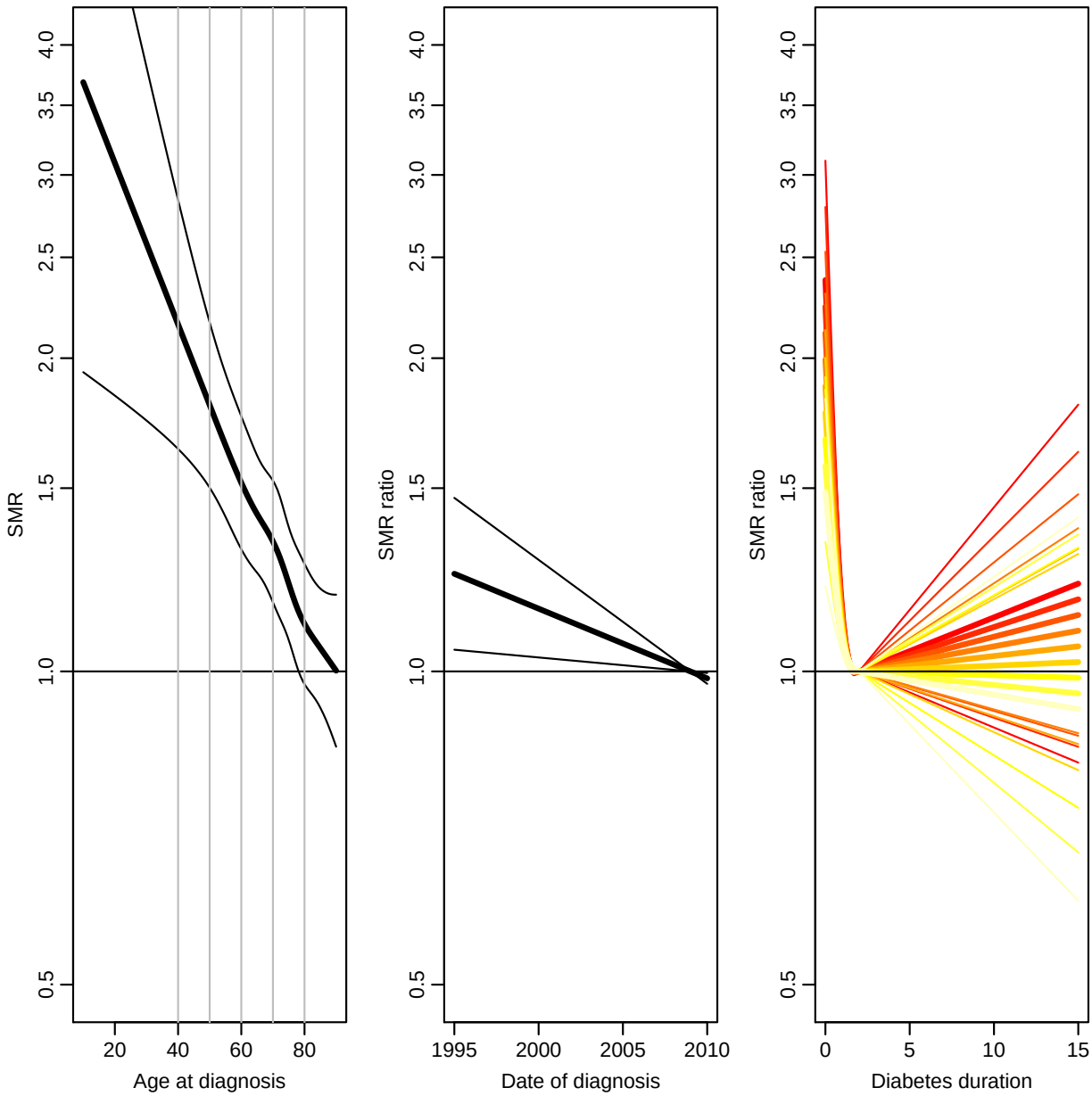


Figure 2.24: *SMR in the diabetic population for both sexes, relative to the (entire) Danish population — simplified model.*

Thus the change in SMR is a 1.5% annuaal decrease (95% c.i.: (0.3-2.7)%).

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,] 0.2676805 -1.369183 1.93171
```

Thus the estimate is an annual increase in SMR or 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.

2.12.2 Interaction models

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

```
> Six <- update( Sx, . ~. + I(A-dur):Ns(dur,knots=kn.dur) )
> anova( Six, 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     64091      18783
2     64093      18789 -2  -5.1995  0.07429
```

```
> ci.exp( Six )
```

```
                                    exp(Est.)         2.5%        97.5%
(Intercept)                      1.190491e+14 4.349094e+03 3.258768e+24
Ns(A - dur, kn = kn.Ad)1         5.952633e-01 4.728864e-01 7.493099e-01
Ns(A - dur, kn = kn.Ad)2         5.200933e-01 4.197792e-01 6.443794e-01
Ns(A - dur, kn = kn.Ad)3         3.244669e-01 2.310045e-01 4.557434e-01
Ns(A - dur, kn = kn.Ad)4         5.027709e-01 4.076581e-01 6.200750e-01
I(P - dur)                       9.846898e-01 9.729425e-01 9.965789e-01
Ns(dur, kn = kn.dur)1            8.385022e-02 2.277759e-02 3.086744e-01
Ns(dur, kn = kn.dur)2            4.585490e-01 3.091840e-01 6.800713e-01
Ns(dur, kn = kn.dur)1:I(A - dur) 1.020444e+00 1.002657e+00 1.038546e+00
Ns(dur, kn = kn.dur)2:I(A - dur) 1.006233e+00 1.000913e+00 1.011582e+00
```

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

```
> Six.A <- ci.exp( Six, ctr.mat=cbind(1,AC,rf.P,dR,dR*pr.A) )
> Six.P <- ci.exp( Six, subset="P"     , ctr.mat=cbind(pr.P-rf.P) )
> Six.d <- ci.exp( Six, subset="kn.dur", ctr.mat=cbind(dC-dR,(dC-dR)*50) )
> for( a in seq(55,90,5) ) Six.d <- cbind( Six.d,
+                  ci.exp( Six, subset="kn.dur", ctr.mat=cbind(dC-dR,(dC-dR)*a) ) )
> dim( Six.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, Six.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, Six.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, Six.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 )
```

29. 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.si <- exp(sapply(predict( Six, newdata=nd, se.fit=TRUE )[1:2],cbind) %*% ci.mat())
> A.sm <- exp(sapply(predict( Sx , newdata=nd, se.fit=TRUE )[1:2],cbind) %*% ci.mat())
```

```
> 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.sm),
+           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 olds , 1.4 for 70 year olds and 1.1 for 80 year olds

30. 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, Six, Sx, test="Chisq" )
```

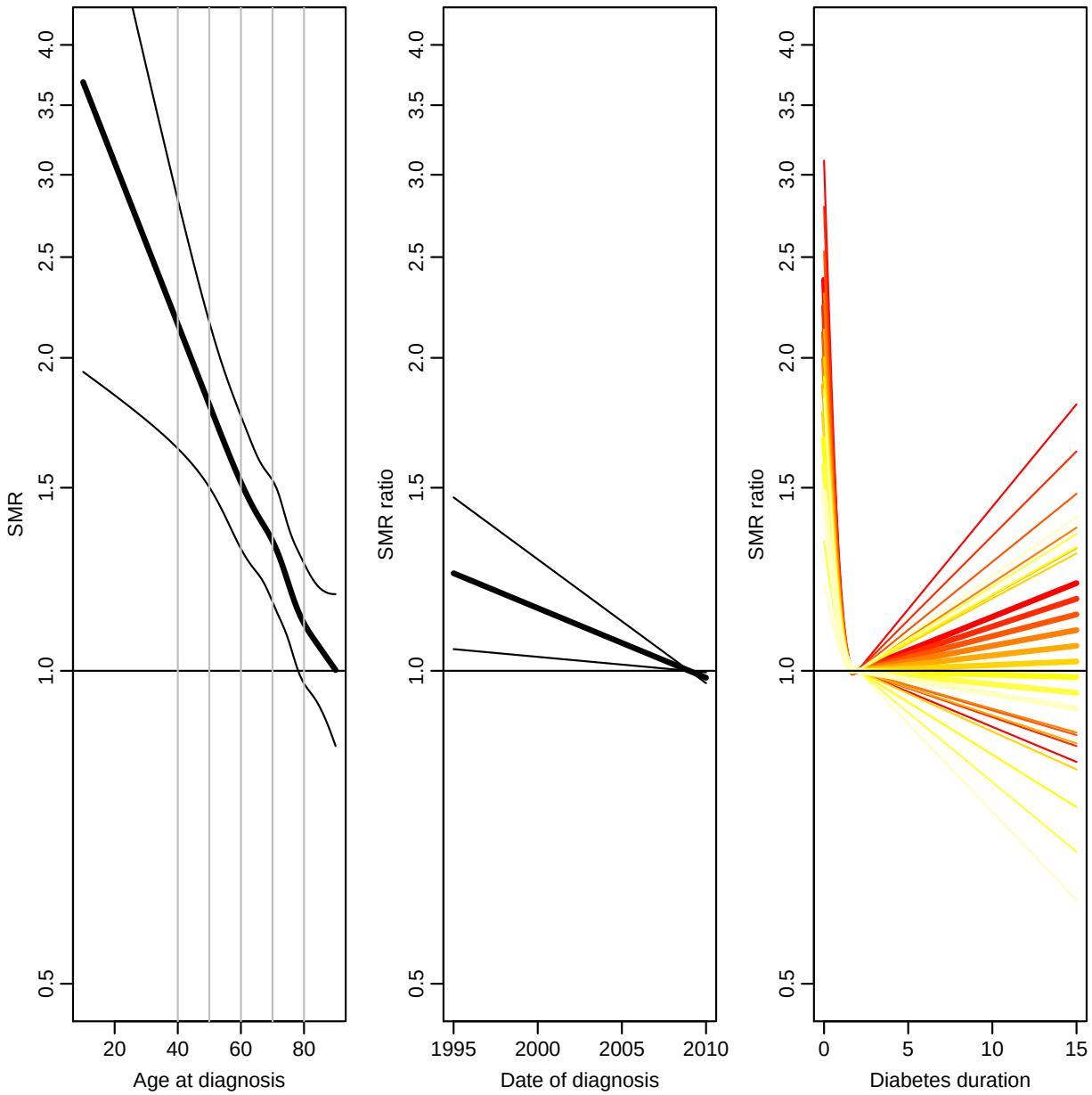


Figure 2.25: *SMR in the diabetic population for both sexes, relative to the (entire) Danish population — interaction model with age-specific duration effects.*

```
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     64085      18777
```

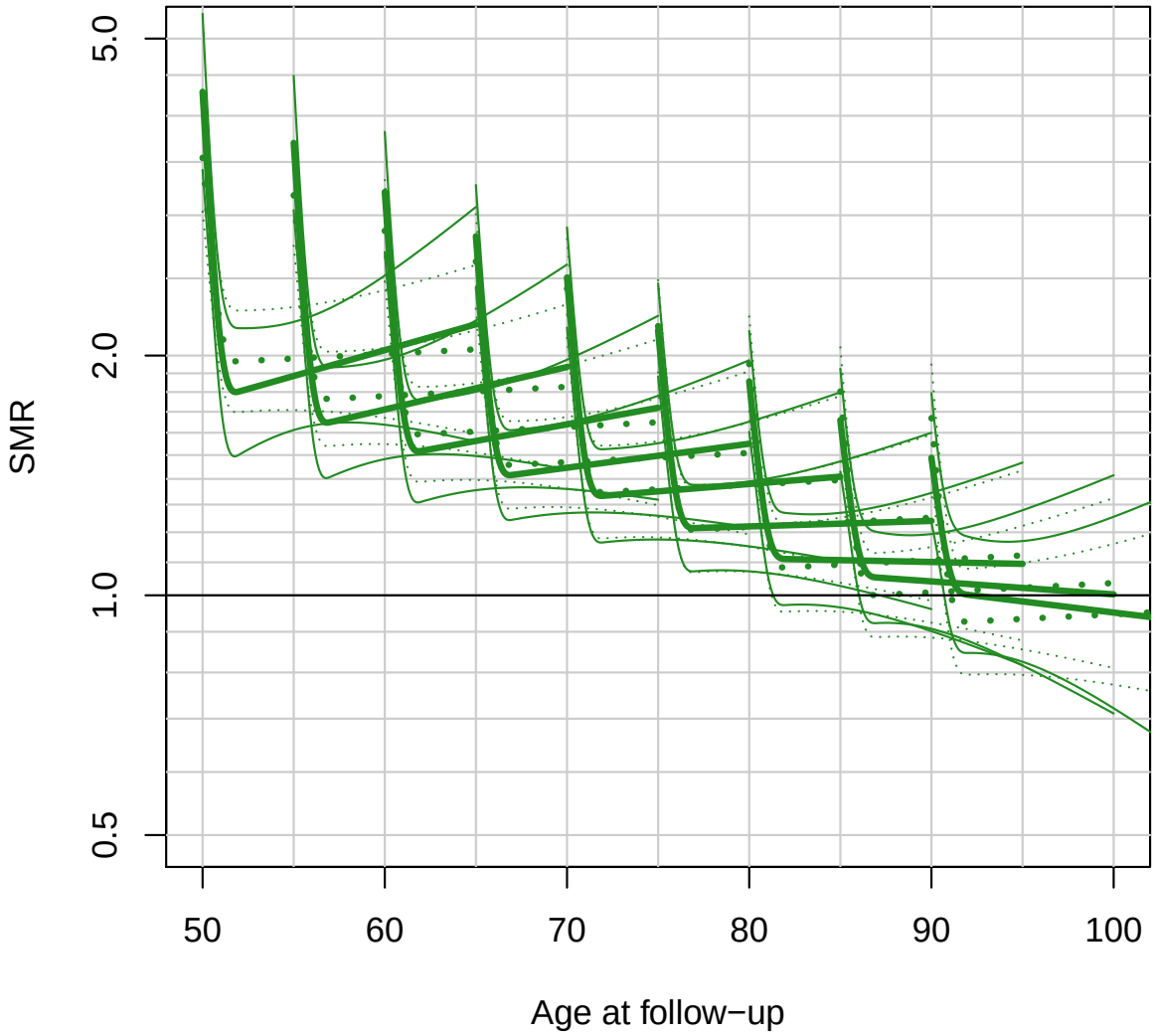


Figure 2.26: *SMR in the diabetic population for both sexes, relative to the (entire) Danish population — interaction model with age-specific duration effects, shown for patients diagnosed at ages 50 to 90.*

```
2     64091      18783 -6  -6.1491  0.40670
3     64093      18789 -2  -5.1995  0.07429
```

And we can use the exact same code to show the interaction and plot it along the others in a similar plot:

```
> A.sX <- exp(sapply(predict( SiX, newdata=nd, se.fit=TRUE )[1:2],cbind) %*% ci.mat())
> 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.si,A.sm),
+           type="l", lty=rep(c(1,3),c(6,3)), lwd=c(3,1,1),
+           col=rep(c("magenta","forestgreen"),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.

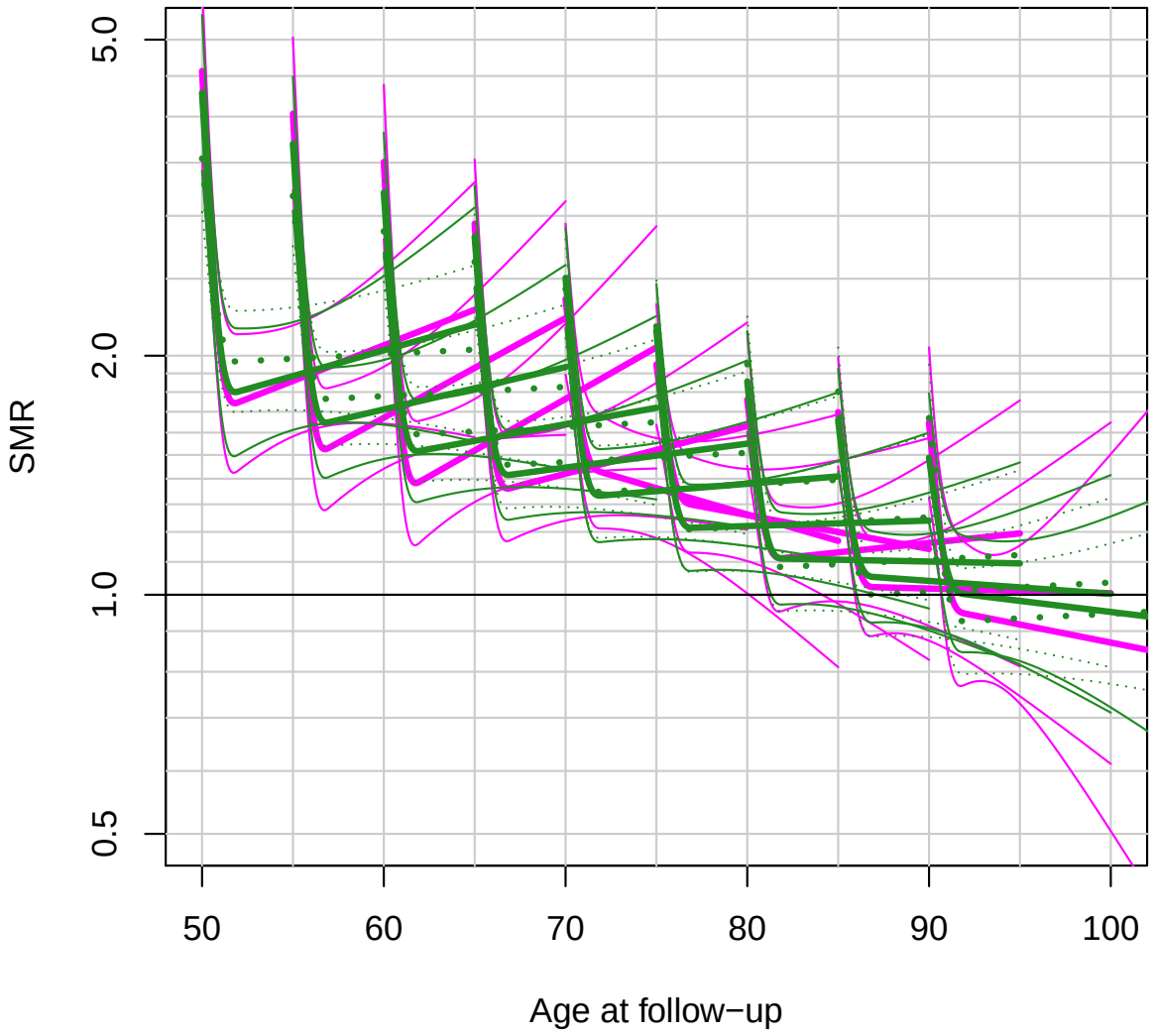


Figure 2.27: *SMR in the diabetic population for both sexes, relative to the (entire) Danish population — interaction model with age-specific duration effects, shown for patients diagnosed at ages 50, 60, 70, 80 and 90. The bright green curves are from the more complex interaction model.*

2.13 Causal inference

2.14 Nested case-control study and case-cohort study: Risk factors of coronary heart disease

In this exercise we shall apply both the nested case-control (NCC) design and the case-cohort (CC) design in sampling control subjects from a defined cohort or closed study population. The case group comprises those cohort members who die from coronary heart disease (CHD) during a > 20 years follow-up of the cohort. The risk factors of interest are cigarette smoking, systolic blood pressure, and total cholesterol level.

Our study population is an occupational cohort comprising 1501 men working in blue-collar jobs in one Nordic country. Eligible subjects had no history of coronary heart disease when recruited to the study in the early 1990s. Smoking habits and many other items were inquired at baseline by a questionnaire, and blood pressure was measured by a research nurse, the values being written down on the questionnaire. Serum samples were also taken from the cohort members at the same time and were stored in a freezer. For some reason, the data in the questionnaires were not entered to any computer file, but the questionnaires were kept in a safe storehouse for further purposes. Also, no biochemical analyses were initially performed for the sera collected from the participants. However, dates of birth and dates of entry to the study were recorded in an electronic file.

In 2010 the study was suddenly reactivated by those investigators of the original team who were still alive then. As the first step mortality follow-up of the cohort members was executed by record linkage to the national population register, from which the dates of death and emigration were obtained. Another linkage was performed with the national register of causes of death in order to get the deaths from coronary heard disease identified. As a result a data file `occoh.txt` was completed containing the following variables:

`id`	= identification number,
`birth`	= date of birth,
`entry`	= date of recruitment and baseline measurements,
`exit`	= date of exit from mortality follow-up,
`death`	= indicator for vital status at the end of follow-up,
	= 1, if dead from any cause, and = 0, if alive,
`chdeath`	= indicator for death from coronary heart disease,
	= 1, if "yes", and 0, if "no".

This exercise is divided into five main parts:

(1) Description of the study base or the follow-up experience of the whole cohort, identification of the cases and illustrating the risk sets.

(2) Nested case-control study within the cohort: (i) selection of controls by risk set or time-matched sampling using function `ccwc()` in package `Epi`, (ii) collection of exposure data for cases and controls from the pertinent data base of the whole cohort to the case-control data set using function `merge()`, and (iii) analysis of case-control data using function `clogit()` in package `survival()`,

(3) Case-cohort study within the cohort: (i) selection of a subcohort by simple random sampling from the cohort, (ii) fitting the Cox model to the data by weighted partial likelihood using function `cch()` in package `survival()`.

(4) Comparison of results from these analyses, also with those from a full cohort design.

(5) Further tasks and homework.

2.14.1 Reading the cohort data, illustrating the study base and risk sets

1. Load the packages `Epi` and `survival`. Read in the cohort data file and name the resulting data frame as `oc`. See its structure and print the univariate summaries.

```
> library(Epi)
> library(survival)
> oc <- read.table("./data/occoh.txt",header=T)
> str(oc)
```

```
'data.frame':        1501 obs. of  6 variables:
 $ id     : int  1 2 3 4 5 6 7 8 9 10 ...
 $ birth  : Factor w/ 1349 levels "1929-08-01","1929-12-23",..: 811 230 537 566 266 33
 $ entry  : Factor w/ 302 levels "1990-08-14","1990-08-15",..: 1 1 1 1 1 1 2 2 2 2 ...
 $ exit   : Factor w/ 290 levels "1992-02-25","1992-04-06",..: 290 290 290 290 203 229
 $ death  : int  0 0 0 0 1 1 1 1 0 0 ...
 $ chdeath: int  0 0 0 0 0 0 0 1 0 0 ...
```

```
> summary(oc)
```

```
       id              birth             entry              exit
 Min.   :   1   1931-02-19:   3   1990-08-18:  12   2009-12-31:1205
 1st Qu.: 376   1931-08-24:   3   1991-04-10:  12   2000-01-23:   2
 Median : 751   1933-02-28:   3   1991-04-24:  11   2000-10-04:   2
 Mean   : 751   1939-04-25:   3   1991-12-18:  11   2001-10-13:   2
 3rd Qu.:1126   1941-07-01:   3   1990-11-07:  10   2008-02-09:   2
 Max.   :1501   1943-04-16:   3   1991-03-30:  10   2008-03-23:   2
                (Other)   :1483   (Other)   :1435   (Other)   : 286
     death           chdeath
 Min.   :0.0000   Min.   :0.00000
 1st Qu.:0.0000   1st Qu.:0.00000
 Median :0.0000   Median :0.00000
 Mean   :0.1972   Mean   :0.07995
 3rd Qu.:0.0000   3rd Qu.:0.00000
 Max.   :1.0000   Max.   :1.00000
```

2. It is convenient to change all the dates into fractional calendar years

```
> oc$ybirth <- cal.yr(oc$birth)
> oc$yentry <- cal.yr(oc$entry)
> oc$yexit <- cal.yr(oc$exit)
```

We shall also compute the age at entry and at exit, respectively, as age will be the main time scale in our analyses.

```
> oc$agentry <- oc$yentry - oc$ybirth
> oc$agexit <- oc$yexit - oc$ybirth
```

3. As the next step we shall create a `lexis` object from the data frame along the calendar period and age axes, and as the outcome event we specify the coronary death.

```
> oc.lex <- Lexis( entry = list( per = yentry,
+                                age = yentry - ybirth ),
+                  exit = list( per = yexit),
+            exit.status = chdeath,
+                     id = id, data = oc)
> str(oc.lex)
```

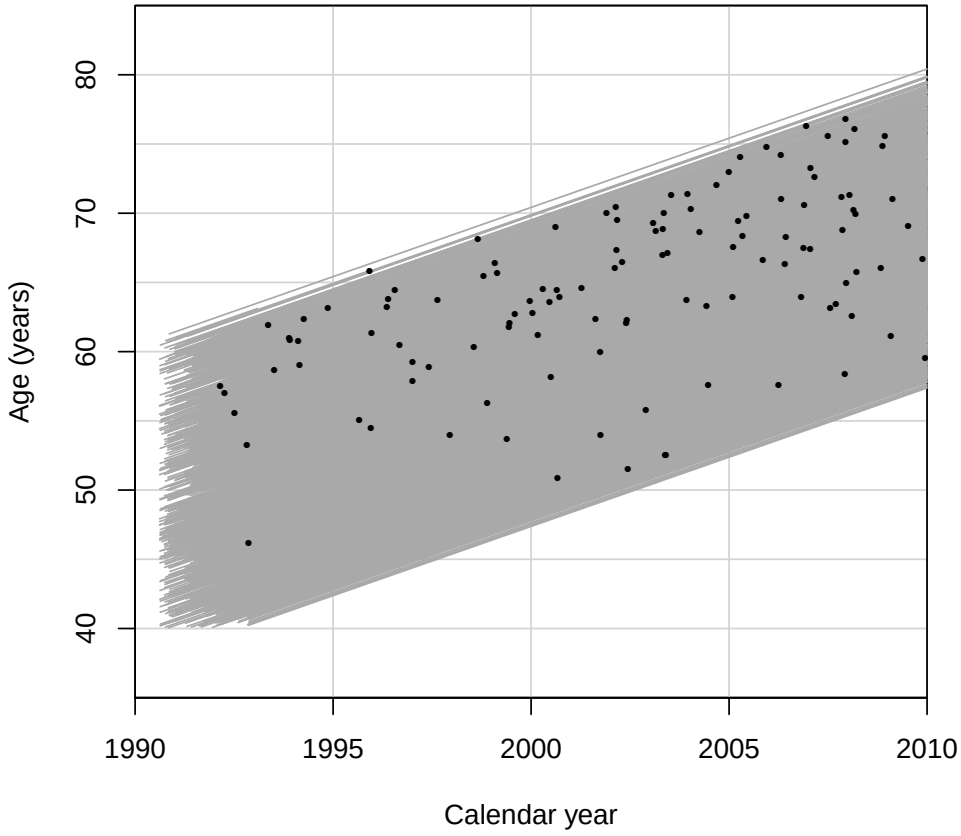
```
Classes 'Lexis' and 'data.frame':        1501 obs. of  17 variables:
 $ per    :Classes 'cal.yr', 'numeric'  num [1:1501] 1991 1991 1991 1991 1991 ...
 $ age    :Classes 'cal.yr', 'numeric'  num [1:1501] 47.5 56.1 51.4 51.1 55.5 ...
 $ lex.dur:Classes 'cal.yr', 'numeric'  num [1:1501] 19.4 19.4 19.4 19.4 15.6 ...
 $ lex.Cst: num  0 0 0 0 0 0 0 0 0 0 ...
 $ lex.Xst: int  0 0 0 0 0 0 0 1 0 0 ...
 $ lex.id : int  1 2 3 4 5 6 7 8 9 10 ...
 $ id     : int  1 2 3 4 5 6 7 8 9 10 ...
 $ birth  : Factor w/ 1349 levels "1929-08-01","1929-12-23",..: 811 230 537 566 266 33
 $ entry  : Factor w/ 302 levels "1990-08-14","1990-08-15",..: 1 1 1 1 1 1 2 2 2 2 ...
 $ exit   : Factor w/ 290 levels "1992-02-25","1992-04-06",..: 290 290 290 290 203 229
 $ death  : int  0 0 0 0 1 1 1 1 0 0 ...
 $ chdeath: int  0 0 0 0 0 0 0 1 0 0 ...
 $ ybirth :Classes 'cal.yr', 'numeric'  num [1:1501] 1943 1935 1939 1940 1935 ...
 $ yentry :Classes 'cal.yr', 'numeric'  num [1:1501] 1991 1991 1991 1991 1991 ...
 $ yexit  :Classes 'cal.yr', 'numeric'  num [1:1501] 2010 2010 2010 2010 2006 ...
 $ agentry:Classes 'cal.yr', 'numeric'  num [1:1501] 47.5 56.1 51.4 51.1 55.5 ...
 $ agexit :Classes 'cal.yr', 'numeric'  num [1:1501] 66.9 75.5 70.8 70.5 71.1 ...
 - attr(*, "time.scales")= chr  "per" "age"
 - attr(*, "time.since")= chr  "" ""
 - attr(*, "breaks")=List of 2
  ..$ per: NULL
  ..$ age: NULL
```

```
> summary(oc.lex)
```

```
Transitions:
     To
From    0   1  Records:  Events: Risk time:  Persons:
   0 1381 120       1501      120   25280.91      1501
```

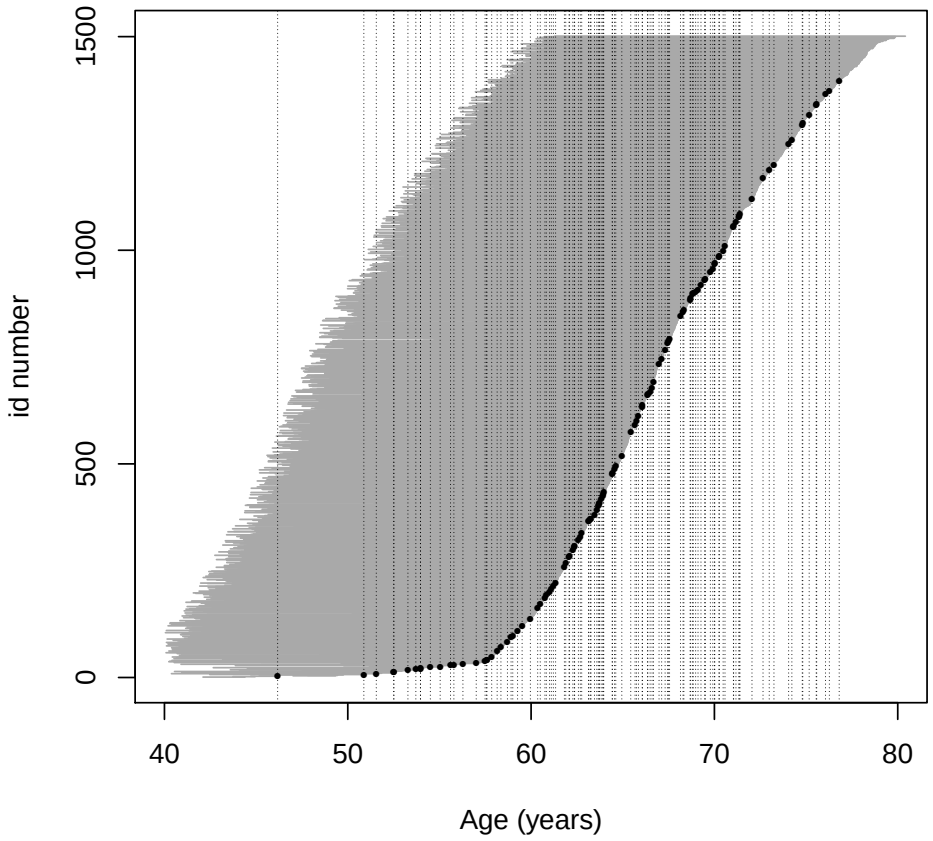
Now we plot the follow-up lines and outcome cases in a conventional Lexis diagram

```
> plot(oc.lex,
+   grid = list( seq(1990, 2010, 5), seq( 35, 85, 5)),
+          xlim = c(1990, 2010), ylim = c(35, 85),
+          lty.grid = 1, xaxs='i', yaxs = 'i',
+          xlab = "Calendar year", ylab = "Age (years)")
> points( oc.lex,  pch = c(NA, 16)[oc.lex$lex.Xst+1], cex=0.5)
```



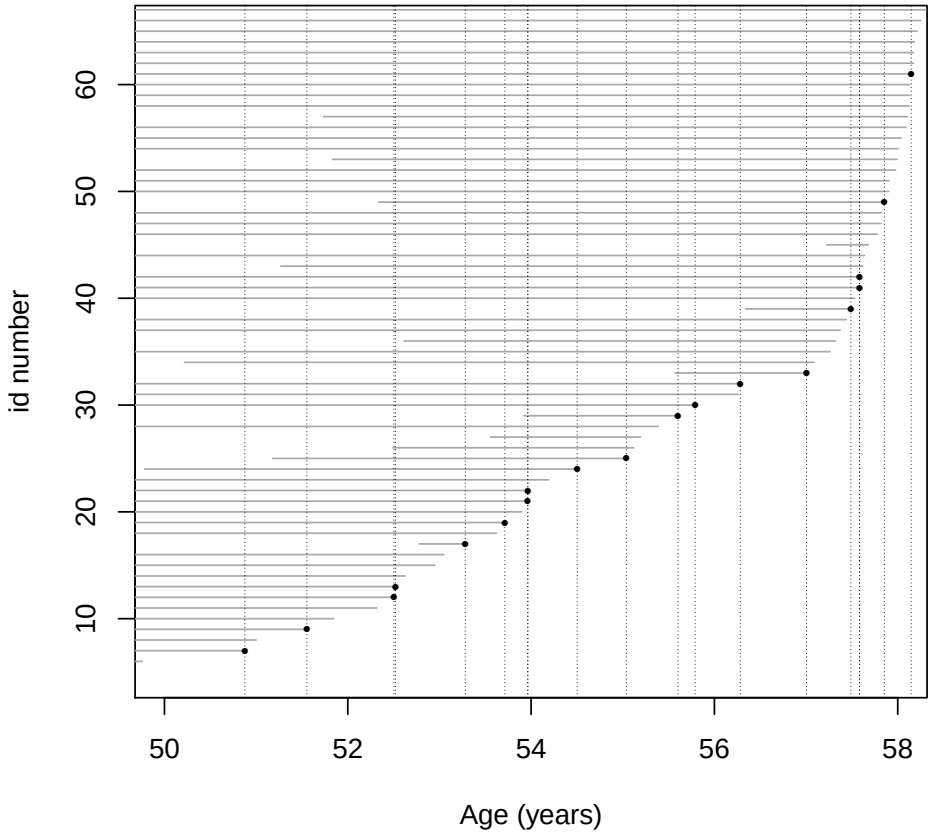
4. As age is here the main time axis, we shall illustrate the *study base* or the follow-up lines and outcome events along the age scale, being ordered by age at exit. Vertical lines at those ages when new coronary deaths occur are drawn to identify the pertinent *risk sets*. For that purpose it is useful first to sort the data frame and the `lexis` object jointly by age at exit & age at entry, and to give a new ID number according to that order.

```
> oc.ord <- cbind(ID = 1:1501, oc[ order( oc$agexit, oc$agentry), ] )
> oc.lexord <- Lexis( entry = list( age = agentry ),
+                        exit = list( age = agexit),
+                exit.status = chdeath,
+                          id = ID, data = oc.ord)
> plot(oc.lexord,
+    time.scale = "age",
+          xlim = c(40, 80),
+          xlab = "Age (years)" )
> points(oc.lexord, time.scale = "age",
+     pch = c(NA, 16)[oc.lexord$lex.Xst+1], cex=0.5)
> with( subset(oc.lexord, lex.Xst==1),
+    abline( v = agexit, lty = 3, lwd = 0.5 ))
```



Now we zoom the graphical illustration of the risk sets into event times occurring between 50 to 58 years.

```
> plot(oc.lexord,
+    time.scale = "age", ylim = c(5, 65),
+          xlim = c(50, 58),
+          xlab = "Age (years)" )
> points(oc.lexord, time.scale = "age",
+     pch = c(NA, 16)[oc.lexord$lex.Xst+1], cex=0.5)
> with( subset(oc.lexord, lex.Xst==1),
+    abline( v = agexit, lty = 3, lwd = 0.5 ) )
```



2.14.2 Nested case-control study

We shall now employ the strategy of *risk-set sampling* or *time-matched* sampling of controls, *i.e.* we are conducting a *nested case-control study* within the cohort.

1. The risk sets are defined according to the age at diagnosis of the case. Further matching is applied for age at entry by 1-year agebands. For this purpose we first generate a categorical variable `agen2` for age at entry

   ```
   > oc.lex$agen2 <- cut(oc.lex$agentry, br = seq(40, 62, 1) )
   ```

   Matched sampling from risk sets may be carried out using function `ccwc()` found in the `Epi` package. Its main arguments are the times of `entry` and `exit` which specify the time at risk along the main time scale (here age), and the outcome variable to be given in the `fail` argument. The number of controls per case is set to be two, and the additional matching factor is given. – After setting the RNG seed (with your own number), make a call of this function and see the structure of the resulting data frame `cactrl` containing the cases and the chosen individual controls.

   ```
   > set.seed(9863157)
   > cactrl <-
   ```

```
+    ccwc(entry=agentry, exit=agexit, fail=chdeath,
+         controls = 2, match= agen2,
+         include = list(id, agentry),
+         data=oc.lex, silent=F)

Sampling risk sets: ..............................................................

> str(cactrl)

'data.frame':        360 obs. of  7 variables:
 $ Set    : num  1 1 1 2 2 2 3 3 3 4 ...
 $ Map    : num  8 1155 614 95 204 ...
 $ Time   : num  63.9 63.9 63.9 66.7 66.7 ...
 $ Fail   : num  1 0 0 1 0 0 1 0 0 1 ...
 $ agen2  : Factor w/ 22 levels "(40,41]","(41,42]",..: 8 8 8 8 8 8 8 8 8 8 ...
 $ id     : int  8 1155 614 95 204 292 115 351 526 504 ...
 $ agentry: num  47.7 47 47.4 47.5 47.6 ...
```

Check the meaning of the four first columns of the case-control data frame from the help page of function `ccwc()`.

2. Now we shall start collecting data on the risk factors for the cases and their matched controls, including determination of the total cholesterol levels from the frozen sera! The storehouse of the risk factor measurements for the whole cohort is file `occoh-Xdata.txt`. It contains values of the following variables.

`id` =	identification number, the same as in `occoh.txt`,
`smok` =	cigarette smoking with categories, 1: "never", 2: "former", 3: "1-14/d", 4: "15+/d",
`sbp` =	systolic blood pressure (mmHg),
`tchol` =	total cholesterol level (mmol/l).

```
> ocX <- read.table("./data/occoh-Xdata.txt", header=T)
> str(ocX)

'data.frame':        1501 obs. of  6 variables:
 $ id   : int  1 2 3 4 5 6 7 8 9 10 ...
 $ birth: Factor w/ 1349 levels "1929-08-01","1929-12-23",..: 811 230 537 566 266 336
 $ entry: Factor w/ 302 levels "1990-08-14","1990-08-15",..: 1 1 1 1 1 1 2 2 2 2 ...
 $ smok : int  4 3 3 1 2 2 1 2 1 1 ...
 $ sbp  : int  130 128 157 102 138 119 155 154 164 124 ...
 $ tchol: num  7.56 6.55 8.13 5.93 7.92 5.9 7.28 7.43 5.34 6.24 ...
```

3. In the next step we collect the values of the risk factors for our cases and controls by merging the case-control data frame and the storehouse file. In this operation we use the `Map` variable of `cactrl` and `id` variable in `ocX` as the keys to link each individual case and control with his own data on risk factors.

```
> oc.ncc <- merge(cactrl, ocX[, c("id", "smok", "tchol", "sbp")],
+    by.x = "Map", by.y = "id")
> str(oc.ncc)

'data.frame':        360 obs. of  10 variables:
 $ Map    : num  2 4 8 13 15 37 41 42 43 47 ...
 $ Set    : num  16 93 1 25 50 5 5 9 18 96 ...
 $ Time   : num  59 63.7 63.9 70.2 60.8 ...
 $ Fail   : num  0 0 1 0 0 0 1 1 1 0 ...
 $ agen2  : Factor w/ 22 levels "(40,41]","(41,42]",..: 17 12 8 15 19 1 1 17 15 21 ...
 $ id     : int  2 4 8 13 15 37 41 42 43 47 ...
 $ agentry: num  56.1 51.1 47.7 54.9 59 ...
 $ smok   : int  3 1 2 2 2 3 4 2 3 1 ...
 $ tchol  : num  6.55 5.93 7.43 5.28 6.03 5.15 6.09 5.41 5.72 6.22 ...
 $ sbp    : int  128 102 154 153 147 116 125 156 128 154 ...
```

4. We shall treat all the risk factors as categorical, such that grouping of total cholesterol and systolic blood pressure will be as follows:

   `cholgrp` = cholesterol class, 1: "<5", 2: "5-<6.5", 3: ">=6.5",
   `sbpgrp` = blood pressure class, 1: "<130", 2: "130-<150", 3: "150-<170", 4: ">=170".

   Create the factors for cholesterol and blood pressure according to the above categorization and convert the smoking variable into a factor, too.

```
> oc.ncc$cholgrp <- cut( oc.ncc$tchol, br = c(2.4, 5, 6.5, 13),
+           include.lowest = T, right = F )
> oc.ncc$sbpgrp <- cut( oc.ncc$sbp, br = c(95, 130, 150, 170, 240),
+           include.lowest = T, right = F )
> oc.ncc$smok <- factor(oc.ncc$smok,
+     labels = c("never", "ex", "1-14/d", ">14/d"))
```

5. It is useful to start the analysis of case-control data by simple tabulations by the categorized risk factors. Crude estimates of the rate ratios associated with them, in which matching is ignored, can be obtained as instructed in Bendix's lecture on Poisson and logistic models on Saturday 26 May.

```
> stat.table( index = list( smok, Fail ),
+           contents = list( count(), percent(smok) ),
+            margins = T, data = oc.ncc )
```

	Fail		
smok	0	1	Total
never	89	31	120
	37.1	25.8	33.3
ex	44	19	63
	18.3	15.8	17.5
1-14/d	73	42	115
	30.4	35.0	31.9

```
 >14/d          34       28       62
              14.2     23.3     17.2


 Total         240      120      360
             100.0    100.0    100.0
 ----------------------------------
```

```
> smok.crncc <- glm( Fail ~ smok, family=binomial, data = oc.ncc)
> round(ci.exp(smok.crncc ), 3)
```

```
            exp(Est.)  2.5% 97.5%
(Intercept)     0.348 0.231 0.524
smokex          1.240 0.631 2.437
smok1-14/d      1.652 0.946 2.885
smok>14/d       2.364 1.239 4.511
```

Do the same for categorized total cholesterol and systolic blood pressure.

```
 -------------------------------------
           ----------Fail-----------
 cholgrp          0       1   Total
 -------------------------------------
 [2.4,5)         68      21      89
               28.3    17.5    24.7

 [5,6.5)        126      62     188
               52.5    51.7    52.2

 [6.5,13]        46      37      83
               19.2    30.8    23.1


 Total          240     120     360
              100.0   100.0   100.0
 -------------------------------------
```

```
                exp(Est.)  2.5% 97.5%
(Intercept)         0.309 0.189 0.504
cholgrp[5,6.5)      1.593 0.896 2.835
cholgrp[6.5,13]     2.605 1.355 5.005
```

```
 -------------------------------------
            ----------Fail-----------
 sbpgrp           0       1   Total
 -------------------------------------
 [95,130)        51      27      78
               21.2    22.5    21.7

 [130,150)      104      36     140
               43.3    30.0    38.9
```

```
 [150,170)        59      31      90
                 24.6    25.8    25.0

 [170,240]        26      26      52
                 10.8    21.7    14.4


 Total           240     120     360
                100.0   100.0   100.0
 -------------------------------------


                exp(Est.)  2.5% 97.5%
(Intercept)         0.529 0.332 0.844
sbpgrp[130,150)     0.654 0.358 1.193
sbpgrp[150,170)     0.992 0.525 1.878
sbpgrp[170,240]     1.889 0.923 3.866
```

6. A proper analysis takes into account matching that was employed in the selection of controls for each case from the pertinent risk set further restricted to subjects who were about the same age at entry as the case was. In this analysis function `clogit()` in `survival` package is utilized. It is in fact a wrapper of function `coxph()`.

```
> m.clogit <- clogit( Fail ~ smok + sbpgrp + cholgrp +
+        strata(Set), data = oc.ncc )
> summary(m.clogit)


Call:
coxph(formula = Surv(rep(1, 360L), Fail) ~ smok + sbpgrp + cholgrp +
    strata(Set), data = oc.ncc, method = "exact")

  n= 360, number of events= 120

                    coef exp(coef) se(coef)      z Pr(>|z|)
smokex           0.03128   1.03178  0.35127  0.089  0.92903
smok1-14/d       0.49922   1.64743  0.30761  1.623  0.10461
smok>14/d        0.91320   2.49228  0.33566  2.721  0.00652
sbpgrp[130,150) -0.29427   0.74507  0.32496 -0.906  0.36517
sbpgrp[150,170)  0.05999   1.06182  0.35328  0.170  0.86517
sbpgrp[170,240]  0.95570   2.60050  0.40644  2.351  0.01870
cholgrp[5,6.5)   0.46672   1.59476  0.29528  1.581  0.11396
cholgrp[6.5,13]  1.13499   3.11114  0.38661  2.936  0.00333

                exp(coef) exp(-coef) lower .95 upper .95
smokex             1.0318     0.9692    0.5183     2.054
smok1-14/d         1.6474     0.6070    0.9015     3.011
smok>14/d          2.4923     0.4012    1.2909     4.812
sbpgrp[130,150)    0.7451     1.3421    0.3941     1.409
sbpgrp[150,170)    1.0618     0.9418    0.5313     2.122
sbpgrp[170,240]    2.6005     0.3845    1.1724     5.768
cholgrp[5,6.5)     1.5948     0.6271    0.8940     2.845
cholgrp[6.5,13]    3.1111     0.3214    1.4583     6.637

Rsquare= 0.077   (max possible= 0.519 )
```

```
Likelihood ratio test= 28.68  on 8 df,   p=0.0003606
Wald test            = 24.43  on 8 df,   p=0.00194
Score (logrank) test = 27.61  on 8 df,   p=0.0005546
```

```
> round(ci.exp(m.clogit ), 3)
```

```
                exp(Est.)  2.5% 97.5%
smokex              1.032 0.518 2.054
smok1-14/d          1.647 0.902 3.011
smok>14/d           2.492 1.291 4.812
sbpgrp[130,150)     0.745 0.394 1.409
sbpgrp[150,170)     1.062 0.531 2.122
sbpgrp[170,240]     2.600 1.172 5.768
cholgrp[5,6.5)      1.595 0.894 2.845
cholgrp[6.5,13]     3.111 1.458 6.637
```

Compare these with the crude estimates obtained above.

2.14.3 Case-cohort study

Now we start applying the second major outcome-selective sampling strategy for collecting exposure data from a big study population

1. The subcohort is selected as a simple random sample ($n = 260$) from the whole cohort. The `id`-numbers of the individuals that are selected will be stored in vector `subcids`, and `subcind` is an indicator for inclusion to the subcohort.

   ```
   > N <- 1501; n <- 260
   > set.seed(1579863)
   > subcids <- sample(N, n )
   > oc.lex$subcind <- 1*(oc.lex$id %in% subcids)
   ```

2. We form the data frame `oc.cc` to be used in the subsequent analysis selecting the union of the subcohort members and the case group from the data frame of the full cohort. After that we collect the data of the risk factors from the data storehouse for the subjects in the case-cohort data

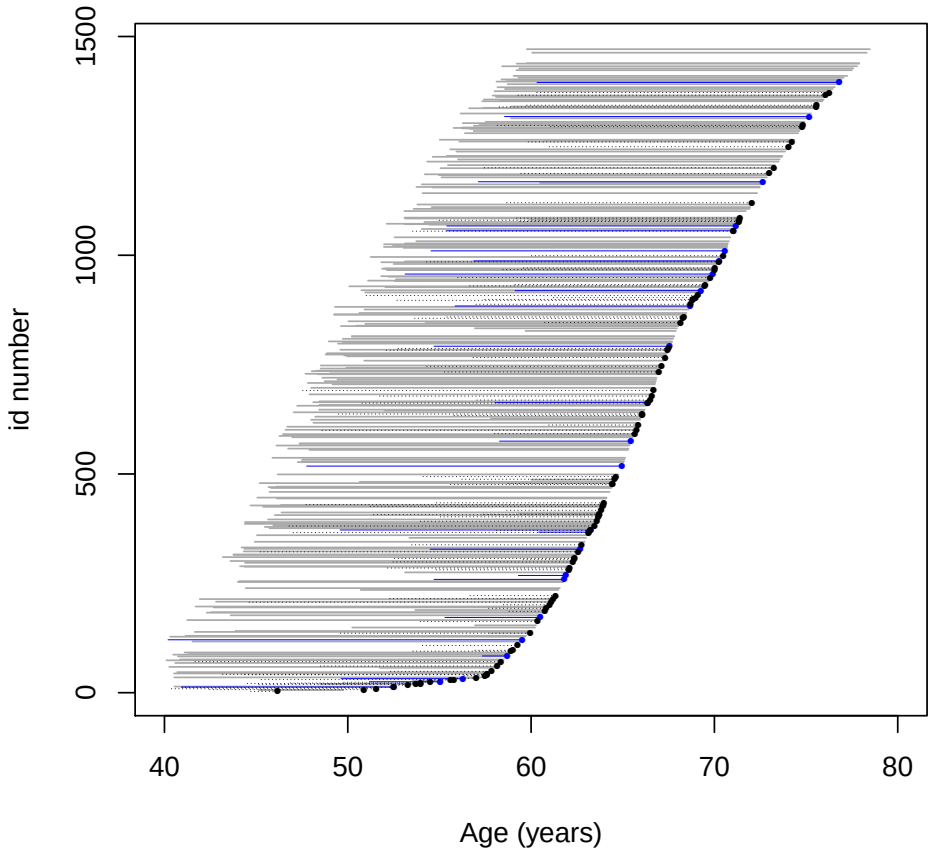
   ```
   > oc.cc <- subset( oc.lex, subcind==1 | chdeath ==1)
   > oc.cc <- merge( oc.cc, ocX[, c("id", "smok", "tchol", "sbp")],
   +    by.x = "id", by.y = "id")
   > str(oc.cc)
   ```

   ```
   Classes 'Lexis' and 'data.frame':        355 obs. of  22 variables:
    $ id     : int  6 8 18 27 28 37 40 41 42 43 ...
    $ per    : num  1991 1991 1991 1991 1991 ...
    $ age    : num  54.4 47.7 50.1 49.3 58.4 ...
    $ lex.dur: num  16.8 16.2 19.4 19.4 19.4 ...
    $ lex.Cst: num  0 0 0 0 0 0 0 0 0 0 ...
    $ lex.Xst: int  0 1 0 0 0 0 0 1 1 1 ...
    $ lex.id : int  6 8 18 27 28 37 40 41 42 43 ...
    $ birth  : Factor w/ 1349 levels "1929-08-01","1929-12-23",..: 336 790 639 680 101 12
   ```

```
$ entry  : Factor w/ 302 levels "1990-08-14","1990-08-15",..: 1 2 3 4 4 6 6 6 6 6 ...
$ exit   : Factor w/ 290 levels "1992-02-25","1992-04-06",..: 229 218 290 290 290 290
$ death  : int  1 1 0 0 0 0 1 1 1 1 ...
$ chdeath: int  0 1 0 0 0 0 0 1 1 1 ...
$ ybirth : num  1936 1943 1941 1941 1932 ...
$ yentry : num  1991 1991 1991 1991 1991 ...
$ yexit  : num  2007 2007 2010 2010 2010 ...
$ agentry: num  54.4 47.7 50.1 49.3 58.4 ...
$ agexit : num  71.3 63.9 69.5 68.7 77.8 ...
$ agen2  : Factor w/ 22 levels "(40,41]","(41,42]",..: 15 8 11 10 19 1 6 1 17 15 ...
$ subcind: num  1 0 1 1 1 1 1 1 1 0 ...
$ smok   : int  2 2 1 4 1 3 4 4 2 3 ...
$ tchol  : num  5.9 7.43 3.34 8.31 4.56 5.15 5.88 6.09 5.41 5.72 ...
$ sbp    : int  119 154 130 140 230 116 141 125 156 128 ...
- attr(*, "breaks")=List of 2
 ..$ per: NULL
 ..$ age: NULL
- attr(*, "time.scales")= chr  "per" "age"
- attr(*, "time.since")= chr  "" ""
```

3. The next five lines of R script provide a graphical illustration of the lifelines contained in the case-cohort data. Lines for the subcohort non-cases are grey without bullet at exit, those for subcohort cases are blue with blue bullet at exit, and for cases outside the subcohort the lines are black and dotted with black bullets at exit.

```
> plot(subset(oc.lexord, chdeath==0 & id %in% subcids),
+    time.scale = "age",
+          xlim = c(40, 80), # ylim = c(18,210),
+          xlab = "Age (years)" )
> lines(subset(oc.lexord, chdeath==1 & id %in% subcids ),
+    time.scale = "age", lwd=0.5, col = 'blue' )
> points(subset(oc.lexord, chdeath==1 & id %in% subcids ),
+     time.scale = "age", pch = 16, col='blue',  cex=0.5)
> lines(subset(oc.lexord, chdeath==1 & !(id %in% subcids) ),
+    time.scale = "age", lty = 3, lwd=0.5, col = 'black' )
> points(subset(oc.lexord, chdeath==1 & !(id %in% subcids) ),
+     time.scale = "age", pch = 16, col='black',  cex=0.5)
```



4. Define the categorical explanatory variables again.

```
> oc.cc$smok <- factor(oc.cc$smok,
+     labels = c("never", "ex", "1-14/d", ">14/d"))
> oc.cc$cholgrp <- cut( oc.cc$tchol, br = c(2.4, 5, 6.5, 13),
+           include.lowest = T, right = F )
> oc.cc$sbpgrp <- cut( oc.cc$sbp, br = c(95, 130, 150, 170, 240),
+           include.lowest = T, right = F )
```

5. To estimate jointly the rate ratios associated with the categorized risk factors we now fit the pertinent Cox model applying the method of *weighted partial likelihood* as presented by Ling & Ying (1993) and Barlow (1994). The weights for all cases and non-cases in the subcohort are first computed and added to the data frame.

```
> N.nonc <- N-sum(oc.lex$chdeath)  # non-cases in whole cohort
> n.nonc <- sum(oc.cc$subcind * (1-oc.cc$chdeath)) # non-cases in subcohort
> wn <- N.nonc/n.nonc           # weight for non-cases in subcohort
> c(N.nonc, n.nonc, wn)

[1] 1381.000000  235.000000    5.876596

> oc.cc$w <- ifelse(oc.cc$subcind==1 & oc.cc$chdeath==0, wn, 1)
```

6. Fitting the Cox model can now be done using function `cch()` in package `survival` with `method = "LinYing"` as follows:

```
> oc.cc$surob <- with(oc.cc, Surv( agentry, agexit, chdeath) )
> cch.LY <- cch( surob ~  smok + sbpgrp + cholgrp, stratum=NULL,
+    subcoh = ~subcind, id = ~id,  cohort.size = N, data = oc.cc,
+     method ="LinYing" )
> summary(cch.LY)


Case-cohort analysis,x$method, LinYing
 with subcohort of 260 from cohort of 1501

Call: cch(formula = surob ~ smok + sbpgrp + cholgrp, data = oc.cc,
    subcoh = ~subcind, id = ~id, stratum = NULL, cohort.size = N,
    method = "LinYing")

Coefficients:
                   Coef    HR  (95%   CI)     p
smokex          -0.091 0.913 0.460 1.815 0.796
smok1-14/d       0.486 1.626 0.895 2.954 0.111
smok>14/d        1.012 2.752 1.408 5.382 0.003
sbpgrp[130,150) -0.300 0.741 0.394 1.394 0.353
sbpgrp[150,170)  0.066 1.068 0.548 2.081 0.846
sbpgrp[170,240]  0.608 1.837 0.898 3.757 0.096
cholgrp[5,6.5)   0.648 1.912 1.060 3.451 0.031
cholgrp[6.5,13]  1.191 3.289 1.667 6.492 0.001
```

2.14.4 Full cohort analysis and comparisons

Finally, suppose the investigators could afford to collect the data of risk factors from the storehouse for the whole cohort.

1. Let us form the data frame corresponding to the full cohort design and create again the categorical risk factors.

```
> oc.full <- merge( oc.lex, ocX[, c("id", "smok", "tchol", "sbp")],
+    by.x = "id", by.y = "id")
> oc.full$smok <- factor(oc.full$smok,
+     labels = c("never", "ex", "1-14/d", ">14/d"))
> oc.full$cholgrp <- cut( oc.full$tchol, br = c(2.4, 5, 6.5, 13),
+           include.lowest = T, right = F )
> oc.full$sbpgrp <- cut( oc.full$sbp, br = c(95, 130, 150, 170, 240),
+           include.lowest = T, right = F )
```

2. Now we can fit the Cox model to the full cohort, and there is no need to employ extra tricks upon the ordinary `coxph()` fit.

```
> cox.coh <- coxph( Surv(agentry, agexit, chdeath) ~
+         smok + sbpgrp + cholgrp, data = oc.full)
> summary(cox.coh)
```

```
Call:
coxph(formula = Surv(agentry, agexit, chdeath) ~ smok + sbpgrp +
    cholgrp, data = oc.full)

  n= 1501, number of events= 120

                   coef exp(coef) se(coef)      z Pr(>|z|)
smokex           0.1207    1.1283   0.2922  0.413 0.679558
smok1-14/d       0.6844    1.9827   0.2382  2.873 0.004066
smok>14/d        0.9411    2.5628   0.2623  3.587 0.000334
sbpgrp[130,150) -0.2744    0.7600   0.2556 -1.074 0.282927
sbpgrp[150,170)  0.1441    1.1551   0.2650  0.544 0.586453
sbpgrp[170,240]  0.6080    1.8367   0.2785  2.183 0.029028
cholgrp[5,6.5)   0.5780    1.7824   0.2532  2.283 0.022454
cholgrp[6.5,13]  1.0527    2.8653   0.2746  3.833 0.000127

                 exp(coef) exp(-coef) lower .95 upper .95
smokex               1.128     0.8863    0.6363     2.001
smok1-14/d           1.983     0.5044    1.2430     3.163
smok>14/d            2.563     0.3902    1.5326     4.286
sbpgrp[130,150)      0.760     1.3158    0.4605     1.254
sbpgrp[150,170)      1.155     0.8658    0.6871     1.942
sbpgrp[170,240]      1.837     0.5444    1.0641     3.170
cholgrp[5,6.5)       1.782     0.5610    1.0851     2.928
cholgrp[6.5,13]      2.865     0.3490    1.6726     4.909

Concordance= 0.687  (se = 0.029 )
Rsquare= 0.029   (max possible= 0.653 )
Likelihood ratio test= 43.43  on 8 df,   p=7.276e-07
Wald test            = 43.03  on 8 df,   p=8.688e-07
Score (logrank) test = 45.44  on 8 df,   p=3.041e-07
```

3. Lastly, a comparison of the point estimates and standard errors between the different designs, can be performed.

```
> betas <- round(cbind( coef(cox.coh),
+      coef(m.clogit), coef(cch.LY) ), 3)
> colnames(betas) <-  c("coh", "ncc", "cch.LY")
> betas
```

```
                   coh    ncc cch.LY
smokex           0.121  0.031 -0.091
smok1-14/d       0.684  0.499  0.486
smok>14/d        0.941  0.913  1.012
sbpgrp[130,150) -0.274 -0.294 -0.300
sbpgrp[150,170)  0.144  0.060  0.066
sbpgrp[170,240]  0.608  0.956  0.608
cholgrp[5,6.5)   0.578  0.467  0.648
cholgrp[6.5,13]  1.053  1.135  1.191
```

```
> SEs <- round(cbind( sqrt(diag(cox.coh$var)),
+     sqrt(diag(m.clogit$var)),
+     sqrt(diag(cch.LY$var)) ), 3)
```

```
> colnames(SEs) <- c("coh", "ncc", "cch-LY")
> SEs
```

```
                   coh   ncc cch-LY
smokex           0.292 0.351  0.350
smok1-14/d       0.238 0.308  0.305
smok>14/d        0.262 0.336  0.342
sbpgrp[130,150) 0.256 0.325  0.322
sbpgrp[150,170) 0.265 0.353  0.340
sbpgrp[170,240] 0.279 0.406  0.365
cholgrp[5,6.5)   0.253 0.295  0.301
cholgrp[6.5,13] 0.275 0.387  0.347
```

You will notice that the point estimates of the coefficients obtained from the full cohort, nested case-control, and case-cohort analyses, respectively, are somewhat variable.

However, the standard errors from the NCC and CC analyses are relatively similar.

2.14.5 Further exercises and homework

1. If you have time, you could run both the NCC study and CC study again but now with a larger control group or subcohort; for example 4 controls per case in NCC and $n = 520$ as the subcohort size in CC. Remember resetting the seed first. Pay attention in the results to how much closer will be the point estimates and the proper SEs to those obtained from the full cohort design.

2. A popular alternative to weighted partial likelihood in the analysis of case-cohort data is the *pseudo-likelihood method* (Prentice 1986), which is based on "late entry" to follow-up of the case subjects not belonging to the subcohort. This method is implemented in function `cch()` which you can apply directly to the case-cohort data `oc.cc` as before but now with `method = "Prentice"`. – Try this and compare the results with those obtained by weighted partial likelihood in model `cch.LY`.

3. Yet another computational solution for maximizing weighted partial likelihood is provided by a combination of functions `twophase()` and `svycoxph()` of the `survey` package. In this approach, the starting point is a data frame, named e.g. `oc.lex2`, that contains the whole cohort but includes also necessary columns for the interesting risk factors, such that their values are known for the cases and the subcohort members but are missing (`NA`) for non-cases outside the subcohort. The case-cohort design embedded in the cohort is viewed as a special case of a *two-phase sampling design*. The pertinent design features are determined by the key indicator variables in `oc.lex2`: subcohort membership (cf. variable `subcind` above) and case status (`chdeath`). Based on these items, a call of function `twophase()` with appropriate arguments creates an object of class `twophase`, named e.g. `oc.2ph`, that contains the original data frame and all the survey design information (like weights) needed to analyse it. After that, function `svycoxph()`, which is a "survey design version" of `coxph()`, is called with a similar model formula as before but in which, instead of the `data` argument, one would specify `design = oc.2ph`. The approach is illustrated

with an example in a vignette "Two-phase designs in epidemiology" by Thomas Lumley (see `http://cran.r-project.org/web/packages/survey/vignettes/epi.pdf`, p. 4–7). – Try this at home and check that you would obtain similar results as with model `cch.LY`.

2.15 Renal complications: Time-dependent variables and multiple states