

Statistical Practice in Epidemiology

with R

Computer exercises

www.pubhealth.ku.dk/~bxc/SPE

Department of Public Health &
Institute of Mathematical Statistics
University of Tartu, Estonia

May 2007

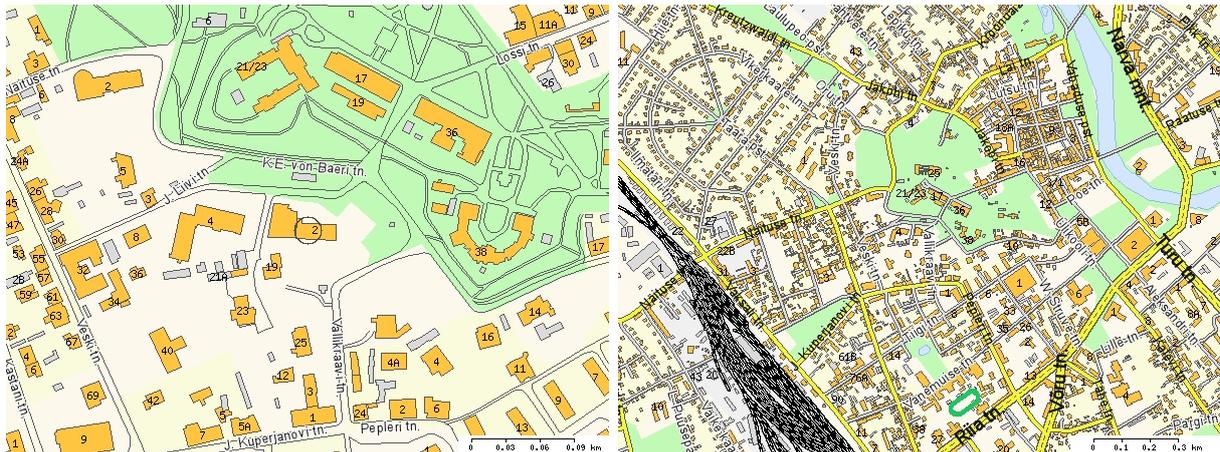
- Bendix Carstensen Steno Diabetes Center, Gentofte, Denmark
& Dept. of Biostatistics, University of Copenhagen, Denmark
bxc@steno.dk
<http://www.biostat.ku.dk/~bxc>
- Peter Dalgaard Dept. of Biostatistics, University of Copenhagen, Denmark
pd@biostat.ku.dk
<http://www.biostat.ku.dk/~pd>
- Krista Fischer Institute of Public Health, University of Tartu, Estonia
Krista.Fischer@ut.ee
- Lyle Gurrin School of Population Health, University of Melbourne, Australia
lgurrin@unimelb.edu.au
<http://www.epi.unimelb.edu.au/about/staff/gurrin-lyle>
- Michael Hills (Retired) Highgate, London, UK
mhills@blueyonder.co.uk
<http://www.mhills.pwp.blueyonder.co.uk>
- 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
plummer@iarc.fr

The course venue is:

University of Tartu, Institute of Mathematical Statistics, J. Liivi 2, Tartu.
(2 in the circle on the map below).

Go to the Tartu homepage (www.tartu.ee), choose → and enter “Liivi 2” in the search box, click and afterwards click on the search results for Liivi 2.

Print the map and appear there Thursday morning at 8:30. See you!



Preliminary program:

Daily timetable

8:45 – 9:15	Recap of yesterday’s practicals
9:15 – 10:00	Lecture
10:00 – 10:30	Coffee break
10:30 – 12:30	Practical
12:30 – 14:00	Lunch at Café Wilde
14:00 – 15:00	Lecture
15:00 – 17:30	Practical (incl. coffee)

Friday 25/05

8:45 – 9:00	Welcome: Introduction of University of Tartu, the course faculty and students (KF).
9:00 – 9:30	History of R. Language concepts. Objects. Functions (PD).
9:30 – 10:00	Interface to other dataformats. Dataframes. Search path. Simple simulation (MP).
10:00 – 10:30	Coffee break
10:30 – 12:30	Practical: Reading data.
12:30 – 14:00	Lunch
14:00 – 14:30	Basic epidemiological concepts (MP).
14:30 – 17:30	Practical: Tabulation of data.
18:30 –	Welcome reception at ...

Saturday 26/05

- 8:45 – 9:15 Recap of yesterday’s practicals
- 9:15 – 9:30 Modelling tools and model objects (PD).
- 9:30 – 10:00 Logistic regression for cc-studies. Poisson regression for follow-up studies (BxC).
- 10:00 – 10:30 Parametrization of models (BxC).
- 10:30 – 11:00 Coffee break
- 11:00 – 12:30 Practical: Logistic regression for melanoma data (EL).
- 12:30 – 14:00 Lunch
- 14:00 – 15:00 Causal inference (KF).
- 15:00 – 17:30 Practical: Simulation illustrating causal inference and logistic regression.

Sunday 27/05

- 8:45 – 9:15 Recap of yesterday’s practicals
- 9:15 – 10:00 Graphics in R. (PD).
- 10:00 – 10:30 Coffee break
- 10:30 – 12:30 Practical: Graphics meccano.
- 13:00 – Lunch / Excursion by boat on Emaljõgi

Monday 28/05

- 8:45 – 9:00 Recap of yesterday’s practicals
- 9:00 – 9:45 Survival analysis in continuous time. Parametric survival models (KF).
- 9:45 – 10:10 Interval censoring (BxC).
- 10:10 – 10:40 Coffee break
- 10:40 – 12:30 Practical: Estonian stroke study.
- 12:30 – 14:00 Lunch
- 14:00 – 14:45 Timesplitting and SMR (BxC).
- 14:45 – 17:30 Practical: Time-splitting and SMR.

Tuesday 29/05

- 8:45 – 9:15 Recap of yesterday’s practicals
- 9:15 – 9:45 Nested and matched cc-studies (BxC).
- 9:45 – 10:15 Case-cohort studies (MP).
- 10:15 – 10:45 Coffee break
- 10:45 – 12:30 Practicals: Matched case-control study.
Case-cohort study (Norwegian malformations).
- 12:30 – 14:00 Lunch
- 14:00 – 16:00 Practical: Computer intensive methods: Bootstrap.
- 16:00 – 16:30 Competing risk models (BxC).
- 16:30 – 17:00 Multistage models (MP).
- 19:00 – Course dinner at ...

Wednesday 30/05

- 9:00 – 10:00 Practicals: Competing risks — histological subtypes of liver cancer.
or: Multistage model data (`heart` from the `msm` package).
- 10:00 – 11:00 Recap of yesterday's and today's practicals
- 11:00 – 11:30 Course evaluation and closure.
- 11:30 – 13:00 Lunch & departure.
- 13:00 – 18:00 Post-mortem (Faculty alone).
- 19:00 – Faculty dinner.

Contents

Introduction to practicals	vi
1 Exercises	1
1.1 Practice with basic R	1
1.2 Reading data into R	6
1.3 Tabulation in R	11
1.4 The measurement of effects	16
1.5 Logistic regression	20
1.6 Statistical simulation and causal inference	25
1.7 Graphics in R	27
1.8 Graphical meccano	31
1.9 Survival analysis: The Stroke dataset	33
1.10 Interval-censored data: Conversion to diabetes	34
1.11 Time-splitting and SMR: Thorotrast	36
1.12 Matched case-control study: <i>Salmonella</i> Typhimurium	41
1.13 Case-cohort study of congenital malformations	43
1.14 Multi-State Markov Models	45
1.15 The heart data set	45
1.16 Competing risks: The Danish Thorotrast study	49
2 Solutions	53
2.1 Practice with basic R	53
2.2 Reading data into R	61
2.3 Tabulation in R	65
2.4 Logistic regression (<code>glm</code>)	75
2.5 Interval-censored data: Conversion to diabetes	88
2.6 Time-splitting and SMR: The Thorotrast study	92
2.7 Matched case-control study: <i>Salmonella</i> Typhimurium	105

Introduction to practicals

Datasets for the practicals in this course will be available on the local machines and on the course homepage in the folder `www.pubhealth.ku.dk/~bxc/SPE/data`

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.diagram  
args( Lexis.diagram )
```

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

When running the exercises it is a good idea to have some text-editor open too, where you keep the R-commands. You can 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.

R has an inbuilt text-editor, accessible from the menu: `File`→`New script` or `File`→`Open script`. Type your R-commands in this, highlight them and press `Ctrl` `R` and they will be run.

Another slightly more fancy option is to install Tinn-R from <http://www.sciviews.org/Tinn-R/>. This is a separate editor with R-syntax highlighting which allows you to send you commands to R too.

Chapter 1

Exercises

1.1 Practice with basic R

Skip this if you are familiar with 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.

1.1.1 Probability functions

R has a set of probability functions for calculating the cumulative probability and its inverse function for calculating quantiles in all the probability distributions you are likely to need. The cumulative probability functions are

`pnorm`, `pchisq`, `pbinom`, `ppois`, etc.

and the quantile functions are

`qnorm`, `qchisq`, `qbinom`, `qpois`, etc.

See `help(pnorm)`, etc., and try

```
> pnorm(1.96)
> qnorm(0.975)
```

1. Find the probability below 1.5 in a Gaussian (normal) distribution.
2. What is the probability between -1.64 and $+1.64$ in a Gaussian distribution?
3. Find the probability below 4.3 in a chi-squared distribution on 1 degree of freedom.
4. Find the probability above 4.3 in a chi-squared distribution on 1 degree of freedom.
5. What is the probability above 10 in a chi-squared distribution on 5 df?
6. What is the 95% quantile in a chi-squared distribution on 1 df?

1.1.2 Vectors

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.

1.1.3 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)
> help(births)
> names(births)
> head(births)
```

1.1.4 Referencing parts of the data frame

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

```
> births[1, "bweight"]
> births[2, "bweight"]
> births[1:10, "bweight"]
```

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`.

1.1.5 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)
```

This makes sure that `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)
```

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

1.1.6 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.

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

1.1.7 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 20–24, 25–29, 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$agegrp <- cut(births$matage, breaks = c(20, 25, 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$agegrp = cut(births$matage, breaks = 5, right = FALSE)
> with(births, table(agegrp))
```

shows 5 intervals of width 4.

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.

1.1.8 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 levels 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 list the values taken by `bweight` for women whose babies have very low birth weight, try

```
> subset(births, vlow)$bweight
```

to create a new dataframe restricted to women with babies of very low birth weight, try

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

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.

1.1.9 Using a text editor with R

When working with R it is best to use a text editor to prepare a batch file (or script) which contains R commands and then to run them from the script. For Windows we recommend using the text editor Tinn-R, but you can use your favourite text editor instead if you prefer. Start up the editor and enter the following lines:

```
library(Epi)
data(births)
births$hyp <- factor(hyp, labels=c("normal", "hyper"))
births$sex <- factor(sex, labels=c("M", "F"))
```

Now save the script and run it. One major advantage of running all your R commands from a script is that you end up with a record of exactly what you did which can be repeated at any time. This will also help you redo the analysis in the (highly likely) event that your data changes before you have finished all analyses.

1. Edit the script to create a factor cutting `matage` at 20, 25, 30, 35, 40, 45 years, and run just this part of the script.
2. Edit the script to create a factor cutting `gestwks` at 20, 35, 37, 39, 45 weeks, and run just this part of the script.
3. Save and run the entire script.

1.1.10 Working with R

When starting R it is always a good idea to use `getwd()` to print the working directory. You may not be where you think you are! The command `dir()` can be used to see what files you have in the working directory.

When exiting from R you are offered the chance of saving all the objects in your current work space. This is not recommended as the work space can fill up with temporary objects, and it is easy to forget what these are when you resume the session. It is better to build up a script file as you work, and to run this at the start of a new session.

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

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

first instructs R to re-direct output away from the R terminal to the file "output.txt" and then summarizes the births data frame, the output from which goes to the sink. While a sink is open all output will go to it. Opening a file with `sink()` will overwrite its contents - to append output to a file, use the `append=TRUE` option with `sink()`. To close a sink, use `sink()` without arguments.

```
> sink()
```

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

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

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

which will save the births data frame in the file `births2.Rdata`. By default the data frame is saved as a binary file, but the option `ascii=TRUE` can be used to save it as a text file. To load the object from the file use

```
> load("births2.Rdata")
```

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

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. You can 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 to copy the following files to your working directory: `fem.dat`, `fem-dot.dat`, `fem.csv`, `fem.dta`.

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 ID
AGE	Age in years
IQ	IQ score
ANXIETY	Anxiety (1=none, 2=mild, 3=moderate,4=severe)
DEPRESS	Depression (1=none, 2=milde, 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("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
```

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.

1.2.4.1 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("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**.

1.2.4.2 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("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.

1.2.4.3 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” (not available).

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

```
> fem4 <- read.table("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("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, ...)
<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. SAS is an important example. R cannot read SAS datasets. However, the SAS XPORT format is well documented and has been adopted as a data interchange format by the US Food and Drug Administration <http://www.sas.com/govedu/fda/faq.html>. Hence there is a function in the `foreign` package for reading SAS XPORT files.

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

```
> library(foreign)
> fem5 <- read.dta("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))
```

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 Tabulation in R

1.3.1 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. The functions `table()`, `tapply()`, `by()`, and `xtabs()` will do this. The appearance of these tables is, however, 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 for publication in paper form, or on the World Wide Web. The package `xtables` 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.

1.3.2 Basic contingency tables

The `bdendo` data set in the `Epi` package contains data on a case-control study of endometrial cancer. Type

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

to create a copy of the data frame `bdendo` in your work space. Use the functions `str()` and `head()` to inspect the data frame.

The study concerns 63 cases of endometrial cancer that occurred in a retirement community in Los Angeles between 1971 and 1975. Each case was matched with 4 healthy controls, who were also living in the community at the time of the case.

The `table()` function can be used to create contingency tables. The following table cross-tabulates case-control status (`d`) with an indicator of whether the women had used estrogens (`est`)

```
> table(bdendo$d, bdendo$est)
```

The tables produced by the `table()` function are very plain. If you want some summary statistics, use the `twoby2()` function from the `Epi` package.

```
> twoby2(bdendo$d, bdendo$est)
```

Tables in R are *objects* that can be passed on to other functions for further manipulation.

```
> est.tab <- table(bdendo$d, bdendo$est)
> pctab(est.tab)
```

The `pctab()` function takes a contingency table as an argument and turns it into a table of percentages.

You can also pass the table to the `fisher.test()` function which will print some summary statistics for the association between the two variables.

```
> fisher.test(est.tab)
```

1.3.3 Manipulating contingency tables

The `UCBAdmissions` data set is a ready-made contingency table that comes as part of the “datasets” package in R. You just need to type

```
> UCBAdmissions
```

to see it, and `help(UCBAdmissions)` to see a description of the data. This is a three-way contingency table of all prospective students who applied to the University of California, Berkeley (UCB) in 1973, classified by sex, department and whether they were accepted or rejected. The five departments are given arbitrary labels “A” to “E”.

1.3.3.1 Flattening tables

The default method of printing three-way (or higher-dimensional) tables is not very easy to comprehend. The `fTable()` function “flattens” contingency tables so that they are human-readable

```
> fTable(UCBAdmissions)
```

A flattened contingency table is an object in its own right:

```
> ucbflat <- fTable(UCBAdmissions)
> ucbflat
```

An `fTable` object can even be written to file

```
> write.fTable(ucbflat, file = "ucb.txt")
```

Open up the file `ucb.txt` in a text editor to view its contents (it will be in the working directory of your R session). You can also read an `fTable` from a file

```
> ucbflat2 <- read.fTable("ucb.txt")
```

The function `all.equal()` can be used to test whether two R objects are the same or not. Use this to ensure that `ucbflat` and `ucbflat2` are the same.

Flattened tables can also be coerced back to three-dimensional contingency tables

```
> as.table(ucbflat2)
```

Use the `all.equal()` function again to ensure that the three dimensional table you have produced is the same as the original `UCBAdmissions` data.

1.3.3.2 Coercing tables to data frames

Another way of converting a three-dimensional table into a two-dimensional structure is to turn it into a data frame

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

This creates a new data frame with one variable for each of the classifying factors and an extra variable “Freq” for the frequency counts.

We can create collapsed contingency tables from this data frame using the `xtabs()` function

```
> xtabs(Freq ~ Gender + Admit, data = ucb.frame)
```

This function uses a formula interface to create a table. It sums over the value of `Freq` within cells defined by cross-classifying `Gender` and `Admit`. Note that `xtabs()` has a `data` argument which tells it where to look for variables. This makes it easier to use than the `table()` function.

Use the `pctab()` function to turn this contingency table into a table of percentages. What does this tell you about the relative success rates of the two genders applying to UCB in 1973?

1.3.4 Mantel-Haenszel testing on tables

The reason why we are using the `UCBAdmissions` data set (even though it is not an epidemiological example) is because it is a nice example of confounding. Females are more often rejected than males. The question is whether this is due to sex discrimination.

Use the `xtabs()` and `pctab()` functions to find

1. the percent of all applicants rejected by each department
2. the distribution of applicants among departments separately for males and females

These tables suggest that the higher rejection probability for females could be the result of confounding by department.

The function `mantelhaen.test()` does Mantel-Haenszel testing for the independence of two factors within strata defined by a third factor. One way to call a function is to supply, as a single argument, a 3-dimensional contingency table, for which the first two dimensions are the factors of interest.

Apply the function `mantelhaen.test()` to the original `UCBAdmissions` three-way table to see if gender is associated with acceptance within strata defined by department.

For 2×2 tables, the function `mantelhaen.test()` not only gives a p -value, but also a summary odds ratio and 95% confidence interval.

There is no significant association between gender and admission after controlling for department. The explanation for the higher success rate of male candidates is that they were more likely to apply for departments with a higher acceptance probability. A nice way to see this visually is with a mosaic plot.

```
> mosaicplot(UCBAdmissions, sort = c(3, 2, 1))
```

The argument `sort` determines what order the margins of the table are taken in. Change the order of the `sort` argument to display other mosaic plots, and try to interpret them.

1.3.5 Tables of summary statistics

The `stat.table()` function in the `Epi` package provides more printer-friendly tables than the functions provided by base R. The `stat.table()` function can be used to produce both contingency tables and tables of summary statistics.

1.3.5.1 One-way tables

You will need to use the data set `nickel` which is contained in the `Epi` package.

```
> data(nickel)
```

This data set is an occupational cohort of workers in the nickel refining industry. Each of the 679 subjects has a numeric exposure index, `exp`, based on their job history, which estimates their life-long exposure to nickel. Create a new variable `exp4` that classifies the exposure into four categories

```
> nickel$exp4 <- cut(nickel$exp, breaks = c(0, 0.5, 4.5, 8.5, Inf),
+   include.lowest = TRUE, right = FALSE)
```

The simplest table is created by

```
> stat.table(index = exp4, data = nickel)
```

This creates a count of individuals, classified by levels of the factor `exp4`. 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 attaching it.

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

```
> stat.table(index = exp4, contents = list(count(), percent(exp4)),
+           data = nickel)
```

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 `nickel`.

1.3.5.2 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 a *tagged* list as the value of the `contents` argument:

```
contents = list("N" = count(), "(%" = percent(exp4))
```

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 `exp4` as the `index` argument to `stat.table()`, use a named list:

```
index = list("Years of exposure" = exp4)
```

1.3.5.3 Cases, Follow-up and Rates

The above examples illustrate the basic features of `stat.table()`. However, our main interest in the Welsh nickel-smelter's study, is not to count subjects, but to evaluate the risk with years of exposure in "high-risk" occupations.

Add two new variables, one indicating death from lung cancer (`d.lung`) and one measuring the follow-up time in years.

```
> nickel <- transform(nickel, d.lung = icd %in% c(162, 163), flwupt = ageout -
+                   agein)
```

A count of cases and follow-up time can be created using the following `contents` argument:

```
contents = list(sum(d.lung), sum(flwupt))
```

Create a table with these contents, providing your own informative labels. To calculate rates, a special function `ratio()` is provided. A call to `ratio(d, y, scale)` calculates $scale * \text{sum}(d) / \text{sum}(y)$ within categories defined by the `index` variable. To calculate incidence rates per 100,000 person-years we therefore use.

```
contents = ratio(d.lung, flwupt, 100000)
```

Add an extra column to your table giving incidence rates, in addition to number of cases and follow-up time.

1.3.5.4 Printing tables

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 `stat.table` object with an explicit call to `print(my.table)`

There are two arguments to the print method for `stat.table` objects. The `width` argument specifies the minimum column width. Use this to print one of the tables you created above, preventing long column headers being folded over too many lines.

The second argument to the print method is `digits` which controls the number of digits printed after the decimal point. This table

```
> case.tab <- stat.table(exp4, list(cases = sum(d.lung)), data = nickel)
```

counts lung cancer cases, but prints them to 2-decimal places. Use the `digits` argument to print the table correctly.

Use `print.default()` instead of `print()` to print one of your tables. This shows the internal structure of the table. You may need to know this if you wish to extract data from a `stat.table` object.

1.3.6 Summary

In this exercise we have seen that tables in R are also *objects* which can be passed to other R functions for further analysis. Table objects can be transformed into other objects without loss of information.

The `stat.table()` function in the `Epi` package provides print-friendly tables of summary statistics. Further information about the capabilities of `stat.table()` can be found in the help page.

1.4 The measurement of effects

1.4.1 Introduction

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

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

The response variable must be numeric.

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

- Primary role: exposure
- Secondary role: confounder

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

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

What other measures of effects might be used?

1.4.2 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, without the need for familiarity with statistical modelling.

We shall use the `births` data in the `Epi` package, which can be inspected with the command `?births`. The variables we shall be interested in are `bweight` (birth weight) and `hyp` (hypertension). An alternative way of characterizing birth weight is shown in `lowbw` which is coded 1 for babies with low birth weight, and 0 otherwise. Other variables of interest are `sex` (of the baby) and `gestwks`, the gestation time. All variables are numeric, so first we need first to do a little housekeeping. To save too much typing these commands are in the file `births-house.r` which can be run with the command `source("births-house.r")` (or from your editor):

```
> source("births-house.r")
```

Now try

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

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

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

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

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

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

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

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

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

1.4.3 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 effects

```
[35,37) vs [20,35) 856.6
```

```
[37,39) vs [20,35) 1360.0
```

```
[39,45) vs [20,35) 1668.0
```

The command

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

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

1.4.4 Stratified effects

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

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

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

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

1.4.5 Controlling the effect of hyp for sex

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

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

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

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

1.4.6 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) or log-linear (for a binary response) we can find the linear effect of `gestwks`.

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

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

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

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

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

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

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

1.4.7 Checking on linearity

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

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

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

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

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

1.4.8 Frequency data

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

```
> UCBAdmissions
```

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

The command

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

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

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

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

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

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

1.5 Logistic regression

1.5.1 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 2.1.

Table 1.1: *Variables in the melanoma dataset.*

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

1.5.2 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 your working directory) to a data frame with name `mm` 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 `mm`.

1.5.3 House keeping

The structure of the data frame `mm` 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. Note that the file starts by reading in the data, so whenever you run it you start with the original data set. 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 `mm` and note the changes. Use the command `summary(mm)` to look at the univariate distributions.

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

1.5.4 One variable at a time

As a first step it is a good idea to start by looking at the effect of each of the variables, controlled for age in 10 year age groups and sex. Try

```
> effx(cc, type = "binary", exposure = skin, list(age.cat, sex), data = mm)
```

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

1.5.5 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.frk <- glm(cc ~ freckles + age.cat + sex, family = "binomial", data = mm)
> summary(m.frk)
> coef(m.frk)
> exp(coef(m.frk))
```

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

```
> ci.lin(m.frk, Exp = TRUE)
> round(ci.lin(m.frk, Exp = TRUE, alpha = 0.1)[, c(5, 6, 7)], 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

1.5.6 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 with `freckles` and one without, and recording the change in deviance.

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

The change in residual deviance is $1785.9 - 1737.1 = 48.8$ on $1389 - 1387 = 2$ degrees of freedom. Use the function `pchisq` to find the probability of exceeding 48.8 on 2df. The test is more easily carried out with

```
> anova(m2, m1, test = "Chisq")
```

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.

1.5.7 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`.

1.5.8 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 = mm)
```

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 = mm)
```

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

1.5.9 Stratification using glm

We shall reproduce the output from

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

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

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

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). You can improve this output with `ci.lin`. Try this.

```
> ci.lin(nested, Exp = T)[, c(5, 6, 7)]
```

1.5.10 Naevi

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

```
> with(mm, stem(nvsmall))
> with(mm, 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(mm, 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 logistic regression models including `sex`, `age.cat` and `nvsm4` in the model formula. Do the same with `nvlar3`.

Now fit a glm containing `age.cat` `sex` `nvsm4` and `nvlar3` and place the result in `m.nvboth`. What is the interpretation of the coefficients for `nvsm4` and `nvlar3`?

1.5.11 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 with

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

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 = mm)
> m2 <- glm(cc ~ fscore + age.cat + sex, family = "binomial", data = mm)
> 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 = mm)
> ci.lin(m1, Exp = TRUE)[c(2, 3), c(5, 6, 7)]
> m2 <- glm(cc ~ fscore + age.cat + sex, family = "binomial", data = mm)
> ci.lin(m2, Exp = TRUE)[2, c(5, 6, 7)]
```

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`).

1.5.12 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:

```
> plotEst(exp(ci.lin(m.nvboth)[- (1:7), -(2:4)]), xlog = T, vref = 1)
```

The `xlog` argument makes the OR axis logarithmic.

1.5.13 Further questions

Investigate some of these questions:

1. Is there still an effect of freckles even for those with dark skins?
2. Is there any effect of eye colour? Is there still an effect after taking account of skin colour?
3. If the main focus of interest is the effect of freckles which variables would you control for? Fit the appropriate model and summarize your conclusions. Plot the coefficients with their confidence intervals.

1.6 Statistical simulation and causal inference

Sometimes it is useful to generate artificial data in order to study, how statistical analysis procedures work. By solving next exercises you will learn to use basic tools for such statistical simulation.

1. Generate a sample of size 1000 having a normal distribution with mean 100 and standard deviation 10 and then obtain summary statistics and a histogram of that sample:

```
x <- rnorm(1000,100,10)
summary(x)
hist(x)
```

Now replace sample size 1000 by 20 and repeat the same commands.

2. Generate data from binomial (Bernoulli) distribution, taking values 1 and 0 with probabilities p and $1 - p$, respectively. Let's take $p = 0.4$:

```
x <- rbinom(500,1,0.4)
table(x)
```

3. Now let's 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)
```

Test the association either by χ^2 -test or logistic regression:

```
chisq.test(table(x,y))
summary(glm(y~x,family="binomial"))
```

4. In the following we are interested in evaluating the effect of an occupational hazard, (job) on the occurrence of depression. Suppose the following “real” causal relationships hold:
 - depression occurs more frequently among females than males.
 - females are more likely to select the risk job category.
 - persons with depression tend to smoke more frequently (i.e. depression “causes” smoking).
 - persons in the risk job tend to smoke more than other professions (i.e. this job “causes” smoking).

The task is to simulate a dataset in accordance with this model, and subsequently analyse it to see how the results come out.

- (a) Sketch a causal graph (not necessarily with R) of the situation.
- (b) Simulate a dataset (size=2000) by assuming all four variables to be binary, thus use `rbinom(2000,1,p)`, where p is a vector of length 2000 with the probabilities of value 1. The latter can be constructed by e.g.:

```
linpred <- b0 + b1 * VAR1 + b2 * VAR2
p <- 1 / ( 1 + exp( -linpred )
```

Here VAR1 and VAR2 are already generated variables (indicators of gender, job, smoking etc. and b0, b1 and b2 the coefficients of your choice (b1 and b2 are the true log odds ratios).

- (c) Example R commands to generate the data:

```
# linear predictors for job and depression
joblp <- -1 + 3*sex
deprlp <- -3 + 2*sex

il <- function(x) 1/(1+exp(-x)) # inverse logit function

# variables job and depression:
job<-rbinom(2000,1,il(joblp))
depr<-rbinom(2000,1,il(deprlp))

# smoking - dependent on job and depression:
smlp <- -2 + 3*depr + 3*job
smok <- rbinom(2000,1,il(smlp))

summary(glm(smok~depr+job,family=binomial))

# look at the data
#(if too many 0-cells, change some coefficients)

ftable(job,smok,sex,depr)
```

- (d) To estimate the effect of job on depression, fit the following logistic regression models for the outcome “depression” depending on “job”:
- unadjusted model
 - adjusted for sex
 - adjusted for smoking
 - adjusted for sex and smoking
- (e) Which (if any) of the models gives an unbiased estimate of the actual causal effect of interest?
- (f) How can the answer be seen from the graph?
- (g) Suppose the job actually causes depression. Change the data-generation algorithm and see, whether you find the right answer by the correct model.
- (h) Now suppose the job increases the probability of depression for females, but decreases the probability of depression for males. Change the algorithm to incorporate that *interaction*. What do you see, if you fit a properly adjusted model, but without that interaction? Can you get the right parameters back when fitting the correct model?

1.7 Graphics in R

This is an exercise that is designed to introduce you to the basic concepts in the way R graphics is used, in order to give you practise with the tools needed to do the Graphics meccano exercise.

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.

1.7.1 Simple plot on the screen

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

```
> data(births)
> str(births)
```

Now attach the dataframe and look at the birthweight distribution with

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

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

```
> `?`(hist)
```

and try some of the options, for example:

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

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

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

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

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

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)")
```

1.7.2 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)
```

1.7.3 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")
```

1.7.3.1 Using indexing for plot elements

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

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

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

1.7.3.2 Generating colours

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

```
> rainbow(4)
```

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

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

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

1.7.4 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, 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), ]
```

1.7.5 Saving your graphs for use in other documents

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

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

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

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

1.7.6 The `par()` command

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

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

```
> `?`(par)
```

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

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

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

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

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

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

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

1.8 Graphical meccano

The plot in figure 1.1 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.

It was originally created by S-PLUS in 1993. 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 the graphical features in R. This will take you through a number of fundamental techniques.

To get started, execute the following R code. You probably should not study the code in too much detail at this point.

```
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])
```

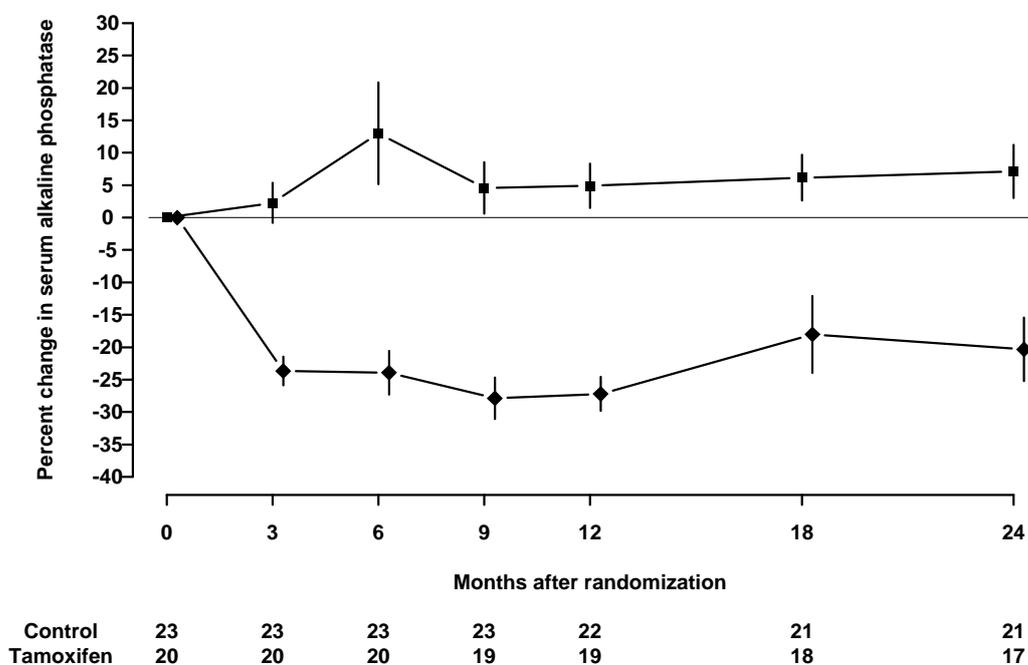


Figure 1.1: *Original figure to be reproduced in R*

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

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 linewidths 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.

1.9 Survival analysis: The Stroke dataset

In the file `stroke.csv` you can find data about all registered cases of stroke in Tartu, during 1991-1993. In the dataset there are the following variables:

- AGE - age in years
- SEX - sex (1-male, 0-female)
- DSTR - date of stroke
- DIED - date of death
- DGN - specific diagnosis, type of stroke (ID - unidentified)
- COMA - indicator, whether the patient was in coma after stroke
- MINF - history of myocardial infarction of the patient
- DIAB - history of diabetes
- HAN - history of hypertension

The follow-up was stopped at 01/01/1996. For subjects who were alive at that time, the value of the variable DIED is missing.

1. Use either `read.table` or `read.csv` to load the dataset. Do not forget to look into the file before to see, what is the field separator.
2. Using the dates of diagnosis and death, define a new variable for time under observation and the censoring indicator.
3. Plot the Kaplan-Meier estimates of survival (use the function `survfit`) function for different specific diagnoses of stroke. Also find the median survival for each of the diagnoses? Do the medians exist? Why?
4. Plot the log-cumulative hazards for different diagnoses. Do hazards look proportional?
5. Plot the Kaplan-Meier estimates of survival function separately for men and woman. Also test the difference using the logrank test. What do you conclude?
6. Fit a Cox regression model with a) age only; b) age and sex as covariates. What do you conclude? How would you make a Kaplan-Meier graph showing both, age and sex effects?
7. Include other significant covariates in the model. For the final model, check, whether the proportionality assumption is fulfilled for each of the variables, using the `cox.zph` function. If not, try whether stratification helps to solve the problem.
8. Plot the predicted survival curves for 75 years old patients for each of the different sub-diagnoses of stroke.
9. Plot the predicted survival curves for patients with and without hypertension.

1.10 Interval-censored data: Conversion to diabetes

A largely a-symptomatic disease as diabetes is diagnosed by a test. When persons are in a trial to prevent diabetes, then tests are taken at prescheduled times, so it is only known whether a person is diseased or not at the visit dates.

If we consider the disease irreversible, then we effectively assume that the person has been well (non-diseased) prior to any visit where he is well. Moreover, disease onset must have been somewhere between the last time the person was seen well and the first time seen ill.

The data we will analyse are from an intervention trial, "Addition" where high-risk persons are followed for development of diabetes. The intervention is at the level of the general practitioner. The aim was to see if education of GPs had any effect on diabetes occurrence among their patients.

1. Load the dataset by `data(DMconv)`, and find out what is in there, using `str(DMconv)` and `?DMconv`.
2. Make a histogram for the observed times of last seen well and first seen ill. Remember to subtract the date of entry to convert dates to time since entry. (Use `cal.yr` if you want time in years).
3. Now fit a model without covariates, exploring how the specification of the intervals for the baseline hazard influence the estimates, e.g. by:

```
m0 <-
Icens( first.well = cal.yr(doe)-cal.yr(doe), # Timescale is time since entry
       last.well = cal.yr(dlw)-cal.yr(doe),
       first.ill = cal.yr(dfi)-cal.yr(doe),
       data = DMconv,
       breaks = c(seq(0,6,2),10) # constant rate in intervals (0:6)
m0
```

You should try exploring different specifications of `breaks=`.

4. Extract the resulting estimated rates using `summary`. Make sure you are aware of the meaning of the parameters. Plot them as a function of time. (You may want to look at the graphics option `type="s"` in `?par`).
5. Choose a sensible division of the baseline (3, for example), and fit a multiplicative relative risk model for the effect for the covariates, by adding the extra parameters:

```
model = "MRR",
formula = ~ gtol + grp,
```

Explain the meaning of the parameters.

6. You can get the parameters from the model by using the generic function `summary` on the model object. Plot the estimated rates for the four groups of patients.
7. Now try to fit an additive excess risk model (`model="AER"`) with the same covariates. Plot the estimated rates in each of the four groups using this model.

8. (*Optional — very time-consuming!*). The parameters in the additive excess risk model are rates, so the normal (i.e. *symmetric*) approximation to their distribution is probably inaccurate. Therefore it is possible to do boot-strap sampling from the dataset and get more reliable confidence intervals for the parameter estimates. Try to fit the model using the parameter `boot=10` (not more!), and see how the result looks.

If you embed the function call in `system.time()` you can see how long it takes:

```
system.time(  
  m0 <-  
  Icens( ..., boot=10 )  
)
```

Then you can decide wheter you want to try with `boot=400` over lunch, or tonight when you go to bed!

1.11 Time-splitting and SMR: Thorotrast

In the period 1935–50 a contrast medium called Thorotrast was used for cerebral angiography (X-ray imaging of the brain). This contrast medium contained ^{232}Th , thorium. It turns out that thorium is not excreted from the body, it is permanently deposited, some 60% in the liver, 20% in the spleen and some 10% in the bone marrow, and a very small fraction in other organs.

Thorium is an α -emitting radionuclide, i.e. it emits α -rays (i.e. He-nuclei) which is ionizing, but not particularly penetrating; it only penetrates 2–3 cell-layers. The half-life of ^{232}Th is 1.4×10^{10} years, so the patients that have been injected with Thorotrast exposed are exposed to a constant, small α -radiation for life.

In the study is 990 Thorotrast patients who had a cerebral angiography in the period 1935–50 and 1480 controls who have had a cerebral angiography in the period 1946–63, on similar indications as the Thorotrast patients, but with another contrast medium.

Persons undergoing cerebral angiography are in many cases seriously ill, they are suspected of cerebral malformations or tumors, so both the Thorotrast group and the control group have very high mortality rates, and a pattern of causes of death that differ substantially from the general population. Especially during the first year after diagnosis there is a very high mortality among the patients, which is entirely associated to the conditions that have lead to to the cerebral angiography. Therefore, only the follow-up of both Thorotrast patients and control patients is only relevant from one year after angiography.

1.11.1 The data sets

There are two sources of data for this exercise, the cohort data and the mortality rates from Denmark. The dataset with the cohort is loaded by `data(thoro)`; and you can get an explanation by typing `?thoro`. The relevant cause-specific mortality figures for Denmark are loaded by `data(gmortDK)`. As well as overall mortality (`rt`), the file also contains the mortality for all cancers, etc. For a complete explanation, use `?gmortDK`.

1. First take a look at the cohort data by e.g. `head(thoro)` and/or `summary(thoro)`.

Note that the date variables are of class “Date”, i.e. they are stored as days since 1 January 1970, you may want to try for example:

```
bd <- thoro$birthdat[1:5]
bd
as.numeric(bd)
cal.yr(bd)
(cal.yr(bd)-1970)*365.25
```

Don't forget to use `?cal.yr`.

2. Declare the follow-up timescales for the dataset, using the `Lexis` command, e.g.:

```
thL <- Lexis( entry = list("per"=cal.yr(injecdat),
                          "age"=cal.yr(injecdat)-cal.yr(birthdat),
                          "tfi"=0),
             exit = list("per"=cal.yr(exitdat)),
             status = exitstat,
             id = id,
             data = thoro )
```

```
str( thL )
head( thL )
```

Explain the meaning of the variables added by `Lexis`, and how they relate to the data variables.

- Note that `thL` has got class “`Lexis`”. Now make a Lexis diagram using the defined object `thL`:

```
plot( thL )
```

This really uses the function `plot.Lexis` to make the plot. Use `?plot.Lexis` to find the available options for this command, and try to improve the plots with indications of the exit-status of the persons in the cohort.

Try to make the life-lines of thorotrast patients and controls different color. Hint: use the indexing facility for a character vector with color names, see the section “Adding to a plot” in “A short introduction to R”.

1.11.2 Rates

- The first analytical task is to look at overall mortality by contrast medium (`contrast`).

Tabulate the number of deaths and person-years from the study by group using `stat.table()`. You will want to convert the dates to fractions of calendar years by the function `cal.yr()` before computing the follow-up time. Remember to start follow-up one year after angiography (`injecdat`) and exclude persons without follow-up beyond one year, for example by:

```
thoro$Y <- pmax( 0, cal.yr(thoro$exitdat)-cal.yr(thoro$injecdat)-1 )
thoro$D <- as.numeric( !is.na(thoro$cause) & thoro$Y > 0 )
thoro <- thoro[thoro$Y>0,]
stat.table( contrast,
            list( D=sum( D ), Y=sum( Y ), Rate=ratio( D, Y/1000 ) ),
            margin=TRUE, data=thoro )
```

- Declare the reduced data as “`Lexis`” using `entrydat` as date of entry. Use the generated dataframe to produce the same table by `stat.table`.
- Compute 95% confidence intervals for the overall rates and for the rate-ratio between the two groups.

Try to do this also by fitting a Poisson-model with `glm` and subsequently use `ci.lin` to compute the rates and the RR.

- (Optional — skip if you are not in the mood for hairy data manipulation):

It is well known that Thorotrast causes liver cancer; try to tabulate the number of liver cancers by patient group.

One may argue that the deaths caused by liver cancer should not be counted, so repeat the mortality calculations above after censoring patients at date of liver cancer diagnosis.

8. An important question is how the mortality rates in the two groups varies with time since injection.

In order to see if the mortality changes the same way in the two groups, split the follow-up time by time since injection using `splitLexis`. Split follow-up in intervals of 1 year during the first years say 5 years from time since angiography and subsequently every 5 years, eg. by:

```
thx <- splitLexis(thL, breaks=list( tfi=c(0:4,seq(5,55,5)) )
```

Take a look at the split data for example by listing the observations with `id==1`, (use for example `subset(thx,id==1)`). Make sure that you understand how they relate to the original record.

9. Compute mortality rates in each interval separately for the two groups, using `stat.table`. How do the rates in the two groups of patients behave by time since injection?
10. (*Optional*) Try to show it in a graph. Hint: Assign the result of `stat.table()` to an object at take a look at the `dimnames()` of this object. Then use `matplot()` to plot the two sets of rates by taking appropriate subsets of the object.
11. The next step is to model the mortality and the rate-ratio in the two groups by a smooth function. Therefore, we split the follow-up in small intervals, and fit a model using natural splines for the mortality as a function of time.

```
thxx <- splitLexis(thL, breaks=list("tfi"=c(0,seq(1,100,0.5))))
dim( thxx )
```

In order to do so we need a *quantitative* variable for each of the intervals, giving the midpoints of the intervals, as well as a failure indicator:

```
thxx$m.tfi <- timeBand( thxx, "tfi", "middle" )
thxx$fail <- (status(thxx) > 0)
```

Remember to consult the help pages for `timeBand`, `status` and `deltat`.

12. A Poisson model can now be used to fit a model for the mortality using natural splines. The point is to fit a separate mortality curve for each contrast group as a function of time since injection:
- Splines are available in the `splines` package, which is loaded by `library(splines)`.
 - The definition of splines requires the definition of internal knots (`knots`) and boundary knots (`Boundary.knots`, abbreviated `Bo`). These are most conveniently defined before the splines.
 - If you want separate splines for each level of `contrast`, use the interaction operator `“:”`.
 - To get the parametrization as log-rates in each of the groups we remove the overall intercept from the model by `“-1”`, and include an intercept with the splines by `intercept=TRUE` (or `i=T`).
 - Finally we scale the person-years by 1000, in order to get results in rates per 1000 person-years.

Now, put these points together in the model specification:

```
kn <- c(4,8,seq(10,40,10))
bk <- c(1,50)
m1 <- glm( fail ~ -1 + contrast:ns( m.tfi, knots=kn, Bo=bk, i=T ) +
          offset( log(lex.deltat)/1000 ),
          family=poisson, data=thxx )
```

Note that we use the midpoint `m.tfi` as defined above as the regression variable in the model.

13. Construct a contrast matrix to multiply with (some of) the coefficients of the model, so that you get the estimated mortality rates at a set of points between 1 and 40 years, say.

You can extract the parameters and multiply them with the contrast matrix in one go by using the facilities of `ci.lin` - remember `?ci.lin`. This will give you estimated mortalities at each of the time-points in `tpt`.

This can be used by pre-multiplying a matrix to the parameters to get estimates of the rates at a number of points:

```
tpt <- seq(1,40,0.5)
CM <- ns( tpt, knots=kn, Bo=bk )
mort1 <- ci.lin( m1, ctr.mat=CM, subset="1:ns", Exp=TRUE )
mort2 <- ci.lin( m1, ctr.mat=CM, subset="2:ns", Exp=TRUE )
```

Because the contrast matrix `CM` is constructed using the `ns` with `tpt` as argument, the result `ci.lin` will be log-rates estimated at each of the timepoints in `tpt`. Now, plot the two sets of estimated mortalities as nice curves with confidence intervals.

14. Use the contrast matrix to construct estimates of the rate-ratio between the groups at the same timepoints:

```
RR <- ci.lin( m1, ctr.mat=cbind(CM,-CM), subset=c("1:ns","2:ns"), Exp=TRUE )
```

1.11.3 SMR

The follow-up of the two groups of patients are in very different time periods and they have differing age-distributions. Therefore it is desirable to control for age and calendar time. This could be done by making an internal comparison of the two contrast groups controlled for age, sex, and calendar period. However, because of the different calendar periods of follow-up, some information would be lost. Instead, the comparison can be standardized for age, sex, and period, using SMRs.

15. The Danish mortality figures are in the dataframe `gmortDK`. Load it by `data(gmortDK)` and inspect it using `?gmortDK`. In order to be able to match up the Danish population mortality rates to the follow-up data these must first be split by current age and calendar time. The names and coding of the age and period variables must be chosen so that they are the same in `gmortDK` and in the split cohort data.
16. Split the dataset, now also along current age and period using cutpoints that correspond to those from the population data:

```
thxx <- splitLexis(thL, breaks=list( age=seq(0,90,5),
                                   per=seq(1938,2038,5),
                                   tfi=seq(0,55,0.5) ) )
```

17. Unlike other packages there is no need in R to sort the dataframe by variables we merge on, or to name them explicitly — R will merge on all variables common to the two dataframes, and only include records in the result that have contributions from both dataframes.

But you must make sure that variables have common names, so define `agr` and `pgr` in the cohort data:

```
thxx$agr <- timeBand(thxx, "age", "left")
thxx$pgr <- timeBand(thxx, "per", "left")
```

and then make `pgr` in the population mortality data match the coding in the cohort data:

```
gmortDK$pgr <- gmortDK$per + 1900
```

Now you can merge the the population data with the follow-up data on the variables `agr`, `pgr` and `sex` (only taking the relevant columns from `gmortDK`):

```
th1ap <- merge(thxx, gmortDK[,c("agr", "pgr", "sex", "rt")],
              by=c("agr", "pgr", "sex"))
```

18. The variable `rt` from `gmortDK` has the population mortality rate in cases per 1000 person-years. Multiply this with the person-years (`lex.deltat`) to form the expected number of cases, `E`, say.
19. Compute the observed and expected number of cases as well as the ratio (SMR) by group using `stat.table`. Further tabulate this by time since injection.
20. Now use the log of `E` as offset-variable to estimate in a model where the SMR in each of the two groups of patients are assumed to depend smoothly on time since injection. Plot the SMR for each of the groups, and the ratio of SMRs as a function of time since injection. (This is parallel to what you did with the rates).
21. Do the ratios of SMRs differ substantially from the rate ratios obtained without using the reference rates?
22. (*Open-ended and complicated*): How would you go about controlling potential confounding by age and calendar time without using SMR? Fit a model where the rate-ratio between thorotrast patients and controls is included as a separate term. Fit the same model using an SMR-analysis and compare the results.

1.12 Matched case-control study: *Salmonella* Typhimurium

In the fall of 1996 an unusually large number of *Salmonella* Typhimurium cases were recorded in Fyn county in Denmark. The Danish Zoonosis Centre set up a matched case-control study to investigate the source of the infection. Cases and two age-, sex- and residency-matched controls were telephone interviewed about their food intake during the previous two weeks.

The data from this study are in the dataframe `S.typh`, which can be accessed after loading the `survival` package by typing `data(S.typh)`. A description of the variables can be seen on the help page for the `S.typh` dataset.

1. Examine the effect of `pork` on the risk of `S.typh` infection, using `coxph`. Remember to use `library(survival)` before using `coxph`.

Alternatively you may use `clogit`, which is just a wrapper for `coxph`, albeit using the algorithm in a slightly less efficient way.

2. Look through the other food exposure variables and find out if any of them have a strong association with the outcome.
3. You should have found in question 2 that `plant7` is a risk factor, while `fruit` is protective. Two questions that this finding raises are

- (a) What is the effect of each variable adjusted for the other in a main effects model?
- (b) What is the effect of `plant7` stratified by `fruit` and *vice versa*, that is, the effect of `fruit` stratified by `plant7`?

For (a), fit the relevant main effects model. Interpret the parameters and make a note of the results.

One possibility for part (b), for example, is that the effect of eating meat from `plant7` on the risk of `S.typh` depends on whether or not the person ate `fruit`. Is this an interaction model? How would you parametrise the model to address this question? (Remember the `I()` function). Fit each of the stratified models and interpret the results.

4. A third alternative in addition to the main effects and stratified models of question 3 is a model featuring an interaction between the two variables `fruit` and `plant7`. One way of studying the interaction of these two factors is to create a new variable with four levels corresponding to the four possible combinations of the two levels of `plant7` and the two levels of `fruit`, using `plant7:fruit` in the model. Try this and see what happens.

If you want the lowest risk category, i.e. `plant7==0` and `fruit==1` as the reference category in the interaction model, you must reparametrise. Do that now and fill in the right hand table below, using the output from the original parametrisation of the interaction model to fill out the left hand table below.

How do the results from this analysis compare to the ones that you found in question 3? In particular, does the interaction model provide a better fit to the data than the main effects model?

log(OR) plant7	fruit	
	0	1
0		
1		

log(OR) plant7	fruit	
	0	1
0		0
1		

5. How would you report the results? (In other words, what model do you prefer *and* how would report the estimates from that model?). Fill in the tables below.

log(OR) plant7	fruit	
	0	1
0		
1		

OR (95% c.i.) plant7	fruit	
	0	1
0		1
1		

1.13 Case-cohort study of congenital malformations

The purpose of this exercise is to study mortality from malformations for children and youth after the first year of life. The data we will use are from the Norwegian Birth Registry and concern the 1.27 million children born in Norway in the period 1967-1989. The data on births are matched to the registry on causes of deaths, and deaths that are due to malformations are recorded.

We will in our analyses use information on all the 672 deaths due to malformations (“cases”) as well as information on a subcohort of 1270 children sampled at random among all the 1.27 million children born in the period. (One of the cases turned out to be a subcohort member.) The variables in the data set are:

sex sex of the child (1=boy; 2=girl)

wgt birth weight in grams

mage mother’s age in years at birth

age child’s age in years at death or censoring

sta status (1=dead from malformations; 0=censored)

subc subcohort status (1=member; 0=not member)

Read the data in the file “malform.txt” into R and name the resulting data frame `malform`.

1.13.1 Exploring the data

Explore the distribution of maternal age, birth weight and sex using graphical and tabular methods. Create factor variables for birth weight and sex by the `cut` function using suitable break points.

NB You should add these derived variables to the data frame `malform`. If you write them to the global environment you will have difficulty in section [1.13.3](#).

1.13.2 Delaying entry for cases

The fundamental problem with case-cohort sampling is that the cases contribute too much follow-up time, relative to the subjects in the subcohort. There are two ways around this.

The first method is to pretend that cases only enter the cohort fractionally before they become a case. In this example, we will assume that cases enter the cohort half a day (1/730th of a year) before they die.

1. Create a new variable `age.entry` that takes the value 0.99 for subjects in the subcohort and `age - 1/730` for the other subjects. (The study only includes cases occurring after the first year of life, but due to rounding error, the first case occurs at age 0.999).
2. Do a Cox regression, analysing the effects of maternal age or birthweight. To get correct estimates define late entry with the `age.entry` variable. Use the `robust` option to `coxph` to get correct standard errors.
3. Compare the results with a naive Cox regression that ignores the fact that the data come from a case-cohort sample, and treats them like a small cohort. The estimates from the naive model are attenuated. Why?
4. Do further modelling with all three risk factors and decide on a final model.

1.13.3 Using an offset

The second method of correcting the Cox regression is to downweight the contribution of the cases using an offset term. A small complication is that a case which occurs in the subcohort has to appear **twice** in the data set: once where it appears down-weighted, and once where it does not.

1. Create a unique identifier variable `id` and add it to the `malform` data frame.
2. Create a data frame `malform.cases` containing only the cases. (hint: `subset`). Add an extra variable `dummy` taking the value `-100`
3. Create a data frame `malform.subcohort` containing only the subjects in the subcohort. The status variable should be reset to 0 (censored) for these subjects. Add an extra variable `dummy` taking the value 0.
4. Combine the data frames `malform.cases` and `malform.subcohort` to form a new data frame `malform2` (hint: `rbind`).
5. Fit a Cox regression to the data frame `malform2`. In order to fit the model correctly *and* get the correct variance estimate you must add the terms `+ offset(dummy) + cluster(id)` to the formula in `coxph`. The cluster term automatically gives robust standard errors, so there is no need to set `robust=TRUE`.

1.13.4 Advanced topic: The Self-Prentice variance estimator

The Self-Prentice estimate of the variance requires some extra computation.

First create a backup copy of the variance

```
if (is.null(cox.fit$naive.var)) {
  cox.fit$naive.var <- cox.fit$var
}
```

where `cox.fit` is the output from `coxph`. Then calculate the variance as follows:

```
dfb <- as.matrix(resid(cox.fit,type="dfbeta"))
d2 <- dfb[malform$subc==1,]
alpha <- 0.001 #Sampling proportion for subcohort
cox.fit$var <- cox.fit$naive.var + (1-alpha)*crossprod(d2)
summary(cox.fit)
```

Compare the result with the previous variance estimate.

Put this sequence of commands in a function, so that you can reapply it without having to type all the commands again.

The Self-Prentice variance estimator uses extra information – the sampling fraction for the subcohort – and therefore should be more accurate than the other method. True or false?

1.14 Multi-State Markov Models

1.14.1 Introduction

The Multi-State Markov transition model is a generalization of the Poisson regression model for diseases that have multiple diagnostic states, instead of a simple dichotomy between “diseased” and “healthy”. Follow-up studies for such diseases generally create “panel data”, in which the disease state of each subject is observed only intermittently, typically at a clinical visit where a diagnostic examination is carried out. Transitions between disease states are therefore interval-censored. There may also be uncertainty about the quality of the disease classification. So-called “hidden” Markov models make a distinction between the true disease state and the observed one, in order to examine the effect of diagnostic misclassification.

The `msm` package by Christopher Jackson allows you to fit multi-state Markov models. In addition to the online help, there is an introductory manual for `msm` in PDF format, which is installed with the package. This practical uses examples taken from the manual, but only shows a selection of the capabilities of the `msm` package.

1.15 The heart data set

The data frame `heart` is provided with the `msm` package. It describes the follow up history of 622 patients following a heart transplant, monitoring the development of a post-transplant complication called *coronary allograft vasculopathy* (CAV). You can make it available by typing

```
> data(heart, package = "msm")
```

and see a description of the contents with `help(heart, package="msm")`. The reason we are using the `package` argument here is that there is another, completely different data set called “heart” in the `survival` package. If both `msm` and `survival` packages are loaded, and you do not use the `package` argument, then you will get whichever one is first on your search path.

We are concerned with modelling the variable `state` which takes values from 1 to 4:

1. healthy
2. mild CAV
3. severe CAV
4. death

and how this evolves with time since transplant (variable `years`).

Before fitting any models, you should spend some time exploring the data in a more informal way. Here is a list of possible questions:

- How old were the subjects when they had their transplant?
- How many were male and how many were female?
- How many visits did the subjects have?
- How long were they followed up for?
- What is the interval between visits?

The `statetable.msm()` function tabulates the transitions that are observed between consecutive clinical visits.

```
> library(msm)
> statetable.msm(state, subject = PTNUM, data = heart)
```

How many deaths occurred in total?

If we were only interested in mortality, and not CAV, we could treat this as a survival analysis problem. Create a new data frame containing only the last visit of each subject. Plot a Kaplan-Meier curve of post-transplant survival.

This analysis is incomplete, since CAV may be an important predictor of mortality. The `msm` package can be used to take this into account.

1.15.1 Fitting a multi-state model

The first task in fitting a multi-state model is to define which transitions are permitted. This can be done by creating a square matrix taking value 1 for the permitted transitions and 0 for the forbidden ones.

```
qmat0 <- matrix(c(0,1,0,1,
                 1,0,1,1,
                 0,1,0,1,
                 0,0,0,0),
               nrow = 4, ncol = 4, byrow=TRUE,
               dimnames=list(from=1:4,to=1:4))
```

Print this matrix. Draw a graph, by hand, with four nodes, representing the states, and arrows between nodes representing the allowed transitions.

A crude estimate of the transition rates can be made with

```
> qmat1 <- crudeinits.msm(state ~ years, subject = PTNUM, data = heart,
+   qmatrix = qmat0)
> print(qmat1)
```

This function ignores the interval-censoring in the data by assuming that transitions occur at the time of clinical visits. As the function name suggests, it is designed to provide initial values for the model.

The model itself is fitted with a similar function call.

```
> heart.msm <- msm(state ~ years, subject = PTNUM, data = heart,
+   qmatrix = qmat1, death = 4)
```

By giving the argument `death=4`, we specify that state 4 is a special state, whose transition times are not interval censored: unlike the other states, the time of transition to state 4 (i.e. time of death) is known exactly.

The argument `qmatrix` has a dual purpose: it specifies which transitions are allowed **and** gives initial values for the transition intensities.

Print the `heart.msm` object and use the `summary` function. The output may not be especially useful, so some *extractor* functions are provided to abstract useful statistics from the output:

- `sojourn.msm(heart.msm)` gives the mean *sojourn time* for each state. This means the amount of time spent in each disease state before moving on to the next one.
- `plot(heart.msm)` Plots parametric survival curves, stratified by disease state. These show how more severe CAV is associated with higher mortality. Compare these curves with the Kaplan-Meier estimate you previously plotted.

- `pmatrix.msm(heart.msm, t)` creates the transition probability matrix for a time interval t . If P is the transition matrix then $P_{ij}(t)$ gives the probability of a subject being in state j at time t , given that they were in state i at time 0. Use this to calculate the proportion of transplant patients that we expect to be healthy 1, 5, and 10 years after transplant.

1.15.2 Adding covariates

We shall consider whether transition rates are different for males and females. Before doing so, save the estimated transition intensity matrix from the previous model, to act as a new set of initial values

```
> qmat2 <- qmatrix.msm(heart.msm)$estimate
```

To examine the effect of sex, supply a formula to the `covariate` argument

```
> heart.msm.sex <- msm(state ~ years, subject = PTNUM, data = heart,
+   qmatrix = qmat2, death = 4, covariate = ~sex)
```

Then

```
> hazard.msm(heart.msm.sex)
```

gives hazard rate ratios, and confidence intervals, for the allowed transitions.

1.15.3 Applying constraints

By default, the effect of the covariates is assumed to be different for each possible transition. In this case, there are 7 possible transitions, and therefore 7 hazard rate ratio estimates. We can also simplify the model, to answer some more clinically relevant questions:

1. Is the mortality rate higher (or lower) for females?
2. Is the rate of CAV progression higher?
3. Is the rate of CAV recovery higher?

This is done by supplying a `constraint` argument

```
constraint = list(sex = c(1,2,3,1,2,3,2))
```

The `constraint` argument gives a numeric label to the allowed transitions (reading across the rows of the `qmatrix`), and constrains the effect of the covariate to be the same for all transitions with the same label. The labels are shown as super-scripts below:

$$\begin{pmatrix} 0 & 1^1 & 0 & 1^2 \\ 1^3 & 0 & 1^1 & 1^2 \\ 0 & 1^3 & 0 & 1^2 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Fit this constrained model to the `heart` data and answer the above questions.

1.15.4 A misclassification model

So far, we have assumed that any observed improvements in CAV state are real (*i.e.* transitions from state 2 to 1 and from state 3 to 2 really do take place). An alternative interpretation is that CAV is a *progressive* disease, which can only get worse with time, and that any apparent improvements in disease status are due to misclassification. The `msm` package can also fit such models.

Consider a model in which improvements in disease status are no longer possible. Draw a new graph for this model, and write down a new matrix, equivalent to `qmat0` with 1 for allowed transitions and 0 for the forbidden transitions. Create a new `qmatrix` of initial values for the new model, using the estimated transition intensities of the previous model. You also need to create a misclassification matrix `ematrix` with the following values

$$\begin{pmatrix} 0 & 0.1 & 0 & 0 \\ 0.1 & 0 & 0.1 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The `ematrix` has a similar interpretation as `qmatrix`, but applies to misclassification probabilities instead of transition intensities. Zero values indicate impossible misclassification. The fourth row consists entirely of zeros, showing that death cannot be misclassified. Likewise, the fourth column is zero, indicating that states 1-3 cannot be misclassified as death. Non-zero values indicate possible misclassifications, and also supply initial values for the model. Hence, for example: if the true disease state is 1 (healthy), we may observe state 2 (mild CAV) instead. Initially, the misclassification probability is assumed to be 10%, but the model will calculate a new estimate during the course of model fitting.

Fit a new model, without covariates, using the new `qmatrix` and with an extra argument `ematrix = ematrix`. Note that the default method for calculating the maximum likelihood estimates does not work particularly well on this problem. You can improve the estimates by giving the optional arguments `method="BFGS"` and `use.deriv=TRUE` to the `msm()` function. Use the `ematrix.msm` function to extract the estimated misclassification matrix. Plot the new survival curves and compare them with the previous ones.

1.15.5 Summary

Further information about the `msm` package and its capabilities can be found in the `msm` manual. This is a PDF document that can be viewed with the R function call.

```
> RShowDoc("msm-manual", package = "msm")
```

1.16 Competing risks: The Danish Thorotrast study

In the period 1935–50 a contrast medium called Thorotrast was used for cerebral angiography (X-ray imaging of the brain). This contrast medium contained ^{232}Th , thorium. It turns out that thorium is not excreted from the body, it is permanently deposited, some 60% in the liver, 20% in the spleen and some 10% in the bone marrow, and a very small fraction in other organs.

Thorium is an α -emitting radionuclide, i.e. it emits α -rays (i.e. He-nuclei) which is ionizing, but not particularly penetrating; it only penetrates 2–3 cell-layers. The half-life of ^{232}Th is 1.4×10^{10} years, so the patients that have been injected with Thorotrast exposed are to a constant, small α -radiation for life.

A number of studies of persons subjected to Thorotrast have been conducted (Japan, Germany, Portugal, Sweden and Denmark). The data used in this workshop comes from one of the largest studies, the Danish, which incorporates 999 exposed patients injected with Thorotrast between 1935 and 1947, and 1480 controls who have had a cerebral angiography in the period 1946–63, on similar indications as the Thorotrast patients.

Persons undergoing cerebral angiography are in many cases seriously ill, they are suspected of cerebral malformations or tumors, so both the Thorotrast group and the control group have very high mortality rates, and a pattern of causes of death that differ much from the general population. Especially during the first year after diagnosis, there is a very high mortality among the patients, which is entirely associated to the conditions that have led to the cerebral angiography. Therefore, the follow-up of both Thorotrast patients and control patients started one year after the angiography, at which time 811 Thorotrast patients and 1236 control patients were alive.

Since the Thorotrast patients receive a continuous dose to the liver they have very high rates of liver cancer. All 127 liver cancers except 8 have been classified as one of three different subtypes: hepatocellular carcinoma, cholangiocellular carcinoma and haemangiosarcoma.

1.16.1 Cumulative dose

Thorium in the form of Thorotrast has a tendency to form small “lumps” when it deposits in the liver. Because of the limited range of the α -rays this causes the radiation dose per time to be less than proportional to the injected dose, because some of the emitted particles never reach beyond the “lump”. It has been estimated that this give rise to the following conversion factors between injected volume and liver dose:

Inj. volume (ml)	Liver dose rate (Gy/year/dl)
1–9	1.40
10–19	1.25
20–29	1.10
30–39	0.95
40–49	0.85
50–59	0.76
60–69	0.72
70–79	0.69
80–99	0.65

The relationship between injected volume v (measured in dl) and effective radiation dose rate ρ (in Gray/year/dl) can be quite well approximated by the function:

$$\rho = 1.502 - 1.937 \times v + 1.109 \times v^2$$

so the annual dose δ (in Gray/per year) is approximately:

$$\delta = (1.502 - 1.937 \times v + 1.109 \times v^2) \times v$$

1.16.2 The data sets

The dataset is available in the Epi package, so you can load the data and inspect it by:

```
> data(thoro)
```

This will load the dataframe `thoro` with information about 2470 cases of cerebral angiography. See the details and variable description on the help page using `?thoro`.

1.16.3 Competing risks: Tumour histology

1. Now, we will look at the incidence rates of the three different histological subtypes of liver cancer. There are no cases of liver cancer in the control group, so this analysis is only of interest for the Thorotrast group, so start by defining a dataset only containing the Thorotrast group (`contrast==1`, for example by:

```
> tht <- thoro[thoro$contrast == 1, ]
```

Note that there are some liver cancer cases that it has not been possible to type, hence the number events for `hepcc`, `chola` and `hmang` do not add up to that for `liver`.

2. Tabulate the event indicators for these three types of events against the existence of a date of livercancer diagnosis `is.na(liverdat)`. Then define a date of exit, `dox`, for the analysis of these three types of event, using date of death or unknown type of liver cancer as censoring date:

```
> tht$dox <- pmin(tht$liverdat, tht$exitdat, na.rm = T)
> tht <- subset(tht, dox > injecdat)
```

3. Then define the cumulative dose per year:

```
> tht <- transform(tht, dl = volume/100)
> tht <- transform(tht, gpy = (1.502 - 1.937 * dl + 1.109 * dl^2) *
+ dl)
```

4. Now create the dataset needed for Cox-analysis of the three competing risks of the three types of liver cancer.

```
> hepcc <- tht
> hepcc$event <- hepcc$hepcc
> hepcc$type <- "hepcc"
> chola <- tht
> chola$event <- chola$chola
> chola$type <- "chola"
> hmang <- tht
> hmang$event <- hmang$hmang
> hmang$type <- "hmang"
> th.cmp <- rbind(hepcc, chola, hmang)
```

Make sure that you understand the mechanics of what goes on in the construction of the datasets.

5. Now do a stratified Cox-analysis of the three rates, using time since injection as time and injected dose as covariate:

```
> library(survival)
> mi <- coxph(Surv(dox - injecdat, event) ~ volume:type + strata(type),
+ data = th.cmp)
> m1 <- coxph(Surv(dox - injecdat, event) ~ volume + strata(type),
+ data = th.cmp)
> anova(mi, m1, test = "Chisq")
```

What is the difference between the two models, i.e. what is being tested by `anova`?

6. Is there any effect of age at entry? Fit the relevant model to answer this question.
7. If we want to assess the effect of (deterministically) time varying variables we must split time into intervals of length say 1 year. Note that it is immaterial whether we split time before or after we duplicate the dataset for competing risk analysis.

However we first must declare the timescales:

```
> thc.L <- Lexis(entry = list(per = cal.yr(injecdat), tfi = 0),
+   exit = list(per = cal.yr(dox)), exit.status = event, data = th.cmp)
> str(thc.L)
```

Then we can split the data along the time since injection and compute the midpoint of the intervals.

```
> thsplit <- splitLexis(thc.L, breaks = list(tfi = 0:100))
> thsplit$m.tfi <- timeBand(thsplit, "tfi", "middle")
```

Now make an analysis equivalent to the Cox-analysis, using splines to model the underlying hazard (remember that a `Lexis` object allows the use of the extractor functions `status()` and `deltat`):

```
> Pi <- glm(status(thsplit) ~ ns(m.tfi, kn = seq(5, 40, 5), Bo = c(1,
+   50), intercept = T):type + volume:type + offset(log(deltat(thsplit))),
+   family = poisson, data = thsplit)
> P1 <- update(Pi, . ~ . - volume:type + volume)
> anova(Pi, P1)
```

How do the conclusions differ from those from the Cox-model?

8. Now, compute the cumulative dose at the beginning of each interval:

```
> thsplit$cdos <- thsplit$m.tfi * thsplit$gpy
```

You may want to also generate the lagged versions, that is variables which at any one time of follow-up give the cumulative dose as it was, say 10 or 20 years earlier:

```
> thsplit$l10dos <- pmax(thsplit$m.tfi - 10, 0) * thsplit$gpy
```

Fit models that allows you to test whether the cumulative dose has different effects on the three types of liver cancer.

In particular, address the question of whether the effect of cumulative dose is proportional between the three types of liver cancer.

1.16.4 Competing risks: Probability of liver cancer.

It may be of interest to estimate how large a fraction of the thorotrast patients actually get a liver cancer.

1. Estimate this proportion by taking the fraction of the patients that actually acquire a liver cancer (look at the variable `liver`).
2. Work out the Nelson-Aalen estimators for the cumulative incidence of liver cancer and the mortality without liver cancer, as a function of time since injection.

3. Use these two to compute the probability of getting liver cancer before time $t = 1, 2, \dots, 50$ years after injection. (Use the lung cancer example from the lectures).

Remember to make a plot of it.

Chapter 2

Solutions

There is a chapter for each of the exercises that has been used at the course. For each one there is also a printout of the R-program that performs the analyses, as well as the graphs produced by the programs.

2.1 Practice with basic R

Skip this if you are familiar with 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.

2.1.1 Probability functions

R has a set of probability functions for calculating the cumulative probability and its inverse function for calculating quantiles in all the probability distributions you are likely to need. The cumulative probability functions are

`pnorm`, `pchisq`, `pbinom`, `ppois`, etc.

and the quantile functions are

`qnorm`, `qchisq`, `qbinom`, `qpois`, etc.

See `help(pnorm)`, etc., and try

```
> pnorm(1.96)
```

```
[1] 0.9750021
```

```
> qnorm(0.975)
```

```
[1] 1.959964
```

1. Find the probability below 1.5 in a Gaussian (normal) distribution.

```
> pnorm(1.5)
```

```
[1] 0.9331928
```

2. What is the probability between -1.64 and $+1.64$ in a Gaussian distribution?

```
> pnorm(1.64) - pnorm(-1.64)
```

- ```
[1] 0.8989948
```
3. Find the probability below 4.3 in a chi-squared distribution on 1 degree of freedom.  
`> pchisq(4.3, 1)`  
[1] 0.9618876
  4. Find the probability above 4.3 in a chi-squared distribution on 1 degree of freedom.  
`> 1 - pchisq(4.3, 1)`  
[1] 0.03811237
  5. What is the probability above 10 in a chi-squared distribution on 5 df?  
`> 1 - pchisq(10, 5)`  
[1] 0.07523525
  6. What is the 95% quantile in a chi-squared distribution on 1 df?  
`> qchisq(0.95, 1)`  
[1] 3.841459

### 2.1.2 Vectors

1. Create a vector `w` with components 1, -1, 2, -2  
`> w <- c(1, -1, 2, -2)`  
[1] 1 -1 2 -2
2. Display this vector  
`> w`  
[1] 1 -1 2 -2
3. Obtain a description of `w` using `str()`  
`> str(w)`  
num [1:4] 1 -1 2 -2
4. Create the vector `w+1`, and display it.  
`> w + 1`  
[1] 2 0 3 -1
5. Create the vector `v` with components (0, 1, 5, 10, 15, ... , 75) using `c()` and `seq()`.  
`> 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
6. Find the length of this vector.  
`> length(v)`  
[1] 17

### 2.1.3 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)
> 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
```

### 2.1.4 Referencing parts of the data frame

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

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

[1] 2974

> births[2, "bweight"]

[1] 3270

> births[1:10, "bweight"]

[1] 2974 3270 2620 3751 3200 3673 3628 3773 3960 3405
```

1. Display the data on the variable `gestwks` for row 7 in the `births` data frame.

```
> births[7, "gestwks"]

[1] 42.14
```

2. 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
```

3. Display the first 10 rows of the data on the variable `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
```

### 2.1.5 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)

`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 ...
```

This makes sure that `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)

`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 ...
```

1. Convert the variable `sex` into a factor

```
> births$sex <- factor(births$sex)
```

2. Label the levels of `sex` as "M" and "F".

```
> births$sex <- factor(births$sex, labels = c("M", "F"))
```

### 2.1.6 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))
```

```
hyp
normal hyper
 428 72
```

or by specifying the data frame as in

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

```
normal hyper
 428 72
```

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

1. Find the frequency distribution of `sex`.

```
> table(births$sex)
```

```
 M F
264 236
```

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

```
sex
 M F
264 236
```

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

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

```
 hyp
sex normal hyper
 M 221 43
 F 207 29
```

### 2.1.7 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 20–24, 25–29, 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$agegrp <- cut(births$matage, breaks = c(20, 25, 30, 35,
+ 40, 45), right = FALSE)
> with(births, table(agegrp))
```

```
agegrp
[20,25) [25,30) [30,35) [35,40) [40,45)
 2 68 200 194 36
```

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$agegrp = cut(births$matage, breaks = 5, right = FALSE)
> with(births, table(agegrp))
```

```
agegrp
[23,27) [27,31) [31,35) [35,39) [39,43)
 16 83 215 150 36
```

shows 5 intervals of width 4.

1. Summarize the numeric variable `gestwks`, which records the length of gestation for the baby, and make a note of the range of values.

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

 Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
 24.69 37.94 39.13 38.72 40.09 43.16 10.00
```

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`.

```
> births$gest4 <- cut(births$gestwks, breaks = c(20, 35, 37, 39,
+ 45))
```

3. Create a new factor `gest5` which cuts `gestwks` into 5 equal intervals, and make a table of frequencies.

```
> births$gest5 <- cut(births$gestwks, breaks = 5)
> table(births$gest5)

(24.7,28.4] (28.4,32.1] (32.1,35.8] (35.8,39.5] (39.5,43.2]
 5 7 27 237 214
```

## 2.1.8 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)
```

```
`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 3 3 3 3 2 4 4 4 ...
 $ 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 8.00 8.09 7.87 8.23 8.07 ...
 $ vlow : logi FALSE ...
```

creates a logical variable `vlow` (in `births` with levels `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 list the values taken by `bweight` for women whose babies have very low birth weight, try

```
> subset(births, vlow)$bweight
```

```
[1] 1203 1780 1946 1663 1546 628 1999 1791 1019 1402 1880 708 1741 981 693
[16] 1570 864 1764 1618 1500 1595 1801 924 1325 1431 1541 1824 1874 1324 1938
```

to create a new dataframe restricted to women with babies of very low birth weight, try

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

| id             | bweight       | lowbw      | gestwks        | preterm        |
|----------------|---------------|------------|----------------|----------------|
| Min. : 22.0    | Min. : 628    | Min. : 1   | Min. : 24.69   | Min. : 0.000   |
| 1st Qu.: 175.8 | 1st Qu.: 1233 | 1st Qu.: 1 | 1st Qu.: 30.71 | 1st Qu.: 1.000 |
| Median : 257.5 | Median : 1558 | Median : 1 | Median : 32.64 | Median : 1.000 |
| Mean : 249.6   | Mean : 1462   | Mean : 1   | Mean : 32.75   | Mean : 0.862   |
| 3rd Qu.: 326.2 | 3rd Qu.: 1788 | 3rd Qu.: 1 | 3rd Qu.: 35.14 | 3rd Qu.: 1.000 |
| Max. : 476.0   | Max. : 1999   | Max. : 1   | Max. : 40.45   | Max. : 1.000   |
|                |               |            | NA's : 1.00    | NA's : 1.000   |

| 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]: 13 | (37,39]: 3  | (32.1,35.8]: 13 |
| Mean : 33.57   |            |       | [35,39]: 8  | (39,45]: 1  | (35.8,39.5]: 4  |
| 3rd Qu.: 36.75 |            |       | [39,43]: 2  | 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 |               |
| Mean : 7.240   |               |
| 3rd Qu.: 7.489 |               |
| Max. : 7.600   |               |

1. Create a logical variable called `early` according to whether `gestwks` is less than 30 or not. Make a frequency table of `early`.

```
> early <- births$gestwks < 30
> table(early)
```

```
early
FALSE TRUE
 485 5
```

2. Display the `id` numbers of women with `gestwks` less than 30 weeks.

```
> subset(births, early)$id
```

```
[1] 142 181 214 226 275
```

### 2.1.9 Using a text editor with R

When working with R it is best to use a text editor to prepare a batch file (or script) which contains R commands and then to run them from the script. For Windows we recommend using the text editor Tinn-R, but you can use your favourite text editor instead if you prefer. Start up the editor and enter the following lines:

```
library(Epi)
data(births)
births$hyp <- factor(hyp, labels=c("normal", "hyper"))
births$sex <- factor(sex, labels=c("M", "F"))
```

Now save the script and run it. One major advantage of running all your R commands from a script is that you end up with a record of exactly what you did which can be repeated at any time. This will also help you redo the analysis in the (highly likely) event that your data changes before you have finished all analyses.

1. Edit the script to create a factor cutting `matage` at 20, 25, 30, 35, 40, 45 years, and run just this part of the script.
2. Edit the script to create a factor cutting `gestwks` at 20, 35, 37, 39, 45 weeks, and run just this part of the script.
3. Save and run the entire script.

### 2.1.10 Working with R

When starting R it is always a good idea to use `getwd()` to print the working directory. You may not be where you think you are! The command `dir()` can be used to see what files you have in the working directory.

When exiting from R you are offered the chance of saving all the objects in your current work space. This is not recommended as the work space can fill up with temporary objects, and it is easy to forget what these are when you resume the session. It is better to build up a script file as you work, and to run this at the start of a new session.

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

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

```
 id bweight lowbw gestwks
Min. : 1.0 Min. : 628 Min. :0.00 Min. :24.69
1st Qu.:125.8 1st Qu.:2862 1st Qu.:0.00 1st Qu.:37.94
Median :250.5 Median :3188 Median :0.00 Median :39.12
Mean :250.5 Mean :3137 Mean :0.12 Mean :38.72
3rd Qu.:375.2 3rd Qu.:3551 3rd Qu.:0.00 3rd Qu.:40.09
Max. :500.0 Max. :4553 Max. :1.00 Max. :43.16
 NA's :10.00

 preterm matage hyp sex agegrp
Min. : 0.0000 Min. :23.00 normal:428 M:264 [23,27): 16
1st Qu.: 0.0000 1st Qu.:31.00 hyper : 72 F:236 [27,31): 83
Median : 0.0000 Median :34.00 [31,35):215
Mean : 0.1286 Mean :34.03 [35,39):150
3rd Qu.: 0.0000 3rd Qu.:37.00 [39,43): 36
Max. : 1.0000 Max. :43.00
 NA's :10.0000

 gest4 gest5 logbw vlow
(20,35]: 31 (24.7,28.4]: 5 Min. :6.443 Mode :logical
(35,37]: 32 (28.4,32.1]: 7 1st Qu.:7.959 FALSE:470
(37,39]:167 (32.1,35.8]: 27 Median :8.067 TRUE :30
(39,45]:260 (35.8,39.5]:237 Mean :8.023
 NA's : 10 (39.5,43.2]:214 3rd Qu.:8.175
 NA's : 10 Max. :8.424
```

first instructs R to re-direct output away from the R terminal to the file "output.txt" and then summarizes the births data frame, the output from which goes to the sink. While a sink is open all output will go to it. Opening a file with `sink()` will overwrite its contents - to append output to a file, use the `append=TRUE` option with `sink()`. To close a sink, use `sink()` without arguments.

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

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

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

which will save the births data frame in the file `births2.Rdata`. By default the data frame is saved as a binary file, but the option `ascii=TRUE` can be used to save it as a text file. To load the object from the file use

```
> load("births2.Rdata")
```

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

## 2.2 Reading data into R

R 2.5.1

```

Program: data.R
Folder: C:\Bendix\Undervis\SPE\OLD\2007\pracs\r
Started: torsdag 20. september 2007, 16:35:25

```

```
> objects(package:datasets)
Error in try(name) : object "package" not found
 [1] "ability.cov" "airmiles" "AirPassengers"
 [4] "airquality" "anscombe" "attenu"
 [7] "attitude" "austres" "beaver1"
[10] "beaver2" "BJSales" "BJSales.lead"
[13] "BOD" "cars" "ChickWeight"
[16] "chickwts" "co2" "CO2"
[19] "crimtab" "discoveries" "DNase"
[22] "esoph" "euro" "euro.cross"
[25] "eurodist" "EuStockMarkets" "faithful"
[28] "fdeaths" "Formaldehyde" "freeny"
[31] "freeny.x" "freeny.y" "HairEyeColor"
[34] "Harman23.cor" "Harman74.cor" "Indometh"
[37] "infert" "InsectSprays" "iris"
[40] "iris3" "islands" "JohnsonJohnson"
[43] "LakeHuron" "ldeaths" "lh"
[46] "LifeCycleSavings" "Loblolly" "longley"
[49] "lynx" "mdeaths" "morley"
[52] "mtcars" "nhtemp" "Nile"
[55] "nottem" "Orange" "OrchardSprays"
[58] "PlantGrowth" "precip" "presidents"
[61] "pressure" "Puromycin" "quakes"
[64] "randu" "rivers" "rock"
[67] "Seatbelts" "sleep" "stack.loss"
[70] "stack.x" "stackloss" "state.abb"
[73] "state.area" "state.center" "state.division"
[76] "state.name" "state.region" "state.x77"
[79] "sunspot.month" "sunspot.year" "sunspots"
[82] "swiss" "Theoph" "Titanic"
[85] "ToothGrowth" "treering" "trees"
[88] "UCBAdmissions" "UKDriverDeaths" "UKGas"
[91] "USAccDeaths" "USArrests" "USJudgeRatings"
```

```

[94] "USPersonalExpenditure" "uspop" "VADeaths"
[97] "volcano" "warpbreaks" "women"
[100] "WorldPhones" "WWWusage"
> help(Titanic)
> library(Epi)
> data(bdendo)
> fem <- read.table("../data/fem.dat", header = TRUE)
> names(fem)
[1] "ID" "AGE" "IQ" "ANXIETY" "DEPRESS" "SLEEP" "SEX"
[8] "LIFE" "WEIGHT"
> str(fem)
'data.frame': 118 obs. of 9 variables:
 $ ID : int 1 2 3 4 5 6 7 8 9 10 ...
 $ AGE : int 39 41 42 30 35 44 31 39 35 33 ...
 $ IQ : int 94 89 83 99 94 90 94 87 -99 92 ...
 $ ANXIETY: int 2 2 3 2 2 NA 2 3 3 2 ...
 $ DEPRESS: int 2 2 3 2 1 1 2 2 2 2 ...
 $ SLEEP : int 2 2 2 2 1 2 NA 2 2 2 ...
 $ SEX : int 1 1 1 1 1 2 1 1 1 1 ...
 $ LIFE : int 1 1 1 1 2 2 1 2 1 1 ...
 $ WEIGHT : num 2.23 1 1.82 -1.18 -0.14 0.41 -0.68 1.59 -0.55 0.36 ...
> head(fem)
 ID AGE IQ ANXIETY DEPRESS SLEEP SEX LIFE WEIGHT
1 1 39 94 2 2 2 1 1 2.23
2 2 41 89 2 2 2 1 1 1.00
3 3 42 83 3 3 2 1 1 1.82
4 4 30 99 2 2 2 1 1 -1.18
5 5 35 94 2 1 1 1 2 -0.14
6 6 44 90 NA 1 2 2 2 0.41
> fem$IQ[fem$IQ == -99] <- NA
> fem2 <- read.table("../data/fem.dat")
> str(fem2)
'data.frame': 119 obs. of 9 variables:
 $ V1: Factor w/ 118 levels "1","10","100",...: 118 1 30 41 52 63 74 85 96 107 ...
 $ V2: Factor w/ 19 levels "29","30","31",...: 19 11 13 14 2 7 16 3 11 7 ...
 $ V3: Factor w/ 24 levels "-99","100","102",...: 24 18 13 7 23 18 14 18 11 1 ...
 $ V4: Factor w/ 5 levels "1","2","3","4",...: 5 2 2 3 2 2 NA 2 3 3 ...
 $ V5: Factor w/ 4 levels "1","2","3","DEPRESS": 4 2 2 3 2 1 1 2 2 2 ...
 $ V6: Factor w/ 3 levels "1","2","SLEEP": 3 2 2 2 1 2 NA 2 2 ...
 $ V7: Factor w/ 3 levels "1","2","SEX": 3 1 1 1 1 1 2 1 1 1 ...
 $ V8: Factor w/ 3 levels "1","2","LIFE": 3 1 1 1 1 2 2 1 2 1 ...
 $ V9: Factor w/ 68 levels "-0.09","-0.14",...: 68 60 37 52 18 2 27 10 48 9 ...
> head(fem2)
 V1 V2 V3 V4 V5 V6 V7 V8 V9
1 ID AGE IQ ANXIETY DEPRESS SLEEP SEX LIFE WEIGHT
2 1 39 94 2 2 2 1 1 2.23
3 2 41 89 2 2 2 1 1 1
4 3 42 83 3 3 2 1 1 1.82
5 4 30 99 2 2 2 1 1 -1.18
6 5 35 94 2 1 1 1 2 -0.14
> fem2$IQ
NULL
> fem3 <- read.table("../data/fem.dat", sep = "\t")
> str(fem3)
'data.frame': 119 obs. of 1 variable:
 $ V1: Factor w/ 119 levels "1 39 94 2 2 2 1 1 2.23",...: 119 1 31 42 53 64 75 86 97 108 ...
> fem4 <- read.table("../data/fem-dot.dat", header = TRUE)
> str(fem4)
'data.frame': 118 obs. of 9 variables:
 $ ID : int 1 2 3 4 5 6 7 8 9 10 ...
 $ AGE : int 39 41 42 30 35 44 31 39 35 33 ...
 $ IQ : Factor w/ 23 levels ".", "100", "102",...: 18 13 7 23 18 14 18 11 1 16 ...
 $ ANXIETY: Factor w/ 5 levels ".", "1", "2", "3",...: 3 3 4 3 3 1 3 4 4 3 ...
 $ DEPRESS: Factor w/ 4 levels ".", "1", "2", "3": 3 3 4 3 2 2 3 3 3 3 ...
 $ SLEEP : Factor w/ 3 levels ".", "1", "2": 3 3 3 3 2 3 1 3 3 3 ...
 $ SEX : Factor w/ 3 levels ".", "1", "2": 2 2 2 2 2 3 2 2 2 2 ...
 $ LIFE : Factor w/ 3 levels ".", "1", "2": 2 2 2 2 3 3 2 3 2 2 ...
 $ WEIGHT : Factor w/ 68 levels "-0.09", "-0.14",...: 61 38 53 18 2 28 10 49 9 27 ...
> fem4 <- read.table("../data/fem-dot.dat", header = TRUE, na.strings = ".")
> 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, ...)
<environment: namespace:utils>
> library(foreign)
> fem5 <- read.dta("../data/fem.dta")
> head(fem5)
 id age iq anxiety depress sleep sex life weight
1 1 39 94 mild mild no yes yes 2.23
2 2 41 89 mild mild no yes yes 1.00
3 3 42 83 moderate moderate or severe no yes yes 1.82
4 4 30 99 mild mild no yes yes -1.18
5 5 35 94 mild none yes yes no -0.14
6 6 44 90 <NA> none no no no 0.41
> attr(fem5, "var.labels")
[1] "Patient ID"

```

```

[2] "Age in years"
[3] "IQ score"
[4] "Anxiety"
[5] "Depression"
[6] "Sleeping normally"
[7] "Lost interest in sex"
[8] "Considered suicide"
[9] "Weight change (kg) in previous 6 months"
> attributes(fem5)
$datalabel
[1] "Data on 118 female psychiatric patients"

$time.stamp
[1] " 1 Jun 2006 12:36"

$names
[1] "id" "age" "iq" "anxiety" "depress" "sleep" "sex"
[8] "life" "weight"

$formats
[1] "%8.0g" "%8.0g" "%8.0g" "%8.0g" "%18.0g" "%8.0g" "%8.0g" "%8.0g"
[9] "%9.0g"

$types
[1] 252 251 252 251 251 251 251 251 254

$val.labels
[1] "" "" "" "anxiety" "depress" "yesno" "yesno"
[8] "yesno" ""

$var.labels
[1] "Patient ID"
[2] "Age in years"
[3] "IQ score"
[4] "Anxiety"
[5] "Depression"
[6] "Sleeping normally"
[7] "Lost interest in sex"
[8] "Considered suicide"
[9] "Weight change (kg) in previous 6 months"

$row.names
 [1] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10" "11" "12"
[13] "13" "14" "15" "16" "17" "18" "19" "20" "21" "22" "23" "24"
[25] "25" "26" "27" "28" "29" "30" "31" "32" "33" "34" "35" "36"
[37] "37" "38" "39" "40" "41" "42" "43" "44" "45" "46" "47" "48"
[49] "49" "50" "51" "52" "53" "54" "55" "56" "57" "58" "59" "60"
[61] "61" "62" "63" "64" "65" "66" "67" "68" "69" "70" "71" "72"
[73] "73" "74" "75" "76" "77" "78" "79" "80" "81" "82" "83" "84"
[85] "85" "86" "87" "88" "89" "90" "91" "92" "93" "94" "95" "96"
[97] "97" "98" "99" "100" "101" "102" "103" "104" "105" "106" "107" "108"
[109] "109" "110" "111" "112" "113" "114" "115" "116" "117" "118"

$version
[1] -8

$label.table
$label.table$anxiety
 none mild moderate severe
 1 2 3 4

$label.table$depress
 none mild moderate or severe
 1 2 3

$label.table$yesno
yes no
 1 2

$label.table[[4]]
NULL

$label.table[[5]]
NULL

$label.table[[6]]
NULL

$label.table[[7]]
NULL

$label.table[[8]]
NULL

$label.table[[9]]
NULL

```

```
$class
[1] "data.frame"

> names(attributes(fem5))
 [1] "datalabel" "time.stamp" "names" "formats" "types"
 [6] "val.labels" "var.labels" "row.names" "version" "label.table"
[11] "class"
>

Program: data.R
Folder: C:\Bendix\Undervis\SPE\OLD\2007\pracs\r
Ended: torsdag 20. september 2007, 16:35:26
Elapsed: 00:00:01

> proc.time()
 user system elapsed
 1.73 0.16 4.63
```

## 2.3 Tabulation in R

### 2.3.1 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. The functions `table()`, `tapply()`, `by()`, and `xtabs()` will do this. The appearance of these tables is, however, 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 for publication in paper form, or on the World Wide Web. The package `xtables` 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.

### 2.3.2 Basic contingency tables

The `bdendo` data set in the `Epi` package contains data on a case-control study of endometrial cancer. Type

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

to create a copy of the data frame `bdendo` in your work space. Use the functions `str()` and `head()` to inspect the data frame.

The study concerns 63 cases of endometrial cancer that occurred in a retirement community in Los Angeles between 1971 and 1975. Each case was matched with 4 healthy controls, who were also living in the community at the time of the case.

The `table()` function can be used to create contingency tables. The following table cross-tabulates case-control status (`d`) with an indicator of whether the women had used estrogens (`est`)

```
> table(bdendo$d, bdendo$est)

 No Yes
0 125 127
1 7 56
```

The tables produced by the `table()` function are very plain. If you want some summary statistics, use the `twoby2()` function from the `Epi` package.

```
> twoby2(bdendo$d, bdendo$est)
```

```
2 by 2 table analysis:
```

```

Outcome : No
Comparing : 0 vs. 1
```

```
 No Yes P(No) 95% conf. interval
0 125 127 0.4960 0.4347 0.5575
1 7 56 0.1111 0.0539 0.2152
```

```
95% conf. interval
```

```

 Relative Risk: 4.4643 2.1961 9.0752
 Sample Odds Ratio: 7.8740 3.4554 17.9429
Conditional MLE Odds Ratio: 7.8292 3.3856 21.1587
 Probability difference: 0.3849 0.2471 0.4780

 Exact P-value: 0
 Asymptotic P-value: 0

```

Tables in R are *objects* that can be passed on to other functions for further manipulation.

```

> est.tab <- table(bdendo$d, bdendo$est)
> pctab(est.tab)

```

```

 No Yes All N
0 49.6 50.4 100.0 252.0
1 11.1 88.9 100.0 63.0

```

The `pctab()` function takes a contingency table as an argument and turns it into a table of percentages.

You can also pass the table to the `fisher.test()` function which will print some summary statistics for the association between the two variables.

```

> fisher.test(est.tab)

```

```

 Fisher's Exact Test for Count Data

data: est.tab
p-value = 9.133e-09
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 3.385563 21.158729
sample estimates:
odds ratio
 7.829222

```

### 2.3.3 Manipulating contingency tables

The `UCBAdmissions` data set is a ready-made contingency table that comes as part of the “datasets” package in R. You just need to type

```

> UCBAdmissions

```

```

, , Dept = A

```

```

 Gender
Admit Male Female
Admitted 512 89
Rejected 313 19

```

```

, , Dept = B

```

```

 Gender
Admit Male Female
Admitted 353 17
Rejected 207 8

```

```

, , Dept = C

```

```

 Gender

```

```
Admit Male Female
Admitted 120 202
Rejected 205 391
```

```
, , Dept = D
```

```
 Gender
Admit Male Female
Admitted 138 131
Rejected 279 244
```

```
, , Dept = E
```

```
 Gender
Admit Male Female
Admitted 53 94
Rejected 138 299
```

```
, , Dept = F
```

```
 Gender
Admit Male Female
Admitted 22 24
Rejected 351 317
```

to see it, and `help(UCBAdmissions)` to see a description of the data. This is a three-way contingency table of all prospective students who applied to the University of California, Berkeley (UCB) in 1973, classified by sex, department and whether they were accepted or rejected. The five departments are given arbitrary labels “A” to “E”.

### 2.3.3.1 Flattening tables

The default method of printing three-way (or higher-dimensional) tables is not very easy to comprehend. The `ftable()` function “flattens” contingency tables so that they are human-readable

```
> ftable(UCBAdmissions)

 Dept A B C D E F
Admit Gender
Admitted Male 512 353 120 138 53 22
 Female 89 17 202 131 94 24
Rejected Male 313 207 205 279 138 351
 Female 19 8 391 244 299 317
```

A flattened contingency table is an object in its own right:

```
> ucbflat <- ftable(UCBAdmissions)
> ucbflat

 Dept A B C D E F
Admit Gender
Admitted Male 512 353 120 138 53 22
 Female 89 17 202 131 94 24
Rejected Male 313 207 205 279 138 351
 Female 19 8 391 244 299 317
```

An `ftable` object can even be written to file

```
> write.ftable(ucbflat, file = "ucb.txt")
```

Open up the file `ucb.txt` in a text editor to view its contents (it will be in the working directory of your R session). You can also read an `fTable` from a file

```
> ucbflat2 <- read.fTable("ucb.txt")
```

The function `all.equal()` can be used to test whether two R objects are the same or not. Use this to ensure that `ucbflat` and `ucbflat2` are the same.

Flattened tables can also be coerced back to three-dimensional contingency tables

```
> as.table(ucbflat2)
```

```
, , Dept = A
```

|          | Gender |        |
|----------|--------|--------|
| Admit    | Male   | Female |
| Admitted | 512    | 89     |
| Rejected | 313    | 19     |

```
, , Dept = B
```

|          | Gender |        |
|----------|--------|--------|
| Admit    | Male   | Female |
| Admitted | 353    | 17     |
| Rejected | 207    | 8      |

```
, , Dept = C
```

|          | Gender |        |
|----------|--------|--------|
| Admit    | Male   | Female |
| Admitted | 120    | 202    |
| Rejected | 205    | 391    |

```
, , Dept = D
```

|          | Gender |        |
|----------|--------|--------|
| Admit    | Male   | Female |
| Admitted | 138    | 131    |
| Rejected | 279    | 244    |

```
, , Dept = E
```

|          | Gender |        |
|----------|--------|--------|
| Admit    | Male   | Female |
| Admitted | 53     | 94     |
| Rejected | 138    | 299    |

```
, , Dept = F
```

|          | Gender |        |
|----------|--------|--------|
| Admit    | Male   | Female |
| Admitted | 22     | 24     |
| Rejected | 351    | 317    |

Use the `all.equal()` function again to ensure that the three dimensional table you have produced is the same as the original `UCBAdmissions` data.

### 2.3.3.2 Coercing tables to data frames

Another way of converting a three-dimensional table into a two-dimensional structure is to turn it into a data frame

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

```
 Admit Gender Dept Freq
1 Admitted Male A 512
2 Rejected Male A 313
3 Admitted Female A 89
4 Rejected Female A 19
5 Admitted Male B 353
6 Rejected Male B 207
7 Admitted Female B 17
8 Rejected Female B 8
9 Admitted Male C 120
10 Rejected Male C 205
11 Admitted Female C 202
12 Rejected Female C 391
13 Admitted Male D 138
14 Rejected Male D 279
15 Admitted Female D 131
16 Rejected Female D 244
17 Admitted Male E 53
18 Rejected Male E 138
19 Admitted Female E 94
20 Rejected Female E 299
21 Admitted Male F 22
22 Rejected Male F 351
23 Admitted Female F 24
24 Rejected Female F 317
```

This creates a new data frame with one variable for each of the classifying factors and an extra variable “Freq” for the frequency counts.

We can create collapsed contingency tables from this data frame using the `xtabs()` function

```
> xtabs(Freq ~ Gender + Admit, data = ucb.frame)
```

```
 Admit
Gender Admitted Rejected
Male 1198 1493
Female 557 1278
```

This function uses a formula interface to create a table. It sums over the value of `Freq` within cells defined by cross-classifying `Gender` and `Admit`. Note that `xtabs()` has a `data` argument which tells it where to look for variables. This makes it easier to use than the `table()` function.

Use the `pctab()` function to turn this contingency table into a table of percentages. What does this tell you about the relative success rates of the two genders applying to UCB in 1973?

### 2.3.4 Mantel-Haenszel testing on tables

The reason why we are using the `UCBAdmissions` data set (even though it is not an epidemiological example) is because it is a nice example of confounding. Females are more often rejected than males. The question is whether this is due to sex discrimination.

Use the `xtabs()` and `pctab()` functions to find

1. the percent of all applicants rejected by each department
2. the distribution of applicants among departments separately for males and females

These tables suggest that the higher rejection probability for females could be the result of confounding by department.

The function `mantelhaen.test()` does Mantel-Haenszel testing for the independence of two factors within strata defined by a third factor. One way to call a function is to supply, as a single argument, a 3-dimensional contingency table, for which the first two dimensions are the factors of interest.

Apply the function `mantelhaen.test()` to the original `UCBAdmissions` three-way table to see if gender is associated with acceptance within strata defined by department.

```
> mantelhaen.test(UCBAdmissions)

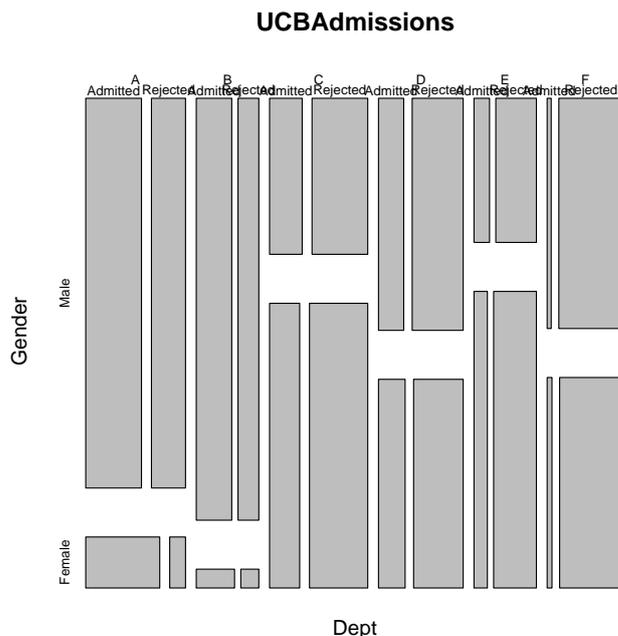
Mantel-Haenszel chi-squared test with continuity correction

data: UCBAdmissions
Mantel-Haenszel X-squared = 1.4269, df = 1, p-value = 0.2323
alternative hypothesis: true common odds ratio is not equal to 1
95 percent confidence interval:
 0.7719074 1.0603298
sample estimates:
common odds ratio
 0.9046968
```

For  $2 \times 2$  tables, the function `mantelhaen.test()` not only gives a  $p$ -value, but also a summary odds ratio and 95% confidence interval.

There is no significant association between gender and admission after controlling for department. The explanation for the higher success rate of male candidates is that they were more likely to apply for departments with a higher acceptance probability. A nice way to see this visually is with a mosaic plot.

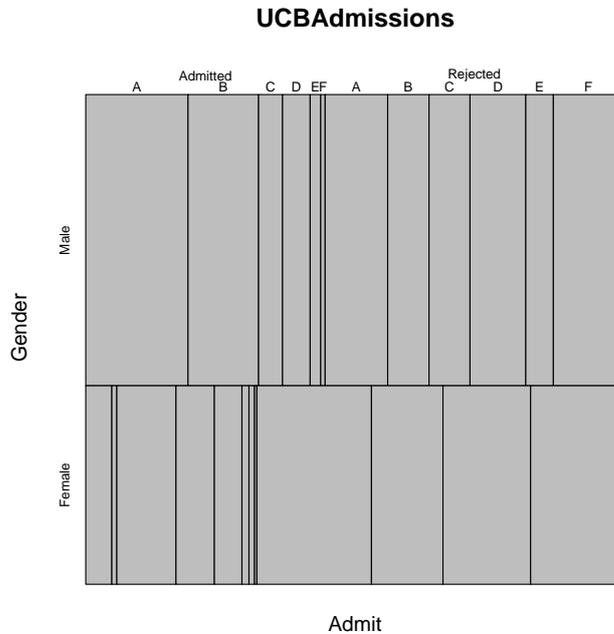
```
> plt("UCB1")
> mosaicplot(UCBAdmissions, sort = c(3, 2, 1))
```



The argument `sort` determines what order the margins of the table are taken in. Change the order of the `sort` argument to display other mosaic plots, and try to interpret them.

Here we also use the `offset` argument to remove the space between the blocks:

```
> plt("UCB2")
> mosaicplot(UCBAdmissions, sort = c(2, 1, 3), off = c(0, 0, 0))
```



### 2.3.5 Tables of summary statistics

The `stat.table()` function in the `Epi` package provides more printer-friendly tables than the functions provided by base R. The `stat.table()` function can be used to produce both contingency tables and tables of summary statistics.

#### 2.3.5.1 One-way tables

You will need to use the data set `nickel` which is contained in the `Epi` package.

```
> data(nickel)
```

This data set is an occupational cohort of workers in the nickel refining industry. Each of the 679 subjects has a numeric exposure index, `exp`, based on their job history, which estimates their life-long exposure to nickel. Create a new variable `exp4` that classifies the exposure into four categories

```
> nickel$exp4 <- cut(nickel$exp, breaks = c(0, 0.5, 4.5, 8.5, Inf),
+ include.lowest = TRUE, right = FALSE)
```

The simplest table is created by

```
> stat.table(index = exp4, data = nickel)
```

```

exp4 count()

[0,0.5) 290
[0.5,4.5) 225
[4.5,8.5) 103
[8.5,Inf] 61

```

This creates a count of individuals, classified by levels of the factor `exp4`. 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 attaching it.

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

```
> stat.table(index = exp4, contents = list(count(), percent(exp4)),
+ data = nickel)
```

```

exp4 count() percent(exp4)

[0,0.5) 290 42.7
[0.5,4.5) 225 33.1
[4.5,8.5) 103 15.2
[8.5,Inf] 61 9.0

```

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 `nickel`.

### 2.3.5.2 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 a *tagged* list as the value of the `contents` argument:

```
contents = list("N" = count(), "(%" = percent(exp4))
```

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 `exp4` as the `index` argument to `stat.table()`, use a named list:

```
index = list("Years of exposure" = exp4)
```

```
> stat.table(index = list("Years of exposure" = exp4), contents = list(N = count(),
+ "(%" = percent(exp4)), data = nickel)
```

```

Years of N (%)
exposure

[0,0.5) 290 42.7
[0.5,4.5) 225 33.1
[4.5,8.5) 103 15.2
[8.5,Inf] 61 9.0

```

### 2.3.5.3 Cases, Follow-up and Rates

The above examples illustrate the basic features of `stat.table()`. However, our main interest in the Welsh nickel-smelter's study, is not to count subjects, but to evaluate the risk with years of exposure in "high-risk" occupations.

Add two new variables, one indicating death from lung cancer (`d.lung`) and one measuring the follow-up time in years.

```
> nickel <- transform(nickel, d.lung = icd %in% c(162, 163), flwupt = ageout -
+ agein)
```

A count of cases and follow-up time can be created using the following contents argument:

```
contents = list(sum(d.lung), sum(flwupt))
```

A table with these contents, with informative labels.

```
> stat.table(index = exp4, contents = list(D = sum(d.lung), Y = sum(flwupt)),
+ data = nickel)
```

```

exp4 D Y

[0,0.5) 42.00 7801.74
[0.5,4.5) 50.00 4924.49
[4.5,8.5) 27.00 1744.06
[8.5,Inf] 18.00 877.77

```

To calculate rates, a special function `ratio()` is provided. A call to `ratio(d, y, scale)` calculates `scale * sum(d) / sum(y)` within categories defined by the `index` variable. To calculate incidence rates per 100,000 person-years we therefore use.

```
contents = ratio(d.lung, flwupt, 100000)
```

Adding an extra column to the table gives the incidence rates in addition to number of cases and follow-up time.

```
> stat.table(index = exp4, contents = list(D = sum(d.lung), Y = sum(flwupt),
+ "Rate/1000" = ratio(d.lung, flwupt, 1e+05)), data = nickel)
```

```

exp4 D Y Rate/1000

[0,0.5) 42.00 7801.74 538.34
[0.5,4.5) 50.00 4924.49 1015.33
[4.5,8.5) 27.00 1744.06 1548.11
[8.5,Inf] 18.00 877.77 2050.66

```

### 2.3.5.4 Printing tables

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 `stat.table` object with an explicit call to `print(my.table)`

There are two arguments to the `print` method for `stat.table` objects. The `width` argument specifies the minimum column width. Use this to print one of the tables you created above, preventing long column headers being folded over too many lines:

```
> print(stat.table(index = exp4, contents = list(D = sum(d.lung),
+ Y = sum(flwupt), "Rate/1000" = ratio(d.lung, flwupt, 1e+05)),
+ data = nickel), width = 9)
```

```

exp4 D Y Rate/1000

[0,0.5) 42.00 7801.74 538.34
[0.5,4.5) 50.00 4924.49 1015.33
[4.5,8.5) 27.00 1744.06 1548.11
[8.5,Inf] 18.00 877.77 2050.66

```

The second argument to the print method is `digits` which controls the number of digits printed after the decimal point. This table

```
> case.tab <- stat.table(exp4, list(cases = sum(d.lung)), data = nickel)
```

counts lung cancer cases, but prints them to 2-decimal places. Use the `digits` argument to print the table correctly.

```
> print(case.tab, digits = 0)
```

```

exp4 cases

[0,0.5) 42
[0.5,4.5) 50
[4.5,8.5) 27
[8.5,Inf] 18

```

Use `print.default()` instead of `print()` to print one of your tables. This shows the internal structure of the table. You may need to know this if you wish to extract data from a `stat.table` object:

```
> print.default(stat.table(index = exp4, contents = list(D = sum(d.lung),
+ Y = sum(flwupt), "Rate/1000" = ratio(d.lung, flwupt, 1e+05)),
+ data = nickel))
```

```

 exp4
contents [0,0.5) [0.5,4.5) [4.5,8.5) [8.5,Inf]
D 42.0000 50.000 27.000 18.0000
Y 7801.7387 4924.492 1744.059 877.7672
Rate/1000 538.3415 1015.333 1548.113 2050.6576
attr(,"table.fun")
[1] "sum" "sum" "ratio"
attr(,"class")
[1] "stat.table" "matrix"
```

### 2.3.6 Summary

In this exercise we have seen that tables in R are also *objects* which can be passed to other R functions for further analysis. Table objects can be transformed into other objects without loss of information.

The `stat.table()` function in the Epi package provides print-friendly tables of summary statistics. Further information about the capabilities of `stat.table()` can be found in the help page.

## 2.4 Logistic regression (glm)

### 2.4.1 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 2.1.

Table 2.1: *Variables in the melanoma dataset.*

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

### 2.4.2 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 your working directory) to a data frame with name `mm` 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 `mm`.

```
'data.frame': 1400 obs. of 9 variables:
 $ cc : int 1 1 1 0 1 0 0 0 0 1 ...
 $ sex : int 2 1 2 2 2 2 2 1 2 2 ...
 $ age : int 71 68 42 66 36 68 68 39 75 49 ...
 $ skin : int 2 2 1 0 1 2 0 2 2 2 ...
 $ hair : int 0 0 1 2 0 2 0 0 0 1 ...
 $ eyes : int 2 2 2 1 2 2 1 2 2 2 ...
 $ freckles: int 2 1 3 2 3 2 2 2 1 2 ...
 $ nvsmall : int 2 3 22 0 1 0 0 3 5 6 ...
 $ nvlarge : int 0 0 1 0 0 0 0 0 0 0 ...
```

|    | cc | sex | age | skin | hair | eyes | freckles | nvsmall | nvlarge |
|----|----|-----|-----|------|------|------|----------|---------|---------|
| 1  | 1  | 2   | 71  | 2    | 0    | 2    | 2        | 2       | 0       |
| 2  | 1  | 1   | 68  | 2    | 0    | 2    | 1        | 3       | 0       |
| 3  | 1  | 2   | 42  | 1    | 1    | 2    | 3        | 22      | 1       |
| 4  | 0  | 2   | 66  | 0    | 2    | 1    | 2        | 0       | 0       |
| 5  | 1  | 2   | 36  | 1    | 0    | 2    | 3        | 1       | 0       |
| 6  | 0  | 2   | 68  | 2    | 2    | 2    | 2        | 0       | 0       |
| 7  | 0  | 2   | 68  | 0    | 0    | 1    | 2        | 0       | 0       |
| 8  | 0  | 1   | 39  | 2    | 0    | 2    | 2        | 3       | 0       |
| 9  | 0  | 2   | 75  | 2    | 0    | 2    | 1        | 5       | 0       |
| 10 | 1  | 2   | 49  | 2    | 1    | 2    | 2        | 6       | 0       |
| 11 | 0  | 1   | 48  | 2    | 1    | 2    | 3        | 4       | 0       |
| 12 | 1  | 2   | 67  | 0    | 0    | 2    | 2        | 1       | 0       |
| 13 | 0  | 1   | 50  | 1    | 0    | 2    | 3        | 4       | 0       |
| 14 | 1  | 2   | 38  | 2    | 0    | 1    | 3        | 8       | 0       |
| 15 | 0  | 2   | 33  | 2    | 1    | 2    | 2        | 3       | 0       |
| 16 | 0  | 2   | 39  | 1    | 0    | 1    | 3        | 0       | 2       |
| 17 | 0  | 2   | 39  | 1    | 1    | 2    | 3        | 0       | 0       |
| 18 | 1  | 1   | 50  | 0    | 1    | 1    | 1        | 3       | 1       |
| 19 | 0  | 2   | 35  | 2    | 0    | 2    | 2        | 1       | 0       |
| 20 | 0  | 2   | 35  | 2    | 0    | 1    | 3        | 5       | 0       |

### 2.4.3 House keeping

The structure of the data frame `mm` 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. Note that the file starts by reading in the data, so whenever you run it you start with the original data set. 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 `mm` and note the changes. Use the command `summary(mm)` to look at the univariate distributions.

```
'data.frame': 1400 obs. of 13 variables:
 $ cc : int 1 1 1 0 1 0 0 0 0 1 ...
 $ sex : Factor w/ 2 levels "M","F": 2 1 2 2 2 2 2 1 2 2 ...
 $ age : int 71 68 42 66 36 68 68 39 75 49 ...
```

```

$ skin : Factor w/ 3 levels "dark","medium",...: 3 3 2 1 2 3 1 3 3 3 ...
$ hair : Factor w/ 4 levels "dark","light_brown",...: 1 1 2 3 1 3 1 1 1 2 ...
$ eyes : Factor w/ 3 levels "brown","grey-green",...: 3 3 3 2 3 3 2 3 3 3 ...
$ freckles : Factor w/ 3 levels "none","some",...: 2 3 1 2 1 2 2 2 3 2 ...
$ nvsmall : int 2 3 22 0 1 0 0 3 5 6 ...
$ nvlarge : int 0 0 1 0 0 0 0 0 0 0 ...
$ age.cat : Factor w/ 6 levels "[20,30)","[30,40)",...: 6 5 3 5 2 5 5 2 6 3 ...
$ hair2 : Factor w/ 2 levels "dark","other": 1 1 2 2 1 2 1 1 1 2 ...
$ nvsm4 : Factor w/ 4 levels "[0,1)","[1,2)",...: 3 3 4 1 2 1 1 3 4 4 ...
$ nvlar3 : Factor w/ 3 levels "[0,1)","[1,2)",...: 1 1 2 1 1 1 1 1 1 1 ...

```

| cc             | sex   | age           | skin       | hair            | eyes           | freckles |
|----------------|-------|---------------|------------|-----------------|----------------|----------|
| Min. :0.0000   | M:584 | Min. :21.00   | dark :318  | dark :690       | brown :187     | none:633 |
| 1st Qu.:0.0000 | F:816 | 1st Qu.:42.00 | medium:594 | light_brown:548 | grey-green:450 | some:526 |
| Median :0.0000 |       | Median :53.00 | light :478 | blonde : 61     | blue :757      | many:237 |
| Mean :0.3386   |       | Mean :52.89   | NA's : 10  | red :101        | NA's : 6       | NA's: 4  |
| 3rd Qu.:1.0000 |       | 3rd Qu.:64.00 |            |                 |                |          |
| Max. :1.0000   |       | Max. :81.00   |            |                 |                |          |

| nvsmall        | nvlarge         | age.cat     | hair2     | nvsm4      | nvlar3      |
|----------------|-----------------|-------------|-----------|------------|-------------|
| Min. : 0.000   | Min. : 0.0000   | [20,30): 61 | dark :690 | [0,1) :922 | [0,1) :1263 |
| 1st Qu.: 0.000 | 1st Qu.: 0.0000 | [30,40):202 | other:710 | [1,2) :192 | [1,2) : 95  |
| Median : 0.000 | Median : 0.0000 | [40,50):347 |           | [2,5) :176 | [2,15): 35  |
| Mean : 1.163   | Mean : 0.1565   | [50,60):296 |           | [5,50):103 | NA's : 7    |
| 3rd Qu.: 1.000 | 3rd Qu.: 0.0000 | [60,70):307 |           | NA's : 7   |             |
| Max. :46.000   | Max. :14.0000   | [70,85):187 |           |            |             |
| NA's : 7.000   | NA's : 7.0000   |             |           |            |             |

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

#### 2.4.4 One variable at a time

As a first step it is a good idea to start by looking at the effect of each of the variables, controlled for age in 10 year age groups and sex. Try

```
> effx(cc, type = "binary", exposure = skin, control = list(age.cat, sex), data = mm)
```

```

response : cc
type : "binary"
exposure : skin
control vars : age.cat sex

```

```

skin is a factor with levels: dark / medium / light
baseline is dark
effects are measured as odds ratios

```

```

effect of skin on cc
controlled for age.cat sex

```

```
number of observations 1390
```

```

 Effect 2.5% 97.5%
medium vs dark 1.39 1.03 1.88
light vs dark 1.65 1.21 2.26

```

```
Test for no effects of exposure on 2 df: p= 0.00595
```

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

```

response : cc

```

```

type : "binary"
exposure : skin
control vars : age.cat sex

```

```

skin is a factor with levels: dark / medium / light
baseline is dark
effects are measured as odds ratios

```

```

effect of skin on cc
controlled for age.cat sex

```

```

number of observations 1390

```

|                | Effect | 2.5% | 97.5% |
|----------------|--------|------|-------|
| medium vs dark | 1.39   | 1.03 | 1.88  |
| light vs dark  | 1.65   | 1.21 | 2.26  |

```

Test for no effects of exposure on 2 df: p= 0.00595

```

```

response : cc
type : "binary"
exposure : hair
control vars : age.cat sex

```

```

hair is a factor with levels: dark / light_brown / blonde / red
baseline is dark
effects are measured as odds ratios

```

```

effect of hair on cc
controlled for age.cat sex

```

```

number of observations 1400

```

|                     | Effect | 2.5%  | 97.5% |
|---------------------|--------|-------|-------|
| light_brown vs dark | 1.50   | 1.180 | 1.91  |
| blonde vs dark      | 1.68   | 0.981 | 2.88  |
| red vs dark         | 1.78   | 1.160 | 2.75  |

```

Test for no effects of exposure on 3 df: p= 0.00148

```

```

response : cc
type : "binary"
exposure : eyes
control vars : age.cat sex

```

```

eyes is a factor with levels: brown / grey-green / blue
baseline is brown
effects are measured as odds ratios

```

```

effect of eyes on cc
controlled for age.cat sex

```

```

number of observations 1394

```

|                     | Effect | 2.5%  | 97.5% |
|---------------------|--------|-------|-------|
| grey-green vs brown | 0.835  | 0.580 | 1.20  |
| blue vs brown       | 1.050  | 0.751 | 1.48  |

Test for no effects of exposure on 2 df: p= 0.189

```

response : cc
type : "binary"
exposure : freckles
control vars : age.cat sex
```

```
freckles is a factor with levels: none / some / many
baseline is none
effects are measured as odds ratios

```

effect of freckles on cc  
controlled for age.cat sex

number of observations 1396

|              | Effect | 2.5% | 97.5% |
|--------------|--------|------|-------|
| some vs none | 1.51   | 1.17 | 1.95  |
| many vs none | 3.07   | 2.24 | 4.22  |

Test for no effects of exposure on 2 df: p= 2.55e-11

## 2.4.5 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.frk <- glm(cc ~ freckles + age.cat + sex, family = "binomial", data = mm)
> summary(m.frk)
```

Call:

```
glm(formula = cc ~ freckles + age.cat + sex, family = "binomial",
 data = mm)
```

Deviance Residuals:

| Min     | 1Q      | Median  | 3Q     | Max    |
|---------|---------|---------|--------|--------|
| -1.2630 | -0.9123 | -0.7732 | 1.3876 | 1.6914 |

Coefficients:

|                | Estimate | Std. Error | z value | Pr(> z ) |
|----------------|----------|------------|---------|----------|
| (Intercept)    | -0.89385 | 0.29048    | -3.077  | 0.00209  |
| frecklessome   | 0.41206  | 0.13030    | 3.163   | 0.00156  |
| frecklesmany   | 1.12242  | 0.16184    | 6.936   | 4.05e-12 |
| age.cat[30,40) | -0.10515 | 0.31393    | -0.335  | 0.73767  |
| age.cat[40,50) | -0.09830 | 0.29765    | -0.330  | 0.74120  |
| age.cat[50,60) | -0.02955 | 0.30155    | -0.098  | 0.92194  |
| age.cat[60,70) | -0.20094 | 0.30216    | -0.665  | 0.50604  |
| age.cat[70,85) | -0.11732 | 0.31666    | -0.370  | 0.71101  |
| sexF           | -0.06223 | 0.11910    | -0.523  | 0.60132  |

(Dispersion parameter for binomial family taken to be 1)

```
Null deviance: 1787.6 on 1395 degrees of freedom
Residual deviance: 1737.1 on 1387 degrees of freedom
(4 observations deleted due to missingness)
AIC: 1755.1
```

Number of Fisher Scoring iterations: 4

```
> coef(m.frk)
```

```

(Intercept) frecklessome frecklesmany age.cat[30,40] age.cat[40,50] age.cat[50,60] age.cat[60,70] age.cat[70,85]
-0.89385163 0.41206202 1.12242246 -0.10514744 -0.09830352 -0.02954865 -0.20093865 -0.11732065
age.cat[70,85] sexF
-0.11732065 -0.06223165

```

```
> exp(coef(m.frk))
```

```

(Intercept) frecklessome frecklesmany age.cat[30,40] age.cat[40,50] age.cat[50,60] age.cat[60,70] age.cat[70,85]
0.4090771 1.5099281 3.0722877 0.9001918 0.9063738 0.9708836 0.8179626 0.8893000
age.cat[70,85] sexF
0.8893000 0.9396652

```

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

```
> ci.lin(m.frk, Exp = TRUE)
```

|                | Estimate    | StdErr    | z           | P            | exp(Est.) | 2.5%      | 97.5%     |
|----------------|-------------|-----------|-------------|--------------|-----------|-----------|-----------|
| (Intercept)    | -0.89385163 | 0.2904794 | -3.07716003 | 2.089831e-03 | 0.4090771 | 0.2314987 | 0.7228725 |
| frecklessome   | 0.41206202  | 0.1302954 | 3.16252098  | 1.564095e-03 | 1.5099281 | 1.1696303 | 1.9492338 |
| frecklesmany   | 1.12242246  | 0.1618354 | 6.93557961  | 4.045653e-12 | 3.0722877 | 2.2372129 | 4.2190672 |
| age.cat[30,40] | -0.10514744 | 0.3139260 | -0.33494338 | 7.376678e-01 | 0.9001918 | 0.4865425 | 1.6655181 |
| age.cat[40,50] | -0.09830352 | 0.2976535 | -0.33026163 | 7.412023e-01 | 0.9063738 | 0.5057597 | 1.6243156 |
| age.cat[50,60] | -0.02954865 | 0.3015530 | -0.09798826 | 9.219416e-01 | 0.9708836 | 0.5376316 | 1.7532730 |
| age.cat[60,70] | -0.20093865 | 0.3021576 | -0.66501273 | 5.060423e-01 | 0.8179626 | 0.4524144 | 1.4788716 |
| age.cat[70,85] | -0.11732065 | 0.3166598 | -0.37049423 | 7.110143e-01 | 0.8893000 | 0.4780870 | 1.6542062 |
| sexF           | -0.06223165 | 0.1191019 | -0.52250772 | 6.013169e-01 | 0.9396652 | 0.7440351 | 1.1867325 |

```
> round(ci.lin(m.frk, Exp = TRUE, alpha = 0.1)[, c(5, 6, 7)], 2)
```

|                | exp(Est.) | 5.0% | 95.0% |
|----------------|-----------|------|-------|
| (Intercept)    | 0.41      | 0.25 | 0.66  |
| frecklessome   | 1.51      | 1.22 | 1.87  |
| frecklesmany   | 3.07      | 2.35 | 4.01  |
| age.cat[30,40] | 0.90      | 0.54 | 1.51  |
| age.cat[40,50] | 0.91      | 0.56 | 1.48  |
| age.cat[50,60] | 0.97      | 0.59 | 1.59  |
| age.cat[60,70] | 0.82      | 0.50 | 1.34  |
| age.cat[70,85] | 0.89      | 0.53 | 1.50  |
| sexF           | 0.94      | 0.77 | 1.14  |

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  |

## 2.4.6 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 with `freckles` and one without, and recording the change in deviance.

```

> m1 <- glm(cc ~ freckles + age.cat + sex, family = "binomial", data = mm)
> m2 <- glm(cc ~ age.cat + sex, family = "binomial", data = mm, subset = !is.na(freckles))
> summary(m1)

```

Call:

```
glm(formula = cc ~ freckles + age.cat + sex, family = "binomial",
 data = mm)
```

Deviance Residuals:

| Min | 1Q | Median | 3Q | Max |
|-----|----|--------|----|-----|
|-----|----|--------|----|-----|

```
-1.2630 -0.9123 -0.7732 1.3876 1.6914
```

Coefficients:

|                | Estimate | Std. Error | z value | Pr(> z ) |
|----------------|----------|------------|---------|----------|
| (Intercept)    | -0.89385 | 0.29048    | -3.077  | 0.00209  |
| frecklesome    | 0.41206  | 0.13030    | 3.163   | 0.00156  |
| frecklesmany   | 1.12242  | 0.16184    | 6.936   | 4.05e-12 |
| age.cat[30,40) | -0.10515 | 0.31393    | -0.335  | 0.73767  |
| age.cat[40,50) | -0.09830 | 0.29765    | -0.330  | 0.74120  |
| age.cat[50,60) | -0.02955 | 0.30155    | -0.098  | 0.92194  |
| age.cat[60,70) | -0.20094 | 0.30216    | -0.665  | 0.50604  |
| age.cat[70,85) | -0.11732 | 0.31666    | -0.370  | 0.71101  |
| sexF           | -0.06223 | 0.11910    | -0.523  | 0.60132  |

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1787.6 on 1395 degrees of freedom  
 Residual deviance: 1737.1 on 1387 degrees of freedom  
 (4 observations deleted due to missingness)  
 AIC: 1755.1

Number of Fisher Scoring iterations: 4

```
> summary(m2)
```

Call:

```
glm(formula = cc ~ age.cat + sex, family = "binomial", data = mm,
 subset = !is.na(freckles))
```

Deviance Residuals:

| Min     | 1Q      | Median  | 3Q     | Max    |
|---------|---------|---------|--------|--------|
| -0.9524 | -0.9037 | -0.8859 | 1.4422 | 1.5160 |

Coefficients:

|                | Estimate | Std. Error | z value | Pr(> z ) |
|----------------|----------|------------|---------|----------|
| (Intercept)    | -0.67853 | 0.28168    | -2.409  | 0.016    |
| age.cat[30,40) | -0.08943 | 0.30900    | -0.289  | 0.772    |
| age.cat[40,50) | -0.08072 | 0.29291    | -0.276  | 0.783    |
| age.cat[50,60) | 0.07473  | 0.29628    | 0.252   | 0.801    |
| age.cat[60,70) | -0.05427 | 0.29647    | -0.183  | 0.855    |
| age.cat[70,85) | 0.07490  | 0.31000    | 0.242   | 0.809    |
| sexF           | 0.04837  | 0.11574    | 0.418   | 0.676    |

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1787.6 on 1395 degrees of freedom  
 Residual deviance: 1785.9 on 1389 degrees of freedom  
 AIC: 1799.9

Number of Fisher Scoring iterations: 4

The change in residual deviance is  $1785.9 - 1737.1 = 48.8$  on  $1389 - 1387 = 2$  degrees of freedom. Use the function `pchisq` to find the probability of exceeding 48.8 on 2df. The test is more easily carried out with

```
> anova(m2, m1, test = "Chisq")
```

Analysis of Deviance Table

Model 1: `cc ~ age.cat + sex`

Model 2: `cc ~ freckles + age.cat + sex`

|   | Resid. Df | Resid. Dev | Df | Deviance | P(> Chi ) |
|---|-----------|------------|----|----------|-----------|
| 1 | 1389      | 1785.89    |    |          |           |
| 2 | 1387      | 1737.10    | 2  | 48.79    | 2.549e-11 |

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.

### 2.4.7 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.

```

response : cc
type : "binary"
exposure : hair2
```

```
hair2 is a factor with levels: dark / other
baseline is dark
effects are measured as odds ratios

```

```
effect of hair2 on cc
number of observations 1400
```

```
Effect 2.5% 97.5%
 1.54 1.23 1.92
```

Test for no effects of exposure on 1 df: p= 0.000143

Reproduce these results by fitting an appropriate glm.

### 2.4.8 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 = mm)
```

```

response : cc
type : "binary"
exposure : freckles
control vars : age.cat sex
stratified by : hair2
```

```
freckles is a factor with levels: none / some / many
baseline is none
hair2 is a factor with levels: dark/other
effects are measured as odds ratios

```

```
effect of freckles on cc
controlled for age.cat sex
```

```
stratified by hair2
```

```
number of observations 1396
```

```
Effect 2.5% 97.5%
```

```
strata dark level some vs none 1.61 1.11 2.34
strata other level some vs none 1.42 1.00 2.01
strata dark level many vs none 2.84 1.76 4.58
strata other level many vs none 3.15 2.06 4.80
```

Test for effect modification on 2 df: p= 0.757

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

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

## 2.4.9 Stratification using glm

We shall reproduce the output from

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

```

response : cc
type : "binary"
exposure : freckles
control vars : age.cat sex
stratified by : hair2
```

```
freckles is a factor with levels: none / some / many
baseline is none
hair2 is a factor with levels: dark/other
effects are measured as odds ratios

```

```
effect of freckles on cc
controlled for age.cat sex
```

```
stratified by hair2
```

```
number of observations 1396
```

```

 Effect 2.5% 97.5%
strata dark level some vs none 1.61 1.11 2.34
strata other level some vs none 1.42 1.00 2.01
strata dark level many vs none 2.84 1.76 4.58
strata other level many vs none 3.15 2.06 4.80
```

Test for effect modification on 2 df: p= 0.757

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

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

```

(Intercept) hair2other age.cat [30,40) age.cat [40,50)
0.3169581 1.5639083 0.9286674 0.9573093
age.cat [50,60) age.cat [60,70) age.cat [70,85) sexF
1.0464308 0.8495081 0.9351315 0.9012339
hair2dark:frecklessome hair2other:frecklessome hair2dark:frecklesmany hair2other:frecklesmany
1.6123583 1.4196216 2.8378600 3.1469251
```

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). You can improve this output with `ci.lin`. Try this.



```

[0,1) 93.9 4.9 1.2
[1,2) 89.6 8.3 2.1
[2,5) 85.8 9.7 4.5
[5,50) 71.8 16.5 11.7

```

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 logistic regression models including `sex`, `age.cat` and `nvsma4` in the model formula.

```

response : cc
type : "binary"
exposure : nvsma4
control vars : age.cat sex
```

```
nvsma4 is a factor with levels: [0,1) / [1,2) / [2,5) / [5,50)
baseline is [0,1)
effects are measured as odds ratios
```

```

effect of nvsma4 on cc
controlled for age.cat sex
```

```
number of observations 1393
```

|                 | Effect | 2.5% | 97.5% |
|-----------------|--------|------|-------|
| [1,2) vs [0,1)  | 1.59   | 1.15 | 2.21  |
| [2,5) vs [0,1)  | 2.47   | 1.77 | 3.43  |
| [5,50) vs [0,1) | 5.06   | 3.28 | 7.81  |

```
Test for no effects of exposure on 3 df: p= <2e-16
```

|                | exp(Est.) | 2.5%      | 97.5%     |
|----------------|-----------|-----------|-----------|
| (Intercept)    | 0.3564499 | 0.1993832 | 0.6372481 |
| nvsma4[1,2)    | 1.5936186 | 1.1472356 | 2.2136867 |
| nvsma4[2,5)    | 2.4653503 | 1.7722804 | 3.4294527 |
| nvsma4[5,50)   | 5.0584685 | 3.2777723 | 7.8065531 |
| age.cat[30,40) | 0.9595575 | 0.5110162 | 1.8018030 |
| age.cat[40,50) | 1.0172163 | 0.5595729 | 1.8491407 |
| age.cat[50,60) | 1.1624871 | 0.6351686 | 2.1275867 |
| age.cat[60,70) | 1.0682888 | 0.5832709 | 1.9566227 |
| age.cat[70,85) | 1.1719606 | 0.6228091 | 2.2053172 |
| sexF           | 0.9553673 | 0.7552541 | 1.2085027 |

Do the same with `nvlar3`.

```

response : cc
type : "binary"
exposure : nvlar3
control vars : age.cat sex
```

```
nvlar3 is a factor with levels: [0,1) / [1,2) / [2,15)
baseline is [0,1)
effects are measured as odds ratios
```

```

effect of nvlar3 on cc
controlled for age.cat sex
```

```
number of observations 1393
```

|                 | Effect | 2.5% | 97.5% |
|-----------------|--------|------|-------|
| [1,2) vs [0,1)  | 1.82   | 1.19 | 2.78  |
| [2,15) vs [0,1) | 3.58   | 1.78 | 7.21  |

Test for no effects of exposure on 2 df:  $p = 4.81e-05$

|                | exp(Est.) | 2.5%      | 97.5%     |
|----------------|-----------|-----------|-----------|
| (Intercept)    | 0.4882029 | 0.2801567 | 0.8507456 |
| nvlar3[1,2)    | 1.8182303 | 1.1908578 | 2.7761179 |
| nvlar3[2,15)   | 3.5840528 | 1.7809624 | 7.2126367 |
| age.cat[30,40) | 0.9020223 | 0.4907297 | 1.6580293 |
| age.cat[40,50) | 0.9172969 | 0.5151752 | 1.6332962 |
| age.cat[50,60) | 1.0666074 | 0.5953609 | 1.9108599 |
| age.cat[60,70) | 0.8927764 | 0.4975674 | 1.6018929 |
| age.cat[70,85) | 0.9985714 | 0.5410858 | 1.8428590 |
| sexF           | 1.0292748 | 0.8185439 | 1.2942577 |

Now fit a glm containing `age.cat`, `sex`, `nvsma4` and `nvlar3` and place the result in `m.nvboth`. What is the interpretation of the last two coefficients?

### 2.4.11 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 with

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

```

response : cc
type : "binary"
exposure : fscore
control vars : age.cat sex
```

```
fscore is numeric
effects are measured as odds ratios

```

```
effect of an increase of 1 unit in fscore on cc
controlled for age.cat sex
```

```
number of observations 1396
```

| Effect | 2.5% | 97.5% |
|--------|------|-------|
| 1.72   | 1.47 | 2.00  |

Test for no effects of exposure on 1 df:  $p = 6.2e-12$

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

Analysis of Deviance Table

| Model                                               | Resid. Df | Resid. Dev | Df | Deviance | P(> Chi ) |
|-----------------------------------------------------|-----------|------------|----|----------|-----------|
| Model 1: <code>cc ~ fscore + age.cat + sex</code>   | 1388      | 1738.62    |    |          |           |
| Model 2: <code>cc ~ freckles + age.cat + sex</code> | 1387      | 1737.10    | 1  | 1.52     | 0.22      |

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 = mm)
> ci.lin(m1, Exp = TRUE)[c(2, 3), c(5, 6, 7)]
 exp(Est.) 2.5% 97.5%
C(freckles, contr.cum)2 1.509928 1.169630 1.949234
C(freckles, contr.cum)3 2.034725 1.487967 2.782389

> m2 <- glm(cc ~ fscore + age.cat + sex, family = "binomial", data = mm)
> ci.lin(m2, Exp = TRUE)[2, c(5, 6, 7)]
exp(Est.) 2.5% 97.5%
1.715535 1.468647 2.003927

```

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`).

### 2.4.12 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:

```
> plotEst(exp(ci.lin(m.nvboth)[- (1:7), -(2:4)]), xlog = T, vref = 1)
```

The `xlog` argument makes the OR axis logarithmic.

### 2.4.13 Further questions

Investigate some of these questions:

1. Is there still an effect of freckles even for those with dark skins?
2. Is there any effect of eye colour? Is there still an effect after taking account of skin colour?
3. If the main focus of interest is the effect of freckles which variables would you control for? Fit the appropriate model and summarize your conclusions. Plot the coefficients with their confidence intervals.

## 2.5 Interval-censored data: Conversion to diabetes

### 2.5.1 R-program for the analysis

This

```
R 2.4.1

Program: int-cens.R
Folder: C:\Bendix\Undervis\SPE\pracs\r
Started: søndag 13. maj 2007, 16:32:11

> plt <- function(file, ...) pdf(paste(file, "pdf", sep="."), ...)
>
> library(Epi)
> library(help=Epi)
> source("c:/stat/r/bxc/library.sources/Epi/data/DMconv.R")
> source("c:/stat/r/bxc/library.sources/Epi/R/print.Icens.R")
> source("c:/stat/r/bxc/library.sources/Epi/R/summary.Icens.R")
>
> #-----
> # 1:
> # Load data and inspect them
>
> #data(DMconv)
> str(DMconv)
'data.frame': 1519 obs. of 6 variables:
 $ id : int 1 2 3 4 5 6 7 8 9 10 ...
 $ doe :Class 'Date' num [1:1519] 12215 11866 11568 11841 11288 ...
 $ dlw :Class 'Date' num [1:1519] 13525 13278 13058 13150 13011 ...
 $ dfi :Class 'Date' num [1:1519] NA NA NA NA NA NA NA NA NA ...
 $ gtol: Factor w/ 2 levels "IFG","IGT": 2 1 1 2 2 1 2 2 2 1 ...
 $ grp : Factor w/ 2 levels "Intervention",...: 2 1 1 1 1 2 2 1 2 1 ...
> head(DMconv, 20)
 id doe dlw dfi gtol grp
1 1 2003-06-12 2007-01-12 <NA> IGT Control
2 2 2002-06-28 2006-05-10 <NA> IFG Intervention
3 3 2001-09-03 2005-10-02 <NA> IFG Intervention
4 4 2002-06-03 2006-01-02 <NA> IGT Intervention
5 5 2000-11-27 2005-08-16 <NA> IGT Intervention
6 6 2001-08-31 2005-07-08 <NA> IFG Control
7 7 2001-09-19 2004-11-17 <NA> IGT Control
8 8 2001-07-03 2005-09-22 <NA> IGT Intervention
9 9 2001-08-08 2006-03-19 <NA> IGT Control
10 10 2002-01-12 2005-04-19 <NA> IFG Intervention
11 11 2001-08-15 2002-02-09 2002-08-20 IGT Control
12 12 2001-09-24 2002-01-14 2003-04-20 IFG Intervention
13 13 2001-04-30 2001-09-19 2003-11-10 IGT Control
14 14 2000-12-19 2004-12-20 <NA> IGT Intervention
15 15 2001-07-01 2001-12-18 2003-10-26 IFG Control
16 16 2001-09-23 2005-12-14 <NA> IFG Intervention
17 17 2003-01-19 2003-03-18 2006-08-17 IGT Intervention
18 18 2001-10-13 2002-01-11 2004-02-09 IGT Intervention
19 19 2002-11-28 2006-06-01 <NA> IFG Intervention
20 20 2002-08-11 2005-07-19 <NA> IGT Intervention
>
> #-----
> # 2:
> # Make a histogram over the examination times in years since entry
>
> par(mfrow=c(2,1))
> with(DMconv, hist(cal.yr(dfi)-cal.yr(doe)))
> with(DMconv, hist(cal.yr(dlw)-cal.yr(doe)))
>
> # With a few more bells and whistles
> plt("int-cens-1")
> par(mfrow=c(2,1))
> with(DMconv, hist(cal.yr(dfi)-cal.yr(doe),
+ xlim=c(0,5), col=gray(0.6), breaks=50, main="", xlab="Conversion"))
> with(DMconv, hist(cal.yr(dlw)-cal.yr(doe),
+ xlim=c(0,5), col=gray(0.6), breaks=50, main="", xlab="Well"))
> dev.off()
windows
 2
>
> #-----
> # 3:
> # Make a simple model without covariates,
> # estimating the conversion rates in intervals
>
> mA <-
+ Icens(first.well= cal.yr(doe)-cal.yr(doe), # Timescale is time since entry
+ last.well= cal.yr(dlw)-cal.yr(doe),
+ first.ill = cal.yr(dfi)-cal.yr(doe),
+ data = DMconv,
+ breaks = c(seq(0,6,2),10) # constant rate in intervals (0:6)
```

```

+)
>
> mB <-
+ Icens(first.well= cal.yr(doe)-cal.yr(doe), # Timescale is time since entry
+ last.well= cal.yr(dlw)-cal.yr(doe),
+ first.ill = cal.yr(df1)-cal.yr(doe),
+ data = DMconv,
+ breaks = c(seq(0,6,1.5),10) # constant rate in intervals (0:6)
+)
>
> mC <-
+ Icens(first.well= cal.yr(doe)-cal.yr(doe), # Timescale is time since entry
+ last.well= cal.yr(dlw)-cal.yr(doe),
+ first.ill = cal.yr(df1)-cal.yr(doe),
+ data = DMconv,
+ breaks = c(seq(0,6,1),10) # constant rate in half as wide intervals
+)
Warning message:
algorithm did not converge in: glm.fit(x = X, y = Y, weights = weights, start = start, etastart = etastart,
>
> #-----
> # 4:
> # Extract the estimates and plot them in the same frame.
>
> mAest <- summary(mA)
> mBest <- summary(mB)
> mCest <- summary(mC)
>
> plt("int-cens-2")
> plot(seq(0,6,2),mAest[,1],type="s",ylim=c(0,1.5),lwd=3)
> matlines(seq(0,6,2) ,mAest[,c(1,5,6)],type="s",lwd=c(3,1,1),col="black",lty=1)
> matlines(seq(0,6,1.5),mBest[,c(1,5,6)],type="s",lwd=c(3,1,1),lty="22",col="blue")
> matlines(seq(0,6,1) ,mCest[,c(1,5,6)],type="s",lwd=c(3,1,1),lty="22",col="red")
> dev.off()
windows
 2
>
> #-----
> # 5:
> # Choosing a baseline specification we fit a MRR model
>
> mrr <-
+ Icens(first.well= cal.yr(doe)-cal.yr(doe), # Timescale is time since entry
+ last.well= cal.yr(dlw)-cal.yr(doe),
+ first.ill = cal.yr(df1)-cal.yr(doe),
+ data = DMconv,
+ model = "MRR", # Multiplicative Rate Ratio ~ "Cox model"
+ formula = ~ gtol + grp,
+ breaks = c(0,1,2,6)
+)
> mrr

```

|            | Estimate | StdErr | z       | P     | RR     | 2.5%   | 97.5%  |
|------------|----------|--------|---------|-------|--------|--------|--------|
| ^(0,1)^    | 0.1272   | 0.0089 | 14.3132 | 0.000 | NA     | 0.1098 | 0.1447 |
| ^(1,2)^    | 0.0957   | 0.0088 | 10.8835 | 0.000 | NA     | 0.0785 | 0.1129 |
| ^(2,6)^    | 0.0784   | 0.0080 | 9.8285  | 0.000 | NA     | 0.0628 | 0.0940 |
| gtolIGT    | 0.3232   | 0.0698 | 4.6324  | 0.000 | 1.3815 | 1.2050 | 1.5839 |
| grpControl | 0.0841   | 0.0842 | 0.9985  | 0.318 | 1.0877 | 0.9222 | 1.2829 |

```

>
> #-----
> # 6:
> # Extract the parameters form the MRR model and plot the estimated rates
>
> plt("int-cens-3")
> RRest <- summary(mrr)
> tm <- c(0,1,2,6)
> plot(tm, RRest[c(1:3,3),1], ylim=c(0,0.3), type="s", lwd=2)
> lines(tm, RRest[c(1:3,3),1]*RRest[4,5], col="red" , type="s", lwd=2)
> lines(tm, RRest[c(1:3,3),1]*RRest[5,5], col="green", type="s", lwd=2)
> lines(tm, RRest[c(1:3,3),1]*RRest[4,5]*RRest[5,5], col="blue" , type="s", lwd=2)
> dev.off()
windows
 2
>
> #-----
> # 7:
> # Fit the excess risk model, extract the parameters plot the estimated rates
>
> maer <-
+ Icens(first.well= cal.yr(doe)-cal.yr(doe), # Timescale is time since entry
+ last.well= cal.yr(dlw)-cal.yr(doe),
+ first.ill = cal.yr(df1)-cal.yr(doe),
+ data = DMconv,
+ model = "AER", # Additive excess risk model
+ formula = ~ gtol + grp,
+ breaks = c(0,1,2,6)
+)
> maer

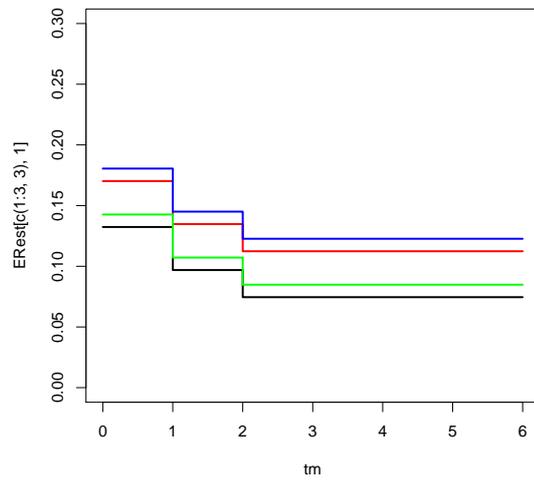
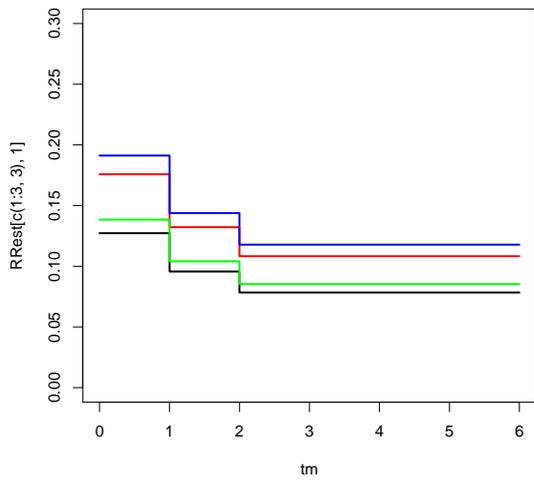
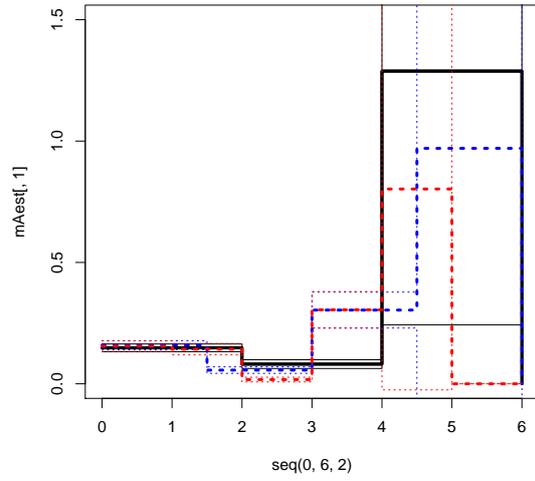
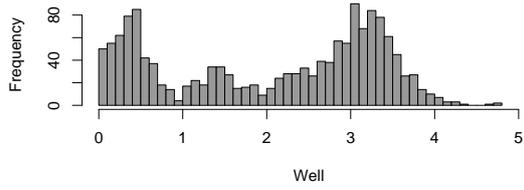
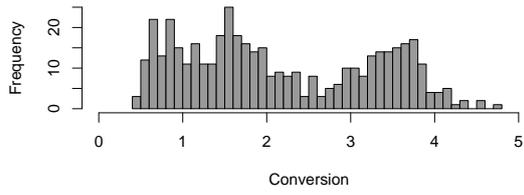
```

```

 Estimate StdErr z P 2.5% 97.5%
` (0,1)` 0.1324 0.0135 9.7786 0.0000 0.1058 0.1589
` (1,2)` 0.0969 0.0132 7.3222 0.0000 0.0710 0.1228
` (2,6)` 0.0745 0.0121 6.1854 0.0000 0.0509 0.0982
gtolIGT 0.0379 0.0118 3.2084 0.0013 0.0147 0.0610
grpControl 0.0103 0.0120 0.8544 0.3929 -0.0133 0.0339
>
> plt("int-cens-4")
> ERest <- summary(maer)
> tm <- c(0,1,2,6)
> plot(tm, ERest[c(1:3,3),1], ylim=c(0,0.3), type="s", lwd=2)
> lines(tm, ERest[c(1:3,3),1]+ERest[4,1], col="red" , type="s", lwd=2)
> lines(tm, ERest[c(1:3,3),1]+ERest[5,1], col="green", type="s", lwd=2)
> lines(tm, ERest[c(1:3,3),1]+ERest[4,1]+ERest[5,1], col="blue" , type="s", lwd=2)
> dev.off()
windows
 2
>
> #-----
> # 8:
> # Fit the excess risk model using bottstrap sampling
>
> system.time(
+ mer <-
+ Icens(first.well= cal.yr(doe)-cal.yr(doe), # Timescale is time since entry
+ last.well= cal.yr(dlw)-cal.yr(doe),
+ first.ill = cal.yr(dfi)-cal.yr(doe),
+ data = DMconv,
+ model = "MRR",
+ formula = ~ gtol,
+ breaks = c(0,1,2,6),
+ boot = 10
+))
[1] 54.55 0.04 57.98 NA NA
> mer
 Estimate StdErr z P RR 2.5% 97.5% Boot-med lo hi
` (0,1)` 0.1316 0.0092 14.3105 0 NA 0.1136 0.1497 0.1397 0.1118 0.1634
` (1,2)` 0.0992 0.0091 10.8929 0 NA 0.0814 0.1171 0.1018 0.0725 0.1144
` (2,6)` 0.0811 0.0082 9.8253 0 NA 0.0649 0.0972 0.0846 0.0688 0.1060
model.matrix.formula..data....1. 0.3266 0.0581 5.6245 0 1.3862 1.2371 1.5532 0.2654 0.1886 0.4438
>

Program: int-cens.R
Folder: C:\Bendix\Undervis\SPE\pracs\r
Ended: s ndag 13. maj 2007, 16:33:26
Elapsed: 00:01:15

```



## 2.6 Time-splitting and SMR: The Thorotrast study

### 2.6.1 R-program for the analysis

This

```
R 2.5.0

Program: thoro.R
Folder: C:\Bendix\Undervis\SPE\pracs\r
Started: mandag 28. maj 2007, 13:46:24

> # All graphs are output to a device called by the function "plt"
> # Choose the appropriate definition of plt here by uncommenting:
> #
> plt <- function(file, ...) pdf(paste(file,"pdf",sep="."), ...)
> # plt <- function(file, ...) win.metafile(paste(file,"emf",sep="."), ...)
> # plt <- function(file, ...) postscript(paste(file,"eps",sep="."), ...)
> # plt <- function(file, ...) X11() # Output to the screen
>
> # Load the relevant libraries
> library(Epi)
> library(splines)
>
> #-----
> # 1:
> # Get the data and look at it
> #
> data(thoro)
> str(thoro)
'data.frame': 2470 obs. of 14 variables:
 $ id : num 1 2 3 4 5 6 7 8 9 10 ...
 $ sex : num 2 2 1 1 1 2 1 2 1 1 ...
 $ birthdat:Class 'Date' num [1:2470] -19501 -15398 -24553 -18862 -24497 ...
 $ contrast: num 1 1 1 1 1 1 1 1 1 1 ...
 $ injecdat:Class 'Date' num [1:2470] -11399 -9531 -12554 -12274 -11912 ...
 $ volume : num 22 80 10 10 10 20 10 40 34 10 ...
 $ exitdat :Class 'Date' num [1:2470] 2479 -1450 -3755 2669 -8990 ...
 $ exitstat: num 1 1 1 1 1 1 1 3 1 1 ...
 $ cause : num 2 8 2 2 14 14 3 NA 2 2 ...
 $ liverdat:Class 'Date' num [1:2470] -1450 -1450 NA 2669 NA ...
 $ liver : num 1 1 0 1 0 0 0 0 0 1 0 ...
 $ hepcc : num 0 0 0 0 0 0 0 0 0 0 ...
 $ chola : num 0 0 0 0 0 0 0 0 0 0 ...
 $ hmang : num 1 1 0 1 0 0 0 0 1 0 ...
> head(thoro)
 id sex birthdat contrast injecdat volume exitdat exitstat cause liverdat liver hepcc chola hmang
1 1 2 1916-08-11 1 1938-10-17 22 1976-10-15 1 2 1966-01-12 1 0 0 1
2 2 2 1927-11-05 1 1943-11-28 80 1966-01-12 1 8 1966-01-12 1 0 0 1
3 3 1 1902-10-12 1 1935-08-19 10 1959-09-21 1 2 <NA> 0 0 0 0
4 4 1 1918-05-12 1 1936-05-25 10 1977-04-23 1 2 1977-04-23 1 0 0 1
5 5 1 1902-12-07 1 1937-05-22 10 1945-05-22 1 14 <NA> 0 0 0 0
6 6 2 1903-09-19 1 1937-04-26 20 1944-09-27 1 14 <NA> 0 0 0 0
>
> # Explore the functionality of cal.yr:
> bd <- thoro$birthdat[1:5]
> bd
[1] "1916-08-11" "1927-11-05" "1902-10-12" "1918-05-12" "1902-12-07"
> as.numeric(bd)
[1] -19501 -15398 -24553 -18862 -24497
> cal.yr(bd)
[1] 1916.609 1927.843 1902.778 1918.359 1902.931
attr(,"class")
[1] "cal.yr" "numeric"
> (cal.yr(bd)-1970)*365.25
[1] -19501 -15398 -24553 -18862 -24497
attr(,"class")
[1] "cal.yr" "numeric"
>
> #-----
> # x:
> # Draw a Lexis diagram for 10% of the data
> #
> Lexis.diagram(age=c(0,100), date=c(1935,1995))
> Lexis.lines(birth.date=birthdat, entry.date=injecdat, exit.date=exitdat,
+ col.life=c("red","blue")[contrast], lwd.life=1,
+ fail=!is.na(cause), col.fail=c("red","blue")[contrast],
+ pch.fail=c(1,16),
+ data=thoro[runif(nrow(thoro))<0.1,])
> dev.off()
null device
 1
>
> #-----
> # 2:
> # Define a Lexis object with the follow-up in the cohort using timescales
```

```

> # calendar time (per), age (age) and time from contrast injection (tfi):
> # Define three time-scales for the follow-up
> # Starting points for all timescales and end point for one of them.
> #
> thL <- Lexis(entry = list("per"=cal.yr(injecdat),
+ "age"=cal.yr(injecdat) - cal.yr(birthdat),
+ "tfi"=0),
+ exit = list("per"=cal.yr(exitdat)),
+ entry.status = 0,
+ exit.status = (exitstat==1),
+ id = id,
+ data = thoro)
> str(thL)
Classes 'Lexis' and 'data.frame': 2470 obs. of 21 variables:
 $ per :Classes 'cal.yr', 'numeric' num [1:2470] 1939 1944 1936 1936 1937 ...
 $ age :Classes 'cal.yr', 'numeric' num [1:2470] 22.2 16.1 32.9 18.0 34.5 ...
 $ tfi : num 0 0 0 0 0 0 0 0 0 0 ...
 $ lex.deltat:Classes 'cal.yr', 'numeric' num [1:2470] 38.0 22.1 24.1 40.9 8.0 ...
 $ lex.status1: num 0 0 0 0 0 0 0 0 0 0 ...
 $ lex.status2: logi TRUE TRUE TRUE TRUE TRUE TRUE ...
 $ lex.id : num 1 2 3 4 5 6 7 8 9 10 ...
 $ id : num 1 2 3 4 5 6 7 8 9 10 ...
 $ sex : num 2 2 1 1 1 2 1 2 1 1 ...
 $ birthdat :Class 'Date' num [1:2470] -19501 -15398 -24553 -18862 -24497 ...
 $ contrast : num 1 1 1 1 1 1 1 1 1 1 ...
 $ injecdat :Class 'Date' num [1:2470] -11399 -9531 -12554 -12274 -11912 ...
 $ volume : num 22 80 10 10 10 20 10 40 34 10 ...
 $ exitdat :Class 'Date' num [1:2470] 2479 -1450 -3755 2669 -8990 ...
 $ exitstat : num 1 1 1 1 1 1 1 3 1 1 ...
 $ cause : num 2 8 2 2 14 14 3 NA 2 2 ...
 $ liverdat :Class 'Date' num [1:2470] -1450 -1450 NA 2669 NA ...
 $ liver : num 1 1 0 1 0 0 0 0 1 0 ...
 $ hepcc : num 0 0 0 0 0 0 0 0 0 0 ...
 $ chola : num 0 0 0 0 0 0 0 0 0 0 ...
 $ hmgang : num 1 1 0 1 0 0 0 0 1 0 ...
- attr(*, "time.scales")= chr "per" "age" "tfi"
- attr(*, "breaks")=List of 3
..$ per: NULL
..$ age: NULL
..$ tfi: NULL
> head(thL)
 per age tfi lex.deltat lex.status1 lex.status2 lex.id id sex birthdat contrast injecdat
1 1938.791 22.18207 0 37.995893 0 TRUE 1 1 2 1916-08-11 1 1938-10-17
2 1943.906 16.06297 0 22.124572 0 TRUE 2 2 2 1927-11-05 1 1943-11-28
3 1935.629 32.85147 0 24.090349 0 TRUE 3 3 1 1902-10-12 1 1935-08-19
4 1936.396 18.03696 0 40.911704 0 TRUE 4 4 1 1918-05-12 1 1936-05-25
5 1937.387 34.45585 0 8.000000 0 TRUE 5 5 1 1902-12-07 1 1937-05-22
6 1937.316 33.60164 0 7.422313 0 TRUE 6 6 2 1903-09-19 1 1937-04-26
 volume exitdat exitstat cause liverdat liver hepcc chola hmgang
1 22 1976-10-15 1 2 1966-01-12 1 0 0 1
2 80 1966-01-12 1 8 1966-01-12 1 0 0 1
3 10 1959-09-21 1 2 <NA> 0 0 0 0
4 10 1977-04-23 1 2 1977-04-23 1 0 0 1
5 10 1945-05-22 1 14 <NA> 0 0 0 0
6 20 1944-09-27 1 14 <NA> 0 0 0 0
>
> #-----
> # 3:
> # Plot a Lexis diagram using the default method for Lexis objects.
> # Life lines are dark grey by default to avoid producing a solid block
> # of black.
> #
> plt("thoro-0")
> plot(thL)
>
> # Add some colored points for the different exits.
> points(subset(thL,exitstat==1), col="blue")
> points(subset(thL,exitstat==2), col="green")
> points(subset(thL,exitstat==3), col="red")
> dev.off()
null device
1
>
> # Lexis diagram with more detail
> plt("thoro-1")
> thL.sub <- subset(thL, runif(nrow(thL)) < 0.1)
> plot(thL.sub, col=c("red","blue")[thL.sub$contrast],
+ xaxs="i", xlim=c(1920,2020),
+ yaxs="i", ylim=c(0,100), las=1,
+ xlab="Date of follow-up", ylab="Age",
+ grid=0:20*5, lty.grid="14")
> points(thL.sub, col=c("red","blue")[thL.sub$contrast],
+ pch=c(16,NA,NA)[thL.sub$exitstat])
> dev.off()
null device
1
>

```

```

> ## If you only want the failures and not the lines in color by group
> plt("thoro-2")
> plot(thL.sub,
+ xaxs="i", xlim=c(1920,2020),
+ yaxs="i", ylim=c(0,100),
+ las=1, grid=seq(0,100,5), lty.grid=1, col=gray(0.4))
> points(thL.sub, col=c("red","blue")[thL.sub$contrast],
+ pch=c(16,NA,1)[thL.sub$exitstat], lwd=2)
> # or:
> with(thL.sub,
+ points(thL.sub, col=c("red","blue")[contrast],
+ pch=c(NA,16)[(exitstat==1)+1])
+)
> # or:
> points(subset(thL.sub, exitstat==1 & contrast==1), col="red", pch=16)
> points(subset(thL.sub, exitstat==1 & contrast==2), col="blue", pch=16)
> dev.off()
null device
1
>
> #-----
> # 4:
> # Definition and tabulation of deaths and person-years from original dataset
> #
> thoro$entrydat <- thoro$injecdat + 366
> thoro$Y <- pmax(0, cal.yr(thoro$exitdat)-cal.yr(thoro$entrydat))
> thoro$D <- as.numeric(thoro$exitstat==1 & thoro$Y > 0)
> # Remove persons without follow-up after one year.
> thoro <- thoro[thoro$Y>0,]
> str(thoro)
'data.frame': 2042 obs. of 17 variables:
 $ id : num 1 2 3 4 5 6 7 9 10 11 ...
 $ sex : num 2 2 1 1 1 2 1 1 1 1 ...
 $ birthdat:Class 'Date' num [1:2042] -19501 -15398 -24553 -18862 -24497 ...
 $ contrast: num 1 1 1 1 1 1 1 1 1 1 ...
 $ injecdat:Class 'Date' num [1:2042] -11399 -9531 -12554 -12274 -11912 ...
 $ volume : num 22 80 10 10 10 20 10 34 10 10 ...
 $ exitdat :Class 'Date' num [1:2042] 2479 -1450 -3755 2669 -8990 ...
 $ exitstat: num 1 1 1 1 1 1 1 1 1 1 ...
 $ cause : num 2 8 2 2 14 14 3 2 2 2 ...
 $ liverdat:Class 'Date' num [1:2042] -1450 -1450 NA 2669 NA ...
 $ liver : num 1 1 0 1 0 0 0 1 0 0 ...
 $ hepcc : num 0 0 0 0 0 0 0 0 0 0 ...
 $ chola : num 0 0 0 0 0 0 0 0 0 0 ...
 $ hmang : num 1 1 0 1 0 0 0 1 0 0 ...
 $ entrydat:Class 'Date' num [1:2042] -11033 -9165 -12188 -11908 -11546 ...
 $ Y : num 37.0 21.1 23.1 39.9 7.0 ...
 $ D : num 1 1 1 1 1 1 1 1 1 1 ...
> st <-
+ stat.table(contrast,
+ list(D=sum(D), Y=sum(Y), Rate=ratio(D, Y/1000)),
+ margin=TRUE, data=thoro)
> st

contrast D Y Rate

1 748.00 19242.20 38.87
2 796.00 30515.03 26.09

Total 1544.00 49757.22 31.03

>
> #-----
> # 5:
> # This could also be completed using a Lexis version
> # --- remember the reference date is now one year after injection
> #
> thL <- Lexis(entry = list("per"=cal.yr(entrydat),
+ "age"=cal.yr(entrydat) - cal.yr(birthdat),
+ "tfi"=1),
+ exit = list("per"=cal.yr(exitdat)),
+ entry.status = 0,
+ exit.status = (exitstat==1),
+ id = id,
+ data = thoro)
> stat.table(contrast,
+ list(D = sum(lex.status2),
+ Y = sum(lex.deltat),
+ Rate = ratio(lex.status2, lex.deltat/1000)),
+ margin=TRUE, data=thL)

contrast D Y Rate

1 748.00 19242.20 38.87
2 796.00 30515.03 26.09

Total 1544.00 49757.22 31.03

```

```

>
> #-----
> # 6:
> # Overall rates and Rate-Ratio with c.i.
> # First use tabulated data from stat.table
> #
> RR <- st[3,1]/st[3,2]
> erf <- exp(1.96*sqrt(1/st[1,1]+1/st[1,2]))
> c(RR,RR/erf,RR*erf)
[1] 1.490211 1.348655 1.646624
>
> # Readable form:
> round(c(RR,RR/erf,RR*erf),3)
[1] 1.490 1.349 1.647
>
> # A simple GLM to do the same task
> m0 <- glm(D ~ factor(contrast) + offset(log(Y/1000)),
+ family=poisson, data=thoro)
> # summary(m0)
> round(ci.lin(m0, Exp=TRUE), 3)
 Estimate StdErr z P exp(Est.) 2.5% 97.5%
(Intercept) 3.660 0.037 100.108 0 38.873 36.185 41.761
factor(contrast)2 -0.399 0.051 -7.834 0 0.671 0.607 0.741
> round(1/ci.lin(m0, Exp=TRUE)[2,5:7,drop=FALSE], 3)
 exp(Est.) 2.5% 97.5%
factor(contrast)2 1.49 1.647 1.349
> # Use a contrast matrix to compute the desired parameter functions
> # Each row in CM is multiplied onto the parameters of m0
> CM <- rbind(c(1,0),c(1,1),c(0,-1),c(0,1))
> rownames(CM) <- c("Thoro","Control","RR","1/RR")
> CM
 [,1] [,2]
Thoro 1 0
Control 1 1
RR 0 -1
1/RR 0 1
> round(ci.lin(m0, ctr.mat=CM, Exp=TRUE)[5:7],3)
 exp(Est.) 2.5% 97.5%
Thoro 38.873 36.185 41.761
Control 26.086 24.335 27.962
RR 1.490 1.349 1.647
1/RR 0.671 0.607 0.741
>
> #-----
> # 7:
> # Remove follow-up after liver cancer
> #
> with(thoro, table(contrast, liver))
 contrast 0 1
 1 677 130
 2 1235 0
> thoro$exitdat <- pmin(thoro$exitdat, thoro$liverdat, na.rm=TRUE)
> thoro$exitstat[thoro$exitdat==thoro$liverdat] <- 0
>
> #-----
> # 8:
> # Renew the definition of thL using the smaller version of thoro
> #
> thL <- Lexis(entry = list("per"=cal.yr(injecdat),
+ "age"=cal.yr(injecdat) - cal.yr(birthdat),
+ "tfi"=0),
+ exit = list("per"=cal.yr(exitdat)),
+ entry.status = 0,
+ exit.status = (exitstat==1),
+ id = id,
+ data = thoro)
>
> # Split in 5-year intervals along age and period axes
> thx <- splitLexis(thL, breaks=list(tfi=c(0:4,seq(5,55,5)),
+ age=seq(0,100,5),
+ per=seq(1900,2000,5)))
> str(thx)
Classes 'Lexis' and 'data.frame': 39879 obs. of 24 variables:
 $ lex.id : num 1 1 1 1 1 1 1 1 1 ...
 $ per : num 1939 1940 1940 1941 1942 ...
 $ age : num 22.2 23.2 23.4 24.2 25.0 ...
 $ tfi : num 0.00 1.00 1.21 2.00 2.82 ...
 $ lex.deltat : num 1.000 0.209 0.791 0.818 0.182 ...
 $ lex.status1: num 0 0 0 0 0 0 0 0 0 ...
 $ lex.status2: num 0 0 0 0 0 0 0 0 0 ...
 $ id : num 1 1 1 1 1 1 1 1 1 ...
 $ sex : num 2 2 2 2 2 2 2 2 2 ...
 $ birthdat : Class 'Date' num [1:39879] -19501 -19501 -19501 -19501 -19501 ...
 $ contrast : num 1 1 1 1 1 1 1 1 1 ...
 $ injecdat : Class 'Date' num [1:39879] -11399 -11399 -11399 -11399 -11399 ...

```

```

$ volume : num 22 22 22 22 22 22 22 22 22 22 ...
$ exitdat :Class 'Date' num [1:39879] -1450 -1450 -1450 -1450 -1450 -1450 -1450 -1450 -1450 ...
$ exitstat : num 0 0 0 0 0 0 0 0 0 0 ...
$ cause : num 2 2 2 2 2 2 2 2 2 2 ...
$ liverdat :Class 'Date' num [1:39879] -1450 -1450 -1450 -1450 -1450 -1450 -1450 -1450 -1450 ...
$ liver : num 1 1 1 1 1 1 1 1 1 1 ...
$ hepcc : num 0 0 0 0 0 0 0 0 0 0 ...
$ cholera : num 0 0 0 0 0 0 0 0 0 0 ...
$ hmang : num 1 1 1 1 1 1 1 1 1 1 ...
$ entrydat :Class 'Date' num [1:39879] -11033 -11033 -11033 -11033 -11033 ...
$ Y : num 37 37 37 37 37 ...
$ D : num 1 1 1 1 1 1 1 1 1 1 ...
- attr(*, "breaks")=List of 3
..$ per: num 1900 1905 1910 1915 1920 ...
..$ age: num 0 5 10 15 20 25 30 35 40 45 ...
..$ tfi: num 0 1 2 3 4 5 10 15 20 25 ...
- attr(*, "time.scales")= chr "per" "age" "tfi"
> subset(thx, id==1)[,1:10]
 lex.id per age tfi lex.deltat lex.status1 lex.status2 id sex birthdat
1 1 1938.791 22.18207 0.000000 1.0000000 0 0 1 2 1916-08-11
2 1 1939.791 23.18207 1.000000 0.2087611 0 0 1 2 1916-08-11
3 1 1940.000 23.39083 1.208761 0.7912389 0 0 1 2 1916-08-11
4 1 1940.791 24.18207 2.000000 0.8179329 0 0 1 2 1916-08-11
5 1 1941.609 25.00000 2.817933 0.1820671 0 0 1 2 1916-08-11
6 1 1941.791 25.18207 3.000000 1.0000000 0 0 1 2 1916-08-11
7 1 1942.791 26.18207 4.000000 1.0000000 0 0 1 2 1916-08-11
8 1 1943.791 27.18207 5.000000 1.2087611 0 0 1 2 1916-08-11
9 1 1945.000 28.39083 6.208761 1.6091718 0 0 1 2 1916-08-11
10 1 1946.609 30.00000 7.817933 2.1820671 0 0 1 2 1916-08-11
11 1 1948.791 32.18207 10.000000 1.2087611 0 0 1 2 1916-08-11
12 1 1950.000 33.39083 11.208761 1.6091718 0 0 1 2 1916-08-11
13 1 1951.609 35.00000 12.817933 2.1820671 0 0 1 2 1916-08-11
14 1 1953.791 37.18207 15.000000 1.2087611 0 0 1 2 1916-08-11
15 1 1955.000 38.39083 16.208761 1.6091718 0 0 1 2 1916-08-11
16 1 1956.609 40.00000 17.817933 2.1820671 0 0 1 2 1916-08-11
17 1 1958.791 42.18207 20.000000 1.2087611 0 0 1 2 1916-08-11
18 1 1960.000 43.39083 21.208761 1.6091718 0 0 1 2 1916-08-11
19 1 1961.609 45.00000 22.817933 2.1820671 0 0 1 2 1916-08-11
20 1 1963.791 47.18207 25.000000 1.2087611 0 0 1 2 1916-08-11
21 1 1965.000 48.39083 26.208761 1.0301164 0 0 1 2 1916-08-11
>
> # Compare the two datasets:
> with(thL, table(lex.status2>0, contrast))
 contrast
 1 2
FALSE 188 439
TRUE 619 796
> with(thx, table(lex.status2>0, contrast))
 contrast
 1 2
FALSE 14857 23607
TRUE 619 796
>
> ## Show breaks in split Lexis object
> plt("thoro-3")
> plot(thx, "tfi")
> dev.off()
null device
1
>
> # Tabulation of follow-up time and events:
> stat.table(list(contrast=thx$contrast),
+ list(D = sum(status(thx) > 0),
+ Y = sum(deltat(thx)),
+ Rate = ratio(status(thx) > 0, deltat(thx)/1000)),
+ margin = TRUE,
+ data = thx)

contrast D Y Rate

1 619.00 20031.33 30.90
2 796.00 31752.56 25.07

Total 1415.00 51783.89 27.33

>
> # Direct tabulation from original dataset
> stat.table(list(contrast,
+ list(D=sum(D), Y=sum(Y), Rate=ratio(D, Y/1000)),
+ margin=TRUE, data=thoro)

contrast D Y Rate

1 748.00 19242.20 38.87
2 796.00 30515.03 26.09

Total 1544.00 49757.22 31.03

```

```

>
> #-----
> # 9:
> #
> # Compute the interval midpoints and use these for tabulation
> thx$m.tfi <- timeBand(thx, "tfi", "middle")
> stx <-
+ stat.table(list(m.tfi, contrast),
+ list(D=sum(status(thx) > 0),
+ Y=sum(deltat(thx)),
+ Rate=ratio(status(thx) > 0, deltat(thx)/1000)),
+ margin=TRUE, data=thx)
> stx
-----contrast-----
m.tfi 1 2 Total

0.5 0.00 0.00 0.00
 807.00 1235.00 2042.00
 0.00 0.00 0.00

1.5 35.00 52.00 87.00
 785.50 1205.46 1990.96
 44.56 43.14 43.70

2.5 19.00 35.00 54.00
 761.34 1160.03 1921.37
 24.96 30.17 28.10

3.5 14.00 28.00 42.00
 744.52 1125.11 1869.62
 18.80 24.89 22.46

4.5 17.00 26.00 43.00
 728.84 1101.78 1830.62
 23.32 23.60 23.49

7.5 69.00 92.00 161.00
 3423.77 5216.21 8639.99
 20.15 17.64 18.63

12.5 70.00 94.00 164.00
 3040.81 4705.23 7746.03
 23.02 19.98 21.17

17.5 68.00 89.00 157.00
 2707.53 4260.68 6968.20
 25.12 20.89 22.53

22.5 73.00 101.00 174.00
 2305.88 3777.01 6082.89
 31.66 26.74 28.60

27.5 79.00 94.00 173.00
 1837.09 3271.58 5108.67
 43.00 28.73 33.86

32.5 66.00 91.00 157.00
 1345.66 2688.94 4034.60
 49.05 33.84 38.91

37.5 51.00 74.00 125.00
 855.55 1562.24 2417.80
 59.61 47.37 51.70

42.5 39.00 20.00 59.00
 499.82 442.26 942.09
 78.03 45.22 62.63

47.5 17.00 0.00 17.00
 168.56 1.04 169.59
 100.86 0.00 100.24

52.5 2.00 NA 2.00
 19.44 NA 19.44
 102.86 NA 102.86

Total 619.00 796.00 1415.00
 20031.33 31752.56 51783.89
 30.90 25.07 27.33

> # More compact print, annotation slightly better, but crap formatting of numbers
> round(ftable(stx, row.vars=2), 1)
 contents D Y Rate
contrast 1 2 Total 1 2 Total 1 2 Total

```

```

m.tfi
0.5 0.0 0.0 0.0 807.0 1235.0 2042.0 0.0 0.0 0.0
1.5 35.0 52.0 87.0 785.5 1205.5 1991.0 44.6 43.1 43.7
2.5 19.0 35.0 54.0 761.3 1160.0 1921.4 25.0 30.2 28.1
3.5 14.0 28.0 42.0 744.5 1125.1 1869.6 18.8 24.9 22.5
4.5 17.0 26.0 43.0 728.8 1101.8 1830.6 23.3 23.6 23.5
7.5 69.0 92.0 161.0 3423.8 5216.2 8640.0 20.2 17.6 18.6
12.5 70.0 94.0 164.0 3040.8 4705.2 7746.0 23.0 20.0 21.2
17.5 68.0 89.0 157.0 2707.5 4260.7 6968.2 25.1 20.9 22.5
22.5 73.0 101.0 174.0 2305.9 3777.0 6082.9 31.7 26.7 28.6
27.5 79.0 94.0 173.0 1837.1 3271.6 5108.7 43.0 28.7 33.9
32.5 66.0 91.0 157.0 1345.7 2688.9 4034.6 49.0 33.8 38.9
37.5 51.0 74.0 125.0 855.6 1562.2 2417.8 59.6 47.4 51.7
42.5 39.0 20.0 59.0 499.8 442.3 942.1 78.0 45.2 62.6
47.5 17.0 0.0 17.0 168.6 1.0 169.6 100.9 0.0 100.2
52.5 2.0 NA 2.0 19.4 NA 19.4 102.9 NA 102.9
Total 619.0 796.0 1415.0 20031.3 31752.6 51783.9 30.9 25.1 27.3
>
> #-----
> # 10:
> # Show the two sets of rates in graph
> #
> # What does the stat.table object look like?
> dimnames(stx)
$contents
 D Y Rate
 "D" "Y" "Rate"

$m.tfi
[1] "0.5" "1.5" "2.5" "3.5" "4.5" "7.5" "12.5" "17.5" "22.5" "27.5" "32.5" "37.5" "42.5"
[14] "47.5" "52.5" "Total"

$contrast
[1] "1" "2" "Total"

> # The relevant 2-column matrix:
> stx[3,1:13,-3]
 contrast
m.tfi 1 2
0.5 0.00000 0.00000
1.5 44.55739 43.13716
2.5 24.95584 30.17168
3.5 18.80407 24.88656
4.5 23.32465 23.59824
7.5 20.15322 17.63731
12.5 23.02019 19.97779
17.5 25.11518 20.88871
22.5 31.65819 26.74071
27.5 43.00282 28.73227
32.5 49.04647 33.84231
37.5 59.61047 47.36785
42.5 78.02734 45.22212
> # Use matplotlib to make nice a graph with correct x-axis
> plt("thoro-4")
> matplotlib(as.numeric(dimnames(stx)[[2]][1:13]),
+ stx[3,1:13,-3], log="y",
+ type="o", lwd=4, lty=1, pch=16,
+ xlab="Time since angiography", ylab="Rate per 1000 P.Y.")
Warning messages:
1: 2 y values <= 0 omitted from logarithmic plot in: xy.coords(x, y, xlabel, ylabel, log = log)
2: 1 y value <= 0 omitted from logarithmic plot in: xy.coords(x, y, xlabel, ylabel, log)
> dev.off()
null device
1
>
> #-----
> # 11:
> # Split data finely and compute mitpoints and exit status
> #
> thxx <- splitLexis(thL, breaks=list("tfi"=c(0,seq(1,100,0.5))))
> dim(thxx)
[1] 102538 24
> # plot(thxx, "tfi")
> table(thxx$tfi)

 0 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6 6.5 7 7.5 8 8.5 9 9.5 10 10.5 11
2042 2042 1988 1952 1920 1894 1866 1851 1831 1807 1792 1772 1758 1751 1734 1714 1698 1680 1659 1639 1618 1594
11.5 12 12.5 13 13.5 14 14.5 15 15.5 16 16.5 17 17.5 18 18.5 19 19.5 20 20.5 21 21.5 22
1578 1563 1554 1535 1513 1497 1482 1470 1458 1444 1428 1413 1396 1379 1362 1340 1326 1309 1290 1271 1258 1237
22.5 23 23.5 24 24.5 25 25.5 26 26.5 27 27.5 28 28.5 29 29.5 30 30.5 31 31.5 32 32.5 33
1213 1197 1177 1163 1144 1123 1106 1080 1062 1041 1020 1004 985 960 936 920 897 886 869 852 829 793
33.5 34 34.5 35 35.5 36 36.5 37 37.5 38 38.5 39 39.5 40 40.5 41 41.5 42 42.5 43 43.5 44
745 724 686 651 612 587 552 514 483 446 418 387 350 320 290 264 242 213 185 153 134 113
44.5 45 45.5 46 46.5 47 47.5 48 48.5 49 49.5 50 50.5 51 51.5 52 52.5 53 53.5
93 76 57 46 39 36 33 26 23 19 15 12 8 6 5 4 4 3 2
>
> ## Note the functions used to extract the relevant quantities

```

```

> ## Get time band
> thxx$m.tfi <- timeBand(thxx, "tfi", "middle")
> ## Set failure variable
> thxx$fail <- (status(thxx) > 0)
>
> #-----
> # 12:
> # Modelling by a smooth function - natural splines
> #
> library(splines)
> # knots, boundary knots and timepoints for prediction
> kn <- c(4,8,seq(10,40,10))
> bk <- c(1,50)
>
> ## The model with natural splines and interaction
> ## Note that since there is no intercept and no main effect of
> ## contrast in the model, the intercept must be included in the
> ## design matrix for the natural splines in ns():
> m1 <- glm(fail ~ -1 + factor(contrast):ns(m.tfi, knots=kn, Bo=bk, i=T) +
+ offset(log(lex.deltat/1000)),
+ family=poisson, data=thxx)
> summary(m1)

Call:
glm(formula = fail ~ -1 + factor(contrast):ns(m.tfi, knots = kn,
 Bo = bk, i = T) + offset(log(lex.deltat/1000)), family = poisson,
 data = thxx)

Deviance Residuals:
 Min 1Q Median 3Q Max
-0.3523 -0.1710 -0.1557 -0.1453 4.6605

Coefficients:
 Estimate Std. Error z value Pr(>|z|)
factor(contrast)1:ns(m.tfi, knots = kn, Bo = bk, i = T)1 2.9121 0.2606 11.172 < 2e-16
factor(contrast)2:ns(m.tfi, knots = kn, Bo = bk, i = T)1 3.0598 0.2086 14.669 < 2e-16
factor(contrast)1:ns(m.tfi, knots = kn, Bo = bk, i = T)2 2.9051 0.2290 12.687 < 2e-16
factor(contrast)2:ns(m.tfi, knots = kn, Bo = bk, i = T)2 2.6665 0.1992 13.386 < 2e-16
factor(contrast)1:ns(m.tfi, knots = kn, Bo = bk, i = T)3 3.1593 0.2327 13.578 < 2e-16
factor(contrast)2:ns(m.tfi, knots = kn, Bo = bk, i = T)3 3.0397 0.2044 14.869 < 2e-16
factor(contrast)1:ns(m.tfi, knots = kn, Bo = bk, i = T)4 3.2602 0.2393 13.622 < 2e-16
factor(contrast)2:ns(m.tfi, knots = kn, Bo = bk, i = T)4 3.2074 0.2228 14.393 < 2e-16
factor(contrast)1:ns(m.tfi, knots = kn, Bo = bk, i = T)5 3.8755 0.2103 18.428 < 2e-16
factor(contrast)2:ns(m.tfi, knots = kn, Bo = bk, i = T)5 3.2949 0.2237 14.728 < 2e-16
factor(contrast)1:ns(m.tfi, knots = kn, Bo = bk, i = T)6 3.0062 0.2391 12.574 < 2e-16
factor(contrast)2:ns(m.tfi, knots = kn, Bo = bk, i = T)6 2.9544 0.3462 8.533 < 2e-16
factor(contrast)1:ns(m.tfi, knots = kn, Bo = bk, i = T)7 8.1812 0.2336 35.022 < 2e-16
factor(contrast)2:ns(m.tfi, knots = kn, Bo = bk, i = T)7 7.2908 0.7093 10.279 < 2e-16
factor(contrast)1:ns(m.tfi, knots = kn, Bo = bk, i = T)8 2.2373 0.3197 6.998 2.6e-12
factor(contrast)2:ns(m.tfi, knots = kn, Bo = bk, i = T)8 0.7053 1.1803 0.598 0.55

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 21684 on 102538 degrees of freedom
Residual deviance: 14852 on 102522 degrees of freedom
AIC: 17714

Number of Fisher Scoring iterations: 7

>
> #-----
> # 13:
> #
> # A set of time-points and a contrast matrix for the rates at these times
> tpt <- seq(1,50,0.2)
> # Note that the knots AND boundary knots must be the same as in the model
> CM <- ns(tpt, knots=kn, Bo=bk, intercept=TRUE)
>
> # Briefly explore how ci.lin works:
> # ci.lin(m1)[,1,drop=F]
> # ci.lin(m1, subset="1:ns")[,1,drop=F]
> # ci.lin(m1, subset="2:ns")[,1,drop=F]
> # ci.lin(m1, subset=c("1:ns","2:ns"))[,1,drop=F]
>
> # Extract the rates for each group and plot them
> plt("thoro=5")
> par(mar=c(3,3,1,3), mgp=c(3,1,0)/1.6, las=1)
> mort1 <- ci.lin(m1, ctr.mat=CM, subset="1:ns", E=T)[,5:7]
> mort2 <- ci.lin(m1, ctr.mat=CM, subset="2:ns", E=T)[,5:7]
> matplot(tpt, cbind(mort1, mort2), type="l", ylim=c(0.2,200),
+ log="y", lty=1, lwd=rep(c(4,1,1),2), col=rep(c("red","blue"),each=3),
+ xlab="Time since angiography (years)",
+ ylab="Mortality rate per 1000 PY")
>
> #-----
> # 14:
> # ci.lin can also be used to extract the rate ratio:

```

```

> #
> rr <- ci.lin(m1, ctr.mat=cbind(CM,-CM), subset=c("1:ns","2:ns"), E=T)[,5:7]
> rr.pl <- 10 # reference level for the RR=1 on the graph
> # - and then add the RR curve to the plot
> matlines(tpt, rr*rr.pl, type="l", lwd=c(4,1,1), col="black", lty=1)
> abline(h=rr.pl)
> axis(side=4,
+ at=rr.pl*(x <- c(c(1,2,5)/10,c(1,2,5),c(1,2,5)*10)),
+ labels=x)
> mtext("Rate ratio", side=4, line=3/1.6, las=0)
> dev.off()
null device
 1

>
> #-----
> # 15:
> # Get the population mortality data.
> #
> data(gmortDK)
> str(gmortDK)
'data.frame': 418 obs. of 21 variables:
 $ agr : num 0 5 10 15 20 25 30 35 40 45 ...
 $ per : num 38 38 38 38 38 38 38 38 38 38 ...
 $ sex : num 1 1 1 1 1 1 1 1 1 1 ...
 $ risk: num 996019 802334 753017 773393 813882 ...
 $ dt : num 14079 726 600 1167 2031 ...
 $ rt : num 14.135 0.905 0.797 1.509 2.495 ...
 $ r1 : num 1.315 0.222 0.151 0.304 0.587 ...
 $ r2 : num 0.127 0.045 0.054 0.07 0.102 0.123 0.205 0.309 0.535 0.99 ...
 $ r3 : num 0.052 0.015 0.019 0.021 0.021 0.038 0.037 0.064 0.086 0.159 ...
 $ r4 : num 0.099 0.006 0.011 0.012 0.022 0.019 0.022 0.044 0.04 0.084 ...
 $ r5 : num 0.033 0.004 0.007 0.008 0.009 0.015 0.01 0.014 0.012 0.031 ...
 $ r6 : num 0.453 0.066 0.045 0.076 0.103 0.081 0.125 0.131 0.147 0.225 ...
 $ r7 : num 0.005 0.001 0.004 0.004 0.009 0.009 0.025 0.027 0.049 0.09 ...
 $ r8 : num 0.067 0.047 0.072 0.078 0.098 ...
 $ r9 : num 2.819 0.045 0.028 0.047 0.084 ...
 $ r10 : num 0.009 0.002 0.004 0.001 0.005 0.013 0.016 0.022 0.056 0.095 ...
 $ r11 : num 1.325 0.076 0.057 0.098 0.118 ...
 $ r12 : num 0.045 0.03 0.027 0.027 0.05 0.051 0.054 0.09 0.124 0.193 ...
 $ r13 : num 0.341 0.009 0.007 0.021 0.021 0.016 0.031 0.03 0.117 0.183 ...
 $ r14 : num 6.777 0.010 0.025 0.036 0.026 ...
 $ r15 : num 0.667 0.327 0.287 0.707 1.241 ...
> head(gmortDK)
 agr per sex risk dt rt r1 r2 r3 r4 r5 r6 r7 r8 r9 r10 r11 r12
1 0 38 1 996019 14079 14.135 1.315 0.127 0.052 0.099 0.033 0.453 0.005 0.067 2.819 0.009 1.325 0.045
2 5 38 1 802334 726 0.905 0.222 0.045 0.015 0.006 0.004 0.066 0.001 0.047 0.045 0.002 0.076 0.030
3 10 38 1 753017 600 0.797 0.151 0.054 0.019 0.011 0.007 0.045 0.004 0.072 0.028 0.004 0.057 0.027
4 15 38 1 773393 1167 1.509 0.304 0.070 0.021 0.012 0.008 0.076 0.004 0.078 0.047 0.001 0.098 0.027
5 20 38 1 813882 2031 2.495 0.587 0.102 0.021 0.022 0.009 0.103 0.009 0.098 0.084 0.005 0.118 0.050
6 25 38 1 789990 1862 2.357 0.586 0.123 0.038 0.019 0.015 0.081 0.009 0.078 0.073 0.013 0.120 0.051
 r13 r14 r15
1 0.341 6.777 0.667
2 0.009 0.010 0.327
3 0.007 0.025 0.287
4 0.021 0.036 0.707
5 0.021 0.026 1.241
6 0.016 0.028 1.106
>
> #-----
> # 16:
> # Now split the data as before but also by age and period in order to
> # be able to match with the population data.
> #
> thxx <- splitLexis(thL, breaks=list(age=seq(0,90,5),
+ per=seq(1938,2038,5),
+ tfi=seq(0,55,0.5)))
> dim(thxx)
[1] 125134 24
> # Show the follow-up and how it is cut by age and period
> plot(thxx)
> # Show the follow-up and how it is cut by time from injection (3rd) and age (2nd)
> plot(thxx,3:2)
>
> #-----
> # 17:
> # Create variables in thxx that can be merged with population data
> #
> thxx$agr <- timeBand(thxx, "age", "left")
> thxx$pgr <- timeBand(thxx, "per", "left")
> thxx$m.tfi <- timeBand(thxx, "tfi", "mid")
>
> # Some additional manipulation required to get the merge on period right...
> gmortDK$pgr <- gmortDK$per + 1900
>
> ## Merge them to the split thorotrast data
> thlap <- merge(thxx, gmortDK[,c("agr","pgr","sex","rt")],
+ by=c("agr","pgr","sex"))

```

```

> dim(thlap)
[1] 125072 28
>
> #-----
> # 18:
> # Now we can compute the deaths, person-years and expected values
> #
> thlap <- transform(thlap,
+ D = (lex.status2 > 0),
+ Y = lex.deltat,
+ E = lex.deltat * rt / 1000)
> str(thlap)
'data.frame': 125072 obs. of 29 variables:
 $ sex : num 1 1 1 1 1 1 1 1 1 ...
 $ agr : num 0 0 0 0 0 0 0 0 0 ...
 $ pgr : num 1943 1948 1948 1948 1948 ...
 $ lex.id : num 3313 3478 3189 3478 3189 ...
 $ per : num 1948 1952 1951 1952 1950 ...
 $ age : num 3.91 4.08 4.42 3.58 3.92 ...
 $ tfi : num 0.0 1.5 1.5 1.0 1.0 ...
 $ lex.deltat : num 0.119 0.500 0.500 0.500 0.500 ...
 $ lex.status1: num 0 0 0 0 0 0 0 0 0 ...
 $ lex.status2: num 0 0 0 0 0 0 0 0 0 ...
 $ id : num 3313 3478 3189 3478 3189 ...
 $ birthdat :Class 'Date' num [1:125072] -9508 -8051 -8555 -8051 -8555 ...
 $ contrast : num 2 2 2 2 2 2 2 2 2 ...
 $ injecdat :Class 'Date' num [1:125072] -8079 -7108 -7489 -7108 -7489 ...
 $ volume : num 0 0 0 0 0 0 0 0 0 ...
 $ exitdat :Class 'Date' num [1:125072] 4077 8085 8085 8085 8085 ...
 $ exitstat : num 1 2 2 2 1 2 2 1 2 ...
 $ cause : num 9 NA NA NA NA 9 NA NA 9 NA ...
 $ liverdat :Class 'Date' num [1:125072] NA ...
 $ liver : num 0 0 0 0 0 0 0 0 0 ...
 $ hepcc : num 0 0 0 0 0 0 0 0 0 ...
 $ chola : num 0 0 0 0 0 0 0 0 0 ...
 $ hmang : num 0 0 0 0 0 0 0 0 0 ...
 $ entrydat :Class 'Date' num [1:125072] -7713 -6742 -7123 -6742 -7123 ...
 $ Y : num 0.119 0.500 0.500 0.500 0.500 ...
 $ D : logi FALSE FALSE FALSE FALSE FALSE ...
 $ m.tfi : num 0.25 1.75 1.75 1.25 1.25 1.25 2.25 2.25 0.25 0.75 ...
 $ rt : num 14.13 7.88 7.88 7.88 7.88 ...
 $ E : num 0.00168 0.00394 0.00394 0.00394 0.00394 ...
>
> #-----
> # 19:
> # Show the SMR overall and by time since injection
> #
> stat.table(contrast,
+ list(D=sum(D), E=sum(E), SMR=ratio(D,E)),
+ margins=TRUE, data=thlap)

contrast D E SMR

1 619.00 221.39 2.80
2 796.00 472.95 1.68

Total 1415.00 694.34 2.04

> stat.table(list(time=floor(tfi/5)*5, contrast),
+ list(D=sum(D), E=sum(E), SMR=ratio(D,E)),
+ margins=TRUE, data=thlap)

time contrast

 1 2 Total
0 85.00 141.00 226.00
 18.67 34.46 53.13
 4.55 4.09 4.25
5 69.00 92.00 161.00
 20.23 40.38 60.61
 3.41 2.28 2.66
10 70.00 94.00 164.00
 22.80 48.83 71.63
 3.07 1.93 2.29
15 68.00 89.00 157.00
 28.35 58.26 86.61
 2.40 1.53 1.81
20 73.00 101.00 174.00
 31.10 64.08 95.18
 2.35 1.58 1.83
25 79.00 94.00 173.00
 28.83 72.16 100.99

```

|       |        |        |         |
|-------|--------|--------|---------|
|       | 2.74   | 1.30   | 1.71    |
| 30    | 66.00  | 91.00  | 157.00  |
|       | 25.89  | 78.33  | 104.22  |
|       | 2.55   | 1.16   | 1.51    |
| 35    | 51.00  | 74.00  | 125.00  |
|       | 22.27  | 55.93  | 78.19   |
|       | 2.29   | 1.32   | 1.60    |
| 40    | 39.00  | 20.00  | 59.00   |
|       | 16.60  | 20.48  | 37.08   |
|       | 2.35   | 0.98   | 1.59    |
| 45    | 17.00  | 0.00   | 17.00   |
|       | 5.89   | 0.06   | 5.95    |
|       | 2.89   | 0.00   | 2.86    |
| 50    | 2.00   | NA     | 2.00    |
|       | 0.76   | NA     | 0.76    |
|       | 2.63   | NA     | 2.63    |
| Total | 619.00 | 796.00 | 1415.00 |
|       | 221.39 | 472.95 | 694.34  |
|       | 2.80   | 1.68   | 2.04    |

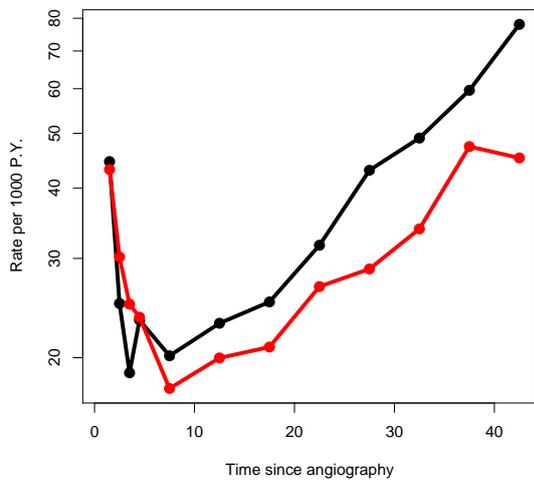
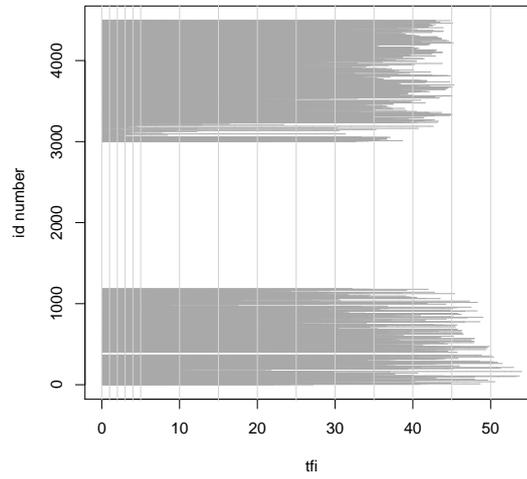
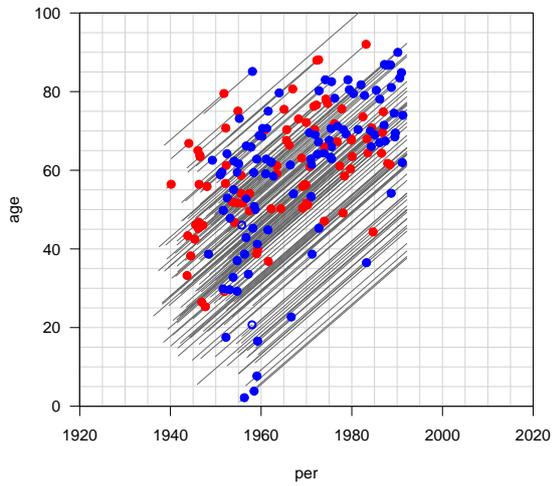
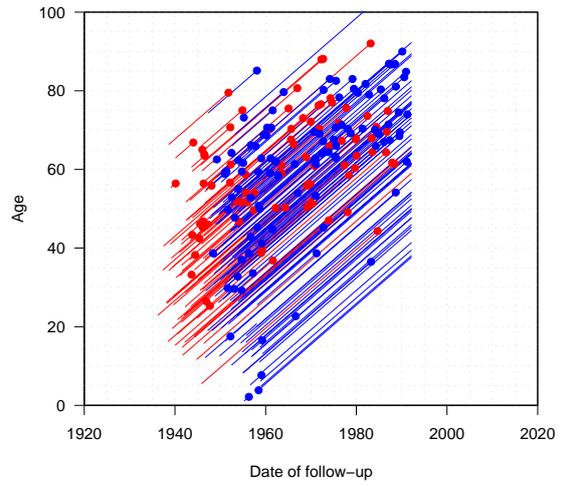
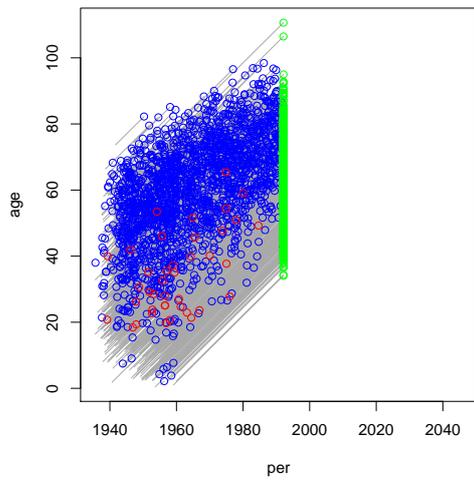
```

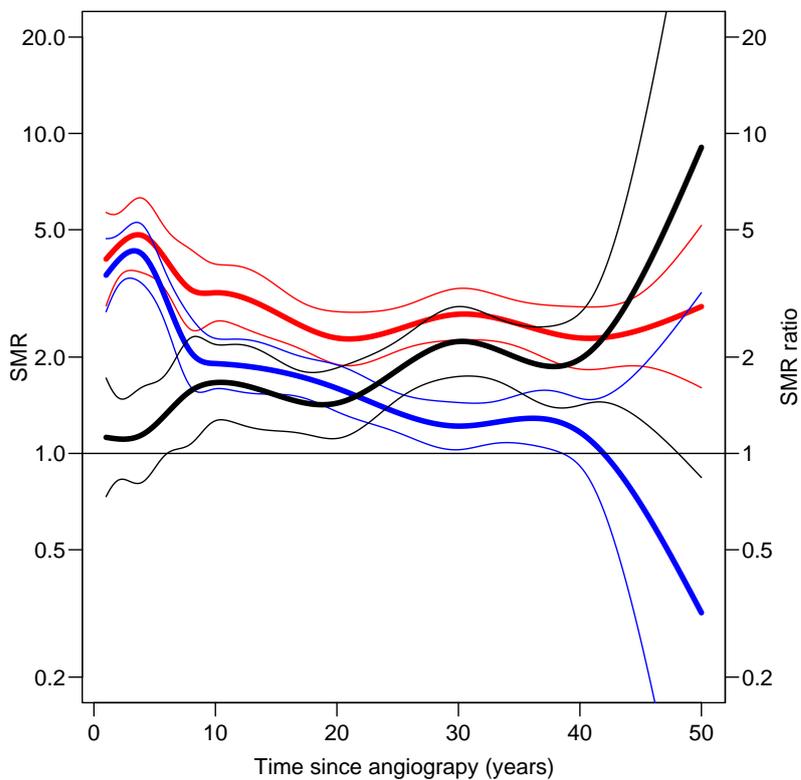
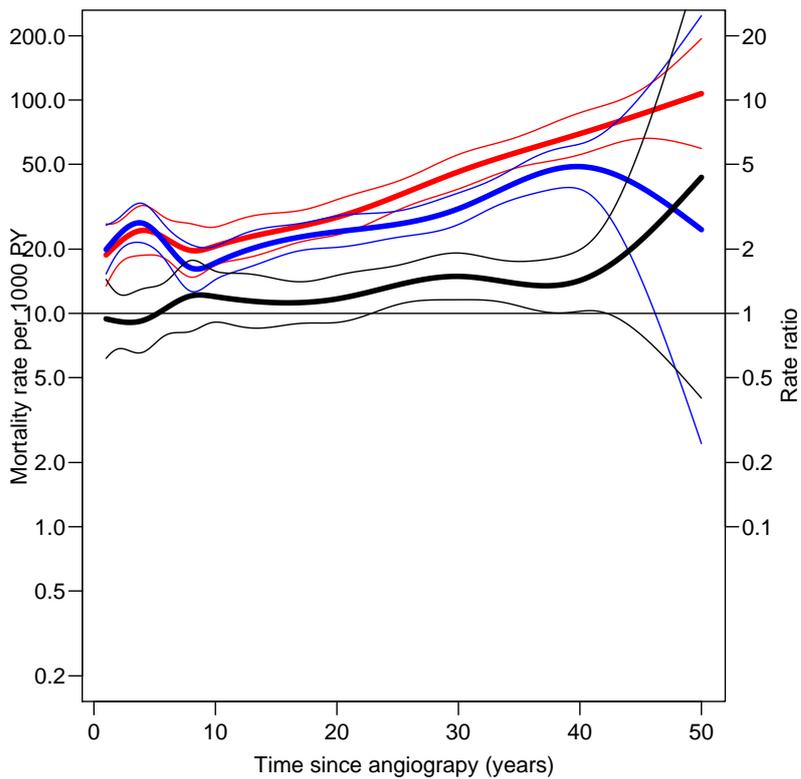
>
> #-----
> # 20:
> # Now the entire analysis can be repeated using E instead of Y
> #
> # The model with natural splines and interaction
> mle <- glm(D ~ -1 +
+ factor(contrast):ns(m.tfi, knots=kn, Bo=bk, intercept=TRUE) +
+ offset(log(E)),
+ family=poisson, data=th1ap)
>
> plt("thoro-6")
> par(mar=c(3,3,1,3), mgp=c(3,1,0)/1.6, las=1)
>
> # Extract the rates for each group (same contrast matrix as before)
> mort1 <- ci.lin(mle, ctr.mat=CM, subset="1:ns", E=T)[,5:7]
> mort2 <- ci.lin(mle, ctr.mat=CM, subset="2:ns", E=T)[,5:7]
> matplot(tpt, cbind(mort1, mort2), type="l", ylim=c(0.2,20),
+ log="y", lty=1, lwd=rep(c(4,1,1),2), col=rep(c("red","blue"),each=3),
+ xlab="Time since angiography (years)",
+ ylab="SMR")
> abline(h=1)
>
> # ci.lin can also be used to extract the ratio of the SMRs:
> rr <- ci.lin(mle, ctr.mat=cbind(CM,-CM), subset=c("1:ns","2:ns"), E=T)[,5:7]
> matlines(tpt, rr, type="l", lwd=c(4,1,1), col="black", lty=1)
> vals <- c(1,2,5)
> axis(side=4,
+ at=(x <- as.vector(outer(vals, -1:1, function(x,y) x*10^y))),
+ labels=x)
> mtext("SMR ratio", side=4, line=3/1.6, las=0)
> dev.off()
windows
 2
>
> #-----
> # 21:
> # The difference between groups using SMRs is larger than using overall
> # mortality. SMR analysis (partially) takes into account age and calendar
> # time which may be confounding the relationship.
>

Program: thoro.R
Folder: C:\Bendix\Undervis\SPE\pracs\r
Ended: mandag 28. maj 2007, 13:48:42
Elapsed: 00:02:17

> proc.time()
 user system elapsed
104.70 14.15 141.95

```





## 2.7 Matched case-control study: *Salmonella* Typhimurium

First we load the dataset and then get an overview:

```
data(S.typh)
str(S.typh)
```

1. We use the `clogit` function to examine the effect of pork:

```
clogit(case ~ pork + strata(set), data = S.typh)
```

which yields

```
Call: clogit(case ~ pork + strata(set), data = S.typh)
```

```
 coef exp(coef) se(coef) z p
pork 0.266 1.30 0.454 0.585 0.56
```

```
Likelihood ratio test=0.35 on 1 df, p=0.554 n=129
(7 observations deleted due to missingness)
```

The odds ratio for eating pork on the risk of *S.typh* infection is 1.30 with 95% confidence interval (0.54,3.18).

2. We can look through the other exposure variables and find out if any of them have a strong association with the outcome by using the following series of statements:

This is done by a sequence of statements:

```
clogit(case ~ beef + strata(set), data = S.typh)
clogit(case ~ veal + strata(set), data = S.typh)
clogit(case ~ poultry + strata(set), data = S.typh)
clogit(case ~ liverp + strata(set), data = S.typh)
clogit(case ~ veg + strata(set), data = S.typh)
clogit(case ~ fruit + strata(set), data = S.typh)
clogit(case ~ egg + strata(set), data = S.typh)
clogit(case ~ plant7 + strata(set), data = S.typh)
```

We find that `plant7` is a risk factor,  $OR = 4.47$  with 95% confidence interval (1.62, 12.39), while `fruit` is protective with  $OR = 0.16$  and 95% C.I. (0.04, 0.60).

3. (a) We fit the model with both `plant7` and `fruit` as main effects using the following syntax:

```
m1 <- clogit(case ~ factor(plant7) + factor(fruit) + strata(set), data=S.typh)
```

which generates the output

```
Call:
clogit(case ~ factor(plant7) + factor(fruit) + strata(set), data = S.typh)
```

```
 coef exp(coef) se(coef) z p
```

```
factor(plant7)1 1.50 4.465 0.586 2.55 0.011
factor(fruit)1 -1.42 0.242 0.753 -1.89 0.059
```

```
Likelihood ratio test=14.2 on 2 df, p=0.000847 n=121
(15 observations deleted due to missingness)
```

The magnitude of the protective effect of `fruit` has been attenuated (from 0.16 with 95% confidence interval (0.04,0.60) to 0.24 with greater imprecision implied by the 95% confidence interval, which is (0.06,1.06)) and the association with the outcome is less strong having adjusted for `plant7`. The estimated odds ratio for `plant7` is unchanged when `fruit` is included in the model (the odds ratio is 4.47) but the 95% confidence interval is wider, having changed from (1.62,12.39) in the single variable model to (1.42,14.13) in the two variable model. There is clearly some evidence of association between the risk of *S.typh* and each of these two exposures adjusted for the other.

- (b) The model where we consider the possible modifying effect of `fruit` among those subjects who have eaten meat from `plant7` is not strictly an interaction model. It is a model that states that there is no effect of `fruit` among people that have not eaten meat from `plant7`, but that the effect of `plant7` is different among fruit eaters and non-fruit eaters.

In order to fit this model we need to generate the indicators of `plant7=1` and `fruit=1` and `plant7=1` and `fruit=0`, which we do using the `I()` facility:

```
m2 <- clogit(case ~ I(plant7*(1-fruit)) + I(plant7*(fruit)) +
strata(set), data=S.typh)
```

The resulting output is:

```
Call: clogit(case ~ I(plant7 * (1 - fruit)) + I(plant7 * (fruit)) +
strata(set), data = S.typh)
```

|                         | coef | exp(coef) | se(coef) | z    | p     |
|-------------------------|------|-----------|----------|------|-------|
| I(plant7 * (1 - fruit)) | 3.05 | 21.21     | 1.170    | 2.61 | 0.009 |
| I(plant7 * (fruit))     | 1.30 | 3.68      | 0.593    | 2.20 | 0.028 |

```
Likelihood ratio test=13.3 on 2 df, p=0.00130 n=121 (15
observations deleted due to missingness)
```

which can be conveniently summarised using the `ci.lin` function:

```
round(ci.lin(m2, Exp=TRUE,
ctr.mat=rbind("No fruit"=c(1,0),"Fruit"=c(0,1),"Ratio"=c(1,-1))),2)
```

|          | Estimate | StdErr | z    | P    | exp(Est.) | 2.5% | 97.5%  |
|----------|----------|--------|------|------|-----------|------|--------|
| No fruit | 3.05     | 1.17   | 2.61 | 0.01 | 21.21     | 2.14 | 209.97 |
| Fruit    | 1.30     | 0.59   | 2.20 | 0.03 | 3.68      | 1.15 | 11.77  |
| Ratio    | 1.75     | 1.13   | 1.54 | 0.12 | 5.76      | 0.62 | 53.16  |

We see that the effect of eating meat from `plant7` is much larger among those who did not eat `fruit` (so `fruit` is indeed protective among those who ate meat from `plant7`), although the evidence for this differential effect is weak since the p-value for the `Ratio` is 0.12.

We also consider the effect of `fruit` stratified by `plant7`:

```
m4 <- clogit(case ~ I(fruit * (1-plant7)) + I(fruit * (plant7)) +
strata(set), data=S.typh)
```

which produces the following output:

```
Call: clogit(case ~ I(fruit * (1 - plant7)) + I(fruit * (plant7)) +
strata(set), data = S.typh)
```

|                         | coef  | exp(coef) | se(coef) | z     | p      |
|-------------------------|-------|-----------|----------|-------|--------|
| I(fruit * (1 - plant7)) | -2.23 | 0.107     | 0.793    | -2.82 | 0.0049 |
| I(fruit * (plant7))     | -0.83 | 0.436     | 0.761    | -1.09 | 0.2800 |

```
Likelihood ratio test=12.5 on 2 df, p=0.00195 n=121
(15 observations deleted due to missingness)
```

Once again the `ci.lin` command is helpful:

```
round(ci.lin(m4, Exp=TRUE,
+ ctr.mat=rbind("No plant7"=c(1,0),"Plant7"=c(0,1),"Ratio"=c(1,-1))),2)
```

|           | Estimate | StdErr | z     | P    | exp(Est.) | 2.5% | 97.5% |
|-----------|----------|--------|-------|------|-----------|------|-------|
| No plant7 | -2.23    | 0.79   | -2.82 | 0.00 | 0.11      | 0.02 | 0.51  |
| Plant7    | -0.83    | 0.76   | -1.09 | 0.28 | 0.44      | 0.10 | 1.94  |
| Ratio     | -1.40    | 0.61   | -2.32 | 0.02 | 0.25      | 0.07 | 0.81  |

There is a protective effect of fruit among both `plant7` strata, that is, among those who did and did not eat meat from `plant7`. The protective effect of fruit is much greater in those who did not eat meat from `plant7` (an odds ratio of 0.11 compared to an odds ratio of 0.44 among those who did eat meat from `plant7`), and there is strong evidence against the null hypothesis of no differential effect since the p-value for the `Ratio` is 0.02.

4. We fit the “default” model with an interaction between `plant7` and `fruit` using the following syntax:

```
m6 <- clogit(case ~ factor(plant7):factor(fruit) + strata(set),
data=S.typh)
```

The output from this model fit is:

Call:

```
clogit(case ~ factor(plant7):factor(fruit) + strata(set), data = S.typh)
```

|                                | coef   | exp(coef) | se(coef) | z     | p     |
|--------------------------------|--------|-----------|----------|-------|-------|
| factor(plant7)0:factor(fruit)0 | -0.306 | 0.737     | 1.133    | -0.27 | 0.790 |
| factor(plant7)1:factor(fruit)0 | 1.706  | 5.506     | 1.138    | 1.50  | 0.130 |
| factor(plant7)0:factor(fruit)1 | -1.424 | 0.241     | 0.615    | -2.31 | 0.021 |
| factor(plant7)1:factor(fruit)1 | NA     | NA        | 0.000    | NA    | NA    |

```
Likelihood ratio test=14.3 on 3 df, p=0.00255 n=121 (15
observations deleted due to missingness)
```

When we fit the interaction model parametrisation in the default way we get the last of the parameters aliased, where `plant7 = 1` and `fruit = 1` corresponds to the reference category. This model thus has the “wrong” category as reference. We would of course prefer the low risk category as reference; i.e. the fruit eaters that are not exposed to `plant7`. The reference in the default model is the “1” category for both factors, hence we want to use `1 - plant7` as the exposure factor to improve the parametrisation and select the correct reference category. The reparametrised interaction model can be fitted using the syntax:

```
m7 <- clogit(case ~ factor(1-plant7):factor(fruit) + strata(set),
data = S.typh)
```

The resulting output is:

Call:

```
clogit(case ~ factor(1 - plant7):factor(fruit) + strata(set),
data = S.typh)
```

|                                    | coef | exp(coef) | se(coef) | z    | p      |
|------------------------------------|------|-----------|----------|------|--------|
| factor(1 - plant7)0:factor(fruit)0 | 3.13 | 22.86     | 1.183    | 2.65 | 0.0081 |
| factor(1 - plant7)1:factor(fruit)0 | 1.12 | 3.06      | 1.116    | 1.00 | 0.3200 |
| factor(1 - plant7)0:factor(fruit)1 | 1.42 | 4.15      | 0.615    | 2.31 | 0.0210 |
| factor(1 - plant7)1:factor(fruit)1 | NA   | NA        | 0.000    | NA   | NA     |

```
Likelihood ratio test=14.3 on 3 df, p=0.00255 n=121
(15 observations deleted due to missingness)
```

We see that all of the estimated odds ratios are greater than 1, so we have been successful in assigning the lowest risk category as the reference. The values in the `coef` column can be used to populate the cells in the tables for display, along with 95% confidence intervals generated manually or using the `ci.lin` command.

We can use the “analysis of deviance” via the `anova` command to compare the interaction model (`m6`) with the main effects model (`m1`):

```
anova(m1, m6, test="Chisq")
```

Analysis of Deviance Table

```
Model 1: Surv(rep(1, 136), case) ~ factor(plant7) + factor(fruit) + strata(set)
Model 2: Surv(rep(1, 136), case) ~ factor(plant7):factor(fruit) + strata(set)
 Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1 119 60.897
2 118 60.765 1 0.132 0.717
```

The p-value of 0.717 provides no evidence against the null hypothesis that there is no interaction between `fruit` and `plant7`. The analysis of deviance can also be used to compare model `m4` (separate effects of `fruit` in those who did and did not eat meat from `plant7`) to the interaction model `m6`. This generates a p-value of 0.18, suggesting again that the interaction model (a three variable model) does not provide a better fit the model `m4`, which is a two variable model allowing for separate effect of `fruit` in the strata defined by `plant7` but no effect of `plant7` alone. For a final model we would have to choose between one of the competing two variables models `m1` or `m4`.

## 2.7.1 R-program for the analysis

R 2.4.1

```

Program: salmonella.R
Folder: C:\Bendix\Undervis\SPE\lg\pracs
Started: søndag 13. maj 2007, 18:51:15

```

```
> library(Epi)
> library(survival)
Loading required package: splines
> data(S.typh)
> str(S.typh)
'data.frame': 136 obs. of 15 variables:
 $ id : num 1 2 3 4 5 6 7 8 9 10 ...
 $ set : num 1 1 1 2 2 2 3 3 3 4 ...
 $ case : num 1 0 0 1 0 0 1 0 0 1 ...
 $ age : num 52 52 52 41 41 41 9 9 9 16 ...
 $ sex : num 1 1 1 1 1 1 1 1 1 1 ...
 $ abroad : num 0 0 0 0 0 0 0 0 0 0 ...
 $ beef : num 1 1 1 1 1 1 0 1 NA 1 ...
 $ pork : num 1 0 1 NA 1 1 0 1 0 0 ...
 $ veal : num 1 0 0 NA 0 0 0 0 0 0 ...
 $ poultry: num 1 0 1 NA 0 0 0 1 0 0 ...
 $ liverp : num 1 1 1 1 1 1 1 1 1 1 ...
 $ veg : num 0 1 1 1 1 1 0 1 NA 1 ...
 $ fruit : num 1 1 1 NA 0 1 1 1 1 0 ...
 $ egg : num 1 0 1 NA 1 1 NA 0 1 NA ...
 $ plant7 : num 1 0 0 NA 1 1 0 1 0 1 ...
> summary(S.typh)
 id set case age sex abroad
Min. : 1.00 Min. : 1.0 Min. :0.0000 Min. : 1.00 Min. :1.0000 Min. :0.00000
1st Qu.: 44.75 1st Qu.:14.0 1st Qu.:0.0000 1st Qu.:15.00 1st Qu.:1.0000 1st Qu.:0.00000
Median : 95.50 Median :28.5 Median :0.0000 Median :18.00 Median :1.0000 Median :0.00000
Mean : 95.83 Mean :28.5 Mean :0.3456 Mean :24.99 Mean :1.456 Mean :0.02206
3rd Qu.:142.25 3rd Qu.:42.0 3rd Qu.:1.0000 3rd Qu.:40.00 3rd Qu.:2.0000 3rd Qu.:0.00000
Max. :205.00 Max. :59.0 Max. :1.0000 Max. :64.00 Max. :2.000 Max. :1.00000
 NA's :19.00
 beef pork veal poultry liverp veg
Min. :0.0000 Min. :0.0000 Min. :0.000000 Min. :0.0000 Min. :0.0000 Min. :0.0000
1st Qu.:0.0000 1st Qu.:1.0000 1st Qu.:0.000000 1st Qu.:0.0000 1st Qu.:1.0000 1st Qu.:1.0000
Median :1.0000 Median :1.0000 Median :0.000000 Median :0.0000 Median :1.0000 Median :1.0000
Mean :0.6769 Mean :0.7597 Mean :0.05344 Mean :0.3178 Mean :0.9701 Mean :0.9242
3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:0.000000 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000
Max. :1.0000 Max. :1.0000 Max. :1.000000 Max. :1.0000 Max. :1.0000 Max. :1.0000
NA's :6.0000 NA's :7.0000 NA's :5.000000 NA's :7.0000 NA's :2.0000 NA's :4.0000
 fruit egg plant7
Min. :0.0000 Min. :0.0000 Min. : 0.0000
1st Qu.:1.0000 1st Qu.:1.0000 1st Qu.: 0.0000
Median :1.0000 Median :1.0000 Median : 1.0000
Mean :0.856 Mean :0.7717 Mean : 0.5041
3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.: 1.0000
Max. :1.0000 Max. :1.0000 Max. : 1.0000
NA's :4.0000 NA's :9.0000 NA's :13.0000
>
> ### Questions 1 and 2: Screen the variables for effect ###
>
> clogit(case ~ beef + strata(set), data=S.typh)
Call:
clogit(case ~ beef + strata(set), data = S.typh)

 coef exp(coef) se(coef) z p
beef -0.119 0.888 0.422 -0.281 0.78

Likelihood ratio test=0.08 on 1 df, p=0.778 n=130 (6 observations deleted due to missingness)
> clogit(case ~ pork + strata(set), data=S.typh)
Call:
clogit(case ~ pork + strata(set), data = S.typh)

 coef exp(coef) se(coef) z p
pork 0.266 1.30 0.454 0.585 0.56

Likelihood ratio test=0.35 on 1 df, p=0.554 n=129 (7 observations deleted due to missingness)
> clogit(case ~ veal + strata(set), data=S.typh)
Call:
clogit(case ~ veal + strata(set), data = S.typh)

 coef exp(coef) se(coef) z p
veal 19.8 4.08e+08 7927 0.0025 1

Likelihood ratio test=7.4 on 1 df, p=0.00651 n=131 (5 observations deleted due to missingness)
Warning message:
Loglik converged before variable 1 ; beta may be infinite. in: fitter(X, Y, strats, offset, init, control, weights = weig
> clogit(case ~ poultry + strata(set), data=S.typh)
```

```

Call:
clogit(case ~ poultry + strata(set), data = S.typh)

 coef exp(coef) se(coef) z p
poultry 0.128 1.14 0.39 0.327 0.74

Likelihood ratio test=0.11 on 1 df, p=0.744 n=129 (7 observations deleted due to missingness)
> clogit(case ~ liverp + strata(set), data=S.typh)
Call:
clogit(case ~ liverp + strata(set), data = S.typh)

 coef exp(coef) se(coef) z p
liverp -1.79 0.167 1.15 -1.55 0.12

Likelihood ratio test=2.9 on 1 df, p=0.0884 n=134 (2 observations deleted due to missingness)
> clogit(case ~ veg + strata(set), data=S.typh)
Call:
clogit(case ~ veg + strata(set), data = S.typh)

 coef exp(coef) se(coef) z p
veg -0.63 0.533 0.687 -0.917 0.36

Likelihood ratio test=0.84 on 1 df, p=0.358 n=132 (4 observations deleted due to missingness)
> clogit(case ~ fruit + strata(set), data=S.typh)
Call:
clogit(case ~ fruit + strata(set), data = S.typh)

 coef exp(coef) se(coef) z p
fruit -1.81 0.163 0.659 -2.75 0.006

Likelihood ratio test=9.47 on 1 df, p=0.00209 n=132 (4 observations deleted due to missingness)
> clogit(case ~ egg + strata(set), data=S.typh)
Call:
clogit(case ~ egg + strata(set), data = S.typh)

 coef exp(coef) se(coef) z p
egg 3.12e-17 1 0.53 5.89e-17 1

Likelihood ratio test=0 on 1 df, p=1 n=127 (9 observations deleted due to missingness)
> clogit(case ~ plant7 + strata(set), data=S.typh)
Call:
clogit(case ~ plant7 + strata(set), data = S.typh)

 coef exp(coef) se(coef) z p
plant7 1.50 4.47 0.519 2.88 0.0039

Likelihood ratio test=10.1 on 1 df, p=0.00149 n=123 (13 observations deleted due to missingness)
>
> ### Question 3 ###
>
> # We found in question 2 that "plant7" is a risk factor
> # while "fruit" is protective. This raises two questions
> # (a) What is the effect of each variable adjusted for the other
> # in a main effects model?
> # (b) What is the effect of plant7 stratified by fruit and vice versa,
> # that is, the effect of fruit stratified by plant 7
> #
>
> # First we tackle part (a) by fitting the model
> # with both "plant7" and "fruit" as main effects
>
> (m1 <- clogit(case ~ factor(plant7) + factor(fruit) + strata(set),
+ data=S.typh))
Call:
clogit(case ~ factor(plant7) + factor(fruit) + strata(set), data = S.typh)

 coef exp(coef) se(coef) z p
factor(plant7)1 1.50 4.465 0.586 2.55 0.011
factor(fruit)1 -1.42 0.242 0.753 -1.89 0.059

Likelihood ratio test=14.2 on 2 df, p=0.000847 n=121 (15 observations deleted due to missingness)
>
> # The magnitude of the protective effect of "fruit" has been attenuated
> # and the association with the outcome is less strong having adjusted for
> # "plant7". The estimated odds ratio for "plant7" is unchanged when "fruit" is
> # included in the model. There is clearly some evidence of association between
> # the risk of S.typh and each of these two exposures adjusted for the other.
>
> # Now for part (b). We fit a stratified model examining the effect of "plant7"
> # within each of the two strata defined by "fruit"

```

```

>
> (m2 <- clogit(case ~ I(plant7*(1-fruit)) + I(plant7*(fruit)) + strata(set),
+ data=S.typh))
Call:
clogit(case ~ I(plant7 * (1 - fruit)) + I(plant7 * (fruit)) +
 strata(set), data = S.typh)

 coef exp(coef) se(coef) z p
I(plant7 * (1 - fruit)) 3.05 21.21 1.170 2.61 0.009
I(plant7 * (fruit)) 1.30 3.68 0.593 2.20 0.028

Likelihood ratio test=13.3 on 2 df, p=0.00130 n=121 (15 observations deleted due to missingness)
>
> # Use the "ci.lin" function to extract the effects of "plant7" among those
> # do and do not eat "fruit", and their ratio. Note that the z-statistic and
> # P-value for the "Ratio" provided a test of the null hypothesis that
> # the effect of "plant7" does not differ between those who do and do not eat
> # "fruit". This comparison could also be achieved by comparing model m2 with
> # a model that has "plant7" as the only exposure variable, which we fit as
> # model m3 below
>
> round(ci.lin(m2, Exp=TRUE,
+ ctr.mat=rbind("No fruit"=c(1,0),"Fruit"=c(0,1),"Ratio"=c(1,-1))),2)
 Estimate StdErr z P exp(Est.) 2.5% 97.5%
No fruit 3.05 1.17 2.61 0.01 21.21 2.14 209.97
Fruit 1.30 0.59 2.20 0.03 3.68 1.15 11.77
Ratio 1.75 1.13 1.54 0.12 5.76 0.62 53.16
>
> # Fit the model with "plant7" as the only exposure variable among those
> # individuals who do not have a missing value for "fruit". Note that
> # we need to use only those observations with non-missing values of "fruit"
> # and "plant7" in order to compare the two models using ANOVA, since
> # this requires that the two models are based on the same number of obs.
>
> (m3 <- clogit(case ~ plant7 + strata(set),
+ data=S.typh, subset = !is.na(fruit)))
Call:
clogit(case ~ plant7 + strata(set), data = S.typh, subset = !is.na(fruit))

 coef exp(coef) se(coef) z p
plant7 1.62 5.04 0.57 2.84 0.0046

Likelihood ratio test=10.2 on 1 df, p=0.00139 n=121 (11 observations deleted due to missingness)
>
> # Compare models m2 and m3 using the "analysis of deviance"
>
> anova (m2, m3, test="Chisq")
Analysis of Deviance Table

Model 1: Surv(rep(1, 136), case) ~ I(plant7 * (1 - fruit)) + I(plant7 *
(fruit)) + strata(set)
Model 2: Surv(rep(1, 136), case) ~ plant7 + strata(set)
 Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1 119 61.760
2 120 64.833 -1 -3.073 0.080
>
> # Now we consider the effect of "fruit" stratified by "plant7"
>
> (m4 <- clogit(case ~ I(fruit*(1-plant7)) + I(fruit*(plant7)) + strata(set),
+ data=S.typh))
Call:
clogit(case ~ I(fruit * (1 - plant7)) + I(fruit * (plant7)) +
 strata(set), data = S.typh)

 coef exp(coef) se(coef) z p
I(fruit * (1 - plant7)) -2.23 0.107 0.793 -2.82 0.0049
I(fruit * (plant7)) -0.83 0.436 0.761 -1.09 0.2800

Likelihood ratio test=12.5 on 2 df, p=0.00195 n=121 (15 observations deleted due to missingness)
>
> round(ci.lin(m4, Exp=TRUE,
+ ctr.mat=rbind("No plant7"=c(1,0),"Plant7"=c(0,1),"Ratio"=c(1,-1))),2)
 Estimate StdErr z P exp(Est.) 2.5% 97.5%
No plant7 -2.23 0.79 -2.82 0.00 0.11 0.02 0.51
Plant7 -0.83 0.76 -1.09 0.28 0.44 0.10 1.94
Ratio -1.40 0.61 -2.32 0.02 0.25 0.07 0.81
>
> # Fit the model with "fruit" as the only exposure variable among those
> # individuals who do not have a missing value for "aplnt7"
>
> (m5 <- clogit(case ~ fruit + strata(set),
+ data=S.typh, subset = !is.na(plant7)))
Call:
clogit(case ~ fruit + strata(set), data = S.typh, subset = !is.na(plant7))

```

```

 coef exp(coef) se(coef) z p
fruit -1.56 0.209 0.681 -2.3 0.022

Likelihood ratio test=6.11 on 1 df, p=0.0134 n=121 (2 observations deleted due to missingness)
>
> # Compare models m2 and m3 using the "analysis of deviance"
>
> anova (m4, m5, test="Chisq")
Analysis of Deviance Table

Model 1: Surv(rep(1, 136), case) ~ I(fruit * (1 - plant7)) + I(fruit *
(plant7)) + strata(set)
Model 2: Surv(rep(1, 136), case) ~ fruit + strata(set)
 Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1 119 62.566
2 120 68.936 -1 -6.370 0.012
>
> ### Question 4 ###
>
> # Now we consider fitting the model with an interaction
> # between "plant7" and "fruit"
>
> (m6 <- clogit(case ~ factor(plant7):factor(fruit) + strata(set),
+ data=S.typh))
Call:
clogit(case ~ factor(plant7):factor(fruit) + strata(set), data = S.typh)

 coef exp(coef) se(coef) z p
factor(plant7)0:factor(fruit)0 -0.306 0.737 1.133 -0.27 0.790
factor(plant7)1:factor(fruit)0 1.706 5.506 1.138 1.50 0.130
factor(plant7)0:factor(fruit)1 -1.424 0.241 0.615 -2.31 0.021
factor(plant7)1:factor(fruit)1 NA NA 0.000 NA NA

Likelihood ratio test=14.3 on 3 df, p=0.00255 n=121 (15 observations deleted due to missingness)
Warning message:
X matrix deemed to be singular; variable 4 in: coxph(formula = Surv(rep(1, 136), case) ~ factor(plant7):factor(fruit) +
>
> # Compare this model to the main effects model
>
> anova(m1, m6, test="Chisq")
Analysis of Deviance Table

Model 1: Surv(rep(1, 136), case) ~ factor(plant7) + factor(fruit) + strata(set)
Model 2: Surv(rep(1, 136), case) ~ factor(plant7):factor(fruit) + strata(set)
 Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1 119 60.897
2 118 60.765 1 0.132 0.717
>
> # The analysis of deviance suggests that there is little evidence against
> # the null hypothesis of no interaction between "fruit" and "plant7"
>
> # The model with an interaction between "plant7" and "fruit"
> # did not use the category with the lowest risk ("plant7" = 0 and "fruit" = 1) as
> # the reference category, making the parameter estimates more difficult to
> # interpret. Here we fit an interaction between "1-plant7" and "fruit"
> # to alias the lowest risk category.
>
> (m7 <- clogit(case ~ factor(1-plant7):factor(fruit) + strata(set),
+ data=S.typh))
Call:
clogit(case ~ factor(1 - plant7):factor(fruit) + strata(set),
 data = S.typh)

 coef exp(coef) se(coef) z p
factor(1 - plant7)0:factor(fruit)0 3.13 22.86 1.183 2.65 0.0081
factor(1 - plant7)1:factor(fruit)0 1.12 3.06 1.116 1.00 0.3200
factor(1 - plant7)0:factor(fruit)1 1.42 4.15 0.615 2.31 0.0210
factor(1 - plant7)1:factor(fruit)1 NA NA 0.000 NA NA

Likelihood ratio test=14.3 on 3 df, p=0.00255 n=121 (15 observations deleted due to missingness)
Warning message:
X matrix deemed to be singular; variable 4 in: coxph(formula = Surv(rep(1, 136), case) ~ factor(1 - plant7):factor(fruit) +
>
> # What do we interpret the parametrisation for this model?
> # (m8 <- clogit(case ~ plant7*fruit + strata(set),
+ data=S.typh))
>
> # The results are summarized using "ci.lin"
>
> # round(exp(ci.lin(m5, subset=1:3)[,c(1,5,6)]), 2)
>

Program: salmonella.R

```

Folder: C:\Bendix\Undervis\SPE\lg\pracs  
Ended: søndag 13. maj 2007, 18:51:16  
Elapsed: 00:00:01  
-----