Statistical Analysis in the Lexis Diagram: Age-Period-Cohort models

Max Planck Institut for Demographic Research, Rostock
March 2009

www.biostat.ku.dk/~bxc/APC/MPIDR-2009

(Compiled Sunday 29th March, 2009 at 01:23)

Bendix Carstensen Steno Diabetes Center, Gentofte, Denmark
& Department of Biostatistics, University of Copenhagen
bxc@steno.dk
http://www.biostat.ku.dk/~bxc/

Eva Gelnarova Institute of Biostatistics and Analyses, Masaryk University, Brno
Czech Republic
gelnarova@cba.muni.cz

Contents

Preface

The course in Age-Period-Cohort models at Max Planck Institute for Demographic Research was initiated by an invitation from Jutta Gambe to me. I accepted on the condition that I was allowed to draw on the assistance for preparation and tutoring of practicals from Eva Gelnarova, whom I met when she was a student at the course "Statistical Practise in Epidemiology with R" in Tartu, Estonia in the summer of 2008.

This set of practical exercises were deveoped in collaboration between us during the first months of 2009. I did the final editing so any faults and errors are my responsibility.

Bendix Carstensen

Course program

The daily program will have one lecture and one practical each morning and each afternoon.

Lectures will be between 45 and 90 minutes; normally with one or two breaks.

The practicals will follow the lecture to fill the 3-hour slot. Sometimes we may need to push over some of the practical computing to take a bit of the beginning of the next slot.

Monday 23rd	
09:00 – 09:15	Welcome and introduction.
09:15 – 12:15	Morning slot:
– L1:	Follow-up time and rates from register data (`surv-rate`)
–	Lexis machinery in `Epi` (`lifetable`)
–	Follow-up time and rates from population data (`tab-mod`)
– P1:	Danish prime ministers (`pm`)
13:15 – 16:15	Afternoon slot:
– L2:	Likelihood for rates: Cox and Poisson
–	Cox as limit of the Poisson (`WntCma`)
–	Poisson model for rates: Factor models
–	Practical handling of linear contrasts in `R` using `ci.lin()` (`tab-mod`)
– P2:	Rates and survival (`rates`)
–	Tabulation (`tabulate`)

Tuesday 24th

09:00 – 09:30		**Recap of Monday**
09:30 – 12:15		Morning slot:
–	L3:	The age-period and the age-cohort model. (`AP-AC`)
–		The Age-drift model
–	P3:	Age-period model (`age-per`)
–		Age-cohort model (`age-coh`)
–		Age-drift model (`age-drift`)
13:15 – 16:15		Afternoon slot:
–	L4:	The Age-period-cohort model
–		Parametrizations
–	P4:	Age-period-cohort model (`age-per-coh`)

Wednesday 25th

09:00 – 09:30		**Recap of Tuesday**
09:30 – 12:15		Morning slot:
–	L5:	Data tabulated by age period *and* cohort.
–		Parametrization revisted: The general case.
–	P5:	Age-period-cohort model for triangles (`apc-tri`)
–	L5:	The implementation of `apc.fit`.
–		Parametrizations.
–		The residual parametrization.
–	P5:	Using the `apc`-functions from `Epi`. (`apc-fit`)
13:15 – 16:15		Afternoon slot:
–	L6:	Several rates compared with APC-models:
–		Estimation and reporting of effects.
–		Parametrization options for several rates.
–	P6:	Lung cancer differences by sex (`lung-sex`)

Thursday 26th

09:00 – 16:00	Study free: Working with the assignment.

Friday 27th

09:00 – 09:30		**Recap of Wednesday**
09:30 – 12:15		Morning slot:
–	L7:	Predictions based on APC models
–		Managing splines for prediction
–	P7:	Predicting lung cancer (`lung-pred`)
–		Predicting breast cancer (`breast-pred`)
13:15 – 15:15		Afternoon slot:
–	L8:	Overview of course:
–		What can the models be used for and what not.
–	P8:	Assignments.
15:15 – 15:30		Wrapping up, closure, evaluation and farewell

Chapter 1

Introduction to computing and practicals

1.1 Reading

It would be helpful if you had read the papers which cover the essentials of the models that we will cover: [4, 2, 3, 1]

1.2 Introduction to exercises

Most of the following exercises all require basic skills in computing, in R, in particular the use of the graphical facilities.

1.2.1 Datasets and how to access them.

All the datasets for the exercises in this section are in the folder `APC\data`. This can be accessed through the homepage of the course, as:
`http://www.biostat.ku.dk/~bxc/APC/data`.

The datasets with `.txt` extension are plain text files where variable names are found in the first line. Such datasets can be read into R with the command `read.table`, and into ***SAS*** by using in `%read_table` macro supplied in the `APC\sas\sasmacro` folder. This can be accessed through the homepage of the course, as:
`http://www.biostat.ku.dk/~bxc/APC/sas/sasmacro`.

1.2.2 R-functions

All the relevant functions for this course (and several more) are supplied in the R-package Epi, which can be downloaded from CRAN (on the R-website).

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

The latter command will list the package information, including names of all the functions available with brief descriptions.

1.2.3 Solutions

This document also contains some suggestions for solutions of teh assignments. They should *not* be taken as the *only* let alone exhuastive solutions to the practicals.

It is a good idea to give it a shot to do the practicals before you look in the solutions. However, the odd solution proposal may contain a twist to the analyses that you may find useful. Any suggestions for improving the solutions would be most welcome.

1.3 Probability concepts in follow-up studies

This section briefly summarizes relations between various quantities used in analysis of follow-up studies. They are used all the time in the analysis and reporting of results. Hence it is important to be familiar with all of them.

You should know them by heart to the extent that you can derive all of them from scratch:

- Survival function.

$$\begin{aligned} S(t) &= \mathrm{P}\left\{\text{survival at least till } t\right\} \\ &= \mathrm{P}\left\{T > t\right\} = 1 - \mathrm{P}\left\{T \leq t\right\} = 1 - F(t) \end{aligned}$$

$$\begin{aligned} S(t|t_\text{entry}) &= \mathrm{P}\left\{\text{survival at least till } t| \text{ alive at } t_\text{entry}\right\} \\ &= S(t)/S(t_\text{entry}) \end{aligned}$$

- Distribution function.

$$\begin{aligned} F(t) &= \mathrm{P}\left\{\text{death before } t\right\} \\ &= \mathrm{P}\left\{T \leq t\right\} = 1 - S(t) \end{aligned}$$

- Intensity:

$$\begin{aligned} \lambda(t) &= \lim_{h \to 0} \mathrm{P}\left\{\text{event in } (t, t+h] \mid \text{alive at } t\right\}/h \\ &= \lim_{h \to 0} \frac{F(t+h) - F(t)}{S(t)h} = \frac{f(t)}{S(t)} \\ &= \lim_{h \to 0} - \frac{S(t+h) - S(t)}{S(t)h} = - \frac{\mathrm{d} \log S(t)}{\mathrm{d}t} \end{aligned}$$

The intensity is also known as the hazard function, hazard rate, rate, mortality/morbidity rate.

- Relationships:

$$\begin{aligned} -\frac{\mathrm{d}\log S(t)}{\mathrm{d}t} &= \lambda(t) \\ &\Updownarrow \\ S(t) &= \exp\left(-\int_0^t \lambda(u)\,\mathrm{d}u\right) = \exp\bigl(-\Lambda(t)\bigr) \end{aligned}$$

$\Lambda(t) = \int_0^t \lambda(s)\,\mathrm{d}s$ is called the *integrated intensity* or the cumulative rate. It is *not* an intensity, it is dimensionless.

$$\lambda(t) = -\frac{\mathrm{d}\log(S(t))}{\mathrm{d}t} = -\frac{S'(t)}{S(t)} = \frac{F'(t)}{1 - F(t)} = \frac{f(t)}{S(t)}$$

- The cumulative *risk* of an event (to time t) is:

$$F(t) = \mathrm{P}\left\{\text{Event before time } t\right\} = \int_0^t \lambda(u)S(u)\,\mathrm{d}u = 1 - S(t) = 1 - \mathrm{e}^{-\Lambda(t)}$$

  For small $|x|$ (< 0.05), we have that $1 - \mathrm{e}^{-x} \approx x$, so for small values of the integrated intensity:

$$\text{Cumulative risk to time } t \approx \Lambda(t) = \text{Cumulative rate}$$

- Likelihood from one person:
  The likelihood from a number of small pieces of follow-up from one individual is a product of conditional probabilities:

$$\begin{aligned}\mathrm{P}\left\{\text{event at } t_4 | \text{entry at } t_0\right\} \quad = \quad & \mathrm{P}\left\{\text{event at } t_4 | \text{ alive at } t_3\right\} \times \\ & \mathrm{P}\left\{\text{survive } (t_2, t_3) | \text{ alive at } t_2\right\} \times \\ & \mathrm{P}\left\{\text{survive } (t_1, t_2) | \text{ alive at } t_1\right\} \times \\ & \mathrm{P}\left\{\text{survive } (t_0, t_1) | \text{ alive at } t_0\right\}\end{aligned}$$

  Each term in this expression corresponds to one *empirical rate*[1]
  $(d, y) = (\#\text{deaths}, \#\text{risk time})$, i.e. the data obtained from the follow-up opf one person in the interval of length y. Each person can contribute many empirical rates, most with $d = 0$; d can only be 1 for the *last* empirical rate for a person.

- Log-likelihood for an empirical rate (d, y):

$$\ell(\lambda) = d\log(\lambda) - \lambda y$$

  This is under the assumption the the underlying rate is assumed to be constant over the interval that the empirical rates refers to.

- Adding these over a group of persons (assuming identical and constant rates) gives:

$$D\log(\lambda) - \lambda Y,$$

  where Y is the total follow-up time, and D is the total number of failures.

  Note: The Poisson log-likelihood for an observation D with mean λY is:

$$D\log(\lambda Y) - \lambda Y = D\log(\lambda) + D\log(Y) - \lambda Y$$

  The term $D\log(Y)$ does not involve the parameter λ, so the likelihood for an observed rate can be maximized by pretending that the no. of cases D is Poisson with mean λY. But this does *not* imply that D follows a Poisson-distribution. It is entirely a likelihood based computational convenience. Anything that is not likelihood based is not justified.

- A linear model for the log-rate, $\log(\lambda) = X\beta$ implies
  $\lambda Y = \exp\big(\log(\lambda) + \log(Y)\big) = \exp\big(X\beta + \log(Y)\big)$. Therefore, in order to get a linear model we must require that $\log(Y)$ appear as a variable in the model for the log-rate with the regression coefficient fixed to 1, a so-called offset-term in the linear predictor.

[1] This is a concept coined by BxC, and so is not necessarily generally recognized.

- Competing risks:

  If there is more than one cause of death, occurring with (cause-specific) rates λ_1, λ_2, λ_3, the survival function is:

  $$S(t) = \exp\left(-\int_0^t \lambda_1(u) + \lambda_2(u) + \lambda_3(u)\,\mathrm{d}u\right)$$

  The probability of dying from cause 1 before time t is:

  $$\int_0^t \lambda_1(u)S(u)\,\mathrm{d}u \neq 1 - \exp\left(-\int_0^t \lambda_1(u)\,\mathrm{d}u\right)$$

  The second part of the term on the right hand side (sometimes referred to as the "cause-specific survival") does not have any probabilistic interpretation.

Bibliography

[1] B Carstensen. Age-Period-Cohort models for the Lexis diagram. *Statistics in Medicine*, 26(15):3018–3045, July 2007.

[2] D. Clayton and E. Schifflers. Models for temporal variation in cancer rates. I: Age-period and age-cohort models. *Statistics in Medicine*, 6:449–467, 1987.

[3] D. Clayton and E. Schifflers. Models for temporal variation in cancer rates. II: Age-period-cohort models. *Statistics in Medicine*, 6:469–481, 1987.

[4] TR Holford. The estimation of age, period and cohort effects for vital rates. *Biometrics*, 39:311–324, 1983.

Chapter 2

Practical exercises

2.1 Danish primeministers

The following table shows all Danish prime ministers in office since the war. They are ordered by the period in office, hence some appear twice. Entry end exit refer to the office of prime minister. A missing date of death means that the person was alive at the end of 2008.

Name	Birth	Death	Entry	Exit
Vilhelm Buhl	16/10/1881	18/12/1954	05/05/1945	07/11/1945
Knud Kristensen	26/10/1880	29/09/1962	07/11/1945	13/11/1947
Hans Hedtoft	21/04/1903	29/01/1955	13/11/1947	30/10/1950
Erik Eriksen	20/11/1902	07/10/1972	30/10/1950	30/09/1953
Hans Hedtoft	21/04/1903	29/01/1955	30/09/1953	29/01/1955
H C Hansen	08/11/1906	19/02/1960	01/02/1955	19/02/1960
Viggo Kampmann	21/07/1910	03/06/1976	21/02/1960	03/09/1962
Jens Otto Kragh	15/09/1914	22/06/1978	03/09/1962	02/02/1968
Hilmar Baunsgaard	26/02/1920	30/06/1989	02/02/1968	11/10/1971
Jens Otto Kragh	15/09/1914	22/06/1978	11/10/1971	05/10/1972
Anker Jorgensen	13/07/1922	.	05/10/1972	19/12/1973
Poul Hartling	14/08/1914	30/04/2000	19/12/1973	13/02/1975
Anker Jorgensen	13/07/1922	.	13/02/1975	10/09/1982
Poul Schlüter	03/04/1929	.	10/09/1982	25/01/1993
Poul Nyrup Rasmussen	15/06/1943	.	25/01/1993	27/11/2001
Anders Fogh Rasmussen	26/01/1953	.	27/11/2001	31/12/2007

The data in the table can be fould in the file `pm-dk.txt`.

```
> st <- read.table( "../data/pm-dk.txt", header=T, as.is=T,
+                   na.strings="." )
> st
> str( st )
```

1. Draw a Lexis diagram with life-lines of the persons.

   ```
   > # Change the character variables with dates to fractional calendar
   > # years
   > for( i in 2:5 ) st <- cal.yr( st, format="%d/%m/%Y" )
   > # Attach the data for those still alive
   > st$fail <- !is.na(st$death)
   > st[!st$fail,"death"] <- 2009
   ```

```
> st
> attach( st )
> # Lexis object
> L <- Lexis( entry = list(per=birth),
+             exit = list(per=death, age=death-birth),
+             exit.status=fail,
+             data=st )
> # Plot Lexis diagram
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6, xaxt="n" ) # Omit x-labels
> plot( L, xlim=c(1945,2010), ylim=c(32,88), lwd=3, las=1,
+       grid=0:20*5, col="black", xlab = "Calendar time", ylab="Age" )
> points( L, pch=c(NA,16)[L$lex.Xst+1] )
> #put names of the prime ministers on the plot
> text( death, death-birth, Name, adj=c(1.05,-0.05), cex=0.7 )
> par( xaxt="s" )
> axis( side=1, at=seq(1950,2010,10) ) # x-labels at nice places
```

2. Mark with a different color the periods where they have been in office.

```
> # New Lexis object describing periods in an office
> # and lines added to a picture
> in_office <- c( rep(FALSE,nrow(st)-1),TRUE )
> st <- cbind( st, in_office )
> Lo <- Lexis( entry = list(per=entry),
+              exit = list(per=exit, age=exit-birth),
+              exit.status=in_office,
+              data = st )
> lines( Lo, lwd=3, las=1, col="red" )
> # the same may be plotted using command segments
> box()
> segments( birth, 0, death, death-birth, lwd=2 )
> segments( entry, entry-birth, exit, exit-birth, lwd=4, col="red" )
```

3. Draw the line representing age 50 years.

```
> abline( h=50 )
```

4. How many 50th birthdays have been celebrated in office since the war?

```
> age_entry <- Lo$age
> age_exit <- Lo$age + Lo$lex.dur
> n_birthday <- sum( ( age_entry<50 ) & ( age_exit>50 ) )
> n_birthday
```

5. Draw the line representing 2 October 1972. (Why just that?)

```
> abline( v=cal.yr( "2/10/1972", format="%d/%m/%Y" ) )
```

6. How many present and former prime ministes were alive at 31st December 2008?

```
> alive <- (L$death >=2004)
> n_alive <- sum( alive )
> n_alive
> #Anker Jorgensen - 1 person has got 2 lex.id's
> levels( as.factor( subset( L$Name, alive==T ) ) )
```

7. Which period(s) since the war has seen the maximal number of former post-war prime ministers alive?

```
> # New lexis object - since entry to the office to the death
> Ln <- Lexis( entry=list(per=entry), exit=list(per=death,age=death-entry),
+       exit.status=fail, data=st )
```

```
> ny <- 2008-1945
> n_alive <- vector( "numeric", ny )
> for (i in 1:ny)
+ {
+ alive <- ( (Ln$death >=(1944+i))&(Ln$entry<=(1944+i)) )
+ n_alive[i] <- nlevels( as.factor( subset( Ln$Name, alive==T ) ) )
+ }
> plot( n_alive~seq(1945,(1945+ny-1),1), type="l", xlab="Calendar year",
+       ylab = "Maximal numbers of former prime ministers alive" )
```

8. Mark the area in the diagram with person years lived by persons aged 50 to 70 in the period 1 January 1970 through 1 January 1990.

```
> rect( 1970, 50, 1990, 70, lwd=2, border="green",col="lightgreen"  )
```

9. Mark the area for the lifetime experience of those who were between 10 and 20 years old in 1945.

```
> polygon( c(1955,2010,2010,1965,1955), c(30,85,75,30,30), lwd=2,
+      border="blue", col="lightblue" )
```

10. How many prime-minister-years have been spent time in each of these sets? And in the intersection of them?

```
> # Prime-minister years lived by persons
> #  aged 50 to 70 in the period 1 January 1970 through 1 January 1990.
> x1 <- splitLexis( Lo ,breaks = c(0,50,70,100), time.scale="age" )
> x2 <- splitLexis( x1, breaks = c(1900,1970,1990,2010), time.scale="per" )
> summary( x2 )
> tapply( status(x2,"exit")==1, list( timeBand(x2,"age","left"),
+                                     timeBand(x2,"per","left") ), sum )
> tapply( dur(x2),  list( timeBand(x2,"age","left"),
+                         timeBand(x2,"per","left") ), sum )
> # Computing the person-years in the 1925-35 cohort
> x3 <- subset( Lo, birth>1925 & birth<=1935 )
> summary( x3 )
> dur( x3 )
> # Computing person years in the intersection
> x4 <- subset( x2 , birth>1925 & birth<=1935 )
> summary( x4 )
> dur( x4 )
```

2.2 Reading and tabulating data

The following exercise is aimed at tabulating and displaying the data typically involved in age-period-cohort analysis.

1. Read the data in the file `lung5-M.txt`, and print the data. What does each line refer to?

   ```
   > lung <- read.table( "../data/lung5-M.txt", header=T )
   > lung
   > head(lung)

   > attach( lung )
   ```

2. Print the no. cases in a nice tabular form, and likewise with the person-years. Is there someything special about the last period?

   ```
   > D_table_nice <- stat.table( index=list(A,P), sum(D), data=lung, margin=T )
   > print( D_table_nice, digits=c(sum=0) )
   > Y_table_nice <- stat.table( index=list(A,P), sum(Y), data=lung, margin=T )
   > print( Y_table_nice, digits=c(sum=2) )
   ```

3. Compute the empirical rates, and print them in a table too.

   ```
   > R_table_nice <- stat.table( index=list(A,P), list(Rate=ratio(D,Y,100000)),
   +                             data=lung, margin=T )
   > print( R_table_nice, digits=c(sum=2) )
   > # another way of computation - not using Epi library
   > D_table <- tapply( D, list(A,P), sum )
   > Y_table <- tapply( Y, list(A,P), sum )
   > R_table <- D_table/Y_table*(10^5)
   ```

4. Make the four classical graphs of the data. Consider whether a log-scale for the y-axis is appropriate. Think about where on the x-axis each age-class is located.

   (a) Age-specific rates for each period. (Rates from the same period connected).

   ```
   > rateplot( R_table, which=c("AP"), ann=TRUE )
   ```

   (b) Age-specific rates for each cohort. (Rates from the same cohort connected).

   ```
   > rateplot( R_table, which=c("AC"), ann=TRUE )
   ```

   (c) Rates for each age-class versus period. (Rates from the same age-class connected).

   ```
   > rateplot( R_table, which=c("PA"), ann=TRUE )
   ```

   (d) Rates for each age-class versus cohort. (Rates from the same age-class connected).

   ```
   > rateplot( R_table, which=c("CA"), ann=TRUE )
   ```

5. How would each of these curves look if:

   (a) age-specific rates did not change at all by time?

   ```
   > # age-specific rates remain still the same as in period 1943
   > source( "rateplot2.R" )
   > R_table_no_change<-matrix( R_table[,1], dim(R_table)[1], dim(R_table)[2] )
   > colnames( R_table_no_change ) <- colnames( R_table )
   > rownames( R_table_no_change ) <- rownames( R_table )
   > R_table_no_change

   > par( mfrow=c(2,2) )
   > rateplot2( R_table_no_change)
   ```

   (b) age-specific rates were only influenced by period?

```
> #age-specific rates are only influence by period
> step <- 2
> change_p <- matrix(rep(seq(1,11*step,step),10),10,11,byrow=T)
> change_p
> R_table_p <- R_table_no_change+change_p
> colnames( R_table_p ) <- colnames( R_table )
> rownames( R_table_p ) <- rownames( R_table )
> R_table_p

> par( mfrow=c(2,2) )
> rateplot2( (R_table_p) )
```

(c) age-specific rates were only influenced by cohort?

```
> #age-specific rates are only influence by cohort
> nr <- nrow(R_table)
> nc <- 10
> p <- c( rep(NA,nc), R_table[,1] )
> np <- length( p )
> R_table_c <- cbind( p[(np-nr+1):np], p[(np-nr):(np-1)],
+                     p[(np-nr-1):(np-2)],p[(np-nr-2):(np-3)],
+                     p[(np-nr-3):(np-4)], p[(np-nr-4):(np-5)],
+                     p[(np-nr-5):(np-6)], p[(np-nr-6):(np-7)],
+                     p[(np-nr-7):(np-8)], p[(np-nr-8):(np-9)],
+                     p[(np-nr-9):(np-10)] )
> colnames( R_table_c ) <- colnames( R_table )
> rownames( R_table_c ) <- rownames( R_table )
> R_table_c

> par( mfrow=c(2,2) )
> rateplot2( R_table_c )
```

2.3 Rates and survival

1. Consider the following data:

Year of birth	Year of death		Age at death
	1994	1995	
1994	2,900	500	0
1993	120	130	1
1992	50	60	2
1991	45	55	3
1990	40	40	4

(a) Represent these data in a Lexis diagram.

```
> # Enter the data from the table into a matrix
> D <- matrix( c(2900,120,50,45,40,500,130,60,55,40), 5, 2 )
> D

> # Make a Lexis diagram and represent the numbers there
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> Lexis.diagram( age=c(0,5), date=c(1991,1996), int=1, lab.int=1,
+           coh.grid=T )
> box()
> text( 1994+rep( c(2,4)/3, c(5,5) ),c(0:4+1/3,0:4+2/3), paste( D ) )
```

(b) On the basis of these data, can you calculate the age-specific death rate for two-year-olds (${}_1m_2$) in 1994? If you can, do it. If you cannot, explain what additional information you would need.

(c) On the basis of these data, can you calculate the probability of surviving from age 2 to age 3 (${}_1q_2$) in for the cohort born in 1992?

If you can, do it. If you cannot, explain what additional information you would need.

2. Consider the following data:

- Live births during 1991: 142,000
- Number of infants born in 1991 who did not survive until the end of 1991: 2,900
- Number of infants born in 1991 who survived to the end of 1991, but did not reach their first birthday: 500
- Live births during 1992: 138,000
- Number of infants born in 1992 who did not survive until the end of 1992: 2,600
- Number of infants born in 1992 who survived to the end of 1992, but did not reach their first birthday: 450

(a) Represent the data on a Lexis diagram.

```
> # Enter the information in two data structures
> B <- c(142, 138)*1000
> D <- c(2900, 500, 2600, 450)

> # Make a Lexis diagram and represent the numbers there
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> Lexis.diagram( age=c(0,5), date=c(1991,1996), int=1, lab.int=1,
+           coh.grid=T )
> text( 1991+c(2,4,5,7)/3, c(1,2,1,2)/3, paste( D ) )
> text( 1991.5+0:1, 0, paste( B ), adj=c(0.5,-0.2), col="red" )
```

(b) Calculate the infant mortality rate (IMR) for 1992 under the assumption that you were only able to observe events occurring in 1992, and that you did not know the birth dates of infants dying during that year.

(c) Same as above, except that now you do know the birth dates of infants dying during 1992.

(d) Assume all data are known: Calculate the IMR.

(e) What is the IMR for the 1992 birth cohort?

2.4 Age-period model

The following exercise is aimed at familiarizing you with the parametrization of the age-period model. It will give you the opportunity explore how to extract and and plot parameter estimates from models. It is based on Danish male lung cancer incidence data in 5-year classes.

1. Read the data in the file `lung5-M.txt` as in the tabulation exercise:

   ```
   > lung <- read.table( "../data/lung5-M.txt", header=T )
   > lung
   > attach( lung )
   > table( A )
   > table( P )
   > tapply( Y, list(A,P), sum )
   ```

   What do these tables show?

2. Fit a Poisson model with effects of age (A) and period (P) as class variables:

   ```
   > ap.1 <- glm( D ~ factor(A) + factor(P) + offset(log(Y)), family=poisson )
   > summary( ap.1 )
   ```

   What do the parameters refer to, i.e. which ones are log-rates and which ones are rate-ratios?

3. Fit the same model without intercept (use `-1` in the model formula); call it `ap.0` — we shall refer to this subsequently. What do the parameters now refer to?

4. Fit the same model, using the period 1968–72 as the reference period, by using the `relevel` command for factors to make 1968 the first level:

   ```
   > ap.3 <- glm( D ~ factor(A) - 1 + relevel(factor(P),"1968") + offset(log(Y)),
   +                  family=poisson )
   ```

5. Extract the prameters from the model, by doing:

   ```
   > ap.cf <- summary( ap.3 )$coef
   ```

6. Now plot the estimated age-specific incidence rates, remembering to annoatte them with the correct scale. We need the first 10 parameters, with their standard errors:

   ```
   > age.cf <- ap.cf[1:10,1:2]
   ```

   This means that we take rows 1–10 and columns 1–2. The corresponding age classes are $40,\ldots,85$. The midpoints of these age-classes are 2.5 years higher. The ages can be generated in R by saying `seq(40,85,5)+2.5`.

   Now put confidence limits on the curves by taking $\pm 1.96 \times$ s.e.. The line of the estimates can be over-drawn once more in a thicker style:

   ```
   > lines( seq(40,85,5)+2.5, exp(age.cf[,1]), lwd=3 )
   ```

7. Now for the rate-ratio-parameters, take the rest of the coefficients:

   ```
   > RR.cf <- ap.cf[11:20,1:2]
   ```

   But the reference group is missing, so we must stick two 0s in the correct place. We use the command `rbind` (row-bind):

   ```
   > RR.cf <- rbind( RR.cf[1:5,], c(0,0), RR.cf[6:10,] )
   ```

Now we have the same situation as for the age-specific rates, and can plot the relative risks (relative to 1968) in precisely the same way as for the agespecific rates.

Make a line-plot of the relative risks with confidence intervals.

8. However, the relevant parameters may also be extracted directly from the model without intercept, using the function `ci.lin` (remember to read the documentation for this!)

   The point is to define a *contrast matrix*, which multiplied to (a subset of) the parameters gives the rates in the reference period. The log-rates in the reference period (the first level of `factor(P)` are the age-parameters. The log-rates in the period labelled `1968` are these *plus* the period estimate from `1968`.

   Now construct the following matrix and look at it:

   ```
   > cm.A <- cbind( diag( nlevels( factor(A) ) ), 1 )
   ```

   Now look at the parameters extracted by `ci.lin`, using the `subset=` argument:

   ```
   > ci.lin( ap.0, subset=c("A","1968") )
   ```

   Now use the argument `ctr.mat=` in `ci.lin` to produce the rates in period 1968 and plot them on a log-scale.

9. Save the estimates of age aned period effects along with the age-points and period-points, using `save` (look up the help page if you are not familiar with it. You will need these in the next exercise on the age-cohort model.

10. We can also use the same machinery to extract the rate-ratios relative to 1968. The contrast matrix to use is the difference between two: The first one is the one that extracts the rate-ratios with a prefixed 0:

    ```
    > cm.P <- rbind(0,diag( nlevels(factor(P))-1 ) )
    > cm.P
    > ci.lin( ap.0, subset="P", ctr.mat=cm.P )
    ```

    In order to subtract the value corresponding to `1968`, we must subtract a 11 × 10 matrix, that just selects the `1968` column:

    ```
    > cm.Pref <- cm.P * 0
    > cm.Pref[,5] <- 1
    > cm.Pref
    ```

    The contrast matrix to use is the difference between these two:

    ```
    > cm.P - cm.Pref
    > ci.lin( ap.0, subset="P", ctr.mat=cm.P-cm.Pref )
    ```

    Use the `Exp=TRUE` argument to get the rate-ratios and plot these with confidence intervals on a log-scale.

11. *For the* **real** *nerds:* Plot the rates and the rate ratios beside each other, and make sure that the physical extent of the units on both the x-axis and the y-axis are the same.

    Hint: You may want to use `par(mar=c(0,0,0,0), oma=)`, the function `layout` as well as the `xaxs="i"` argument to `plot`.

2.5 Age-cohort model

This exercise is aimed at familiarizing you with the parametrization of the age-cohort model. It will give you the opportunity explore how to extract and and plot parameter estimates from models. It is parallel to the exercise on the age-period model and is therefor less detailed.

1. Read the data in the file `lung5-M.txt` as in the tabulation exercise:

   ```
   > lung <- read.table( "../data/lung5-M.txt", header=T )
   > lung
   > attach( lung )
   > table( A )
   > table( P )
   > table( P-A )
   ```

   What do these tables show?

2. Fit a Poisson model with effects of age (A) and cohort (C) as class variables. You will need to form the variable C (cohort) as P−A first.

   What do the parameters refer to ?

3. Fit the same model without intercept. What do the parameters now refer to ?

   (Use `-1` in the model formula.)

4. Fit the same model, using the cohort 1908 as the reference cohort. What do the parameters represent now?

   (Use the `relevel` command for factors to make 1968 the first level.)

5. What is the range of birth dates represented in the cohort 1908?

6. Extract the age-specific incidence parameters from the model and plot then against age. Remember to annotate them with the correct units. Add 95% confidence intervals.

7. Extract the cohort-specific rate-ratio parameters and plot then against the date of birth (cohort). Add 95% confidence intervals.

8. Now load the estimates from the age-period model, and plot the estimated age-specific rates from the two models on top of each other.

   Why are they different; in particular, why do they have different slopes?

2.6 Age-drift model

This exercise is aimed at introducing the age-drift model and make you familiar with the two different ways of parametrizing this model. Like the two previous exercises it is based on the male lung cancer data.

1. First read the data in the file `lung5-M.txt` and create the cohort variable:

   ```
   > lung <- read.table( "../data/lung5-M.txt", header=T )
   > lung$C <- lung$P - lung$A
   ```

2. Fit a Poisson model with effects of age as class variable and period P as continuous variable.

   What do the parameters refer to ?

3. Fit the same model without intercept. What do the parameters now refer to?

4. Fit the same model, using the period 1968–72 as the reference period.

   Now what do the parameters represent?

5. Fit a model with cohort as a continuous variable, using 1908 as the reference, and without intercept. What do the resulting parameters represent?

6. Compare the deviances and the slope estimates from the models with cohort drift and period drift.

7. What is the relationship between the estimated age-effects in the two models?

   Verify this empirically by converting one set of age-parameters to the other.

8. Plot the age-specific incidence rates from the two different models in the same panel.

9. The rates from the model are:

   $$\log(\lambda_{ap}) = \alpha_p + \delta(p - 1970.5)$$

   Therefore, with an x-variable: (1943,...,1993) + 2.5, the log rate ratio relative to 1970.5 will be:

   $$\log \mathrm{RR} = \hat{\delta} \times x$$

   and the upper and lower confidence bands:

   $$\log \mathrm{RR} = (\hat{\delta} \pm 1.96 \times \text{s.e.}(\delta)) \times x$$

   Now extract the slope parameter, and plot the rate-ratio functions as a function of period.

2.7 Age-period-cohort model

The following exercise is aimed at familiarizing you with the parametrization of the age-period-cohort model and with the realtionship of the APC-model to the other model that you have been working with, so we will refer back to those, and assume that you have the results from them at hand.

1. Read the data in the file `lung5-M.txt` as in the tabulation exercise:

   ```
   > lung <- read.table( "../data/lung5-M.txt", header=T )
   > lung
   > attach( lung )
   ```

2. Fit a Poisson model with effects of age (A), period (P) and cohort (C) as class variables. Also fit a model with age alone as a class variable. Write down a scheme showing the deviances and degrees of freedom for the 5 models you have models fitted to this dataset.

3. Compare the models that can be compared, with likelihood-ratio tetsts. You will want to use `anova` (or specifically `anova.glm`) with the argument `test="Chisq"`.

4. Next, fit the same model without intercept, and with the first and last period parameters and the 1908 cohort parameter set to 0. Before you do so a few practical things must be fixed:

   You can merge the first and the last period level using the `Relevel` function (look at the documentation for it).

   ```
   >   lung$Pr <- Relevel( factor(lung$P), list("first-last"=c("1943","1993") ) )
   ```

   You can also use this function to make the 1908 cohort the first level of the cohort factor:

   ```
   >   lung$Cr <- Relevel( factor(lung$P-lung$A), "1908" )
   ```

   It is a good idea to tabulate the new factor against the old one (i.e. that variable from which it was created) in order to meake sure that the relevelling actually is as you intended it to be.

5. Now you can fit the model, using the factors you just defined. What do the parameters now refer to?

6. Make a graph of the parameters. Remember to take the exponential to convert the age-parameters to rates (and find out what the units are) and the period and cohort parameters to rate ratios. Also use a log-scale for the y-axis. You may want to use `ci.lin` to facilitate this.

7. Fit the same model, using the period 1968–72 as the reference period and two cohorts of your choice as references. To decide which of the cohorts to alias it may be useful to see how many observations there are in each:

   ```
   > with( lung, table(P-A) )
   > with( lung, tapply(D,list(P-A),sum) )
   ```

   Having fitted the model, now what do the parameters in it represent?

8. Make a plot of these parameters.

   Add the parameters from the previous parametrization to the same graph.

2.8 Age-period-cohort model for triangles

The following exercise is aimed at showing the problems associated with age-period-cohort modelling for triangular data.

Also you will learn how to overcome these problems by parametric modelling of the three effects.

1. Read the Danish male lung cancer data tabulated by age period *and* birth cohort, `lung5-Mc.txt`. List the first few lines of the dataset and make sure you understand what the variables refer to. Also define nthe synthetic cohorts as `P5-A5`:

   ```
   > ltri <- read.table( "../data/lung5-Mc.txt", header=T )
   > ltri$S5 <- ltri$P5 - ltri$A5
   > attach( ltri )
   ```

2. Make a Lexis diagram showing the subdivision of the follow-data. You will explore the function `Lexis.diagram`.

   ```
   > Lexis.diagram( age=c(40,90), date=c(1943,1998), coh.grid=TRUE )
   ```

3. Use the variables `A5` and `P5` to fit a traditional age-period-cohort model with synthetic cohort defined by `P5-A5`:

   ```
   > ms <- glm( D ~ -1 + factor(A5) + factor(P5) + factor(S5) + offset(log(Y)),
   +            family=poisson, data=ltri )
   > summary( ms )$df

   [1]  38 182  39
   ```

   How many parameters does this model have?

4. Now try to fit the model with the "real" cohort:

   ```
   > mc <- glm( D ~ -1 + factor(A5) + factor(P5) + factor(C5) + offset(log(Y)),
   +            family=poisson, data=ltri )
   > summary( mc )$df

   [1]  40 180  40
   ```

   How many parameters does this model have?

5. Plot the parameter estimates from the two models on top of each other, with confidence intervals. Remember to put the rigt scales on the plots.

   ```
   > par( mfrow=c(1,3) )
   > a.pt <- as.numeric( levels(factor(A5)) )
   > p.pt <- as.numeric( levels(factor(P5)) )
   > s.pt <- as.numeric( levels(factor(S5)) )
   > c.pt <- as.numeric( levels(factor(C5)) )
   > matplot( a.pt, ci.lin( ms, subset="A5", Exp=TRUE )[,5:7]/10^5,
   +          type="l", lty=1, lwd=c(3,1,1), col="black",
   +          xlab="Age", ylab="Rates", log="y" )
   > matlines( a.pt, ci.lin( mc, subset="A5", Exp=TRUE )[,5:7]/10^5,
   +           type="l", lty=1, lwd=c(3,1,1), col="blue" )
   > matplot( p.pt, rbind( c(1,1,1), ci.lin( ms, subset="P5",Exp=TRUE )[,5:7] ),
   +          type="l", lty=1, lwd=c(3,1,1), col="black",
   +          xlab="Period", ylab="RR", log="y" )
   > matlines( p.pt, rbind( c(1,1,1), ci.lin( mc, subset="P5",Exp=TRUE )[,5:7] ),
   +           type="l", lty=1, lwd=c(3,1,1), col="blue" )
   > matplot( s.pt, rbind(c(1,1,1),ci.lin( ms, subset="S5", Exp=TRUE )[,5:7]),
   ```

```
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          xlab="Cohort", ylab="RR", log="y" )
> matlines( c.pt, rbind(c(1,1,1),ci.lin( mc, subset="C5", Exp=TRUE )[,5:7]),
+           type="l", lty=1, lwd=c(3,1,1), col="blue" )
```

It is seen that the confidence bands are much wider for the age and cohort effects but narrower for the period effects.

6. Now fit the model using the proper midpoints of the triangles as factor levels. How many parameters does this model have?

```
> mt <- glm( D ~ -1 + factor(Ax) + factor(Px) + factor(Cx) + offset(log(Y)),
+            family=poisson, data=ltri )
> summary( mt )$df

[1]  76 144  80
```

7. Plot the parameters from this model in three panels as for the previous two models.

```
> par( mfrow=c(1,3) )
> a.pt <- as.numeric( levels(factor(Ax)) )
> p.pt <- as.numeric( levels(factor(Px)) )
> c.pt <- as.numeric( levels(factor(Cx)) )
> matplot( a.pt, ci.lin( mt, subset="Ax", Exp=TRUE )[,5:7]/10^5,
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          xlab="Age", ylab="Rates", log="y" )
> matplot( p.pt, rbind( c(1,1,1), ci.lin( mt, subset="Px",Exp=TRUE )[,5:7] ),
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          xlab="Period", ylab="RR", log="y" )
> matplot( c.pt, rbind(c(1,1,1),ci.lin( mt, subset="Cx", Exp=TRUE )[,5:7]),
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          xlab="Cohort", ylab="RR", log="y" )
```

We see that the parameters clearly do not convey a reasonable picture of the effects; som severe indeterminacy has crept in.

8. What is the residual deviance of this model?

```
> summary( mt )$deviance

[1] 284.7269
```

9. The dataset also has a variable up, which indicates whether the observation comes from an upper or lower triangle. Try to tabulate it against P5-A5-C5.

```
> table( up, P5-A5-S5 )

up    0
  0 110
  1 110
```

10. Fit an age-period cohort model separately for the subset of the dataset from the upper triangles and from the lowere triangles. What is the residual deviance from each of these models and what is the sum of these. Compare to the model using the proper midpoints as factor levels.

```
> m.up <- glm( D ~ -1 + factor(A5) + factor(P5) + factor(S5) + offset(log(Y)),
+              family=poisson, data=subset(ltri,up==1) )
> summary( m.up )$deviance
```

```
[1] 150.2703

> m.lo <- glm( D ~ -1 + factor(A5) + factor(P5) + factor(S5) + offset(log(Y)),
+              family=poisson, data=subset(ltri,up==0) )
> summary( m.lo )$deviance

[1] 134.4566

> summary( m.lo )$deviance + summary( m.up )$deviance

[1] 284.7269

> summary( mt )$deviance

[1] 284.7269
```

11. Next, repeat the plots of the parameters from the model using the proper midpoints as factor levels, but now super-posing the estimates (in different color) from each of the two models just fitted. What goes on?

```
> par( mfrow=c(1,3) )
> a.pt <- as.numeric( levels(factor(Ax)) )
> p.pt <- as.numeric( levels(factor(Px)) )
> c.pt <- as.numeric( levels(factor(Cx)) )
> a5.pt <- as.numeric( levels(factor(A5)) )
> p5.pt <- as.numeric( levels(factor(P5)) )
> s5.pt <- as.numeric( levels(factor(S5)) )
> matplot( a.pt, ci.lin( mt, subset="Ax", Exp=TRUE )[,5:7]/10^5,
+          type="l", lty=1, lwd=c(2,1,1), col=gray(0.7),
+          xlab="Age", ylab="Rates", log="y" )
> matpoints( a5.pt, ci.lin( m.up, subset="A5", Exp=TRUE )[,5:7]/10^5,
+            pch=c(16,3,3), col="blue" )
> matpoints( a5.pt, ci.lin( m.lo, subset="A5", Exp=TRUE )[,5:7]/10^5,
+            pch=c(16,3,3), col="red" )
> matplot( p.pt, rbind( c(1,1,1), ci.lin( mt, subset="Px",Exp=TRUE )[,5:7] ),
+          type="l", lty=1, lwd=c(2,1,1), col=gray(0.7),
+          xlab="Period", ylab="RR", log="y" )
> matpoints( p5.pt[-1], ci.lin( m.up, subset="P5", Exp=TRUE )[,5:7],
+            pch=c(16,3,3), col="blue" )
> matpoints( p5.pt[-1], ci.lin( m.lo, subset="P5", Exp=TRUE )[,5:7],
+            pch=c(16,3,3), col="red" )
> matplot( c.pt, rbind(c(1,1,1),ci.lin( mt, subset="Cx", Exp=TRUE )[,5:7]),
+          type="l", lty=1, lwd=c(2,1,1), col=gray(0.7),
+          xlab="Cohort", ylab="RR", log="y" )
> matpoints( s5.pt[-1], ci.lin( m.up, subset="S5", Exp=TRUE )[,5:7],
+            pch=c(16,3,3), col="blue" )
> matpoints( s5.pt[-1], ci.lin( m.lo, subset="S5", Exp=TRUE )[,5:7],
+            pch=c(16,3,3), col="red" )
```

The model fitted with the “correct” factor levels is actually two different models. This is because observations in upper triangles are modelled by one set of the parameters, and those in lower triangel by another set of parameters.

Because of the ordering of the levels, the parametrization is different, but that is all.

There is no way out of the squeeze, except by resorting to parametric models for the actual underlying scales, abandoning the factor modelling.

12. Load the splines package and fit a model using the correct midpoints of the triangles as quantitative variables in restricted cubic splines, using the function `ns`:

```
> library( splines )
> mspl <- glm( D ~ -1 + ns(Ax,df=7,intercept=T)
+                     + ns(Px,df=6,intercept=F)
+                     + ns(Cx,df=6,intercept=F) + offset(log(Y)),
+               family=poisson, data=ltri )
> summary( mspl )

Call:
glm(formula = D ~ -1 + ns(Ax, df = 7, intercept = T) + ns(Px,
    df = 6, intercept = F) + ns(Cx, df = 6, intercept = F) +
    offset(log(Y)), family = poisson, data = ltri)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-3.72761  -0.88692  -0.01217   0.93283   3.47380

Coefficients: (1 not defined because of singularities)
                                Estimate Std. Error z value Pr(>|z|)
ns(Ax, df = 7, intercept = T)1  -8.08248    0.09584 -84.329  < 2e-16
ns(Ax, df = 7, intercept = T)2  -8.81421    0.11261 -78.271  < 2e-16
ns(Ax, df = 7, intercept = T)3  -8.20301    0.11520 -71.209  < 2e-16
ns(Ax, df = 7, intercept = T)4  -7.90599    0.11814 -66.921  < 2e-16
ns(Ax, df = 7, intercept = T)5  -3.98298    0.08558 -46.540  < 2e-16
ns(Ax, df = 7, intercept = T)6 -21.35542    0.24841 -85.967  < 2e-16
ns(Ax, df = 7, intercept = T)7   0.70588    0.05540  12.741  < 2e-16
ns(Px, df = 6, intercept = F)1   0.59989    0.03777  15.883  < 2e-16
ns(Px, df = 6, intercept = F)2   0.94029    0.04319  21.771  < 2e-16
ns(Px, df = 6, intercept = F)3   1.18582    0.04354  27.237  < 2e-16
ns(Px, df = 6, intercept = F)4   1.22421    0.04204  29.122  < 2e-16
ns(Px, df = 6, intercept = F)5   1.46929    0.08247  17.816  < 2e-16
ns(Px, df = 6, intercept = F)6   1.07376    0.04202  25.555  < 2e-16
ns(Cx, df = 6, intercept = F)1   1.57834    0.10334  15.273  < 2e-16
ns(Cx, df = 6, intercept = F)2   1.60219    0.11202  14.303  < 2e-16
ns(Cx, df = 6, intercept = F)3   1.37407    0.10178  13.500  < 2e-16
ns(Cx, df = 6, intercept = F)4   1.03167    0.07211  14.306  < 2e-16
ns(Cx, df = 6, intercept = F)5   1.19310    0.21716   5.494 3.93e-08
ns(Cx, df = 6, intercept = F)6        NA         NA      NA       NA

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1.0037e+08  on 220  degrees of freedom
Residual deviance: 4.3344e+02  on 202  degrees of freedom
AIC: 2026.7

Number of Fisher Scoring iterations: 4

> summary( mt )$deviance - summary( mspl )$deviance

[1] -148.7082

> summary( mt )$df       - summary( mspl )$df

[1]  58 -58  61
```

13. How do the deviances compare?

14. Make a prediction of the terms, using `predict.glm` using the argument `type="terms"`.

2.9 Using apc.fit etc.

This exercise is aimed at introducing the functions for fitting and plotting the results from age-period-cohort models: apc.fit apc.plot apc.lines and apc.frame.

1. Read the testis cancer data and collapse the cases over the histological subtypes:

   ```
   > th <- read.table( "../data/testis-hist.txt", header=T )
   > str( th )
   ```

   Knowing the names of the variables in the dataset, you collapse over the histological subtypes. You may want to use the function aggregate; note that there is no need to tabulate by cohort, because even for the triangular data the relationship $c = p - a$ holds.

   Note that the original data had three subtypes of testis cancer, so while it is OK to sum the number of cases (D), risk time should not be aggregated across histological subtypes — this is aggregation *within* subsets of the Lexis diagram.

2. Present the rates in 5-year age and period classes from age 15 to age 59 using rateplot. Consider the function subset. To this end you must make a table, for example using something like:

   ```
   > with( tc, tapply( D, list(floor(A/5)*5+2.5,
   +                           floor((P-1943)/5)*5+1945.5), sum ) )
   ```

   — assuming your aggregated data is in the data frame tc. and a similar construction for the risk time.

3. Fit an age-period-cohort model to the data using the machinery implemented in apc.fit. The function returns a fitted model *and* a parametrization, hence you must choose how to parametrize it, in this case "ACP" with all the drift included in the cohort effect and the reference cohort being 1918.

   ```
   > tapc <- apc.fit( subset( tc, A>15 & A<60 ), npar=c(10,10,10), parm="ACP", ref.c=1918 )
   ```

   Can any of the effects be omitted from the model?

4. Plot the estimates using the apc.plot function:

   ```
   > apc.plot( tapc, ci=TRUE )
   ```

5. Now explore in more depth the cohort effect by increasing the number of parameters used for it:

   ```
   > tapc <- apc.fit( subset( tc, A>15 & A<60 ), npar=c(10,10,20),
   +                  parm="ACP", ref.c=1918, scale=10^5 )
   > fp <- apc.plot( tapc, ci=TRUE )
   ```

   Do the extra parameters for the cohort effect have any influence on the model fit?

6. Explore the effect of using the residual method instead, and over-plot the estimates from this method on the existing plot[1]:

[1] Unfortunately there is a fatal bug in apc.fit when fitting the period residuals to the age-cohort model — it does not crash but simply fit a totally meaningless model. There is a fix for this in the version 1.0.11 of the Epi package which is available at the course homepage

7. The standard display is not very pretty — it gives an overview, but certainly not anything worth publishing, hence a bit of handwork is needed. Use the `apc.frame` for this, and create a nicer plot of the estimates from the residual model. You may not agree with all the parameters suggested here:

```
> par( mar=c(3,4,1,4), mgp=c(3,1,0)/1.7, las=1 )
> fp <- apc.frame( a.lab=seq(20,60,10),
+                  a.tic=seq(10,60,5),
+                 cp.lab=seq(1900,2000,20),
+                 cp.tic=seq(1885,2000,5),
+                  r.lab=c(c(1,2,5)/10,1,2,5,10),
+                  r.tic=c(1:9/10,1:10),
+                    gap=8,
+                 rr.ref=1)
> apc.lines( tapc, ci=TRUE, col="blue", frame.par=fp )
> apc.lines( tac.p, ci=TRUE, col="red", frame.par=fp )
```

8. Try to repeat the exercise using period as the primary timescale, and add this to the plot as well.

   What is revealed by looking at the data this way?

2.10 Lung cancer: the sex difference

The purpose of this exercise to analyse lung cancer incidence rates in Danish men and women and make comparisons of the effects between the two.

1. Read the lung cancer dataset from the

   ```
   > lung <- read.table("../data/apc-Lung.txt", header=T )
   > str( lung )
   > summary( lung )
   ```

   These data are tabulated by sex, age, period and cohort in 1-year classes, i.e. each observation corresponds to a triangle in the Lexis diagram.

2. The variables `A`, `P` and `C` are the left endpoints of the tabulation intervals. In order to be able to properly analyse data, compute the correct midpoints for each of the triangles.

3. Produce a suitable overview of the rates using the `rateplot` on suitably grouped rates. Make the plots separately for men and women.

4. Fit an age-period-cohort model for male and female rates separately. Plot them in separate displays using `apc.plot`. Use `apc.frame` to set up a display that will accomodate plotting of both sets of estimates.

5. Can you find a way of estimating the ratios of rates and the ratios of RRs between the two sexes (including confidence intervals for them) using only the `apc` objects for males and females separately.

6. Use the function `ns` (from the splines package) to create model matrices describing age, period and cohort effects respectively. Then use the function `detrend` to remove intercept and trend from the cohort and period terms.

   Fit the age-period-cohort model with these terms separately for each sex, for example by introducing an interaction between sex and all the variables (remember that sex must be a factor for this to be meaningful).

7. Are there any of the effects that possibly could be assumed to be similar between males and females?

8. Fit a model where the period effect is assumed to be identical between males and females and plot the resulting fit for the male/female rate-ratios, and comment on this.

2.11 Prediction of breast cancer rates

1. Read the breast cancer data from the text file and take a look at it for example by:

   ```
   > breast <- read.table("../data/breast.txt", header=T )
   > str( breast )
   > summary( breast )
   ```

   These data are tabulated be age, period and cohort, i.e. each observation correspond to a triangle in the Lexis diagram.

2. The variables `A`, `P` and `C` are the left endpoints of the tabulation intervals. In order to be able to proper analyse data, compute the correct midpoints for each of the triangles.

3. Produce a suitable overview of the rates using the `rateplot` on suitably grouped rates.

4. Fit the age-period-cohort model with natural splines and plot it in a age-period-cohort display. Adjust the display to proper quality using `apc.frame`.

5. Based on the model fitted, make a prediction of future rates of breast cancer:

   - at year 2020.
   - in the 1960 generation.

   Use extensions of the estimated period and cohort effects through the last point and a point 30 years earlier. Try also to see how using a distance of 40 and 20 years work too.

   As a start, add the prediction of the period and cohort effects to the plot of the effects.

   You will need to look into the single components of the `apc` object from `apc.fit`, and you should take a look at the function `approx` for linear interpolation.

6. Now use predictions of the period- and cohort effects based on the 30-year differences to make predictions of cross-sectional rates in 2020 and of the (longitudinal) rates in the 1960 cohort.

   Most likely you will need to compute extrapolated values for the period- and cohort-effects anew.

   Show the predicted rates in a plot.

Chapter 3

Solutions to exercises

3.1 Danish primeministers

The following table shows all Danish prime ministers in office since the war. They are ordered by the period in office, hence some appear twice. Entry end exit refer to the office of prime minister. A missing date of death means that the person was alive at the end of 2008.

Name	Birth	Death	Entry	Exit
Vilhelm Buhl	16/10/1881	18/12/1954	05/05/1945	07/11/1945
Knud Kristensen	26/10/1880	29/09/1962	07/11/1945	13/11/1947
Hans Hedtoft	21/04/1903	29/01/1955	13/11/1947	30/10/1950
Erik Eriksen	20/11/1902	07/10/1972	30/10/1950	30/09/1953
Hans Hedtoft	21/04/1903	29/01/1955	30/09/1953	29/01/1955
H C Hansen	08/11/1906	19/02/1960	01/02/1955	19/02/1960
Viggo Kampmann	21/07/1910	03/06/1976	21/02/1960	03/09/1962
Jens Otto Kragh	15/09/1914	22/06/1978	03/09/1962	02/02/1968
Hilmar Baunsgaard	26/02/1920	30/06/1989	02/02/1968	11/10/1971
Jens Otto Kragh	15/09/1914	22/06/1978	11/10/1971	05/10/1972
Anker Jorgensen	13/07/1922	.	05/10/1972	19/12/1973
Poul Hartling	14/08/1914	30/04/2000	19/12/1973	13/02/1975
Anker Jorgensen	13/07/1922	.	13/02/1975	10/09/1982
Poul Schlüter	03/04/1929	.	10/09/1982	25/01/1993
Poul Nyrup Rasmussen	15/06/1943	.	25/01/1993	27/11/2001
Anders Fogh Rasmussen	26/01/1953	.	27/11/2001	31/12/2007

The data in the table can be fould in the file `pm-dk.txt`.

```
> st <- read.table( "../data/pm-dk.txt", header=T, as.is=T,
+                   na.strings="." )
> st

                Name      birth      death      entry       exit
1       Vilhelm Buhl 16/10/1881 18/12/1954 05/05/1945 07/11/1945
2    Knud Kristensen 26/10/1880 29/09/1962 07/11/1945 13/11/1947
3       Hans Hedtoft 21/04/1903 29/01/1955 13/11/1947 30/10/1950
4       Erik Eriksen 20/11/1902 07/10/1972 30/10/1950 30/09/1953
5       Hans Hedtoft 21/04/1903 29/01/1955 30/09/1953 29/01/1955
6         H C Hansen 08/11/1906 19/02/1960 01/02/1955 19/02/1960
7     Viggo Kampmann 21/07/1910 03/06/1976 21/02/1960 03/09/1962
8     Jens Otto Krag 15/09/1914 22/06/1978 03/09/1962 18/02/1968
```

```
9      Hilmar Baunsgaard 26/02/1920 30/06/1989 18/02/1968 09/10/1971
10        Jens Otto Krag 15/09/1914 22/06/1978 09/10/1971 05/10/1972
11       Anker Jørgensen 13/07/1922       <NA> 05/10/1972 18/12/1973
12         Poul Hartling 14/08/1914 30/04/2000 18/12/1973 13/02/1975
13       Anker Jørgensen 13/07/1922       <NA> 13/02/1975 10/09/1982
14         Poul Schlüter 03/04/1929       <NA> 10/09/1982 25/01/1993
15   Poul Nyrup Rasmussen 15/06/1943       <NA> 25/01/1993 27/11/2001
16 Anders Fogh Rasmussen 26/01/1953       <NA> 27/11/2001 21/01/2009
```

```
> str( st )
```

```
'data.frame':        16 obs. of  5 variables:
 $ Name : chr  "Vilhelm Buhl" "Knud Kristensen" "Hans Hedtoft" "Erik Eriksen" ...
 $ birth: chr  "16/10/1881" "26/10/1880" "21/04/1903" "20/11/1902" ...
 $ death: chr  "18/12/1954" "29/09/1962" "29/01/1955" "07/10/1972" ...
 $ entry: chr  "05/05/1945" "07/11/1945" "13/11/1947" "30/10/1950" ...
 $ exit : chr  "07/11/1945" "13/11/1947" "30/10/1950" "30/09/1953" ...
```

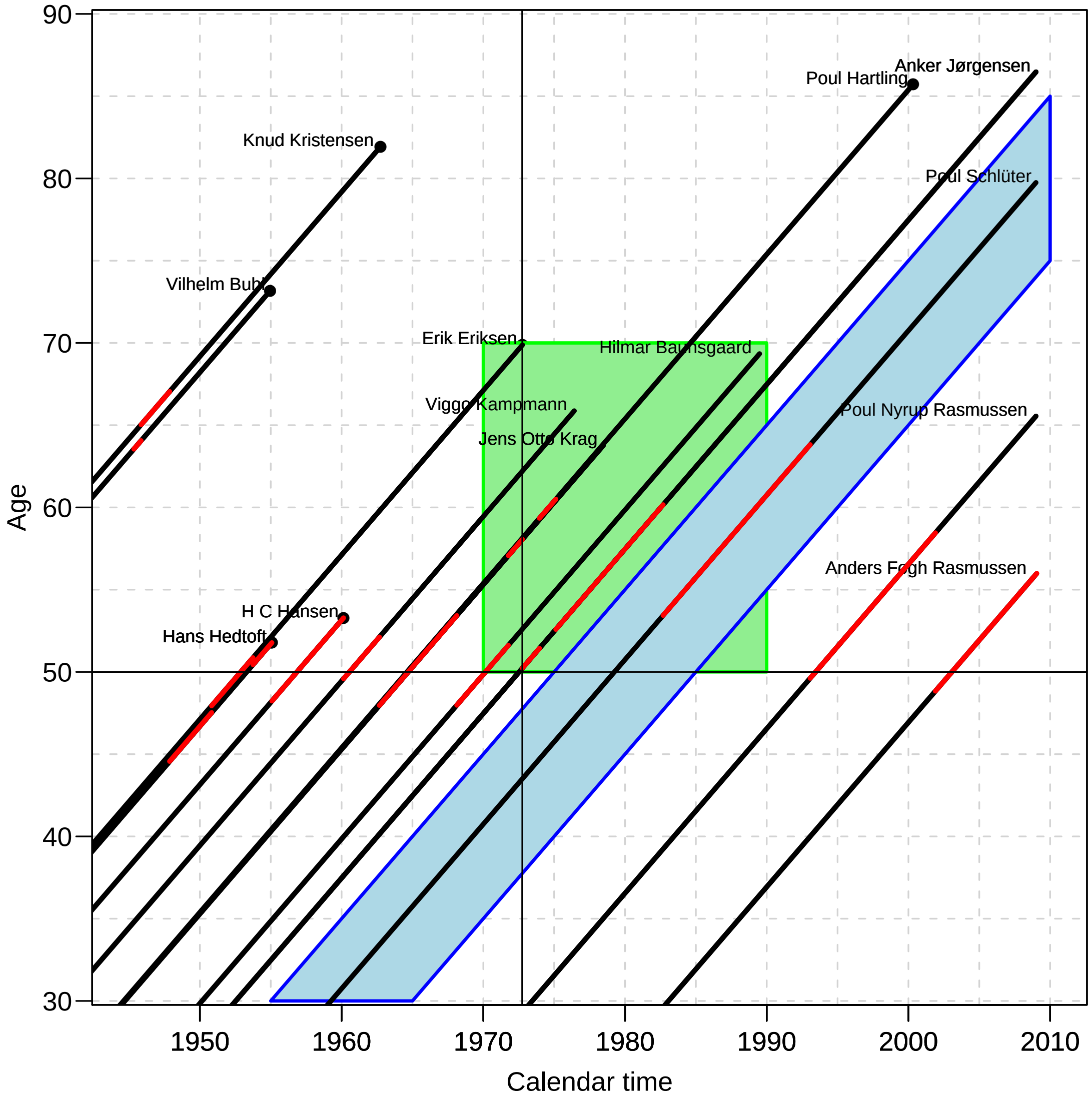

Figure 3.1: *Lexis diagram of life lines of all post-war Danish prime ministers, from 30 years of age.*

1. Draw a Lexis diagram with life-lines of the persons.

```
> # Change the character variables with dates to fractional calendar
> # years
> for( i in 2:5 ) st[,i] <- cal.yr( st[,i], format="%d/%m/%Y" )
> # Attach the data for those still alive
> st$fail <- !is.na(st$death)
> st[!st$fail,"death"] <- 2009
> st

                    Name    birth    death    entry     exit  fail
1           Vilhelm Buhl 1881.792 1954.961 1945.340 1945.849  TRUE
2        Knud Kristensen 1880.820 1962.742 1945.849 1947.864  TRUE
3           Hans Hedtoft 1903.300 1955.076 1947.864 1950.827  TRUE
4           Erik Eriksen 1902.884 1972.765 1950.827 1953.745  TRUE
5           Hans Hedtoft 1903.300 1955.076 1953.745 1955.076  TRUE
6             H C Hansen 1906.851 1960.133 1955.084 1960.133  TRUE
7         Viggo Kampmann 1910.550 1976.420 1960.138 1962.671  TRUE
8         Jens Otto Krag 1914.704 1978.471 1962.671 1968.130  TRUE
9      Hilmar Baunsgaard 1920.152 1989.493 1968.130 1971.769  TRUE
10        Jens Otto Krag 1914.704 1978.471 1971.769 1972.760  TRUE
11       Anker Jørgensen 1922.528 2009.000 1972.760 1973.962 FALSE
12         Poul Hartling 1914.616 2000.327 1973.962 1975.117  TRUE
13       Anker Jørgensen 1922.528 2009.000 1975.117 1982.690 FALSE
14         Poul Schlüter 1929.253 2009.000 1982.690 1993.066 FALSE
15  Poul Nyrup Rasmussen 1943.451 2009.000 1993.066 2001.904 FALSE
16 Anders Fogh Rasmussen 1953.069 2009.000 2001.904 2009.055 FALSE

> attach( st )

> # Lexis object
> L <- Lexis( entry = list(per=birth),
+             exit = list(per=death,age=death-birth),
+             exit.status = fail,
+             data = st )
> # Plot Lexis diagram
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6, xaxt="n" ) # Omit x-labels
> plot( L, xlim=c(1945,2010), ylim=c(32,88), lwd=3, las=1,grid=0:20*5, col="black",
+ xlab="Calendar time", ylab="Age" )
> points( L, pch=c(NA,16)[L$lex.Xst+1] )
> #put names of the prime ministers on the plot
> text( death, death-birth, Name, adj=c(1.05,-0.05), cex=0.7 )
> par( xaxt="s" )
> axis( side=1, at=seq(1950,2010,10) ) # x-labels at nice places
```

2. Mark with a different color the periods where they have been in office.

```
> # New Lexis object describing periods in an office
> # and lines added to a picture
> in_office <- c( rep(FALSE,nrow(st)-1), TRUE )
> st <- cbind( st, in_office )
> Lo <- Lexis( entry = list(per=entry),
+              exit = list(per=exit,age=exit-birth),
+              exit.status = in_office,
+              data = st )
> lines( Lo, lwd=3, las=1, col="red" )
> # the same may be plotted using command segments
> box()
> segments( birth, 0, death, death-birth, lwd=2 )
> segments( entry, entry-birth, exit, exit-birth, lwd=4, col="red" )
```

3. Draw the line representing age 50 years.

```
> abline( h=50 )
```

4. How many 50th birthdays have been celebrated in office since the war?

```
> age_entry <- Lo$age
> age_exit <- Lo$age+Lo$lex.dur
> n_birthday<- sum( ( age_entry<50) & ( age_exit>50 ) )
> n_birthday

[1] 7
```

5. Draw the line representing 2 October 1972. (Why just that?)

```
> abline( v=cal.yr( "2/10/1972", format="%d/%m/%Y" ) )
```

6. How many present and former prime ministes were alive at 31st December 2008?

```
> alive <- (L$death >=2004)
> n_alive <- sum(alive)
> n_alive

[1] 5

> #Anker Jorgensen - 1 person has got 2 lex.id's
> levels( as.factor( subset( L$Name,alive==T ) ) )

[1] "Anders Fogh Rasmussen" "Anker Jørgensen"       "Poul Nyrup Rasmussen"
[4] "Poul Schlüter"
```

7. Which period(s) since the war has seen the maximal number of former post-war prime ministers alive?

```
> # New lexis object - since entry to the office to the death
> Ln <- Lexis( entry=list(per=entry), exit=list(per=death,age=death-entry),
+ exit.status=fail, data=st )
> ny <- 2008-1945
> n_alive <- vector( "numeric", ny )
> for (i in 1:ny)
+ {
+ alive <- ( (Ln$death >=(1944+i)) & (Ln$entry<=(1944+i)) )
+ n_alive[i] <- nlevels( as.factor( subset( Ln$Name, alive==T ) ) )
+ }
```

The maximal number of former post-war prime ministres alive was 5 in 1974-1976 3.2.

8. Mark the area in the diagram with person years lived by persons aged 50 to 70 in the period 1 January 1970 through 1 January 1990.

```
> rect( 1970, 50, 1990, 70, lwd=2, border="green",col="lightgreen"  )
```

9. Mark the area for the lifetime experience of those who were between 10 and 20 years old in 1945.

```
> polygon( c(1955,2005,2005,1965,1955), c(30,80,70,30,30), lwd=2,
+      border="blue", col="lightblue" )
```

The Lexis diagram with all the requested lines etc. is shown in figure 3.1 .

10. How many prime-minister-years have been spent time in each of these sets? And in the intersection of them?

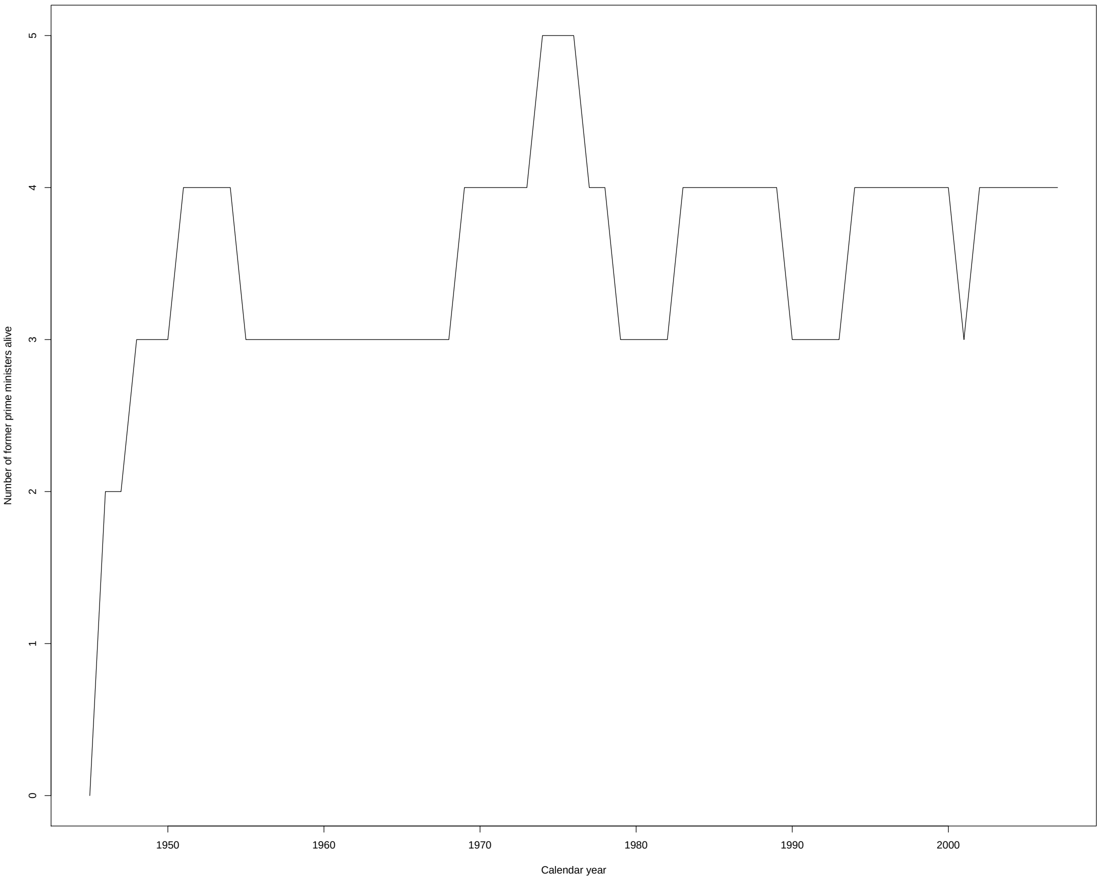

Figure 3.2: *Number of former prime ministers alive.*

```
> # Prime-minister years lived by persons
> #  aged 50 to 70 in the period 1 January 1970 through 1 January 1990.
> x1 <- splitLexis ( Lo ,breaks=c(0,50,70,100), time.scale="age" )
> x2 <- splitLexis ( x1, breaks=c(1900,1970,1990,2010), time.scale="per" )
> summary( x2 )

Transitions:
     To
From    FALSE TRUE  Records:  Events: Risk time:  Persons:
  FALSE    24    1        25        1       63.7        16

Rates:
     To
From    FALSE TRUE Total
  FALSE     0 0.02  0.02

> tapply( status(x2,"exit")==1, list( timeBand(x2,"age","left"),
+                                     timeBand(x2,"per","left") ), sum )

   1900 1970 1990
0     0    0    0
50    0    0    1

> tapply( dur(x2),  list( timeBand(x2,"age","left"),
+                         timeBand(x2,"per","left") ), sum )
```

```
       1900       1970      1990
0  11.10198  0.1519507  1.549624
50 13.54415 19.8480493 17.505818

> # Computing the person-years in the 1925-35 cohort
> x3 <- subset( Lo , birth>1925 & birth<=1935 )
> summary( x3 )

Transitions:
     To
From    FALSE  Records:  Events: Risk time:  Persons:
  FALSE     1         1        0      10.38         1

Rates:
     To
From    FALSE Total
  FALSE     0     0

> dur( x3 )

[1] 10.37645

> # Computing person years in the intersection
> x4 <- subset( x2 , birth>1925 & birth<=1935 )
> summary( x4 )

Transitions:
     To
From    FALSE  Records:  Events: Risk time:  Persons:
  FALSE     2         2        0      10.38         1

Rates:
     To
From    FALSE Total
  FALSE     0     0

> dur( x4 )

[1] 7.310062 3.066393
```

The number of person-years in office in ages 50-69 in the period 1979-1989 is 19.85. The number of prime-minister-years in the 1925-35 cohort is 10.38. The intersection of the two sets holds 7.31 person-years.

3.2 Reading and tabulating data

The following exercise is aimed at tabulating and displaying the data typically involved in age-period-cohort analysis.

1. Read the data in the file `lung5-M.txt`, and print the data. What does each line refer to?

   First we have read the data concerning the lung cancer tabulated in 5 years wide age and period groups. Variables in a data set represent the Age group (A), Period (P), number of cancer cases (D) and person-years (Y). Each line represents number of cancer cases and person-years at risk in for a specific age group and period.

```
> lung <- read.table( "../data/lung5-M.txt", header=T )
> head(lung)

   A    P  D        Y
1 40 1943 80 694046.5
2 40 1948 81 754769.5
3 40 1953 73 769440.7
4 40 1958 99 749264.5
5 40 1963 82 757240.0
6 40 1968 97 709558.5

> attach( lung )
```

2. Print the no. cases in a nice tabular form, and likewise with the person-years. Is there someything special about the last period?

   Table `D_table_nice` represents number of cancer cases in a tabulater form. Similarly, table `Y_table_nice` represents person-years in a tabulater form. While the person-years at risk are constant or slightly increasing for previous periods, in the last period 1993 the person-years and number of cases (for age groups older then 55 years and even more for men older then 65) are slightly smaller. These were born during and before the the second-world war.

```
> D_table_nice <- stat.table(index=list(A,P), sum(D), data=lung, margin=T )
> print( D_table_nice, digits=c(sum=0) )

-----------------------------------------------------------------------------------------------
       ------------------------------------------------P--------------------------------------
A           1943    1948    1953    1958    1963    1968    1973    1978    1983    1988     199
-----------------------------------------------------------------------------------------------
40            80      81      73      99      82      97      86      90     116     149      9
45           135     163     208     226     252     284     263     251     257     265     25
50           197     292     442     508     560     580     657     608     591     493     44
55           261     404     596     772    1052    1075    1115    1218    1090     995     69
60           213     394     577     955    1342    1682    1654    1826    1885    1497    111
65           141     273     491     868    1235    1856    2136    2231    2188    2193    148
70           110     215     300     596     976    1448    1924    2283    2293    2157    169
75            54     126     167     320     514     860    1213    1559    1824    1640    122
80            20      57      87     157     220     390     573     753     881     837     71
85             7      10      23      48      72     110     176     213     307     286     26

Total       1218    2015    2964    4549    6305    8382    9797   11032   11432   10512    797
-----------------------------------------------------------------------------------------------

> Y_table_nice<-stat.table( index=list(A,P), sum(Y), data=lung, margin=T )
> print( Y_table_nice, digits=c(sum=2) )
```

```
 -------------------------------------------------------------------------------------------------------
        -----------------------------------------------------------------P--------------------
 A            1943       1948       1953       1958       1963       1968       1973       197
 -------------------------------------------------------------------------------------------------------
 40       694046.50  754769.50  769440.67  749264.50  757240.00  709558.50  695210.17  756263.0
 45       622256.67  676718.00  738290.50  754357.67  737405.67  747054.83  697976.33  681063.8
 50       538964.17  600506.33  653867.50  715819.83  733590.17  717677.33  724880.33  675371.5
 55       471016.00  512338.00  571270.67  622413.33  681097.00  699103.17  683242.67  686939.5
 60       403172.50  435098.33  474197.50  528106.33  573204.83  627036.33  644142.67  627509.1
 65       328690.50  357694.83  386083.00  419562.00  463265.17  501020.00  548399.50  564173.5
 70       230090.83  269235.83  294786.67  317388.00  341288.33  373577.00  404348.83  442925.0
 75       140110.67  166641.83  195729.83  214930.33  228793.50  245932.00  268415.17  290162.3
 80        67778.83   80587.00   98561.33  116116.67  125697.33  136646.17  150131.83  163433.0
 85        24656.17   28463.83   34280.50   42136.33   49263.33   56018.17   63742.67   71226.5

 Total   3520782.84 3882053.48 4216508.17 4480094.99 4690845.33 4813623.50 4880490.17 4959067.3
 -------------------------------------------------------------------------------------------------------
```

3. Compute the empirical rates, and print them in a table too.

   Table `R_table_nice` represents age-specific incidence rate per 100 000 person-years in a tabulater form. Despite the change in person-years, the age-specific rates for period 1993 do not diverge from the rates of previous ones.

```
> R_table_nice <- stat.table( index=list(A,P), list(Rate=ratio(D,Y,100000)),
+                             data=lung, margin=T )
> print( R_table_nice, digits=c(sum=2) )
```

```
 -------------------------------------------------------------------------------------------------------
        -----------------------------------------------P----------------------------------------
 A            1943     1948     1953     1958     1963     1968     1973     1978     1983     1988     199
 -------------------------------------------------------------------------------------------------------
 40          11.53    10.73     9.49    13.21    10.83    13.67    12.37    11.90    12.32    14.52    12.0
 45          21.70    24.09    28.17    29.96    34.17    38.02    37.68    36.85    34.66    28.67    30.5
 50          36.55    48.63    67.60    70.97    76.34    80.82    90.64    90.02    89.61    68.50    63.6
 55          55.41    78.85   104.33   124.03   154.46   153.77   163.19   177.31   170.10   158.82   127.9
 60          52.83    90.55   121.68   180.83   234.12   268.25   256.78   290.99   299.02   253.44   240.3
 65          42.90    76.32   127.17   206.88   266.59   370.44   389.50   395.45   398.84   396.26   352.3
 70          47.81    79.86   101.77   187.78   285.98   387.60   475.83   515.44   499.75   480.40   462.1
 75          38.54    75.61    85.32   148.89   224.66   349.69   451.91   537.29   571.51   487.43   464.4
 80          29.51    70.73    88.27   135.21   175.02   285.41   381.66   460.74   501.23   426.03   426.1
 85          28.39    35.13    67.09   113.92   146.15   196.36   276.11   299.05   395.51   334.99   351.1

 Total       34.59    51.91    70.30   101.54   134.41   174.13   200.74   222.46   220.12   190.83   174.2
 -------------------------------------------------------------------------------------------------------
```

```
> # another way of computation - not using Epi library
> D_table <- tapply( D, list(A,P), sum )
> Y_table <- tapply( Y, list(A,P), sum )
> R_table <- D_table/Y_table*(10^5)
```

4. Make the four classical graphs of the data. Consider whether a log-scale for the y-axis is appropriate. Think about where on the x-axis each age-class is located.

   (a) Age-specific rates for each period. (Rates from the same period connected).

   ```
   > rateplot( R_table, which=c("AP"), ann=TRUE )
   ```

   (b) Age-specific rates for each cohort. (Rates from the same cohort connected).

   ```
   > rateplot( R_table, which=c("AC"), ann=TRUE )
   ```

   (c) Rates for each age-class versus period. (Rates from the same age-class connected).

```
> rateplot( R_table, which=c("PA"), ann=TRUE )
```

(d) Rates for each age-class versus cohort. (Rates from the same age-class connected).

```
> rateplot( R_table, which=c("CA"), ann=TRUE )
```

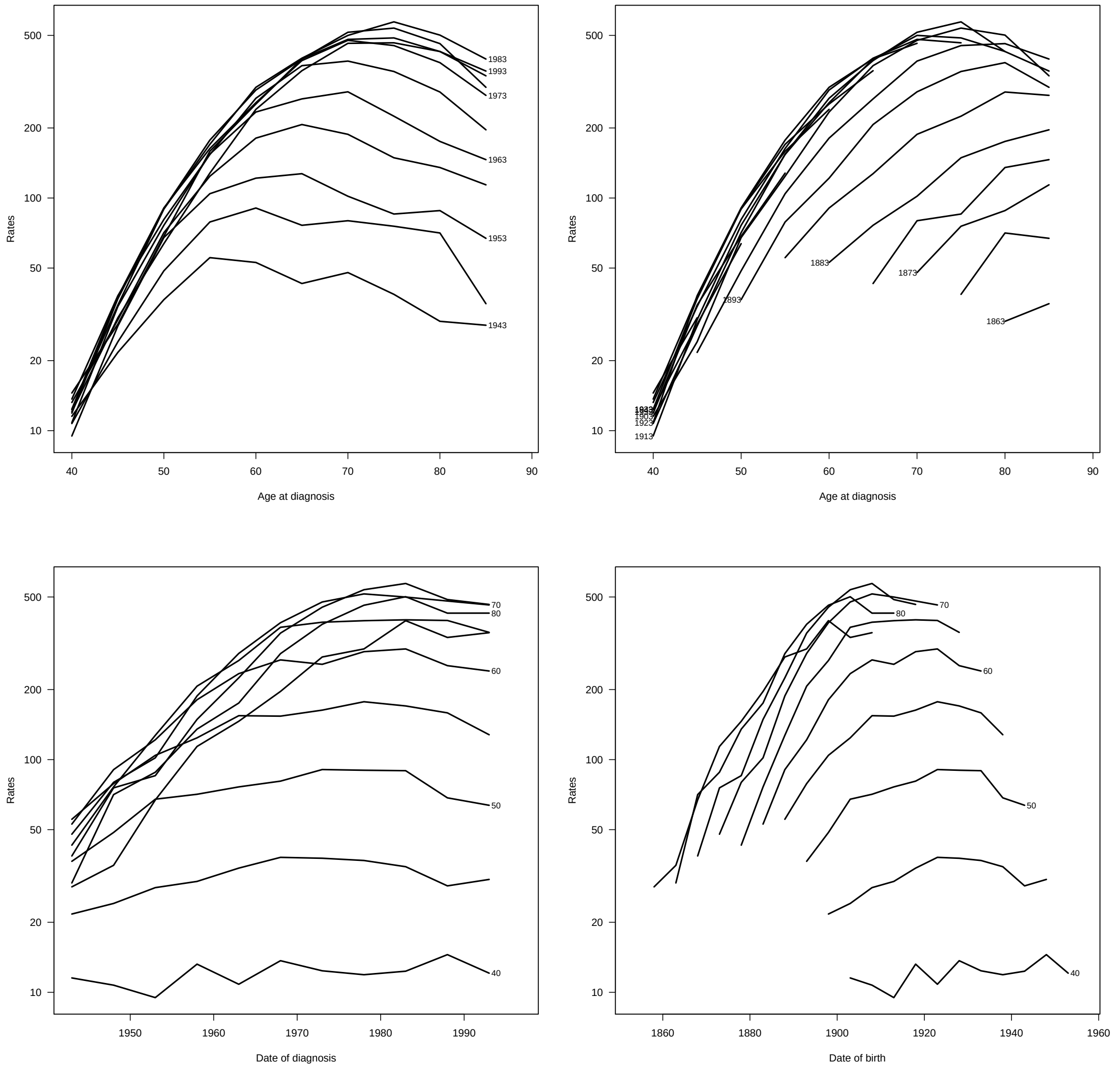

Figure 3.3: *Four rate plots for lung cancer data. Top left: Age on x axis, the rates corresponding to same period are connected by lines. Top right: Age on x axis, the rates corresponding to same cohorts are connected by lines. Bottom left: Period on x axis, the rates corresponding to same age groups are connected by lines. Bottom right: Cohort on x axis, the rates corresponding to same age groups are connected by lines.*

5. How would each of these curves look if:

(a) age-specific rates did not change at all by time?

When age-specific rates did not change at all by time, the age-specific rates are identical for all periods and cohorts. The period and cohort effects are represented by constant horizontal lines. Fig.3.4

```
> # age-specific rates remain still the same as in period 1943
> source( "rateplot2.R" )
> R_table_no_change <- matrix( R_table[,1], dim(R_table)[1], dim(R_table)[2] )
> colnames( R_table_no_change ) <- colnames( R_table )
> rownames( R_table_no_change ) <- rownames( R_table )
> R_table_no_change
```

```
       1943     1948     1953     1958     1963     1968     1973     1978
40 11.52661 11.52661 11.52661 11.52661 11.52661 11.52661 11.52661 11.52661
45 21.69523 21.69523 21.69523 21.69523 21.69523 21.69523 21.69523 21.69523
50 36.55159 36.55159 36.55159 36.55159 36.55159 36.55159 36.55159 36.55159
55 55.41213 55.41213 55.41213 55.41213 55.41213 55.41213 55.41213 55.41213
60 52.83098 52.83098 52.83098 52.83098 52.83098 52.83098 52.83098 52.83098
65 42.89750 42.89750 42.89750 42.89750 42.89750 42.89750 42.89750 42.89750
70 47.80721 47.80721 47.80721 47.80721 47.80721 47.80721 47.80721 47.80721
75 38.54096 38.54096 38.54096 38.54096 38.54096 38.54096 38.54096 38.54096
80 29.50774 29.50774 29.50774 29.50774 29.50774 29.50774 29.50774 29.50774
85 28.39046 28.39046 28.39046 28.39046 28.39046 28.39046 28.39046 28.39046
       1983     1988     1993
40 11.52661 11.52661 11.52661
45 21.69523 21.69523 21.69523
50 36.55159 36.55159 36.55159
55 55.41213 55.41213 55.41213
60 52.83098 52.83098 52.83098
65 42.89750 42.89750 42.89750
70 47.80721 47.80721 47.80721
75 38.54096 38.54096 38.54096
80 29.50774 29.50774 29.50774
85 28.39046 28.39046 28.39046
```

```
> par( mfrow=c(2,2) )
> rateplot2( R_table_no_change )
```

(b) age-specific rates were only influenced by period?

When age-specific rates are influenced only by period, the age-specific rates are parallel for all periods. The period effects are represented by parallel lines. Fig.3.5.

```
> #age-specific rates are only influence by period
> step <- 2
> change_p <- matrix( rep(seq(1,11*step,step),10),10,11, byrow=T )
> change_p
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
 [1,]    1    3    5    7    9   11   13   15   17    19    21
 [2,]    1    3    5    7    9   11   13   15   17    19    21
 [3,]    1    3    5    7    9   11   13   15   17    19    21
 [4,]    1    3    5    7    9   11   13   15   17    19    21
 [5,]    1    3    5    7    9   11   13   15   17    19    21
 [6,]    1    3    5    7    9   11   13   15   17    19    21
 [7,]    1    3    5    7    9   11   13   15   17    19    21
 [8,]    1    3    5    7    9   11   13   15   17    19    21
 [9,]    1    3    5    7    9   11   13   15   17    19    21
[10,]    1    3    5    7    9   11   13   15   17    19    21
```

```
> R_table_p <- R_table_no_change + change_p
> colnames( R_table_p ) <- colnames( R_table )
> rownames( R_table_p ) <- rownames( R_table )
> R_table_p
```

```
       1943     1948     1953     1958     1963     1968     1973     1978
40 12.52661 14.52661 16.52661 18.52661 20.52661 22.52661 24.52661 26.52661
45 22.69523 24.69523 26.69523 28.69523 30.69523 32.69523 34.69523 36.69523
50 37.55159 39.55159 41.55159 43.55159 45.55159 47.55159 49.55159 51.55159
55 56.41213 58.41213 60.41213 62.41213 64.41213 66.41213 68.41213 70.41213
60 53.83098 55.83098 57.83098 59.83098 61.83098 63.83098 65.83098 67.83098
```

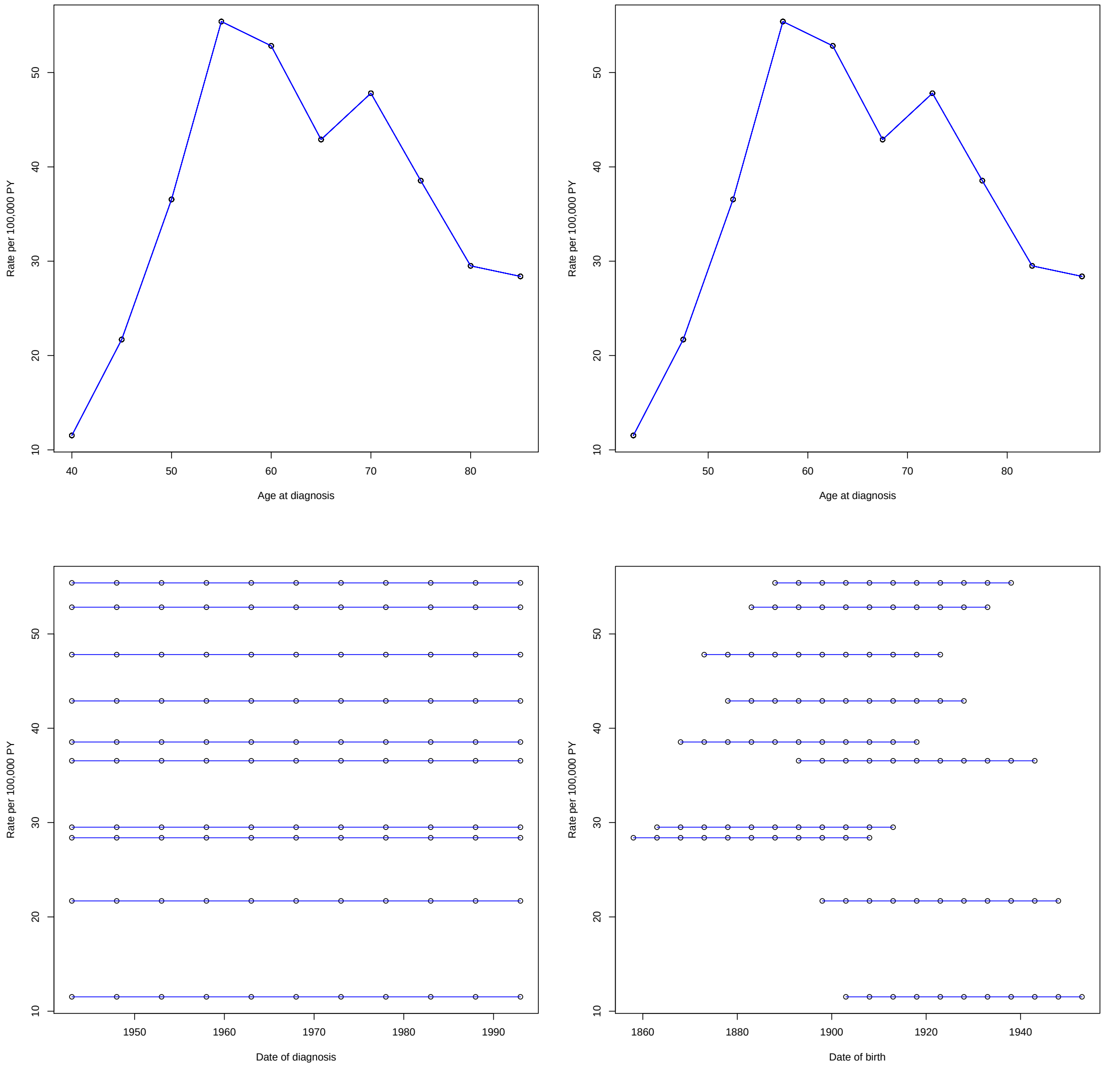

Figure 3.4: *Four rate plots for data with no period and cohort effect. Top left: Age on x axis, the rates corresponding to same period are connected by lines. Top right: Age on x axis, the rates corresponding to same cohorts are connected by lines. Bottom left: Period on x axis, the rates corresponding to same age groups are connected by lines. Bottom right: Cohort on x axis, the rates corresponding to same age groups are connected by lines.*

```
65 43.89750 45.89750 47.89750 49.89750 51.89750 53.89750 55.89750 57.89750
70 48.80721 50.80721 52.80721 54.80721 56.80721 58.80721 60.80721 62.80721
75 39.54096 41.54096 43.54096 45.54096 47.54096 49.54096 51.54096 53.54096
80 30.50774 32.50774 34.50774 36.50774 38.50774 40.50774 42.50774 44.50774
85 29.39046 31.39046 33.39046 35.39046 37.39046 39.39046 41.39046 43.39046
       1983     1988     1993
40 28.52661 30.52661 32.52661
45 38.69523 40.69523 42.69523
50 53.55159 55.55159 57.55159
55 72.41213 74.41213 76.41213
60 69.83098 71.83098 73.83098
65 59.89750 61.89750 63.89750
70 64.80721 66.80721 68.80721
```

```
75 55.54096 57.54096 59.54096
80 46.50774 48.50774 50.50774
85 45.39046 47.39046 49.39046
```

```
> par( mfrow=c(2,2) )
> rateplot2( (R_table_p) )
```

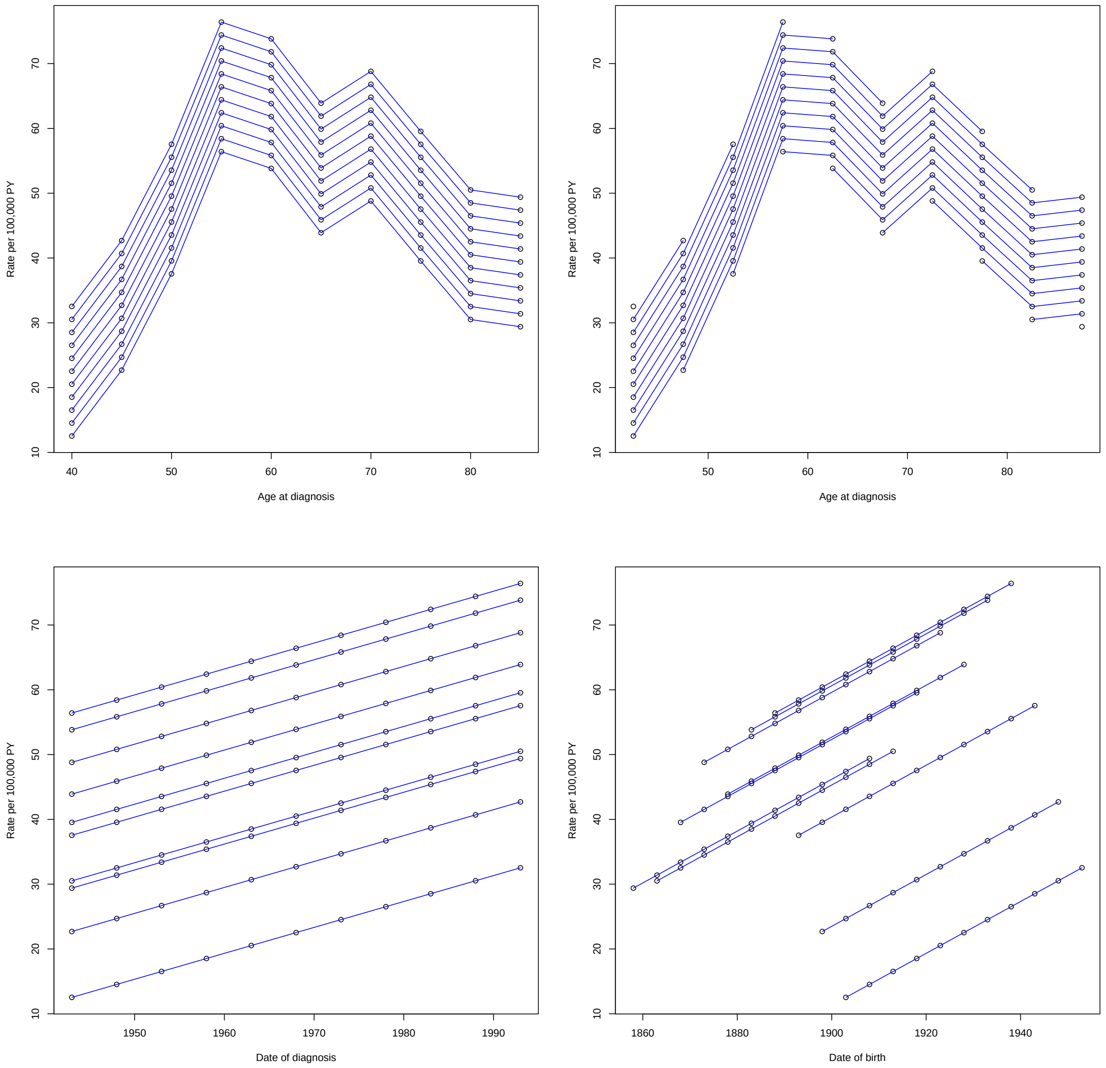

Figure 3.5: *Four rate plots for data with an effect of period. Top left: Age on x axis, the rates corresponding to same period are connected by lines. Top right: Age on x axis, the rates corresponding to same cohorts are connected by lines. Bottom left: Period on x axis, the rates corresponding to same age groups are connected by lines. Bottom right: Cohort on x axis, the rates corresponding to same age groups are connected by lines.*

(c) age-specific rates were only influenced by cohort?

The situation when age-specific rates are influenced only by cohort is demonstrated at Fig.3.6

```
> #age-specific rates are only influence by cohort
> nr <- nrow( R_table )
> nc <- 10
> p <- c( rep(NA,nc ), R_table[,1] )
> np <- length( p )
> R_table_c <- cbind(p[(np-nr+1):np],p[(np-nr):(np-1)],p[(np-nr-1):(np-2)],
+ p[(np-nr-2):(np-3)],p[(np-nr-3):(np-4)],p[(np-nr-4):(np-5)],
+ p[(np-nr-5):(np-6)],p[(np-nr-6):(np-7)],p[(np-nr-7):(np-8)],
+ p[(np-nr-8):(np-9)],p[(np-nr-9):(np-10)]
+ )
> colnames( R_table_c ) <- colnames( R_table )
> rownames( R_table_c ) <- rownames( R_table )
> R_table_c
       1943     1948     1953     1958     1963     1968     1973     1978
40 11.52661       NA       NA       NA       NA       NA       NA       NA
45 21.69523 11.52661       NA       NA       NA       NA       NA       NA
50 36.55159 21.69523 11.52661       NA       NA       NA       NA       NA
55 55.41213 36.55159 21.69523 11.52661       NA       NA       NA       NA
60 52.83098 55.41213 36.55159 21.69523 11.52661       NA       NA       NA
65 42.89750 52.83098 55.41213 36.55159 21.69523 11.52661       NA       NA
70 47.80721 42.89750 52.83098 55.41213 36.55159 21.69523 11.52661       NA
75 38.54096 47.80721 42.89750 52.83098 55.41213 36.55159 21.69523 11.52661
80 29.50774 38.54096 47.80721 42.89750 52.83098 55.41213 36.55159 21.69523
85 28.39046 29.50774 38.54096 47.80721 42.89750 52.83098 55.41213 36.55159
       1983     1988 1993
40       NA       NA   NA
45       NA       NA   NA
50       NA       NA   NA
55       NA       NA   NA
60       NA       NA   NA
65       NA       NA   NA
70       NA       NA   NA
75       NA       NA   NA
80 11.52661       NA   NA
85 21.69523 11.52661   NA
> par( mfrow=c(2,2) )
> rateplot2( R_table_c )
```

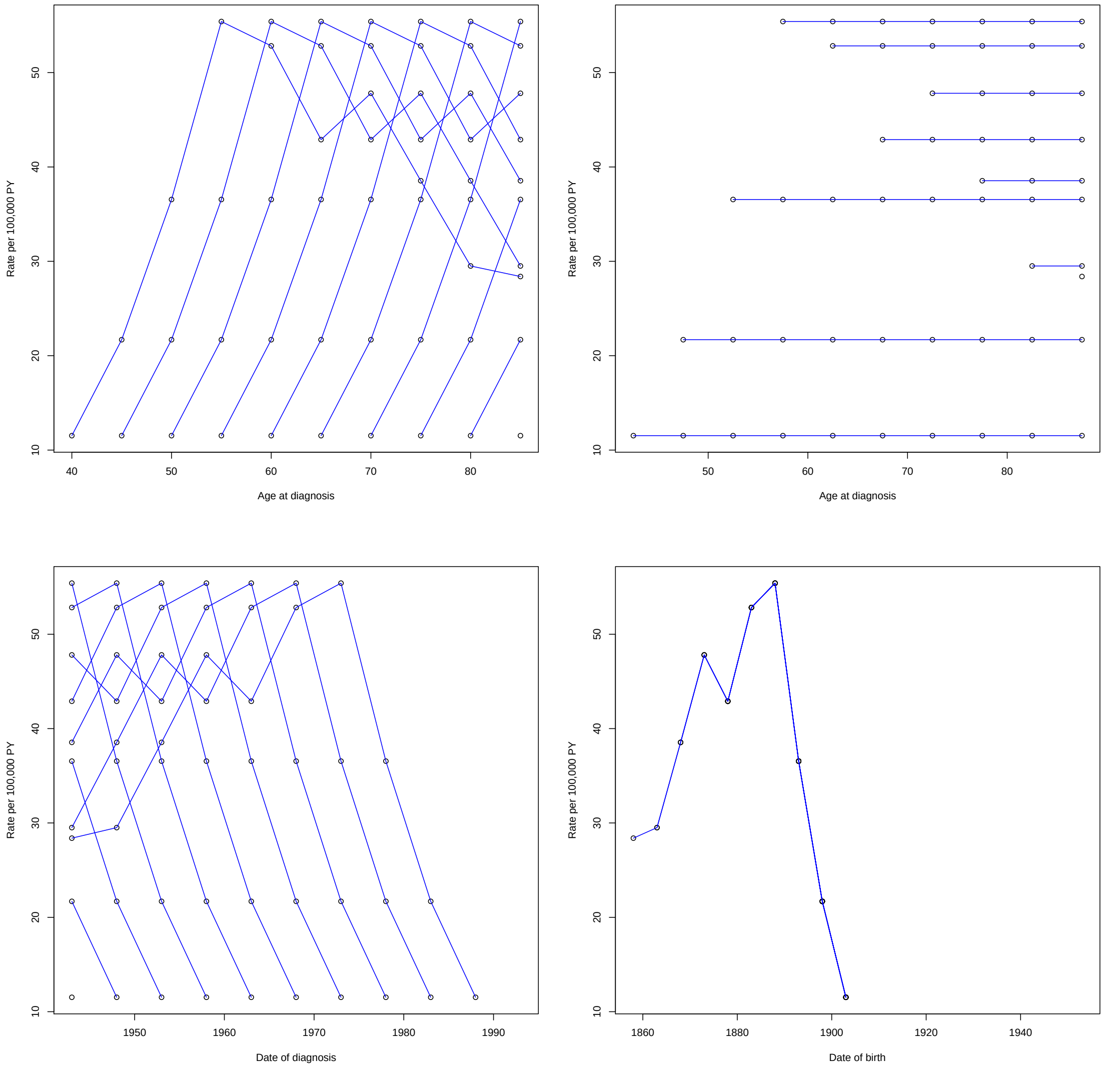

Figure 3.6: *Four rate plots for data with an effect of cohort. Top left: Age on x axis, the rates corresponding to same period are connected by lines. Top right: Age on x axis, the rates corresponding to same cohorts are connected by lines. Bottom left: Period on x axis, the rates corresponding to same age groups are connected by lines. Bottom right: Cohort on x axis, the rates corresponding to same age groups are connected by lines.*

3.3 Rates and survival

1. Consider the following data:

Year of birth	Year of death		Age at death
	1994	1995	
1994	2,900	500	0
1993	120	130	1
1992	50	60	2
1991	45	55	3
1990	40	40	4

(a) Represent these data in a Lexis diagram.

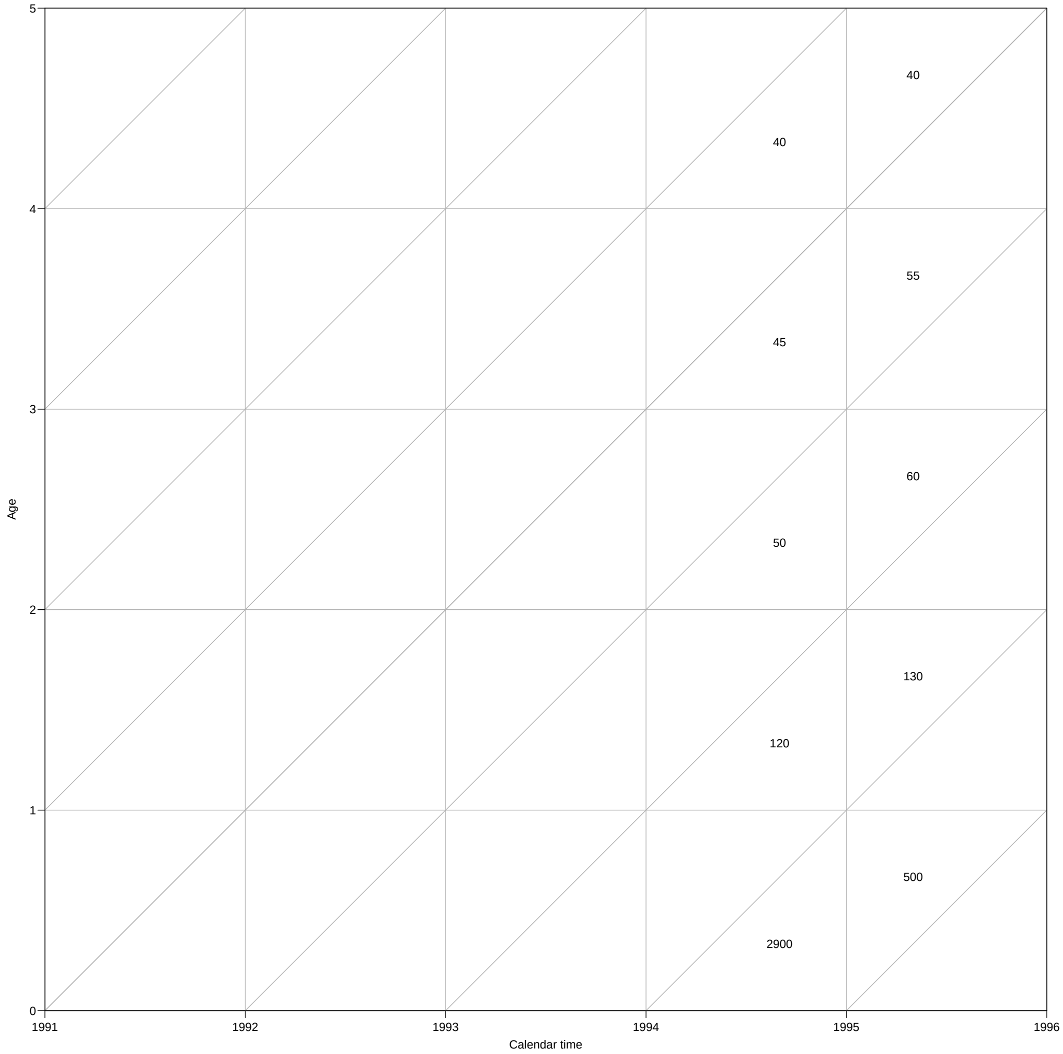

Figure 3.7: *Deaths in age classes 0–4 for the birth cohorts 1990–94, and in age class 0 for the cohorts 1991 and 1992. Fictitious data.*

The given data are shown in the Lexis-diagram in figure 3.7(left). Note that the deaths given are only for one age class for each cohort, so there is no period with complete death count.

(b) On the basis of these data, can you calculate the age-specific death rate for two-year-olds (${}_1m_2$) in 1994? If you can, do it. If you cannot, explain what additional information you would need.

In order to be able to do so one would need the total number of deaths among all two-year olds in 1994. But only the deaths in the 1992 cohort are known, not those in the 1991 cohort. Further one would need to know the risk time in the age-class in 1994. This could be estimated as the average of the number of 2-years olds at the beginning and end of 1994 if these numbers were available. If the number of one-year olds at the beginning of 1994 and the number of three-year olds at the end of 1994 were available a more sophisticated estimate of the risk time would be available.

(c) On the basis of these data, can you calculate the probability of surviving from age 2 to age 3 ($_1q_2$) in for the cohort born in 1992?

If you can, do it. If you cannot, explain what additional information you would need.

It is not possible to compute the probability of surviving from age 2 to age 3 in the 1992 cohort, because the number in this cohort that reach the age of 2 is not known. This number would be the denominator in the fraction estimating the probability where the numerator would be the number of deaths, $50 + 60 = 110$.

2. Consider the following data:

   - Live births during 1991: 142,000
   - Number of infants born in 1991 who did not survive until the end of 1991: 2,900
   - Number of infants born in 1991 who survived to the end of 1991, but did not reach their first birthday: 500
   - Live births during 1992: 138,000
   - Number of infants born in 1992 who did not survive until the end of 1992: 2,600
   - Number of infants born in 1992 who survived to the end of 1992, but did not reach their first birthday: 450

   (a) The data are represented on a Lexis diagram at figure 3.7 (right).

   (b) Calculate the infant mortality rate (IMR) for 1992 under the assumption that you were only able to observe events occurring in 1992, and that you did not know the birth dates of infants dying during that year.

   The infant mortality rate given that we only observe events during 1992, would have to be computed on the assumption that birth rates were constant, i.e. the number of births in 1991 and 1992 were the same. We would then observe the one-year survival probability to be $(500 + 2600)/138000 = 0.02246377$, and hence the IMR to be $-\log(1 - 0.02246377) = 0.22720$.

   Alternatively we could argue that out of the initial 138000, 3100 dies, so a fair bet on the risk time is $138000 - 3100/2 = 136450$, so the rate is estimated as $3100/136450 = 0.022719$

   (c) Same as above, except that now you do know the birth dates of infants dying during 1992.

   If we know the birth date of those dying during 1992 we get extra information that enables us to produce a better estimate of the risk time. If we assume that births occur uniformly over the year, the $138000 - 2600 = 135400$ survivors of the 1992 cohort contribute on average 1/2 person-year. Assuming the 2600 deaths occur uniformly over the triangle, these will contribute 1/3 person-year each[1]. By the same token the 500 deaths in the upper triangle also contribute 1/3 person year each. In order to get the contribution from those surviving through the upper triangle we must again invoke the assumption of constancy of birth and death rates and assume that 135400 0-year olds are alive at the beginning of 1992, so 134900 survive, contributiong 134900/2 person years. Thus the total risk time is:

$$135400/2 + 2600/3 + 134900/2 + 500/3 = 136183.3$$

   giving an estimate of the infant mortality rate of $3100/136183.3 = 0.022763$.

[1] $\int_{p=0}^{p=1} \int_{a=0}^{a=p} 2a \, \mathrm{d}a \, \mathrm{d}p = \int_{p=0}^{p=1} a^2 \, \mathrm{d}p = 1/3$

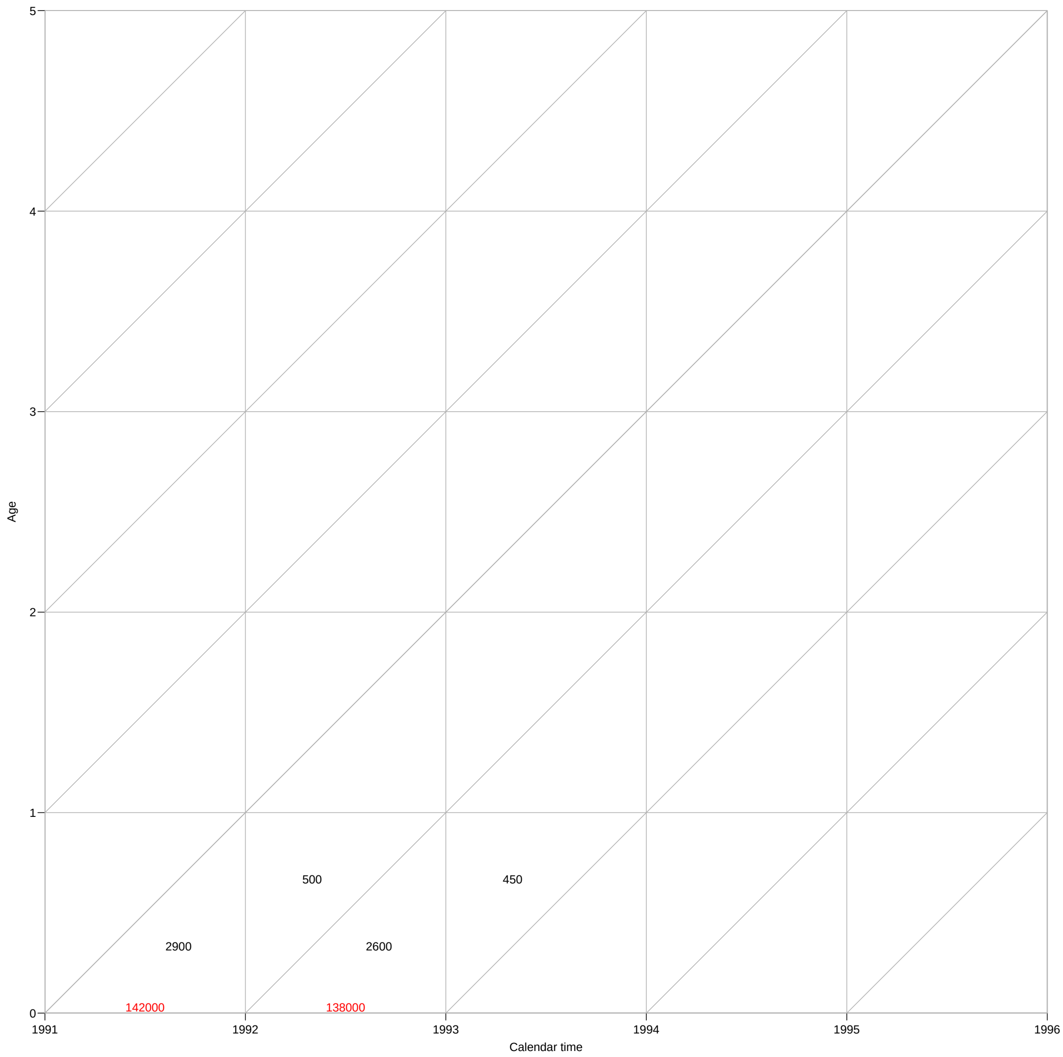

Figure 3.8: *Deaths in the cohorts 1991 and 1992. Fictitious data.*

(d) Assume all data are known: Calculate the IMR.

If we assume all numbers are known, the last calculation must be updated with the correct number of 0-year olds at the beginning of 1992, $142000 - 2900 = 139100$, giving 138600 survivors in the upper triangle:

$$135400/2 + 2600/3 + 138600/2 + 500/3 = 138033.3$$

giving an estimate of the infant mortality rate of $3100/138033.3 = 0.022458$.

Thus we see that the annual variation in birth rates far outweighs the differenecs between the various methodological approaches.

(e) What is the IMR for the 1992 birth cohort?

For the 1992 birth cohort we have two ways of proceeding:

- $2600 + 450 = 3050$ out of 138000 die, thus the one-year survival probability is $3050/138000 = 0.022101$ and hence the infant mortality rate $-\log(1 - 0.022101) = 0.022349$.

- The person-years can be calculated using the same arguments as above:

$$(138000 - 2600)/2 + 2600/3 + (138000 - 2600 - 450)/2 + 450/3 = 136191.7$$

so the rate is estimated as $3050/136191.7 = 0.022395$.

3.4 Age-period model

The following exercise is aimed at familiarizing you with the parametrization of the age-period model. It will give you the opportunity explore how to extract and and plot parameter estimates from models.

1. Read the data in the file `lung5-M.txt` as in the tabulation exercise:

```
> lung <- read.table( "../data/lung5-M.txt", header=T )
> head(lung)

   A    P  D        Y
1 40 1943 80 694046.5
2 40 1948 81 754769.5
3 40 1953 73 769440.7
4 40 1958 99 749264.5
5 40 1963 82 757240.0
6 40 1968 97 709558.5

> attach( lung )
> table( A )

A
40 45 50 55 60 65 70 75 80 85
11 11 11 11 11 11 11 11 11 11

> table( P )

P
1943 1948 1953 1958 1963 1968 1973 1978 1983 1988 1993
  10   10   10   10   10   10   10   10   10   10   10
```

The tables here shows the extent of the data aling the age and period axes, whereas the next table shows the persons years. It is more conveniently rescaled to person-millenia, rounded to one decimal:

```
> round( tapply( Y, list(A,P), sum )/1000, 1 )

    1943  1948  1953  1958  1963  1968  1973  1978  1983   1988  1993
40 694.0 754.8 769.4 749.3 757.2 709.6 695.2 756.3 941.4 1026.5 753.0
45 622.3 676.7 738.3 754.4 737.4 747.1 698.0 681.1 741.6  924.4 821.4
50 539.0 600.5 653.9 715.8 733.6 717.7 724.9 675.4 659.5  719.7 700.9
55 471.0 512.3 571.3 622.4 681.1 699.1 683.2 686.9 640.8  626.5 544.1
60 403.2 435.1 474.2 528.1 573.2 627.0 644.1 627.5 630.4  590.7 463.1
65 328.7 357.7 386.1 419.6 463.3 501.0 548.4 564.2 548.6  553.4 421.5
70 230.1 269.2 294.8 317.4 341.3 373.6 404.3 442.9 458.8  449.0 365.9
75 140.1 166.6 195.7 214.9 228.8 245.9 268.4 290.2 319.2  336.5 262.9
80  67.8  80.6  98.6 116.1 125.7 136.6 150.1 163.4 175.8  196.5 168.0
85  24.7  28.5  34.3  42.1  49.3  56.0  63.7  71.2  77.6   85.4  74.6
```

2. We fit a Poisson model with effects of age (A) and period (P) as class variables:

```
> ap.1 <- glm( D ~ factor(A) + factor(P) + offset(log(Y)), family=poisson )
> summary( ap.1 )

Call:
glm(formula = D ~ factor(A) + factor(P) + offset(log(Y)), family = poisson)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-10.400   -3.728   -0.984    3.685   11.203

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)   -10.34235    0.04192 -246.71   <2e-16
factor(A)45     0.95258    0.03673   25.93   <2e-16
factor(A)50     1.78237    0.03383   52.69   <2e-16
factor(A)55     2.41412    0.03265   73.94   <2e-16
factor(A)60     2.86259    0.03216   89.01   <2e-16
factor(A)65     3.15159    0.03201   98.47   <2e-16
factor(A)70     3.31784    0.03209  103.40   <2e-16
factor(A)75     3.30980    0.03261  101.50   <2e-16
factor(A)80     3.17640    0.03423   92.81   <2e-16
factor(A)85     2.90983    0.04024   72.32   <2e-16
factor(P)1948   0.39206    0.03629   10.80   <2e-16
factor(P)1953   0.67592    0.03404   19.86   <2e-16
factor(P)1958   1.01434    0.03226   31.44   <2e-16
factor(P)1963   1.26666    0.03130   40.47   <2e-16
factor(P)1968   1.48717    0.03067   48.49   <2e-16
factor(P)1973   1.59239    0.03039   52.40   <2e-16
factor(P)1978   1.67994    0.03020   55.62   <2e-16
factor(P)1983   1.69902    0.03015   56.35   <2e-16
factor(P)1988   1.59958    0.03028   52.83   <2e-16
factor(P)1993   1.52558    0.03078   49.57   <2e-16

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 71776.2  on 109  degrees of freedom
Residual deviance:  2723.5  on  90  degrees of freedom
AIC: 3620.5

Number of Fisher Scoring iterations: 5
```

The parameters in this model are: `intercept`: the log-rate in the refence category, which in this model is the first age-category (40: 40–44 years), and the first period (1943: 1943–47), — namely the ones not mentioned in the output from the model. All other parameters are log-rate-ratios relative to this reference category.

3. The same model is now fitted without intercept:

```
> ap.0 <- glm( D ~ -1 +  factor(A) + factor(P) + offset(log(Y)), family=poisson )
> summary( ap.0 )

Call:
glm(formula = D ~ -1 + factor(A) + factor(P) + offset(log(Y)),
    family = poisson)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-10.400   -3.728   -0.984    3.685   11.203

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
factor(A)40   -10.34235    0.04192 -246.71   <2e-16
factor(A)45    -9.38977    0.03454 -271.89   <2e-16
factor(A)50    -8.55998    0.03145 -272.17   <2e-16
factor(A)55    -7.92822    0.03020 -262.48   <2e-16
factor(A)60    -7.47976    0.02970 -251.83   <2e-16
factor(A)65    -7.19075    0.02956 -243.26   <2e-16
factor(A)70    -7.02451    0.02970 -236.53   <2e-16
factor(A)75    -7.03255    0.03031 -232.05   <2e-16
factor(A)80    -7.16595    0.03209 -223.33   <2e-16
factor(A)85    -7.43252    0.03847 -193.22   <2e-16
factor(P)1948   0.39206    0.03629   10.80   <2e-16
factor(P)1953   0.67592    0.03404   19.86   <2e-16
factor(P)1958   1.01434    0.03226   31.44   <2e-16
factor(P)1963   1.26666    0.03130   40.47   <2e-16
factor(P)1968   1.48717    0.03067   48.49   <2e-16
factor(P)1973   1.59239    0.03039   52.40   <2e-16
factor(P)1978   1.67994    0.03020   55.62   <2e-16
factor(P)1983   1.69902    0.03015   56.35   <2e-16
factor(P)1988   1.59958    0.03028   52.83   <2e-16
factor(P)1993   1.52558    0.03078   49.57   <2e-16

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1.0037e+08  on 110  degrees of freedom
Residual deviance: 2.7235e+03  on  90  degrees of freedom
AIC: 3620.5

Number of Fisher Scoring iterations: 5
```

The age-parameters now refer to log-rates as estimated in the reference period, 1943.

4. Now we fit the same model, using the period 1968–72 as the reference period, by using the `relevel` command for factors to make 1968 the first level:

```
> ap.2 <- glm( D ~ factor(A) - 1 + relevel(factor(P),"1968") + offset(log(Y)),
+                  family=poisson )
```

5. Extract the parameters from the model, by doing:

```
> ( ap.cf <- summary( ap.2 )$coef )

                                  Estimate Std. Error     z value      Pr(>|z|)
factor(A)40                    -8.85517346 0.03267181 -271.034040  0.000000e+00
factor(A)45                    -7.90259321 0.02232327 -354.007042  0.000000e+00
factor(A)50                    -7.07280223 0.01707967 -414.106430  0.000000e+00
factor(A)55                    -6.44104968 0.01455119 -442.647633  0.000000e+00
factor(A)60                    -5.99258631 0.01342462 -446.387795  0.000000e+00
factor(A)65                    -5.70357953 0.01312796 -434.460586  0.000000e+00
factor(A)70                    -5.53733722 0.01337568 -413.985515  0.000000e+00
factor(A)75                    -5.54537497 0.01462008 -379.298646  0.000000e+00
factor(A)80                    -5.67877130 0.01794833 -316.395572  0.000000e+00
factor(A)85                    -5.94534410 0.02775505 -214.207677  0.000000e+00
relevel(factor(P), "1968")1943 -1.48717439 0.03066768  -48.493215  0.000000e+00
relevel(factor(P), "1968")1948 -1.09511737 0.02481363  -44.133706  0.000000e+00
relevel(factor(P), "1968")1953 -0.81125051 0.02137233  -37.957983  0.000000e+00
relevel(factor(P), "1968")1958 -0.47283820 0.01841692  -25.674120 2.274664e-145
relevel(factor(P), "1968")1963 -0.22051337 0.01667114  -13.227249  6.108232e-40
relevel(factor(P), "1968")1973  0.10521650 0.01487968    7.071155  1.536496e-12
relevel(factor(P), "1968")1978  0.19276119 0.01449332   13.300001  2.314659e-40
relevel(factor(P), "1968")1983  0.21184343 0.01438727   14.724363  4.496857e-49
relevel(factor(P), "1968")1988  0.11240928 0.01465483    7.670458  1.713837e-14
relevel(factor(P), "1968")1993  0.03840264 0.01565559    2.452966  1.416836e-02
```

6. We plot the estimated age-specific incidence rates, we need the first 10 parameters, with their standard errors:

```
> age.cf <- ap.cf[1:10,1:2]
```

This means that we take rows 1–10 and columns 1–2. The corresponding age classes are 40, . . . , 85. The midpoints of these age-classes are 2.5 years higher. The ages can be generated in R by saying `seq(40,85,5)+2.5`. So we can make the plot in increasing detail:

```
> par( mfrow=c(1,3) )
> am <- seq(40,85,5)+2.5
> plot( am, age.cf[,1] )
> plot( am, exp(age.cf[,1]), log="y" )
> plot( am, exp(age.cf[,1]), type="l", log="y" )
```

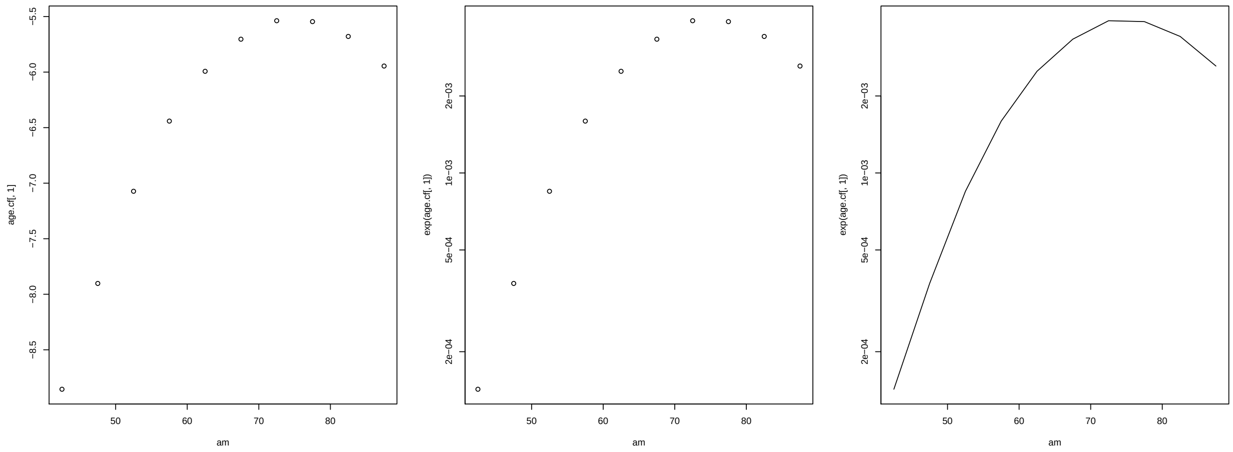

Figure 3.9: *Three versions of the plot of the age-specific rates.*

If we want to put confidence limits on we just take ±1.96 × s.e. on the log-scale. And the s.e.s are in column 2 of `age.cf`. Lines are added to a plot by the command `lines`, or all is made in one go using `matplot`

```
> matplot( am, cbind( exp(age.cf[,1]),
+                     exp(age.cf[,1]-1.96*age.cf[,2]),
+                     exp(age.cf[,1]+1.96*age.cf[,2]) ),
+          type="l", log="y", lwd=c(3,1,1), lty=1, col="black" )
```

The specification of `lty=` and `col=` is necessary in `matplot`, because these otherwise cycles through linetypes and colours, whcih is not desired here.

7. Now for the rate-ratio-parameters, take the rest of the coefficients:

```
> ( RR.cf <- ap.cf[11:20,1:2] )
```

```
                                  Estimate Std. Error
relevel(factor(P), "1968")1943 -1.48717439 0.03066768
relevel(factor(P), "1968")1948 -1.09511737 0.02481363
relevel(factor(P), "1968")1953 -0.81125051 0.02137233
relevel(factor(P), "1968")1958 -0.47283820 0.01841692
relevel(factor(P), "1968")1963 -0.22051337 0.01667114
relevel(factor(P), "1968")1973  0.10521650 0.01487968
relevel(factor(P), "1968")1978  0.19276119 0.01449332
relevel(factor(P), "1968")1983  0.21184343 0.01438727
relevel(factor(P), "1968")1988  0.11240928 0.01465483
relevel(factor(P), "1968")1993  0.03840264 0.01565559
```

But the reference group is missing, so we must stick two 0s in the correct place. We use the command `rbind` (row-bind):

```
> RR.cf <- rbind( RR.cf[1:5,], c(0,0), RR.cf[6:10,] )
> RR.cf

                                 Estimate Std. Error
relevel(factor(P), "1968")1943 -1.48717439 0.03066768
relevel(factor(P), "1968")1948 -1.09511737 0.02481363
relevel(factor(P), "1968")1953 -0.81125051 0.02137233
relevel(factor(P), "1968")1958 -0.47283820 0.01841692
relevel(factor(P), "1968")1963 -0.22051337 0.01667114
                                0.00000000 0.00000000
relevel(factor(P), "1968")1973  0.10521650 0.01487968
relevel(factor(P), "1968")1978  0.19276119 0.01449332
relevel(factor(P), "1968")1983  0.21184343 0.01438727
relevel(factor(P), "1968")1988  0.11240928 0.01465483
relevel(factor(P), "1968")1993  0.03840264 0.01565559
```

Now we have the same situation as for the age-specific rates, and can plot the relative risks (relative to 1968) in precisely the same way as for the age-specific rates:

```
> matplot( as.numeric(levels(factor(P)))+2.5,
+          cbind( exp(RR.cf[,1]),
+                 exp(RR.cf[,1]-1.96*RR.cf[,2]),
+                 exp(RR.cf[,1]+1.96*RR.cf[,2]) ),
+          type="l", log="y", lwd=c(3,1,1), lty=1, col="black" )
```

These rate-ratios are presented beside the corresponding age-specific rates.

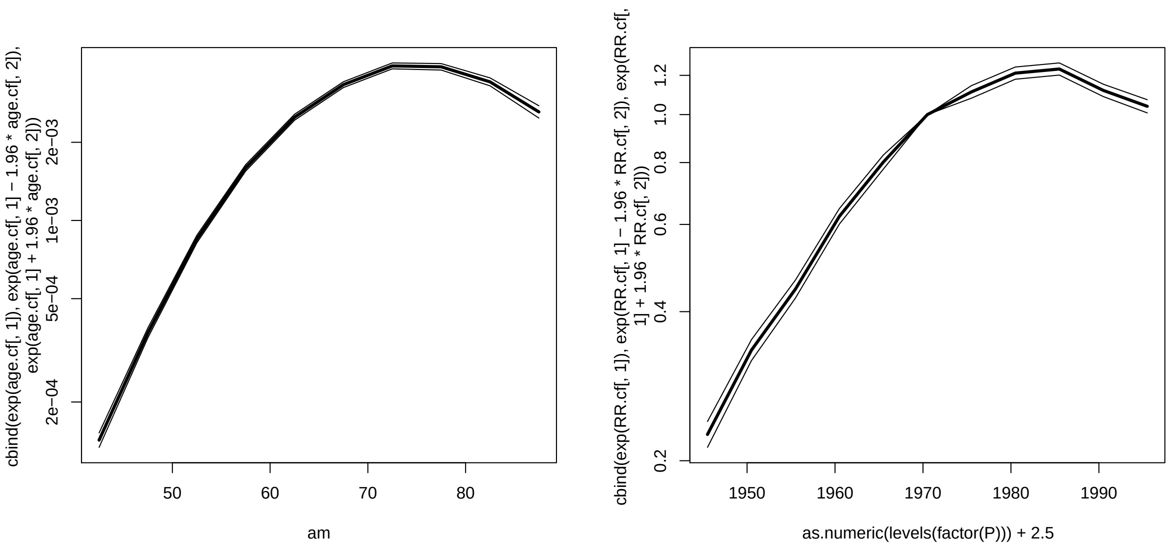

Figure 3.10: *Age-specific rates and rate-ratios relative to the period 1968–72.*

8. The relevant parameters may also be extracted directly from the model without intercept, using the function `ci.lin` which allows selection of a subset of the parameters either by using numbers in the sequence or using character strings through `grep`. Linear functions of selected parameter are computed using a *contrast matrix*, which is multiplied to the selected parameters.

If we want log-rates in the reference period (the first level of `factor(P)` are the age-parameters. The log-rates in the period labelled `1968` are these *plus* the period estimate from `1968`, so to illustrate the workings of the subsetting we select the relevant parameters and just disply these.

```
> ci.lin( ap.0, subset=c("A","1968") )

                 Estimate     StdErr          z P       2.5%      97.5%
factor(A)40    -10.342348 0.04192098 -246.71054 0 -10.424511 -10.260184
factor(A)45     -9.389768 0.03453519 -271.88982 0  -9.457455  -9.322080
factor(A)50     -8.559977 0.03145070 -272.17123 0  -8.621619  -8.498334
factor(A)55     -7.928224 0.03020492 -262.48125 0  -7.987425  -7.869024
factor(A)60     -7.479761 0.02970184 -251.82817 0  -7.537975  -7.421546
factor(A)65     -7.190754 0.02956000 -243.25964 0  -7.248690  -7.132817
factor(A)70     -7.024512 0.02969777 -236.53331 0  -7.082718  -6.966305
factor(A)75     -7.032549 0.03030666 -232.04631 0  -7.091949  -6.973149
factor(A)80     -7.165946 0.03208700 -223.32863 0  -7.228835  -7.103056
factor(A)85     -7.432518 0.03846618 -193.22216 0  -7.507911  -7.357126
factor(P)1968    1.487174 0.03066768   48.49322 0   1.427067   1.547282
```

Since we often need rates as the exponentila of the parameters, there is a `Exp=` argument that gives these too (with c.i.):

```
> ci.lin( ap.0, subset=c("A","1968"), Exp=TRUE )

                 Estimate     StdErr          z P    exp(Est.)         2.5%
factor(A)40    -10.342348 0.04192098 -246.71054 0 3.223854e-05 2.969561e-05
factor(A)45     -9.389768 0.03453519 -271.88982 0 8.357488e-05 7.810509e-05
factor(A)50     -8.559977 0.03145070 -272.17123 0 1.916238e-04 1.801683e-04
factor(A)55     -7.928224 0.03020492 -262.48125 0 3.604259e-04 3.397078e-04
factor(A)60     -7.479761 0.02970184 -251.82817 0 5.643925e-04 5.324747e-04
factor(A)65     -7.190754 0.02956000 -243.25964 0 7.535208e-04 7.111050e-04
factor(A)70     -7.024512 0.02969777 -236.53331 0 8.898020e-04 8.394882e-04
factor(A)75     -7.032549 0.03030666 -232.04631 0 8.826786e-04 8.317744e-04
factor(A)80     -7.165946 0.03208700 -223.32863 0 7.724481e-04 7.253654e-04
factor(A)85     -7.432518 0.03846618 -193.22216 0 5.916955e-04 5.487263e-04
factor(P)1968    1.487174 0.03066768   48.49322 0 4.424576e+00 4.166460e+00
                      97.5%
factor(A)40    3.499924e-05
factor(A)45    8.942772e-05
factor(A)50    2.038076e-04
factor(A)55    3.824076e-04
factor(A)60    5.982235e-04
factor(A)65    7.984666e-04
factor(A)70    9.431313e-04
factor(A)75    9.366982e-04
factor(A)80    8.225870e-04
factor(A)85    6.380294e-04
factor(P)1968 4.698682e+00
```

To get the linear combination of parameters we want we construct the contrast matrix needed to provide the estimates if premultiplied to the selected subset of parameters.

```
> ( cm.A <- cbind( diag( nlevels( factor(A) ) ), 1 ) )

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
 [1,]    1    0    0    0    0    0    0    0    0     0     1
 [2,]    0    1    0    0    0    0    0    0    0     0     1
 [3,]    0    0    1    0    0    0    0    0    0     0     1
 [4,]    0    0    0    1    0    0    0    0    0     0     1
 [5,]    0    0    0    0    1    0    0    0    0     0     1
 [6,]    0    0    0    0    0    1    0    0    0     0     1
 [7,]    0    0    0    0    0    0    1    0    0     0     1
 [8,]    0    0    0    0    0    0    0    1    0     0     1
 [9,]    0    0    0    0    0    0    0    0    1     0     1
[10,]    0    0    0    0    0    0    0    0    0     1     1
```

Using the argument `ctr.mat=` in `ci.lin` to produce the rates in period 1968 we can plot them on a log-scale (note we select only the columns with rates and ci.s:

```
> arates <- ci.lin( ap.0, subset=c("A","1968"), ctr.mat=cm.A, Exp=TRUE )[,5:7]
> matplot( as.numeric( levels( factor(A) ) )+2.5, arates,
+          log="y", type="l", lwd=c(3,1,1), col="black", lty=1 )
```

The rates extracted this way is in the left panel of figure 3.11.

9. Using the same machinery to extract the rate-ratios relative to 1968, we construct the contrast matrix to extract the difference between the RRs with the first period as reference and the RR at 1968; this is the differnece between two metrices: The first one is the one that extracts the rate-ratios with a prefixed 0:

```
> cm.P <- rbind(0,diag( nlevels(factor(P))-1 ) )
> cm.P

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0    0    0    0    0    0    0    0    0     0
 [2,]    1    0    0    0    0    0    0    0    0     0
 [3,]    0    1    0    0    0    0    0    0    0     0
 [4,]    0    0    1    0    0    0    0    0    0     0
 [5,]    0    0    0    1    0    0    0    0    0     0
 [6,]    0    0    0    0    1    0    0    0    0     0
 [7,]    0    0    0    0    0    1    0    0    0     0
 [8,]    0    0    0    0    0    0    1    0    0     0
 [9,]    0    0    0    0    0    0    0    1    0     0
[10,]    0    0    0    0    0    0    0    0    1     0
[11,]    0    0    0    0    0    0    0    0    0     1
```

The second is the matrix with 1s in the column corresponding to 1968.

```
> cm.Pref <- cm.P * 0
> wh.col <- grep( "1968", levels(factor(P)) ) - 1
> cm.Pref[,wh.col] <- 1
> cm.Pref

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0    0    0    0    1    0    0    0    0     0
 [2,]    0    0    0    0    1    0    0    0    0     0
 [3,]    0    0    0    0    1    0    0    0    0     0
 [4,]    0    0    0    0    1    0    0    0    0     0
 [5,]    0    0    0    0    1    0    0    0    0     0
 [6,]    0    0    0    0    1    0    0    0    0     0
 [7,]    0    0    0    0    1    0    0    0    0     0
 [8,]    0    0    0    0    1    0    0    0    0     0
 [9,]    0    0    0    0    1    0    0    0    0     0
[10,]    0    0    0    0    1    0    0    0    0     0
[11,]    0    0    0    0    1    0    0    0    0     0
```

The contrast matrix to use is the difference between these two, and can therefore be directly plotted:

```
> cm.P - cm.Pref

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0    0    0    0   -1    0    0    0    0     0
 [2,]    1    0    0    0   -1    0    0    0    0     0
 [3,]    0    1    0    0   -1    0    0    0    0     0
 [4,]    0    0    1    0   -1    0    0    0    0     0
 [5,]    0    0    0    1   -1    0    0    0    0     0
 [6,]    0    0    0    0    0    0    0    0    0     0
 [7,]    0    0    0    0   -1    1    0    0    0     0
 [8,]    0    0    0    0   -1    0    1    0    0     0
 [9,]    0    0    0    0   -1    0    0    1    0     0
[10,]    0    0    0    0   -1    0    0    0    1     0
[11,]    0    0    0    0   -1    0    0    0    0     1

> RR0 <- ci.lin( ap.0, subset="P", ctr.mat=cm.P-cm.Pref, Exp=TRUE )[,5:7]
> matplot( as.numeric(levels(factor(P)))+2.5, RR0,
+          type="l", log="y", lwd=c(3,1,1), lty=1, col="black" )
```

These RRs are plotted alongside the estimated rates in figure 3.11.

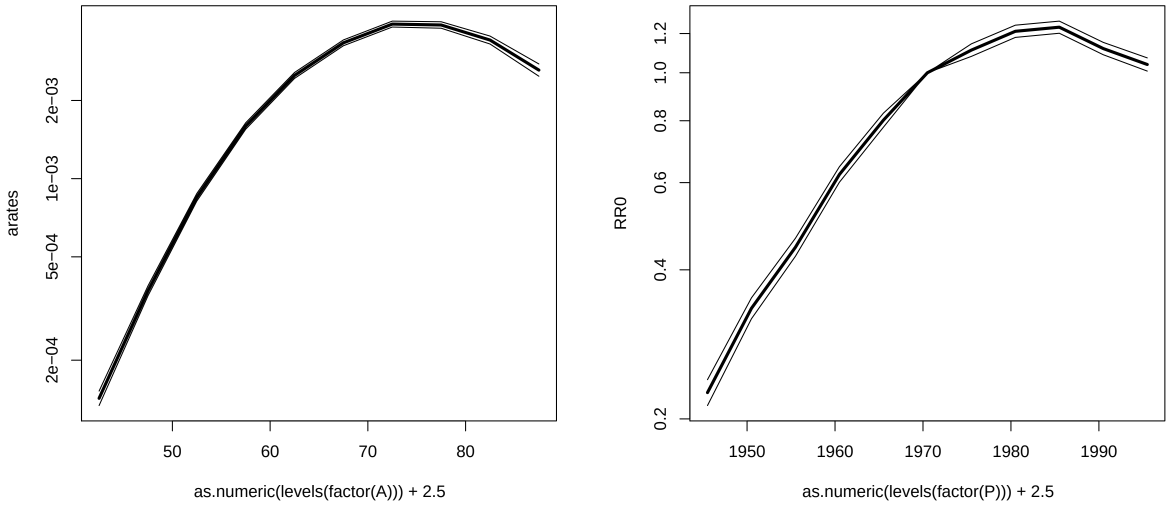

Figure 3.11: *Age-specific rates and rate-ratios relative to the period 1968–72, extracted using* ci.lin.

10. The estimates are saved along with the computed mipoints:

```
> age.pt <- as.numeric(levels(factor(A)))+2.5
>  RR.pt <- as.numeric(levels(factor(P)))+2.5
> save( age.pt, arates,
+       RR.pt, RR0, file="../data/age-per-est.Rdata" )
```

11. If we want to plot the rates and the rate ratios beside each other, and make sure that the physical extent of the units on both the x-axis and the y-axis are the same, we first determine the relative extent of the x-axes for the two plots:

```
> alim <- range( A ) + c(0,5)
> plim <- range( P ) + c(0,5)
```

We then use these to determine the relative width of the two panels, using the `layout` function, and subsequenty adjust the y-axis of the RR-plot to the same physical extent as the rate axis (note that the `par("usr")` returns the $\log_{10}$ of the limits for logaritmic axes):

```
> # Compute limits explicitly
> rlim <- range(arates*10^5)*c(1/1.05,1.05)
> RRlim <- 10^(log10(rlim)-ceiling(mean(log10(rlim))))
> # Determin reltive width of plots
> layout( rbind( c(1,2) ), widths=c(diff(alim),diff(plim)) )
> # No space on the sides of the plots, only outer space
> par( mar=c(4,0,1,0), oma=c(0,4,0,4), mgp=c(3,1,0)/1.5, las=1 )
> matplot( as.numeric(levels(factor(A)))+2.5, arates*10^5,
+          type="l", lwd=c(3,1,1), lty=1, col="black",
+          log="y", xaxs="i", xlim=alim, xlab="Age", ylim=rlim )
> mtext( "Male lung cancer per 100,000", las=0, side=2, outer=T, line=2.5 )
> matplot( as.numeric(levels(factor(P)))+2.5, RR0,
+          type="l", lwd=c(3,1,1), lty=1, col="black",
+          log="y", xlab="Period of follow-up", xlim=plim, yaxt="n", ylim=RRlim, ylab="" )
> abline( h=1 )
> points( 1968+2.5, 1, pch=1, lwd=3 )
> axis( side=4 )
> mtext( "Rate ratio", side=4, outer=T, las=0, line=2.5 )
```

The resulting plot is in figure 3.12

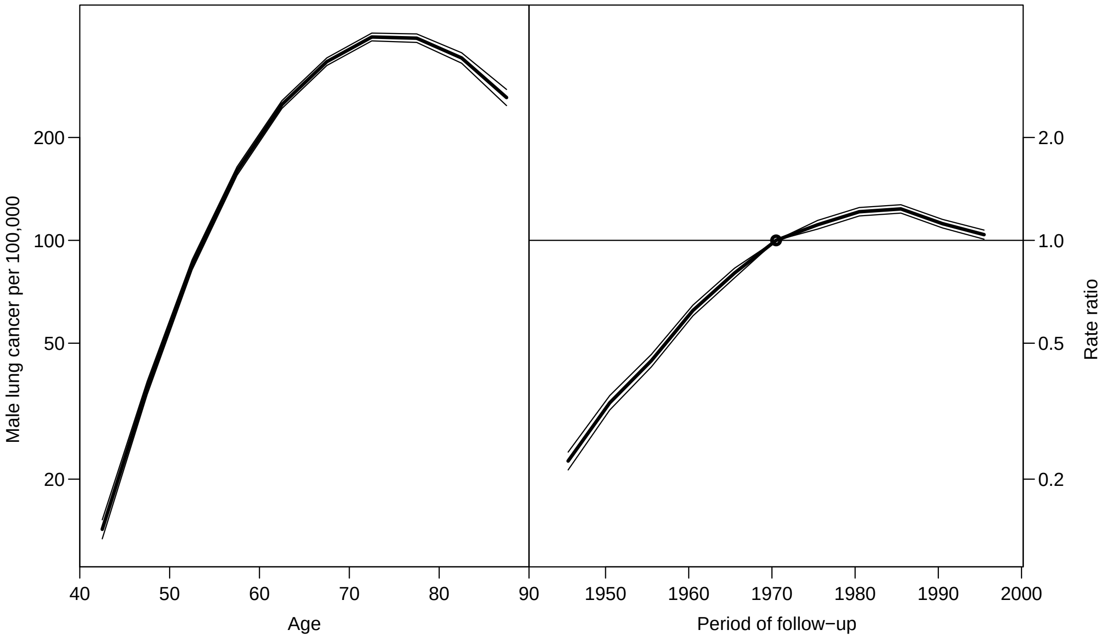

Figure 3.12: *Age-specific rates and rate-ratios relative to the period 1968–72, extracted using* `ci.lin`*, and plotted with scales with physically equal scaling.*

3.5 Age-cohort model

This exercise is parallel to the exercise on the age-period model.

1. First we read the data in the file `lung5-M.txt` and create the cohort variable:

```
> lung <- read.table( "../data/lung5-M.txt", header=T )
> lung$C <- lung$P - lung$A
> attach( lung )
> table( C )

C
1858 1863 1868 1873 1878 1883 1888 1893 1898 1903 1908 1913 1918 1923 1928 1933
   1    2    3    4    5    6    7    8    9   10   10    9    8    7    6    5
1938 1943 1948 1953
   4    3    2    1
```

   It is clear from these tables that the data layout is by age and period, since the outer cohorts are more scarcely represented.

2. We fit a Poisson model with effects of age (A) and cohort (C) as class variables:

```
> ac.1 <- glm( D ~ factor(A) + factor(C) + offset(log(Y)), family=poisson )
> summary( ac.1 )

Call:
glm(formula = D ~ factor(A) + factor(C) + offset(log(Y)), family = poisson)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-7.2822  -2.0274   0.3573   2.0545   5.2834

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)    -11.83501    0.38038 -31.114  < 2e-16
factor(A)45      0.96843    0.03800  25.487  < 2e-16
factor(A)50      1.83467    0.03591  51.087  < 2e-16
factor(A)55      2.51168    0.03508  71.595  < 2e-16
factor(A)60      3.02924    0.03476  87.147  < 2e-16
factor(A)65      3.40740    0.03471  98.156  < 2e-16
factor(A)70      3.67325    0.03487 105.335  < 2e-16
factor(A)75      3.78630    0.03545 106.819  < 2e-16
factor(A)80      3.78402    0.03704 102.165  < 2e-16
factor(A)85      3.66814    0.04280  85.703  < 2e-16
factor(C)1863    0.01046    0.42031   0.025 0.980152
factor(C)1868    0.51345    0.38845   1.322 0.186240
factor(C)1873    0.82684    0.38231   2.163 0.030560
factor(C)1878    1.05336    0.38054   2.768 0.005639
factor(C)1883    1.41904    0.37972   3.737 0.000186
factor(C)1888    1.91197    0.37927   5.041 4.63e-07
factor(C)1893    2.28073    0.37909   6.016 1.78e-09
factor(C)1898    2.55794    0.37900   6.749 1.49e-11
factor(C)1903    2.76315    0.37895   7.292 3.06e-13
factor(C)1908    2.83415    0.37894   7.479 7.48e-14
factor(C)1913    2.81410    0.37901   7.425 1.13e-13
factor(C)1918    2.86228    0.37902   7.552 4.30e-14
factor(C)1923    2.91551    0.37906   7.691 1.45e-14
factor(C)1928    2.86546    0.37917   7.557 4.12e-14
factor(C)1933    2.86314    0.37936   7.547 4.44e-14
factor(C)1938    2.72290    0.37983   7.169 7.57e-13
factor(C)1943    2.68759    0.38066   7.060 1.66e-12
factor(C)1948   2.85099    0.38263   7.451 9.27e-14
factor(C)1953   2.81411    0.39456   7.132 9.87e-13

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 71776.18  on 109  degrees of freedom
Residual deviance:   829.63  on  81  degrees of freedom
AIC: 1744.7

Number of Fisher Scoring iterations: 4
```

The parameters in this model are: `intercept`: the log-rate in the refence category for age (40:40–44), in the reference cohort which in this model is the first cohort (1858 = 1943 − 85 which comprises persons born 5 years on either side of this, i.e. in the years 1853–1862 — but not *all* persons borm in this interval). Note however that there are no observations in the dataset in this category; it is actually a prediction purely outside the dataset. The rest of the parameters are log-rate-ratios relative to thsi category.

3. We now fit the model without intercept,

4. and with 1908 as the reference:

```
> ac.2 <- glm( D ~ factor(A) - 1 + relevel(factor(C),"1908") + offset(log(Y)),
+                  family=poisson )
```

The age-parameters now represent the estimated age-specific log-incidence rates from the 1908 cohort.

5. The range of birth dates represented in the cohort 1908 is from 1.1.1903–31.12.1912. Only those born on 1.1.1908 are not represented in any other cohort. Hence the name “synthetic” cohort.

6. We now extract the age-specific incidence rates with 95% c.i.s from the model using `ci.lin`:

```
> age.cf <- ci.lin( ac.2, subset="A", Exp=TRUE)[,5:7]
> matplot( as.numeric(levels(factor(A)))+2.5, age.cf,
+          log="y", type="l", lty=1, lwd=c(3,1,1), col="black" )
```

7. Simiularly we extract the cohort-specific rate-ratio parameters, but we recall that the 1908 cohiort is missing from the estimates:

```
> RR.cf <- ci.lin( ac.2, subset="C", Exp=TRUE )[,5:7]
> wh <- grep( "1908", levels(factor(C)) ) - 1
> RR.cf <- rbind( RR.cf[1:wh,], c(1,1,1), RR.cf[-(1:wh),] )
> RR.cf

                                 exp(Est.)       2.5%      97.5%
relevel(factor(C), "1908")1858 0.05876855 0.02796331 0.12350977
relevel(factor(C), "1908")1863 0.05938629 0.04146987 0.08504321
relevel(factor(C), "1908")1868 0.09820451 0.08277938 0.11650395
relevel(factor(C), "1908")1873 0.13435012 0.12110391 0.14904520
relevel(factor(C), "1908")1878 0.16850582 0.15647290 0.18146408
relevel(factor(C), "1908")1883 0.24290000 0.22987080 0.25666770
relevel(factor(C), "1908")1888 0.39765267 0.38150319 0.41448578
relevel(factor(C), "1908")1893 0.57498146 0.55558344 0.59505676
relevel(factor(C), "1908")1898 0.75865134 0.73613440 0.78185703
relevel(factor(C), "1908")1903 0.93146302 0.90603144 0.95760844
                               1.00000000 1.00000000 1.00000000
relevel(factor(C), "1908")1913 0.98015018 0.95413843 1.00687107
relevel(factor(C), "1908")1918 1.02853256 1.00032662 1.05753381
relevel(factor(C), "1908")1923 1.08476601 1.05335624 1.11711238
relevel(factor(C), "1908")1928 1.03180855 0.99700213 1.06783011
relevel(factor(C), "1908")1933 1.02941676 0.98736788 1.07325636
relevel(factor(C), "1908")1938 0.89472043 0.84629736 0.94591416
relevel(factor(C), "1908")1943 0.86367228 0.80177907 0.93034332
relevel(factor(C), "1908")1948 1.01698726 0.91442192 1.13105675
relevel(factor(C), "1908")1953 0.98016430 0.78931406 1.21716072

> matplot( as.numeric(levels(factor(C))), RR.cf,
+          type="l", log="y", lwd=c(3,1,1), lty=1, col="black" )
```

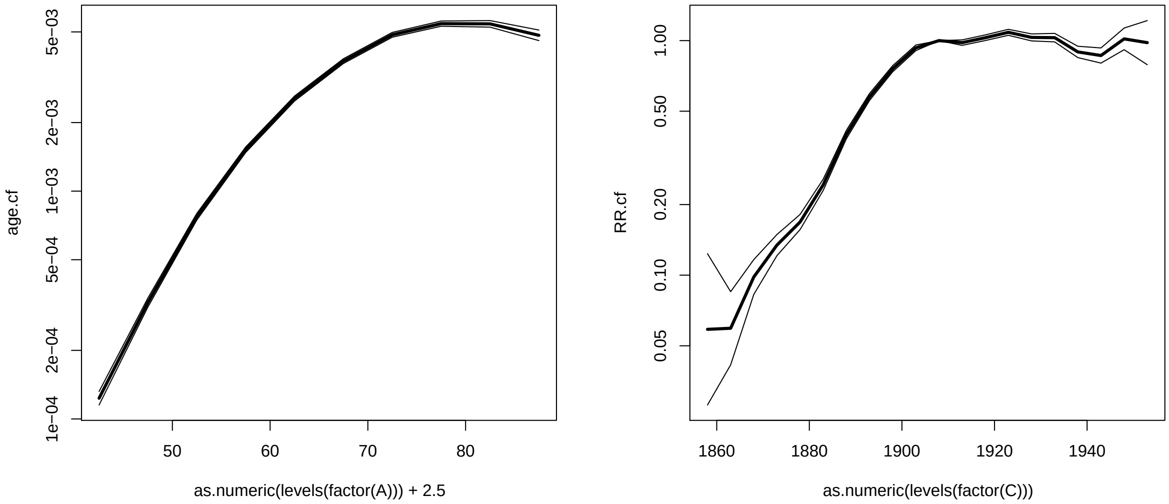

Figure 3.13: *Age-specific rates and rate-ratios relative to the cohort 1908.*

We could of course do as in the precvious exercise and combine the two plots in one which is properly scales on both axes:

```
> alim <- range( A ) + c(0,5)
> clim <- range( C ) + c(-2.5,2.5)
> # Compute limits explicitly
> rlim <- range(age.cf*10^5)*c(1/1.05,1.05)
> RRlim <- 10^(log10(rlim)-ceiling(mean(log10(rlim)))) / 2
> # Determin reltive width of plots
> layout( rbind( c(1,2) ), widths=c(diff(alim),diff(clim)) )
> # No space on the sides of the plots, only outer space
> par( mar=c(4,0,1,0), oma=c(0,4,0,4), mgp=c(3,1,0)/1.5, las=1 )
> matplot( as.numeric(levels(factor(A)))+2.5, age.cf*10^5,
+          type="l", lwd=c(3,1,1), lty=1, col="black",
+          log="y", xaxs="i", xlim=alim, xlab="Age", ylim=rlim )
> mtext( "Male lung cancer per 100,000", las=0, side=2, outer=T, line=2.5 )
> matplot( as.numeric(levels(factor(C))), RR.cf,
+          type="l", lwd=c(3,1,1), lty=1, col="black",
+          log="y", xlab="Date of birth", xlim=clim, yaxt="n", ylim=RRlim, ylab="" )
> abline( h=1 )
> points( 1908, 1, pch=1, lwd=3 )
> axis( side=4 )
> mtext( "Rate ratio", side=4, outer=T, las=0, line=2.5 )
```

The resulting plot is in figure 3.14

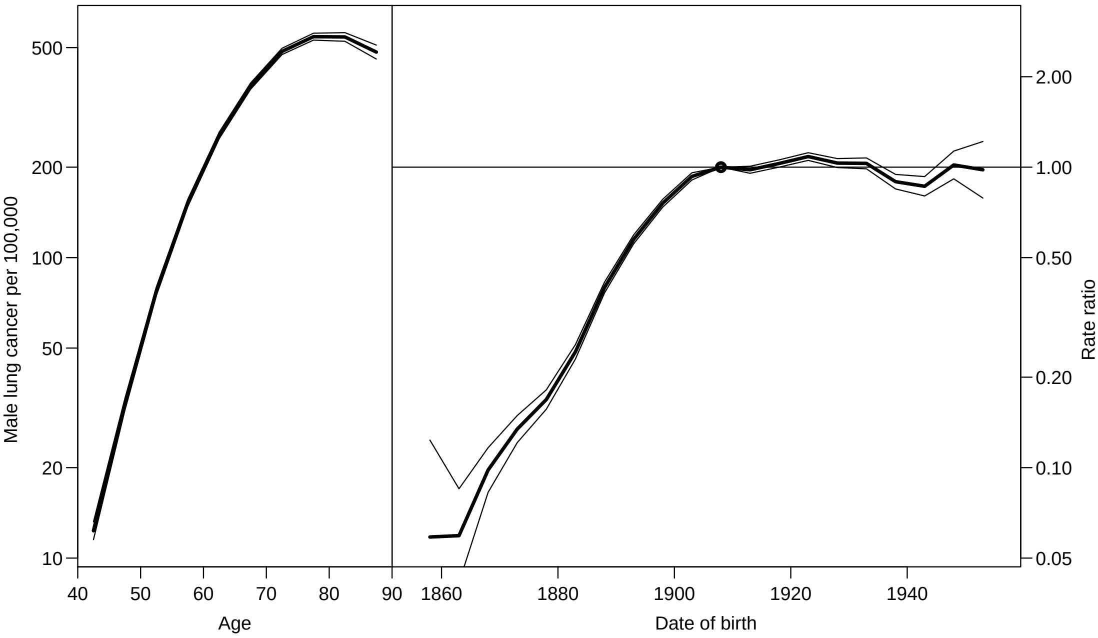

Figure 3.14: *Age-specific rates and rate-ratios relative to the period 1968–72, extracted from the age-cohort model. Note the axes with physically equal scaling.*

8. Now we load the estimates from the age-period model, and plot the estimated age-specific rates from the two models on top of each other. First

```
> load( file = "../data/age-per-est.Rdata" )
> matplot( as.numeric(levels(factor(A)))+2.5, age.cf,
+          log="y", type="l", lty=1, lwd=c(3,1,1), col="black" )
> matlines( age.pt, arates,
+           type="l", lty=1, lwd=c(3,1,1), col="blue" )
```

The difference between the curves in figure 3.15, comes from the fact that the rates are increasing by time. The estimates from the age-cohort model refer to rates in a "true" cohort, whereas those from the age-period model refers to cross-sectional rates, where succcesively older persons are from successively older cohorts (i.e. where rates were lower overall).

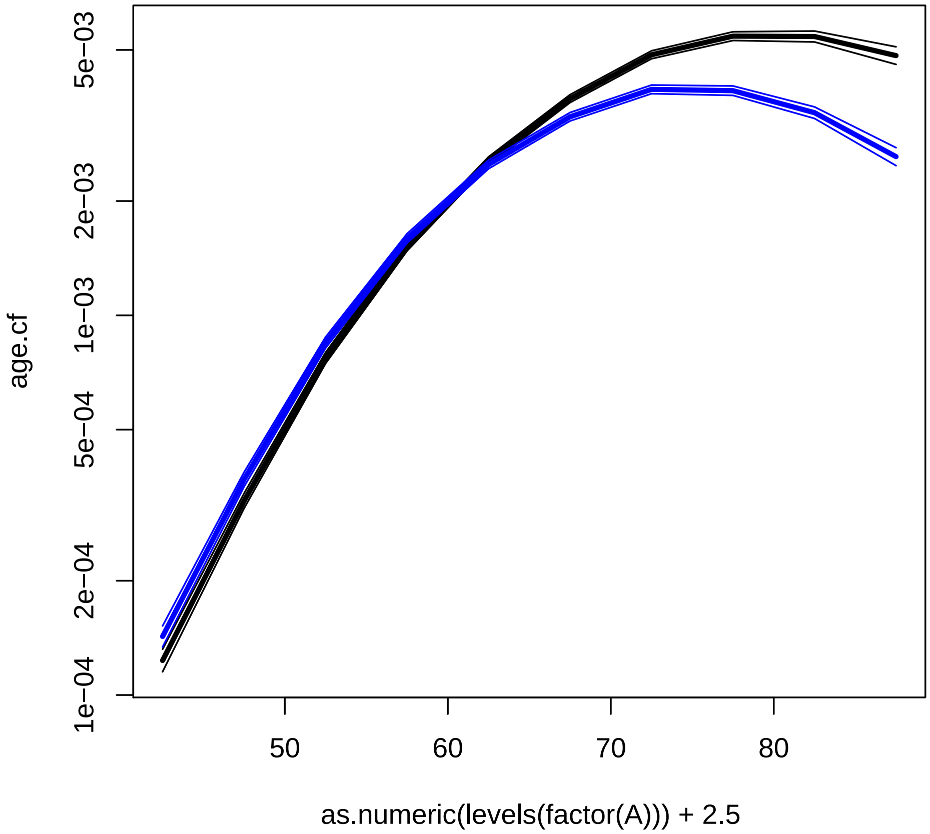

Figure 3.15: *Age-specific rates from the age-cohort model (black) and from the age-period model (blue).*

3.6 Age-drift model

This exercise is aimed at introducing the age-drift model and make you familiar with the two different ways of parametrizing this model. Like the two previous exercises it is based on the male lung cancer data.

1. First we read the data in the file `lung5-M.txt` and create the cohort variable:

```
> lung <- read.table( "../data/lung5-M.txt", header=T )
> lung$C <- lung$P - lung$A
> attach( lung )
> table( C )

C
1858 1863 1868 1873 1878 1883 1888 1893 1898 1903 1908 1913 1918 1923 1928 1933
   1    2    3    4    5    6    7    8    9   10   10    9    8    7    6    5
1938 1943 1948 1953
   4    3    2    1
```

2.

3. We fit the model to have age-parameters that refer to the period 1968–72. The midpoint of this period is 1970.5 (periods are coded by their left endpoint, so we need to add 2.5 years to 1968 to get the midpoint).

```
> mp <- glm( D ~ -1 + factor(A) + I(P-1970.5) + offset( log(Y) ),
+            family=poisson, data=lung )
> ci.lin( mp )[,1:2]
```

```
                  Estimate       StdErr
factor(A)40    -9.05098276 0.0309680655
factor(A)45    -8.10126627 0.0198136394
factor(A)50    -7.25742965 0.0136740899
factor(A)55    -6.61045586 0.0104218606
factor(A)60    -6.15631240 0.0087801489
factor(A)65    -5.87004530 0.0082214194
factor(A)70    -5.70814910 0.0085601026
factor(A)75    -5.71952829 0.0103703662
factor(A)80    -5.85585022 0.0147011698
factor(A)85    -6.12052787 0.0257736805
I(P - 1970.5)  0.02330670 0.0002569689
```

The parameters now represent the log-rates in each of the age-classes at 1970.5, i.e. in the period 1968–72. The period-paramter is the the annual change in log-rates.

4. We now fit the same model, but with cohort as the continuous variable, centered around 1908:

```
> mc <- glm( D ~ -1 + factor(A) + I(C-1908) + offset( log(Y) ),
+            family=poisson, data=lung )
> ci.lin( mc )[,1:2]
```

```
                Estimate       StdErr
factor(A)40 -9.57538357 0.0317010811
factor(A)45 -8.50913356 0.0205578133
factor(A)50 -7.54876343 0.0142616192
factor(A)55 -6.78525612 0.0107586856
factor(A)60 -6.21457915 0.0088754237
factor(A)65 -5.81177854 0.0081553406
factor(A)70 -5.53334883 0.0084736086
factor(A)75 -5.42819451 0.0104021596
factor(A)80 -5.44798293 0.0148625870
factor(A)85 -5.59612707 0.0259850279
I(C - 1908)  0.02330670 0.0002569689
```

5. We see that the estimated slope (the drfit!) is exactly the same as in the period-model, but the age-estimates are not.

   Moreover the two are really the same model just parametrized differently; the residual deviances are the same:

```
> c( summary( mp )$deviance,
+    summary( mc )$deviance )
```

```
[1] 6417.381 6417.381
```

6. If we write how the cohort model is parametrized we have:

$$\begin{aligned} log(\lambda_{ap}) &= \alpha_a + \beta(c - 1908) \\ &= \alpha_a + \beta(p - a - 1908) \\ &= [\alpha_a + \beta(62.5 - a)] + \beta(p - 1970.5) \end{aligned}$$

   The expression in the square brackets are the age-parameters in the age-period model. Hence, the age parameters are linked by a simple linear relation, which is easily verified empirically:

```
> ap <- ci.lin( mp )[1:10,1]
> ac <- ci.lin( mc )[1:10,1]
> c.sl <- ci.lin( mc )[11,1]
> a.pt <- seq(40,85,5)
> cbind( ap, ac + c.sl*(62.5-a.pt) )

                   ap
factor(A)40 -9.050983 -9.050983
factor(A)45 -8.101266 -8.101266
factor(A)50 -7.257430 -7.257430
factor(A)55 -6.610456 -6.610456
factor(A)60 -6.156312 -6.156312
factor(A)65 -5.870045 -5.870045
factor(A)70 -5.708149 -5.708149
factor(A)75 -5.719528 -5.719528
factor(A)80 -5.855850 -5.855850
factor(A)85 -6.120528 -6.120528
```

7.

```
> matplot( a.pt + 2.5, cbind( ci.lin( mp, subset="A", Exp=TRUE )[,5:7],
+                             ci.lin( mc, subset="A", Exp=TRUE )[,5:7] ) * 10^5,
+          log="y", xlab="Age", ylab="Lung cancer incidence rates / 100,000",
+          type="l", lty=1, lwd=c(3,1,1), col=rep(gray(c(0.2,0.7)),each=3) )
```

8. The relative risks are from the model:

$$\log(\lambda_{ap}) = \alpha_p + \delta(p - 1970.5)$$

   Therefore, with an x-variable: (1943,...,1993) + 2.5, the relative risk will be:

$$\text{RR} = \hat{\delta} \times x$$

   and the upper and lower confidence bands:

$$\text{RR} = (\hat{\delta} \pm 1.96 \times \text{s.e.}(\delta)) \times x$$

   We can find the estimated RRs with confidence intervals using a suitable 1-column contrast matrix. We of course need a separate one for period and cohort since these cover different time-spans:

```
> p.pt <- seq(min(P),max(P),,10)+2.5
> c.pt <- seq(min(C),max(C),,10)
> ctr.p <- cbind( p.pt - 1970.5 )
> ctr.c <- cbind( c.pt - 1908   )
> matplot( c.pt, ci.lin( mc, subset="C", ctr.mat=ctr.c, Exp=TRUE )[,5:7],
+          log="y", xlab="Calendar time", ylab="Rate ratio", xlim=c(1850,2000),
+          type="l", lty=1, lwd=c(3,1,1), col=gray(0.2) )
> matlines( p.pt, ci.lin( mp, subset="P", ctr.mat=ctr.p, Exp=TRUE )[,5:7],
+           type="l", lty=1, lwd=c(3,1,1), col=gray(0.7) )
> abline(h=1)
> points( c(1908,1970.5), c(1,1), pch=16 )
```

The effect of time (the drift) is the same for the two parametrizations, but the age-specific rates refer either to cross-sectional rates (period drift) or longitudinal rates (cohort drift).

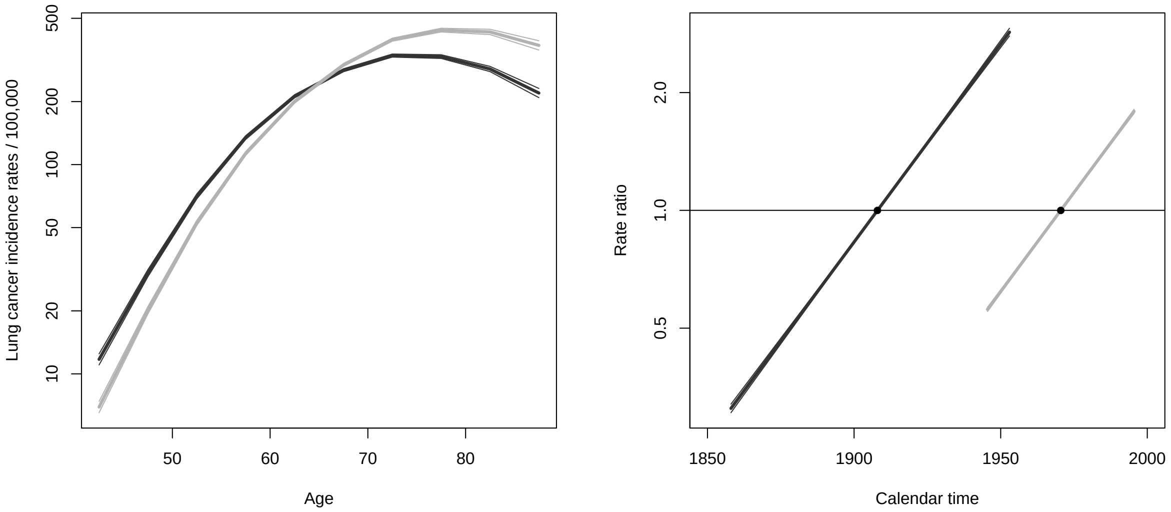

Figure 3.16: *Age-specific rates from the age-drift model (left) and the rate-ratios as esimated under the two diffferent parametrizations.*

3.7 Age-period-cohort model

We will need the results from the age-period, the age-cohort and the age-drift models in this exercise so we briefly fit these models afte we have read data.

1. Read the data in the file `lung5-M.txt` as in the tabulation exercise:

```
> lung <- read.table( "../data/lung5-M.txt", header=T )
> str( lung )

'data.frame':        110 obs. of  4 variables:
 $ A: int  40 40 40 40 40 40 40 40 40 40 ...
 $ P: int  1943 1948 1953 1958 1963 1968 1973 1978 1983 1988 ...
 $ D: int  80 81 73 99 82 97 86 90 116 149 ...
 $ Y: num  694047 754770 769441 749265 757240 ...

> m.AP <- glm( D ~ factor(A) + factor(P) + offset( log(Y) ),
+              family=poisson, data=lung )
> m.AC <- glm( D ~ factor(A) + factor(P-A) + offset( log(Y) ),
+              family=poisson, data=lung )
> m.Ad <- glm( D ~ factor(A) + P + offset( log(Y) ),
+              family=poisson, data=lung )
```

2. We then fit the age-period-cohort model. Note that there is no such variable as the cohort in the dataset; we have to compute this as $P - A$. This is best done on the fly instead of cluttering up the dataframe with another variable. In the same go we fit the simplest model with age alone:

```
> m.APC <- glm( D ~ factor(A) + factor(P) + factor(P-A) + offset( log(Y) ),
+               family=poisson, data=lung )
> m.A   <- glm( D ~ factor(A) + offset( log(Y) ),
+               family=poisson, data=lung )
```

3. We can use `anova.glm` to test the different models in a sequence that gives all the valid comparisons:

```
> anova( m.A, m.Ad, m.AP, m.APC, m.AC, m.Ad, test="Chisq" )

Analysis of Deviance Table

Model 1: D ~ factor(A) + offset(log(Y))
Model 2: D ~ factor(A) + P + offset(log(Y))
Model 3: D ~ factor(A) + factor(P) + offset(log(Y))
Model 4: D ~ factor(A) + factor(P) + factor(P - A) + offset(log(Y))
Model 5: D ~ factor(A) + factor(P - A) + offset(log(Y))
Model 6: D ~ factor(A) + P + offset(log(Y))
  Resid. Df Resid. Dev  Df Deviance  P(>|Chi|)
1       100    15103.0
2        99     6417.4   1   8685.6        0.0
3        90     2723.5   9   3693.9        0.0
4        72      208.5  18   2514.9        0.0
5        81      829.6  -9   -621.1 6.245e-128
6        99     6417.4 -18  -5587.8        0.0
```

The successive test refer to:

   (a) linear effect of period/cohort

   (b) non-linear effect of period

(c) non-linear effect of cohort (in the presence of period)

(d) non-linear effect of period (in the presence of cohort)

(e) non-linear effect of cohort

Clearly, with the large amounts of data that we are dealing with, all of the tests are strongly significant, but comparing the likelihood ratio statistics there is some indication that the period curvature (non-linear component) is stronger than the cohort one.

4. When we want to fit models where some of the factor levels are merged or sorted as the first one, we use the `Relevel` function to do this:

```
> lung$Pr <- Relevel( factor(lung$P), list("first-last"=c("1943","1993") ) )
> lung$Cr <- Relevel( factor(lung$P-lung$A), "1908" )
```

We of course check that the resulst of these opertions are as we like the to be:

```
> with( lung, table(P,Pr) )

      Pr
P      first-last 1948 1953 1958 1963 1968 1973 1978 1983 1988
  1943         10    0    0    0    0    0    0    0    0    0
  1948          0   10    0    0    0    0    0    0    0    0
  1953          0    0   10    0    0    0    0    0    0    0
  1958          0    0    0   10    0    0    0    0    0    0
  1963          0    0    0    0   10    0    0    0    0    0
  1968          0    0    0    0    0   10    0    0    0    0
  1973          0    0    0    0    0    0   10    0    0    0
  1978          0    0    0    0    0    0    0   10    0    0
  1983          0    0    0    0    0    0    0    0   10    0
  1988          0    0    0    0    0    0    0    0    0   10
  1993         10    0    0    0    0    0    0    0    0    0
```

```
> with( lung, table(P-A,Cr) )

      Cr
       1908 1858 1863 1868 1873 1878 1883 1888 1893 1898 1903 1913 1918 1923
  1858    0    1    0    0    0    0    0    0    0    0    0    0    0    0
  1863    0    0    2    0    0    0    0    0    0    0    0    0    0    0
  1868    0    0    0    3    0    0    0    0    0    0    0    0    0    0
  1873    0    0    0    0    4    0    0    0    0    0    0    0    0    0
  1878    0    0    0    0    0    5    0    0    0    0    0    0    0    0
  1883    0    0    0    0    0    0    6    0    0    0    0    0    0    0
  1888    0    0    0    0    0    0    0    7    0    0    0    0    0    0
  1893    0    0    0    0    0    0    0    0    8    0    0    0    0    0
  1898    0    0    0    0    0    0    0    0    0    9    0    0    0    0
  1903    0    0    0    0    0    0    0    0    0    0   10    0    0    0
  1908   10    0    0    0    0    0    0    0    0    0    0    0    0    0
  1913    0    0    0    0    0    0    0    0    0    0    0    9    0    0
  1918    0    0    0    0    0    0    0    0    0    0    0    0    8    0
  1923    0    0    0    0    0    0    0    0    0    0    0    0    0    7
  1928    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  1933    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  1938    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  1943    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  1948    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  1953    0    0    0    0    0    0    0    0    0    0    0    0    0    0
      Cr
       1928 1933 1938 1943 1948 1953
  1858    0    0    0    0    0    0
  1863   0    0    0    0    0    0
  1868   0    0    0    0    0    0
  1873   0    0    0    0    0    0
  1878   0    0    0    0    0    0
  1883   0    0    0    0    0    0
  1888   0    0    0    0    0    0
  1893   0    0    0    0    0    0
  1898   0    0    0    0    0    0
  1903   0    0    0    0    0    0
  1908   0    0    0    0    0    0
  1913   0    0    0    0    0    0
  1918   0    0    0    0    0    0
  1923   0    0    0    0    0    0
  1928   6    0    0    0    0    0
  1933   0    5    0    0    0    0
  1938   0    0    4    0    0    0
  1943   0    0    0    3    0    0
  1948   0    0    0    0    2    0
  1953   0    0    0    0    0    1
```

5. We can now fit the models with these factors:

```
> m.APC1 <- glm( D ~ -1 + factor(A) + factor(Pr) + factor(Cr) + offset( log(Y) ),
+                  family=poisson, data=lung )
> m.APC1$coef
```

```
   factor(A)40    factor(A)45    factor(A)50    factor(A)55    factor(A)60
  -9.328701115   -8.334529816   -7.454972743   -6.769070541   -6.241541847
   factor(A)65    factor(A)70    factor(A)75    factor(A)80    factor(A)85
  -5.849698430   -5.568204628   -5.440013453   -5.424818364   -5.526811866
factor(Pr)1948 factor(Pr)1953 factor(Pr)1958 factor(Pr)1963 factor(Pr)1968
   0.095424116    0.104770778    0.200248212    0.249105289    0.311058535
factor(Pr)1973 factor(Pr)1978 factor(Pr)1983 factor(Pr)1988 factor(Cr)1858
   0.295910526    0.294440825    0.249025339    0.103123244   -2.640060438
factor(Cr)1863 factor(Cr)1868 factor(Cr)1873 factor(Cr)1878 factor(Cr)1883
  -2.646673834   -2.149730193   -1.850593043   -1.645272902   -1.310031751
factor(Cr)1888 factor(Cr)1893 factor(Cr)1898 factor(Cr)1903 factor(Cr)1913
  -0.853337885   -0.520887869   -0.272223872   -0.079090672    0.005457283
factor(Cr)1918 factor(Cr)1923 factor(Cr)1928 factor(Cr)1933 factor(Cr)1938
   0.088513857    0.179650494    0.165997726    0.197699170    0.089012570
factor(Cr)1943 factor(Cr)1948 factor(Cr)1953
   0.086044048    0.293382042    0.307806293
```

The age-coefficients are log-rates (where the rates are in units person-year^{-1}, the cohort parameters are log-rate-ratios relative to a trend from the first to the last period.

6. We can use ci.lin to extract the parameters with confidence limits from this model:

```
> A.eff <- ci.lin( m.APC1, subset="A", Exp=TRUE )[,5:7]
> P.eff <- rbind( c(1,1,1),
+           ci.lin( m.APC1, subset="P", Exp=TRUE )[,5:7],
+                 c(1,1,1) )
> C.ref <- match( "1908", levels( with(lung,factor(P-A)) ) )
> C.eff <- rbind( c(1,1,1),
+           ci.lin( m.APC1, subset="C",
+                  Exp=TRUE )[,5:7] )[c(2:C.ref,1,C.ref:(nlevels(lung$Cr)-1)),]
```

In order to plot these we need the timepoints on the respective scales:

```
> A.pt <- sort( unique( lung$A ) ) + 2.5
> P.pt <- sort( unique( lung$P ) ) + 2.5
> C.pt <- sort( unique( lung$P-lung$A ) )
```

Then we can plot the estomated effects

```
> par( mfrow=c(1,3), las=2 )
> matplot( A.pt, A.eff,
+          xlab="Age", ylab="Rates",
+          log="y", type="l", lty=1, lwd=c(3,1,1), col="black" )
> matplot( P.pt, P.eff,
+          xlab="Period", ylab="RR",
+          log="y", type="l", lty=1, lwd=c(3,1,1), col="black" )
> abline( h=1 )
> matplot( C.pt, C.eff,
+          xlab="Cohort", ylab="RR",
+          log="y", type="l", lty=1, lwd=c(3,1,1), col="black" )
> abline( h=1 )
```

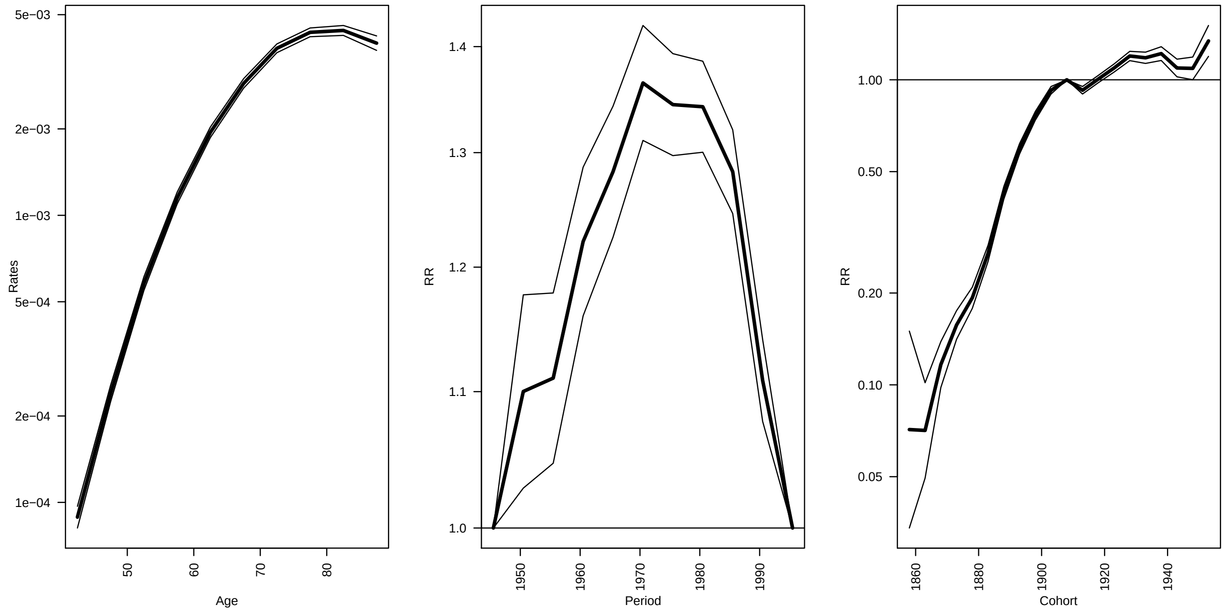

Figure 3.17: *Estimates of the age-period-cohort model estimates — raw as they are.*

This is is not a particularly informative plot, as the scales are all different — the rates are between 10^{-4} and 5×10^{-3}, whereas the cohort RRs are between 0.05 and slightly more than 1. So if we resacle the rate to rates per 1000, and then demand that all display have y-axis from 0.05 to 5, we get comprable displays:

```
> par( mfrow=c(1,3), las=2 )
> matplot( A.pt, A.eff*1000,
+          xlab="Age", ylab="Rates", ylim=c(0.05,5),
+          log="y", type="l", lty=1, lwd=c(3,1,1), col="black" )
> matplot( P.pt, P.eff,
+          xlab="Period", ylab="RR", ylim=c(0.05,5),
+          log="y", type="l", lty=1, lwd=c(3,1,1), col="black" )
> abline( h=1 )
> matplot( C.pt, C.eff,
+          xlab="Cohort", ylab="RR", ylim=c(0.05,5),
+          log="y", type="l", lty=1, lwd=c(3,1,1), col="black" )
> abline( h=1 )
```

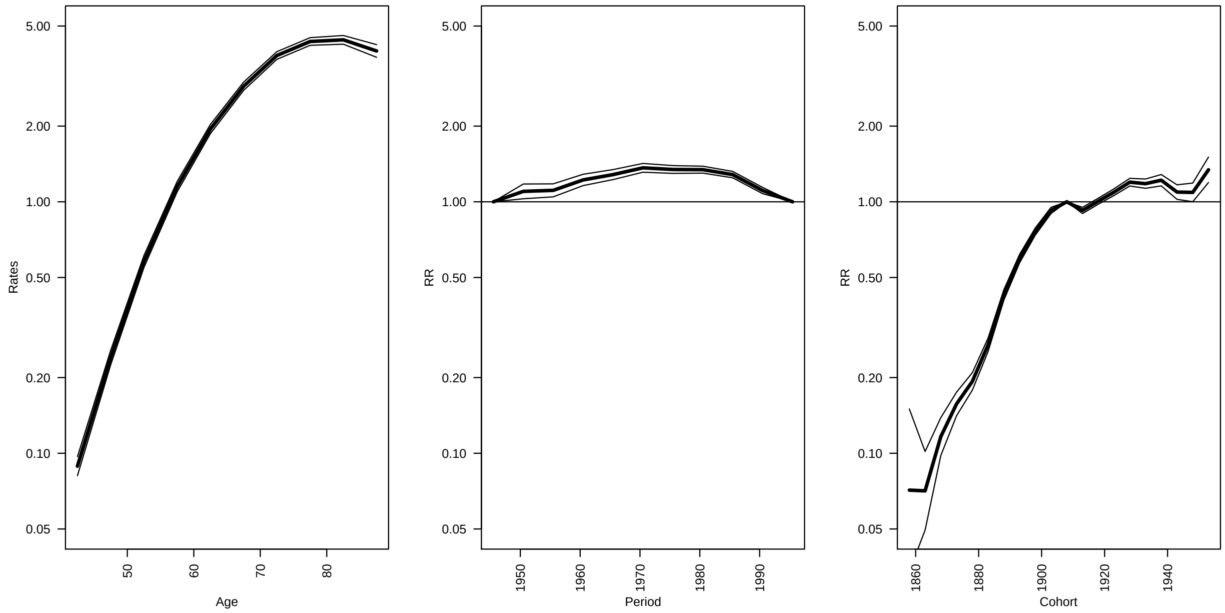

Figure 3.18: *Estimates of the age-period-cohort model estimates, scaled displays.*

The parameters in this model represent age-specific rates, that approximates the rates in the 1980 cohort (as predicted...), cohort RRs relative to this cohort, and finally period ”residual” RRs.

But note an explicit decision has been made as to how the period residuals are defined; namely as the deviations from the line between the periods 1943 and 1993.

7. We now fit the model with two cohorts aliased and one period as fixpoint. To decide which of the cohort to alias (and define as the first level of the factor) we tabulate no of observations and no of cases

```
> with( lung, table(P-A) )

1858 1863 1868 1873 1878 1883 1888 1893 1898 1903 1908 1913 1918 1923 1928 1933
   1    2    3    4    5    6    7    8    9   10   10    9    8    7    6    5
1938 1943 1948 1953
   4    3    2    1

> with( lung, tapply(D,list(P-A),sum) )

 1858  1863  1868  1873  1878  1883  1888  1893  1898  1903  1908  1913  1918
    7    30   134   371   752  1436  2822  4668  6934  9305 10873 10468  9438
 1923  1928  1933  1938  1943  1948  1953
 8010  5040  3036  1536   827   400    91
```

Rater arbitraily we decide on 1878 and 1933; the numbers of these in the cohort numbers are computed by:

```
> C.ref.pos <- with( lung, match( c("1878","1933"), levels( factor(P-A) ) ) )
> P.ref.pos <- with( lung, match( "1973", levels( factor(P) ) ) )

> lung$Cx <- Relevel( factor(lung$P-lung$A), list("first-last"=c("1878","1933") ) )
> lung$Px <- Relevel( factor(lung$P), "1973" )
```

With these definitions we can now fit the model with the alternative parametrization:

```
> m.APC2 <- glm( D ~ -1 + factor(A) + factor(Px) + factor(Cx) + offset( log(Y) ),
+                family=poisson, data=lung )
> m.APC2$coef
```

```
   factor(A)40    factor(A)45    factor(A)50    factor(A)55    factor(A)60
   -8.83509142    -8.00846304    -7.29644888    -6.77808959    -6.41810381
   factor(A)65    factor(A)70    factor(A)75    factor(A)80    factor(A)85
   -6.19380331    -6.07985243    -6.11920417    -6.27155199    -6.54108841
factor(Px)1943 factor(Px)1948 factor(Px)1953 factor(Px)1958 factor(Px)1963
   -1.30116802    -1.03820099    -0.86131141    -0.59829106    -0.38189107
factor(Px)1968 factor(Px)1978 factor(Px)1983 factor(Px)1988 factor(Px)1993
   -0.15239491     0.16607322     0.28820064     0.30984147     0.37426114
factor(Cx)1858 factor(Cx)1863 factor(Cx)1868 factor(Cx)1873 factor(Cx)1883
   -0.32461587    -0.49877219    -0.16937146    -0.03777722     0.16769824
factor(Cx)1888 factor(Cx)1893 factor(Cx)1898 factor(Cx)1903 factor(Cx)1908
    0.45684919     0.62175629     0.70287737     0.72846765     0.64001541
factor(Cx)1913 factor(Cx)1918 factor(Cx)1923 factor(Cx)1928 factor(Cx)1938
    0.47792978     0.39344343     0.31703715     0.13584147    -0.27622952
factor(Cx)1943 factor(Cx)1948 factor(Cx)1953
   -0.44674095    -0.40694587    -0.56006454
```

We note that it is only the parametization that differs; the fitted model is the same:

```
> summary( m.APC  )$deviance

[1] 208.5476

> summary( m.APC1 )$deviance

[1] 208.5476

> summary( m.APC2 )$deviance

[1] 208.5476
```

8. We use the same points for the age, period and cohort as before, but now extract the parameters in a slightly different way:

```
> A.Eff <- ci.lin( m.APC2, subset="A", Exp=TRUE )[,5:7]
> P.Eff <- ci.lin( m.APC2, subset="P", Exp=TRUE )[,5:7]
> nP <- nrow(P.Eff)
> P.Eff <- rbind( P.Eff[1:(P.ref.pos-1),],c(1,1,1),P.Eff[P.ref.pos:nP,])
> C.Eff <- ci.lin( m.APC2, subset="C",Exp=TRUE )[,5:7]
> nC <- nrow(C.Eff)
> C.Eff <- rbind(C.Eff[1:(C.ref.pos[1]-1),],
+               c(1,1,1),
+               C.Eff[(C.ref.pos[1]):(C.ref.pos[2]-2),],
+               c(1,1,1),
+               C.Eff[(C.ref.pos[2]-1):nC,] )
```

We can now plot the two sets of parameters in the same plots:

```
> par( mfrow=c(1,3), las=2 )
> matplot( A.pt, cbind(A.eff,A.Eff)*1000,
+          xlab="Age", ylab="Rates", ylim=c(0.05,5),
+          log="y", type="l", lty=1, lwd=c(3,1,1), col=rep(c("black","blue"),each=3) )
> matplot( P.pt, cbind(P.eff,P.Eff),
+          xlab="Period", ylab="RR", ylim=c(0.05,5),
+          log="y", type="l", lty=1, lwd=c(3,1,1), col=rep(c("black","blue"),each=3) )
```

```
> abline( h=1 )
> matplot( C.pt, cbind(C.eff,C.Eff),
+          xlab="Cohort", ylab="RR", ylim=c(0.05,5),
+          log="y", type="l", lty=1, lwd=c(3,1,1), col=rep(c("black","blue"),each=3) )
> abline( h=1 )
```

It is clear from the estimates that very different displays can be obtained from different parametrizations.

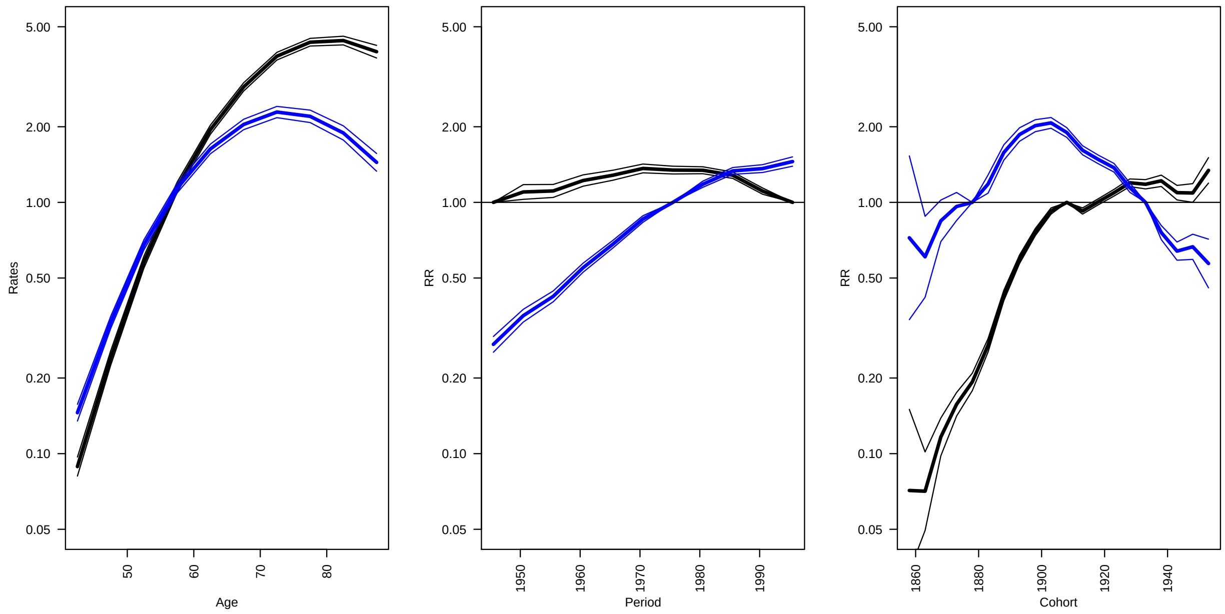

Figure 3.19: *Estimates of the age-period-cohort model estimates, from the two different parametrizations.*

3.8 Age-period-cohort model for triangles

The following exercise is aimed at showing the problems associated with age-period-cohort modelling for triangular data.

Also you will learn how to overcome these problems by parametric modelling of the three effects.

1. Read the Danish male lung cancer data tabulated by age period *and* birth cohort, `lung5-Mc.txt`. List the first few lines of the dataset and make sure you understand what the variables refer to. Also define nthe synthetic cohorts as `P5-A5`:

```
> ltri <- read.table( "../data/lung5-Mc.txt", header=T )
> ltri$S5 <- ltri$P5 - ltri$A5
> attach( ltri )
```

2. Make a Lexis diagram showing the subdivision of the follow-data. You will explore the function `Lexis.diagram`.

   Try as an esoteric exercise to plot the number of cases in each of the triangles.

```
> Lexis.diagram( age=c(40,90), date=c(1943,1998), coh.grid=TRUE )
```

3. Use the variables `A5` and `P5` to fit a traditional age-period-cohort model with synthetic cohort defined by `P5-A5`:

```
> ms <- glm( D ~ -1 + factor(A5) + factor(P5) + factor(S5) + offset(log(Y)),
+            family=poisson, data=ltri )
> summary( ms )$df

[1]  38 182  39
```

   How many parameters does this model have?

4. Now try to fit the model with the "real" cohort:

```
> mc <- glm( D ~ -1 + factor(A5) + factor(P5) + factor(C5) + offset(log(Y)),
+            family=poisson, data=ltri )
> summary( mc )$df

[1]  40 180  40
```

   How many parameters does this model have?

5. Plot the parameter estimates from the two models on top of each other, with confidence intervals. Remember to put the right scales on the plots.

```
> par( mfrow=c(1,3) )
> a.pt <- as.numeric( levels(factor(A5)) )
> p.pt <- as.numeric( levels(factor(P5)) )
> s.pt <- as.numeric( levels(factor(S5)) )
> c.pt <- as.numeric( levels(factor(C5)) )
> matplot( a.pt, ci.lin( ms, subset="A5", Exp=TRUE )[,5:7]/10^5,
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          xlab="Age", ylab="Rates", log="y" )
> matlines( a.pt, ci.lin( mc, subset="A5", Exp=TRUE )[,5:7]/10^5,
+           type="l", lty=1, lwd=c(3,1,1), col="blue" )
> matplot( p.pt, rbind( c(1,1,1), ci.lin( ms, subset="P5",Exp=TRUE )[,5:7] ),
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          xlab="Period", ylab="RR", log="y" )
> matlines( p.pt, rbind( c(1,1,1), ci.lin( mc, subset="P5",Exp=TRUE )[,5:7] ),
```

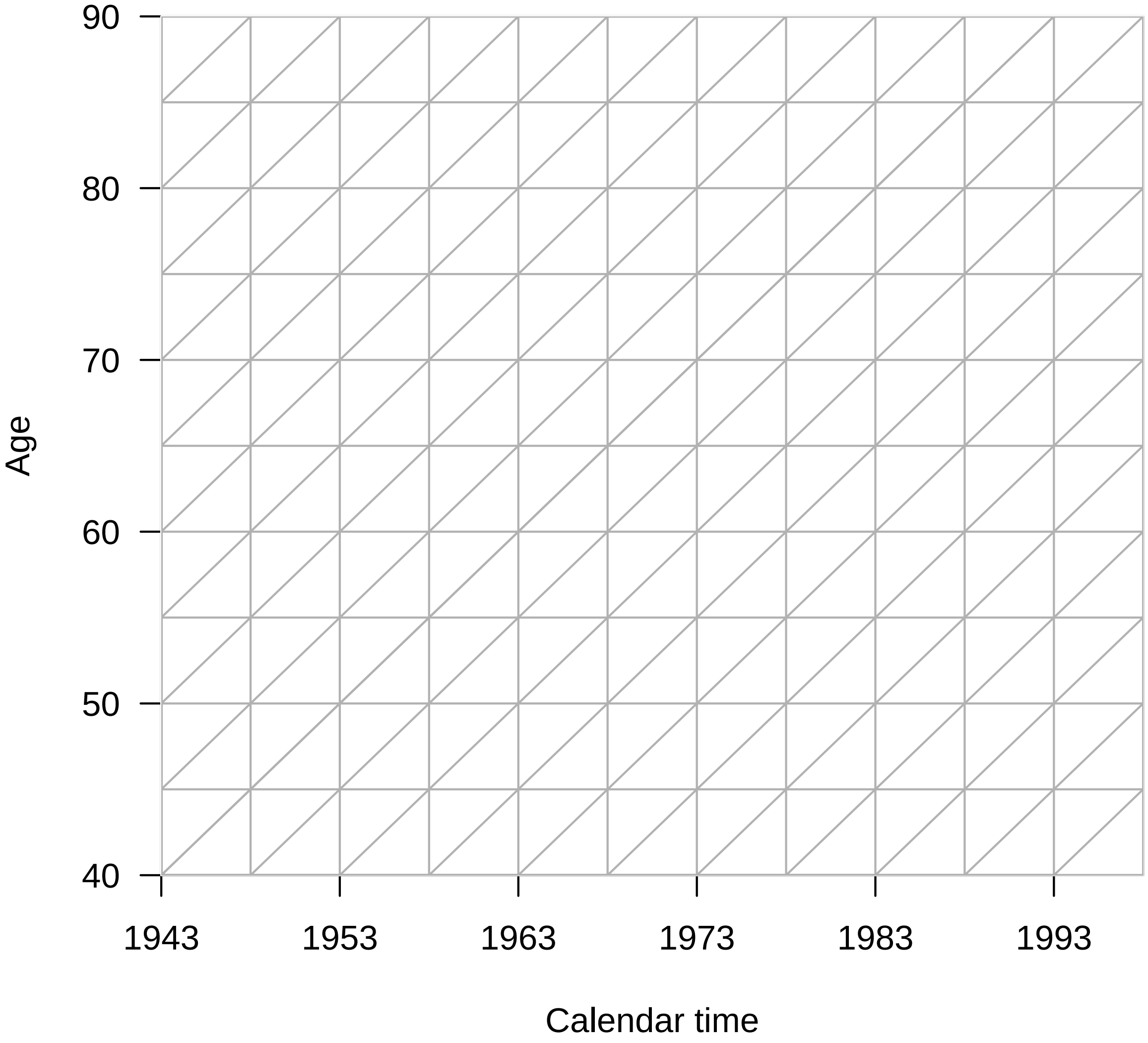

Figure 3.20: *Lexis diagram showing the extent of the data.*

```
+           type="l", lty=1, lwd=c(3,1,1), col="blue" )
> matplot( s.pt, rbind(c(1,1,1),ci.lin( ms, subset="S5", Exp=TRUE )[,5:7]),
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          xlab="Cohort", ylab="RR", log="y" )
> matlines( c.pt, rbind(c(1,1,1),ci.lin( mc, subset="C5", Exp=TRUE )[,5:7]),
+           type="l", lty=1, lwd=c(3,1,1), col="blue" )
```

It is seen that the confidence bands are much wider for the age and cohort effects but narrower for the period effects.

6. Now fit the model using the proper midpoints of the triangles as factor levels. How many parameters does this model have?

```
> mt <- glm( D ~ -1 + factor(Ax) + factor(Px) + factor(Cx) + offset(log(Y)),
```

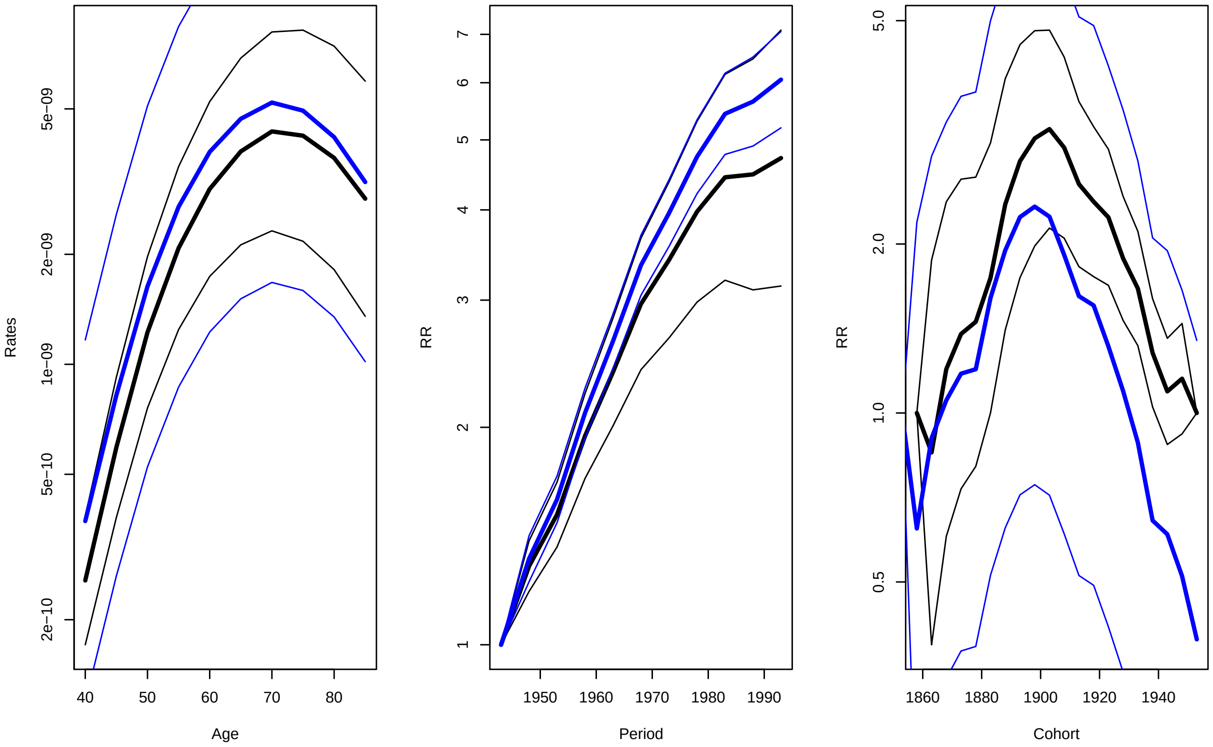

Figure 3.21: *Estimates from.*

```
+             family=poisson, data=ltri )
> summary( mt )$df

[1]  76 144  80
```

7. Plot the parameters from this model in three panels as for the previous two models.

```
> par( mfrow=c(1,3) )
> a.pt <- as.numeric( levels(factor(Ax)) )
> p.pt <- as.numeric( levels(factor(Px)) )
> c.pt <- as.numeric( levels(factor(Cx)) )
> matplot( a.pt, ci.lin( mt, subset="Ax", Exp=TRUE )[,5:7]/10^5,
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          xlab="Age", ylab="Rates", log="y" )
> matplot( p.pt, rbind( c(1,1,1), ci.lin( mt, subset="Px",Exp=TRUE )[,5:7] ),
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          xlab="Period", ylab="RR", log="y" )
> matplot( c.pt, rbind(c(1,1,1),ci.lin( mt, subset="Cx", Exp=TRUE )[,5:7]),
+          type="l", lty=1, lwd=c(3,1,1), col="black",
+          xlab="Cohort", ylab="RR", log="y" )
```

We see that the parameters clearly do not convey a reasonable picture of the effects; som severe indeterminacy has crept in.

8. What is the residual deviance of this model?

```
> summary( mt )$deviance
```

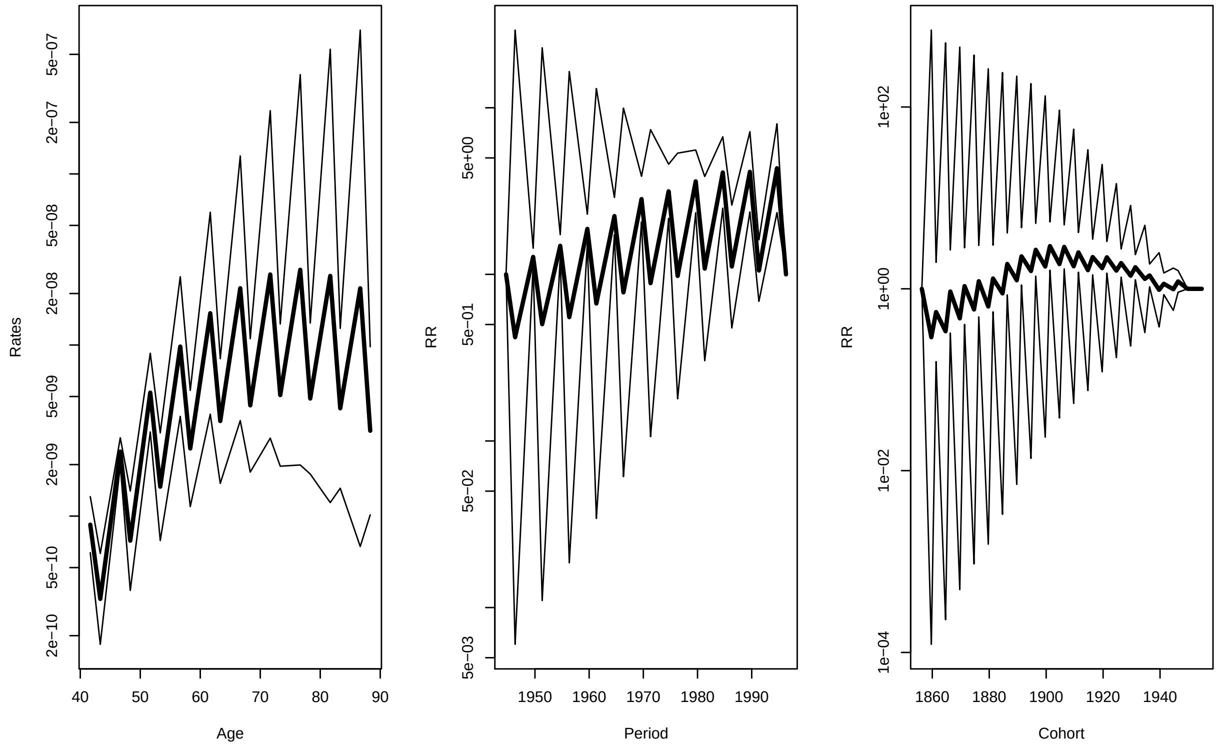

Figure 3.22: *Estimates from.*

```
[1] 284.7269
```

9. The dataset also has a variable `up`, which indicates whether the observation comes from an upper or lower triangle. Try to tabulate it against `P5-A5-C5`.

```
> table( up, P5-A5-S5 )
```

```
up    0
  0 110
  1 110
```

10. Fit an age-period cohort model separately for the subset of the dataset from the upper triangles and from the lowere triangles. What is the residual deviance from each of these models and what is the sum of these. Compare to the model using the proper midpoints as factor levels.

```
> m.up <- glm( D ~ -1 + factor(A5) + factor(P5) + factor(S5) + offset(log(Y)),
+               family=poisson, data=subset(ltri,up==1) )
> summary( m.up )$deviance
```

```
[1] 150.2703
```

```
> m.lo <- glm( D ~ -1 + factor(A5) + factor(P5) + factor(S5) + offset(log(Y)),
+               family=poisson, data=subset(ltri,up==0) )
> summary( m.lo )$deviance
```

```
[1] 134.4566
```

```
> summary( m.lo )$deviance + summary( m.up )$deviance

[1] 284.7269

> summary( mt )$deviance

[1] 284.7269
```

11. Next, repeat the plots of the parameters from the model using the proper midpoints as factor levels, but now super-posing the estimates (in different color) from each of the two models just fitted. What goes on?

```
> par( mfrow=c(1,3) )
> a.pt <- as.numeric( levels(factor(Ax)) )
> p.pt <- as.numeric( levels(factor(Px)) )
> c.pt <- as.numeric( levels(factor(Cx)) )
> a5.pt <- as.numeric( levels(factor(A5)) )
> p5.pt <- as.numeric( levels(factor(P5)) )
> s5.pt <- as.numeric( levels(factor(S5)) )
> matplot( a.pt, ci.lin( mt, subset="Ax", Exp=TRUE )[,5:7]/10^5,
+          type="l", lty=1, lwd=c(2,1,1), col=gray(0.7),
+          xlab="Age", ylab="Rates", log="y" )
> matpoints( a5.pt, ci.lin( m.up, subset="A5", Exp=TRUE )[,5:7]/10^5,
+            pch=c(16,3,3), col="blue" )
> matpoints( a5.pt, ci.lin( m.lo, subset="A5", Exp=TRUE )[,5:7]/10^5,
+            pch=c(16,3,3), col="red" )
> matplot( p.pt, rbind( c(1,1,1), ci.lin( mt, subset="Px",Exp=TRUE )[,5:7] ),
+          type="l", lty=1, lwd=c(2,1,1), col=gray(0.7),
+          xlab="Period", ylab="RR", log="y" )
> matpoints( p5.pt[-1], ci.lin( m.up, subset="P5", Exp=TRUE )[,5:7],
+            pch=c(16,3,3), col="blue" )
> matpoints( p5.pt[-1], ci.lin( m.lo, subset="P5", Exp=TRUE )[,5:7],
+            pch=c(16,3,3), col="red" )
> matplot( c.pt, rbind(c(1,1,1),ci.lin( mt, subset="Cx", Exp=TRUE )[,5:7]),
+          type="l", lty=1, lwd=c(2,1,1), col=gray(0.7),
+          xlab="Cohort", ylab="RR", log="y" )
> matpoints( s5.pt[-1], ci.lin( m.up, subset="S5", Exp=TRUE )[,5:7],
+            pch=c(16,3,3), col="blue" )
> matpoints( s5.pt[-1], ci.lin( m.lo, subset="S5", Exp=TRUE )[,5:7],
+            pch=c(16,3,3), col="red" )
```

The model fitted with the “correct” factor levels is actually two different models. This is because observations in upper triangles are modelled by one set of the parameters, and those in lower triangel by another set of parameters.

Because of the ordering of the levels, the parametrization is different, but that is all.

There is no way out of the squeeze, except by resorting to parametric models for the actual underlying scales, abandoning the factor modelling.

12. Load the splines package and fit a model using the correct midpoints of the triangles as quantitative variables in restricted cubic splines, using the function `ns`:

```
> library( splines )
> mspl <- glm( D ~ -1 + ns(Ax,df=7,intercept=T)
+                     + ns(Px,df=6,intercept=F)
+                     + ns(Cx,df=6,intercept=F) + offset(log(Y)),
+              family=poisson, data=ltri )
> summary( mspl )
```

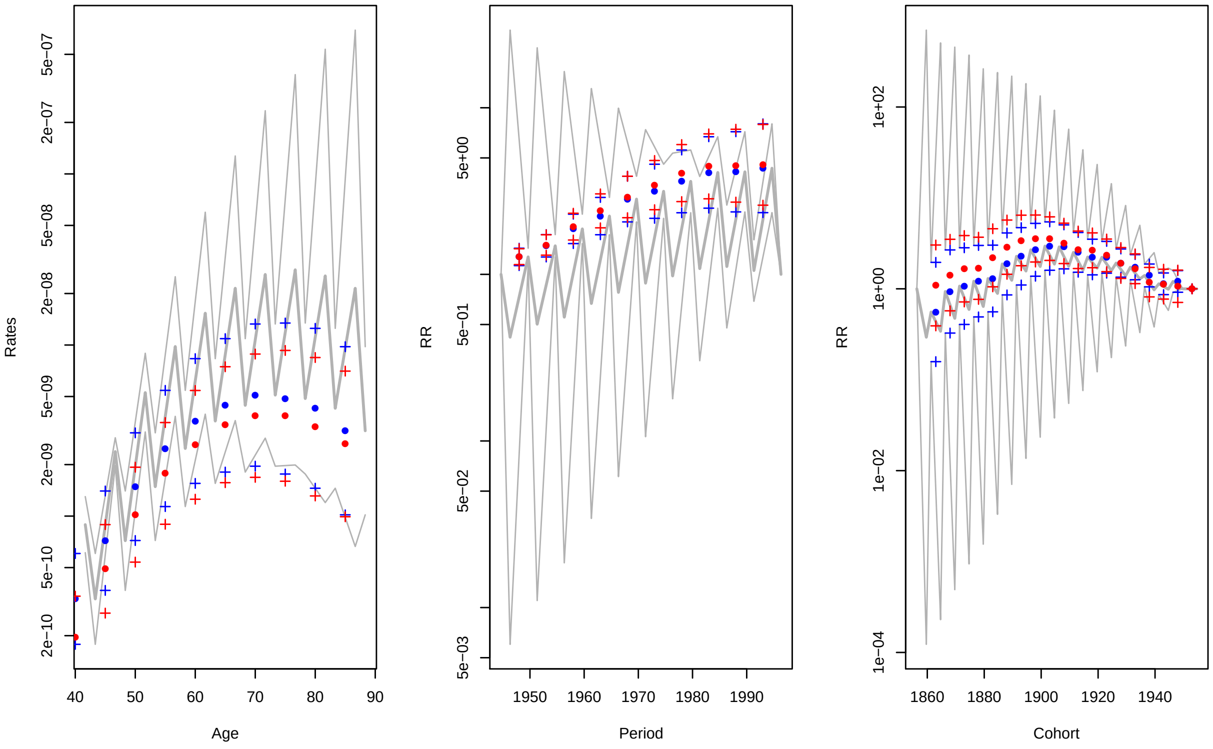

Figure 3.23: *Estimates from.*

```
Call:
glm(formula = D ~ -1 + ns(Ax, df = 7, intercept = T) + ns(Px,
    df = 6, intercept = F) + ns(Cx, df = 6, intercept = F) +
    offset(log(Y)), family = poisson, data = ltri)

Deviance Residuals:
     Min         1Q     Median         3Q        Max
-3.72761   -0.88692   -0.01217    0.93283    3.47380

Coefficients: (1 not defined because of singularities)
                                Estimate Std. Error z value Pr(>|z|)
ns(Ax, df = 7, intercept = T)1  -8.08248    0.09584 -84.329  < 2e-16
ns(Ax, df = 7, intercept = T)2  -8.81421    0.11261 -78.271  < 2e-16
ns(Ax, df = 7, intercept = T)3  -8.20301    0.11520 -71.209  < 2e-16
ns(Ax, df = 7, intercept = T)4  -7.90599    0.11814 -66.921  < 2e-16
ns(Ax, df = 7, intercept = T)5  -3.98298    0.08558 -46.540  < 2e-16
ns(Ax, df = 7, intercept = T)6 -21.35542    0.24841 -85.967  < 2e-16
ns(Ax, df = 7, intercept = T)7   0.70588    0.05540  12.741  < 2e-16
ns(Px, df = 6, intercept = F)1   0.59989    0.03777  15.883  < 2e-16
ns(Px, df = 6, intercept = F)2   0.94029    0.04319  21.771  < 2e-16
ns(Px, df = 6, intercept = F)3   1.18582    0.04354  27.237  < 2e-16
ns(Px, df = 6, intercept = F)4   1.22421    0.04204  29.122  < 2e-16
ns(Px, df = 6, intercept = F)5   1.46929    0.08247  17.816  < 2e-16
ns(Px, df = 6, intercept = F)6   1.07376    0.04202  25.555  < 2e-16
ns(Cx, df = 6, intercept = F)1   1.57834    0.10334  15.273  < 2e-16
ns(Cx, df = 6, intercept = F)2   1.60219    0.11202  14.303  < 2e-16
ns(Cx, df = 6, intercept = F)3   1.37407    0.10178  13.500  < 2e-16
ns(Cx, df = 6, intercept = F)4   1.03167    0.07211  14.306  < 2e-16
ns(Cx, df = 6, intercept = F)5   1.19310    0.21716   5.494 3.93e-08
ns(Cx, df = 6, intercept = F)6        NA         NA      NA       NA

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1.0037e+08  on 220  degrees of freedom
Residual deviance: 4.3344e+02  on 202  degrees of freedom
AIC: 2026.7

Number of Fisher Scoring iterations: 4

> summary( mt )$deviance - summary( mspl )$deviance

[1] -148.7082

> summary( mt )$df       - summary( mspl )$df

[1]  58 -58  61
```

13. How do the deviances compare?

14. Make a prediction of the terms, using `predict.glm` using the argument `type="terms"` and `se.fit=TRUE`. Remember to look up the help page for `predict.glm`.

```
> pspl <- predict( mspl, type="terms", se.fit=TRUE )
> str(pspl)

List of 3
 $ fit           : num [1:220, 1:3] -10.8 -11.1 -10.8 -11.1 -10.8 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:220] "1" "2" "3" "4" ...
  .. ..$ : chr [1:3] "ns(Ax, df = 7, intercept = T)" "ns(Px, df = 6, intercept = F)" "ns(Cx, df
  ..- attr(*, "constant")= num 0
 $ se.fit        : num [1:220, 1:3] 0.107 0.109 0.107 0.109 0.107 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:220] "1" "2" "3" "4" ...
  .. ..$ : chr [1:3] "ns(Ax, df = 7, intercept = T)" "ns(Px, df = 6, intercept = F)" "ns(Cx, df
 $ residual.scale: num 1

> a.ord <- order( ltri$Ax )
> p.ord <- order( ltri$Px )
> c.ord <- order( ltri$Cx )
> par( mfrow=c(1,3) )
> matplot( Ax[a.ord], exp(cbind( pspl$fit[,1], pspl$se.fit[,1] )[a.ord,] %*% ci.mat())*10^5,
+          type="l", lty=1, lwd=c(2,1,1), col=gray(0.2),
+          xlab="Age", ylab="Rates", log="y" )
> matplot( Px[p.ord], exp(cbind( pspl$fit[,2], pspl$se.fit[,2] )[p.ord,] %*% ci.mat()),
+          type="l", lty=1, lwd=c(2,1,1), col=gray(0.2),
+          xlab="Period", ylab="RR", log="y" )
> matplot( Cx[c.ord], exp(cbind( pspl$fit[,3], pspl$se.fit[,3] )[c.ord,] %*% ci.mat()),
+          type="l", lty=1, lwd=c(2,1,1), col=gray(0.2),
+          xlab="Cohort", ylab="RR", log="y" )
```

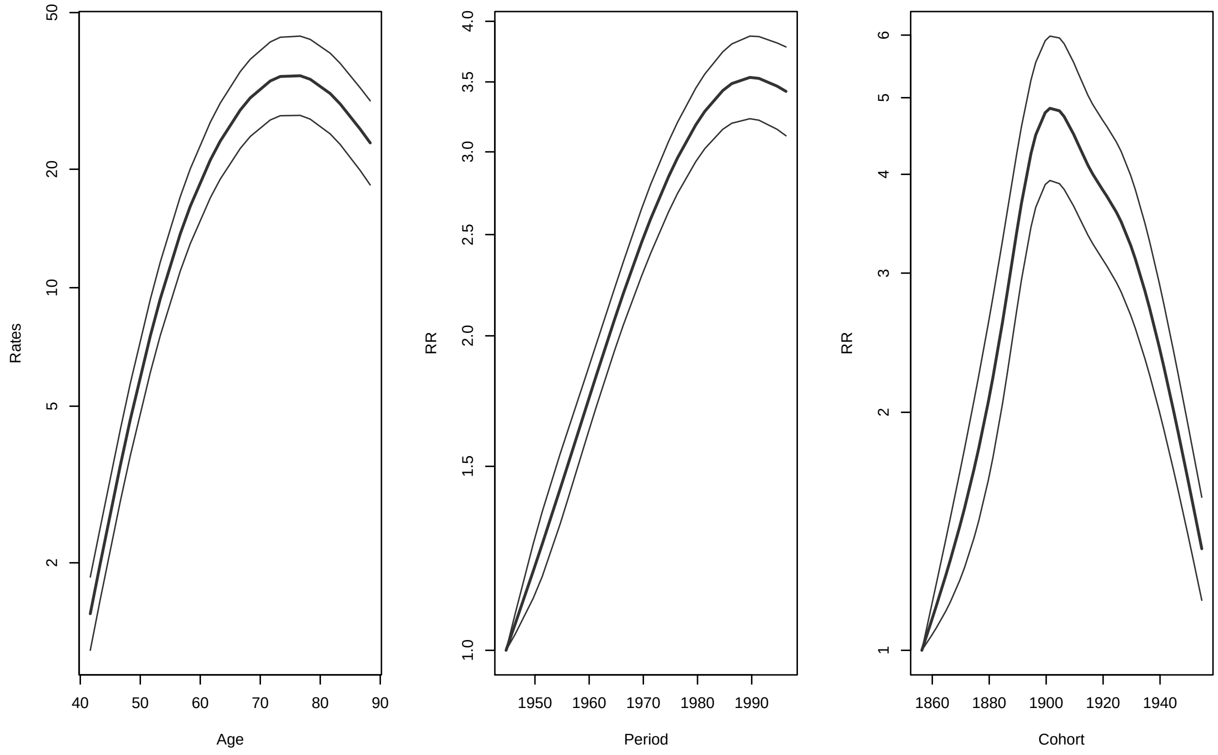

Figure 3.24: *Estimates from.*

3.9 Using apc.fit etc.

This exercise introduces the functions for fitting and plotting the results from age-period-cohort models: apc.fit apc.plot apc.lines and apc.frame.

1. We first read the testis cancer data and collapse the cases over the histological subtypes:

```
> th <- read.table( "../data/testis-hist.txt", header=T )
> str( th )
```

```
'data.frame':        29160 obs. of  9 variables:
 $ a    : int  0 0 0 0 0 0 1 1 1 1 ...
 $ p    : int  1943 1943 1943 1943 1943 1943 1943 1943 1943 1943 ...
 $ c    : int  1942 1942 1942 1943 1943 1943 1941 1941 1941 1942 ...
 $ y    : num  18853 18853 18853 20797 20797 ...
 $ age  : num  0.667 0.667 0.667 0.333 0.333 ...
 $ diag : num  1943 1943 1943 1944 1944 ...
 $ birth: num  1943 1943 1943 1943 1943 ...
 $ hist : int  1 2 3 1 2 3 1 2 3 1 ...
 $ d    : int  0 1 0 0 0 0 0 0 0 0 ...
```

Knowing the names of the variables in the dataset, we can now collapse over the histological subtypes. There is no need to tabulate by cohort as well, because even for the triangular data the relationship $c = p - a$ holds. For aesthetic reasons we get rid of the variable we do not need:

```
> tc <- aggregate( th[,c("age","diag","d","y")], list(A=th$age,P=th$diag), sum )
> str( tc )
```

```
'data.frame':       9720 obs. of  6 variables:
 $ A   : num  0.667 1.667 2.667 3.667 4.667 ...
 $ P   : num  1943 1943 1943 1943 1943 ...
 $ age : num  2 5 8 11 14 ...
 $ diag: num  5830 5830 5830 5830 5830 ...
 $ d   : int  1 0 0 0 0 0 0 0 0 0 ...
 $ y   : num  56559 51319 49931 49083 48376 ...

> names( tc ) <- toupper( names(tc) )
> tc <- tc[,c("A","P","D","Y")]
```

Now the original data had three subtypes of testis cancer, so while it is OK to sum the number of cases (D), the amount of risk time has been aggregated erroneously, so we must divide by 3:

```
> tc$Y <- tc$Y/3
> tc$C <- tc$P - tc$A
> str( tc )

'data.frame':       9720 obs. of  5 variables:
 $ A: num  0.667 1.667 2.667 3.667 4.667 ...
 $ P: num  1943 1943 1943 1943 1943 ...
 $ D: int  1 0 0 0 0 0 0 0 0 0 ...
 $ Y: num  18853 17106 16644 16361 16125 ...
 $ C: num  1943 1942 1941 1940 1939 ...

> head( tc )

          A        P D        Y        C
1 0.6666667 1943.333 1 18853.00 1942.667
2 1.6666667 1943.333 0 17106.33 1941.667
3 2.6666667 1943.333 0 16643.50 1940.667
4 3.6666667 1943.333 0 16361.00 1939.667
5 4.6666667 1943.333 0 16125.17 1938.667
6 5.6666667 1943.333 0 15728.50 1937.667
```

2. If we want to present the rates in 5-year age and period classes from age 15 to age 59 using rateplot, we must make a table as input to the rateplot function. Note that in this case we aggregate *across* subsets of the Lexis diagram and not as above *within*, and hence we must use the sum both for events and risk time:

```
> par( mfrow=c(2,2), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> rateplot(
+ with( subset( tc, A>15 & A<60 ),
+       tapply( D, list(floor(A/5)*5+2.5,
+                       floor((P-1943)/5)*5+1945.5), sum ) /
+       tapply( Y, list(floor(A/5)*5+2.5,
+                       floor((P-1943)/5)*5+1945.5), sum ) * 10^5 ),
+           col=topo.colors(12) )
```

3. We now fit an age-period-cohort model to the data using the machinery implemented in apc.fit. The function returns a fitted model *and* a parametrization, hence we must choose how to parametrize it, in this case "ACP" with all the drift included in the cohort effect and the reference cohort being 1918.

```
> tapc <- apc.fit( subset( tc, A>15 & A<60 ), npar=c(10,10,10), parm="ACP", ref.c=1918 )
```

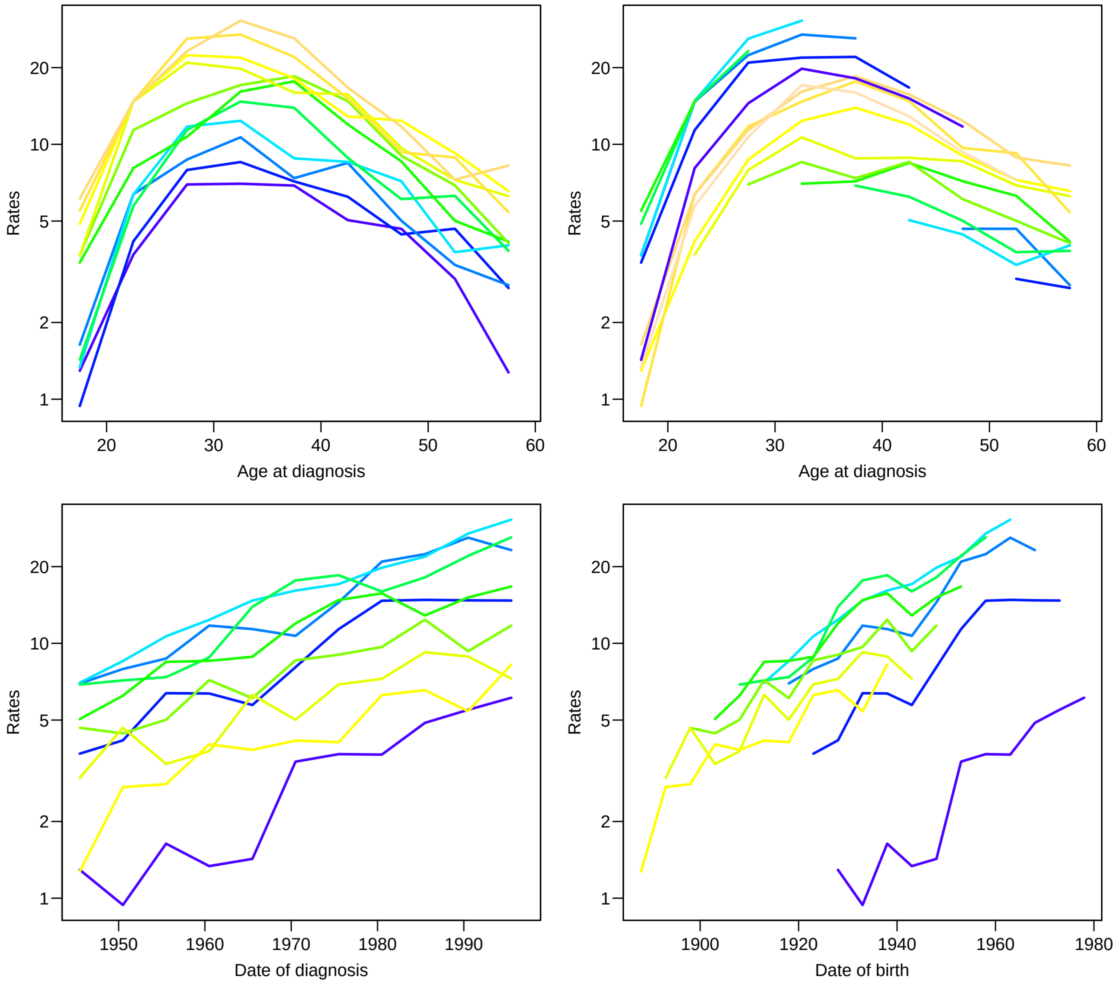

Figure 3.25: *Age-specific rates for testis cancer in Denmark.*

```
[1] "ML of APC-model Poisson with log(Y) offset : ( ACP ):\n"

Analysis of deviance for Age-Period-Cohort model

                  Resid. Df Resid. Dev   Df Deviance  P(>|Chi|)
Age                    4849     6513.1
Age-drift              4848     5313.6    1   1199.5 7.990e-263
Age-Cohort             4839     5244.4    9     69.2  2.147e-11
Age-Period-Cohort      4830     5193.9    9     50.5  8.633e-08
Age-Period             4839     5290.5   -9    -96.6  7.652e-17
Age-drift              4848     5313.6   -9    -23.2  5.867e-03
```

It is seen that the period effect is weaker (deviance=50.5) than the cohort effect (deviance=96.6), although still *formally* strongly significant.

4. We can plot the estimates using the apc.plot function:

```
> apc.plot( tapc, ci=TRUE )

 cp.offset     RR.fac
1823.33333    0.00001
```

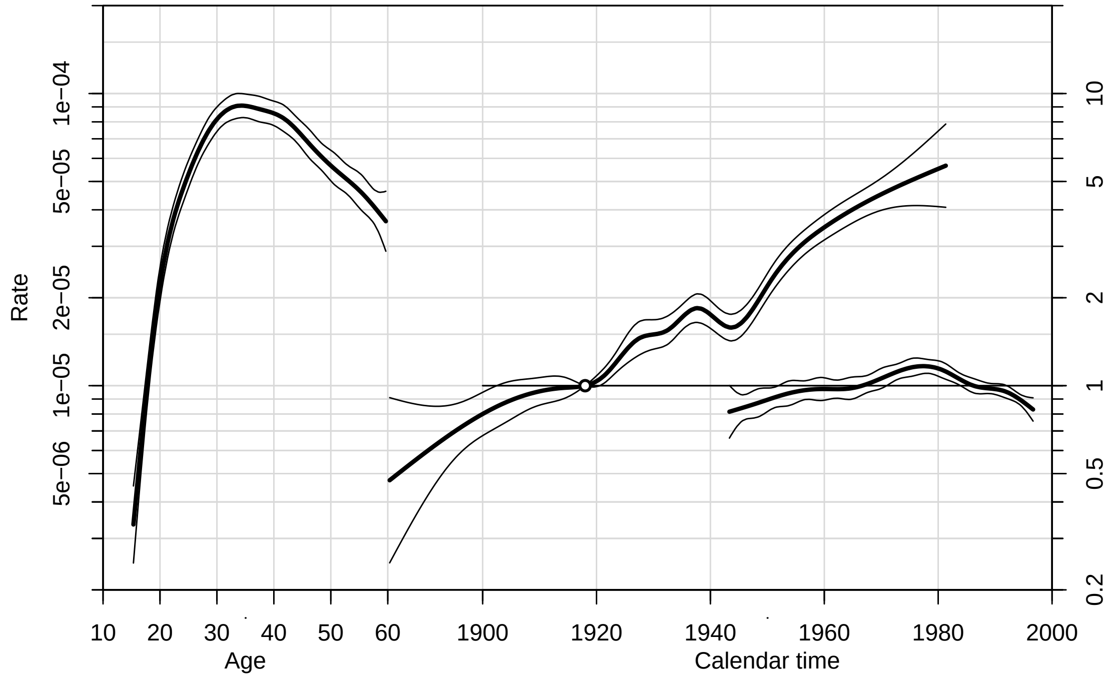

Figure 3.26: *The default plot for the fit of an Age-Period-Cohort model for testis cancer in Denmark. 10 parameters for all effects.*

5. Now explore in more depth the cohort effect by increasing the number of parameters used for it:

```
> tapc <- apc.fit( subset( tc, A>15 & A<60 ), npar=c(10,10,20),
+                  parm="ACP", ref.c=1918, scale=10^5 )

[1] "ML of APC-model Poisson with log(Y) offset : ( ACP ):\n"

Analysis of deviance for Age-Period-Cohort model

                  Resid. Df Resid. Dev   Df Deviance  P(>|Chi|)
Age                    4849     6513.1
Age-drift              4848     5313.6    1   1199.5 7.990e-263
Age-Cohort             4829     5233.1   19     80.6  1.484e-09
Age-Period-Cohort      4820     5182.6    9     50.5  8.811e-08
Age-Period             4839     5290.5  -19   -107.9  1.955e-14
Age-drift              4848     5313.6   -9    -23.2  5.867e-03

> fp <- apc.plot( tapc, ci=TRUE )
```

6. We now explore the effect of using the residual method instead, and over-plot the estimates from this method on the existing plot[2]:

[2] Unfortunately there is a fatal bug in `apc.fit` when fitting the period residuals to the age-cohort model — it does not crash but simply fit a totally meaningless model. There is a fix for this in the version 1.0.11 of the `Epi` package which is available at the course homepage

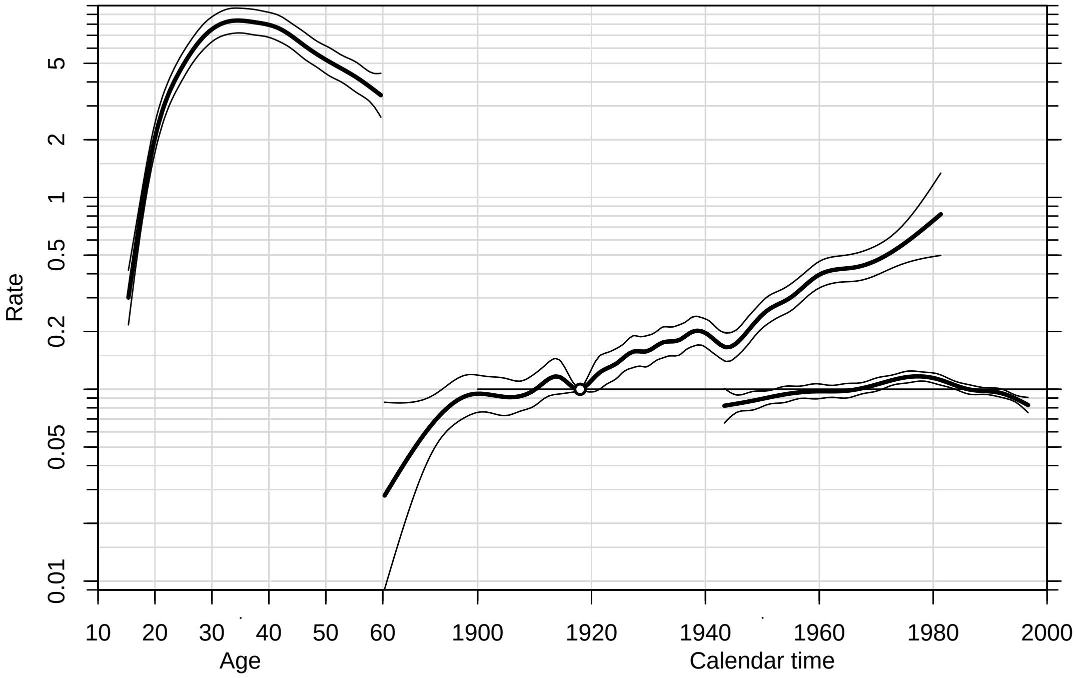

Figure 3.27: *The default plot for the fit of an AGe-Period-Cohort model for testis cancer in Denmark. 20 parameters for the cohort effect, 10 for age and period.*

```
> tac.p <- apc.fit( subset( tc, A>15 & A<60 ), npar=c(10,10,20),
+                  parm="AC-P", ref.c=1918, scale=10^5 )

[1] "Sequential modelling Poisson with log(Y) offset : ( AC-P ):\n"

Analysis of deviance for Age-Period-Cohort model

                  Resid. Df Resid. Dev   Df Deviance  P(>|Chi|)
Age                    4849     6513.1
Age-drift              4848     5313.6    1   1199.5 7.990e-263
Age-Cohort             4829     5233.1   19     80.6  1.484e-09
Age-Period-Cohort      4820     5182.6    9     50.5  8.811e-08
Age-Period             4839     5290.5  -19   -107.9  1.955e-14
Age-drift              4848     5313.6   -9    -23.2  5.867e-03

> fp <- apc.plot( tapc, ci=TRUE )
> apc.lines( tac.p, ci=TRUE, col="red", frame.par=fp )
```

7. The standard display is not very pretty — it gives an overview, but certainly not anything worth publishing, hence a bit of handwork is needed. We can use the `apc.frame` for this, and create a nicer plot of the estimates from the residual model:

```
> par( mar=c(3,4,1,4), mgp=c(3,1,0)/1.7, las=1 )
> fp <- apc.frame( a.lab=seq(20,60,10),
+                  a.tic=seq(10,60,5),
+                 cp.lab=seq(1900,2000,20),
```

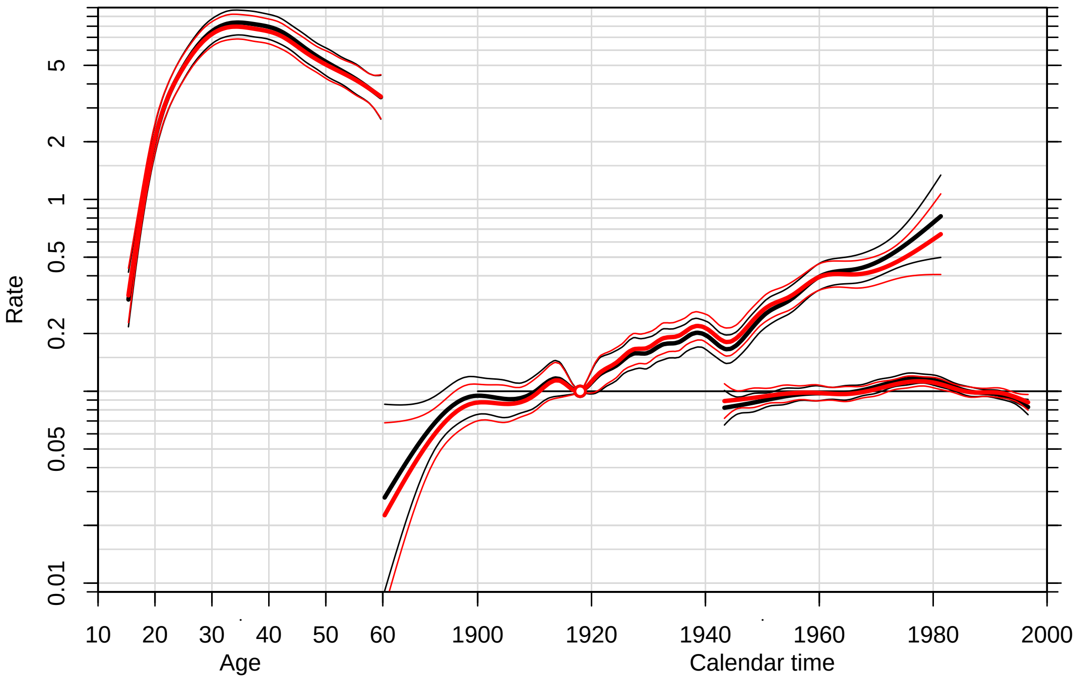

Figure 3.28: *Comparing the ML-method with the residual method for the Danish testis cancer cases.*

```
+                  cp.tic=seq(1885,2000,5),
+                   r.lab=c(c(1,2,5)/10,1,2,5,10),
+                   r.tic=c(1:9/10,1:10),
+                     gap=8,
+                  rr.ref=1)
> apc.lines( tapc, ci=TRUE, col="blue", frame.par=fp )
> apc.lines( tac.p, ci=TRUE, col="red", frame.par=fp )
```

8. We now try to use period as the primary timescale, and add this to the plot as well:

```
> tap.c <- apc.fit( subset( tc, A>15 & A<60 ), npar=c(10,10,20),
+                    parm="AP-C", ref.p=1950, scale=10^5 )

[1] "Sequential modelling Poisson with log(Y) offset : ( AP-C ):\n"

Analysis of deviance for Age-Period-Cohort model

                  Resid. Df Resid. Dev   Df Deviance  P(>|Chi|)
Age                    4849     6513.1
Age-drift              4848     5313.6    1   1199.5 7.990e-263
Age-Cohort             4829     5233.1   19     80.6  1.484e-09
Age-Period-Cohort      4820     5182.6    9     50.5  8.811e-08
Age-Period             4839     5290.5  -19   -107.9  1.955e-14
Age-drift              4848     5313.6   -9    -23.2  5.867e-03

> apc.lines( tap.c, ci=TRUE, col=c("black","gray","black"), frame.par=fp )
```

From the black (and gray) curves in figure 3.29, the dips in incidence rates for the

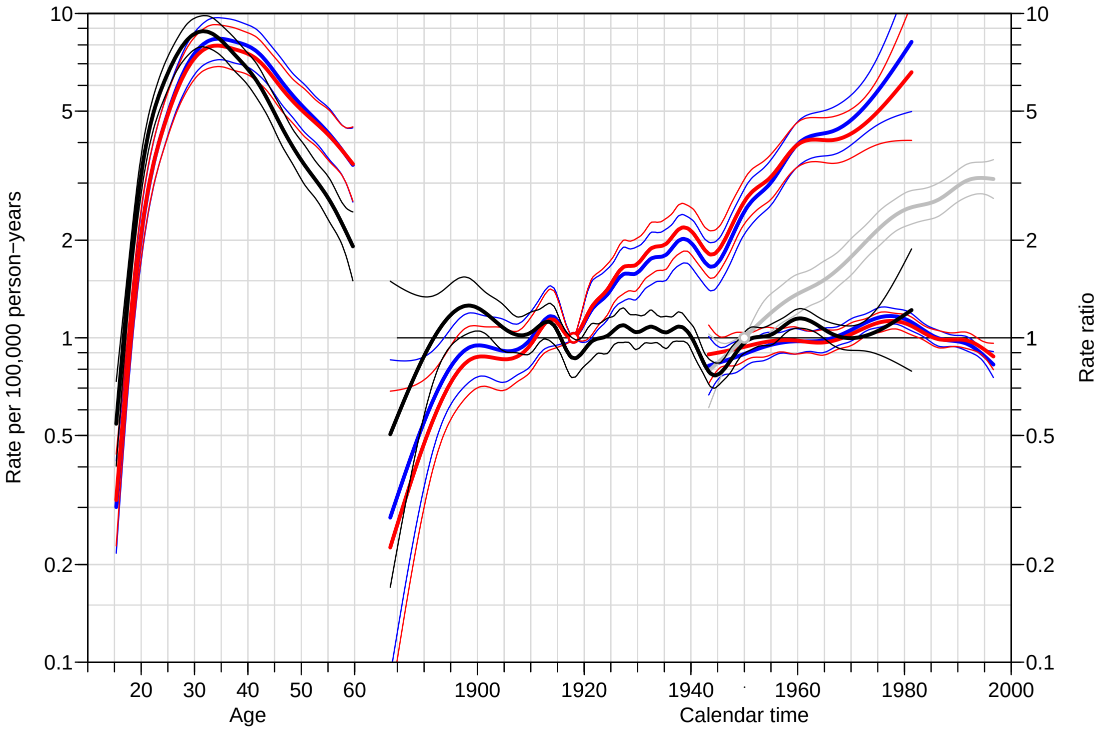

Figure 3.29: *Comparing the ML-method with the residual method for the Danish testis cancer cases. Additionally, the parametrization of the residual method for the age-period model is shown.*

generations born during the world wars is quite remarkable, but it also seen that the shift to a period-primary model shifts the age-specific rates to peak at a slightly earlier age, 30 instead of 35.

The former figure is an indication of the age-distribution of next years cases (when multiplied by the population distribution . . .), whereas the latter is a reasonable statement about the natural history of the disease; men are at increasing risk until age 35, and there after it decreases.

3.10 Lung cancer: the sex difference

The following exercise is aimed at investigation of the effect of age, period and cohort on the lung cancer incidence for both sexes using one complex age-period-cohort model. First, we will use 5-year triangular data to xxxx and build separate models for males and females. Further the complex model will be built for 1-year triangular data.

1. First we read 1-year triangular data from data set `apc-Lung.txt`

```
> lung <- read.table( "../data/apc-Lung.txt", header=T )
> head( lung)

  sex A    P    C D       Y
1   1 0 1943 1942 0 19546.2
2   1 0 1943 1943 0 20796.5
3   1 0 1944 1943 0 20681.3
4   1 0 1944 1944 0 22478.5
5   1 0 1945 1944 0 22369.2
6   1 0 1945 1945 0 23885.0
```

2. The variables `A`, `P` and `C` are the left endpoints of the tabulation intervals, so the value of the variable `P-A-C` is 0 for lower triangles and 1 for upper triangles in the Lexis diagram. This can the be used to compute the correct values of the mean age and period (and cohort) in the dataset.

```
> lung <- transform( lung, up = P-A-C, At = A, Pt = P, Ct = C )
> lung <- transform( lung, A = At + 1/3 + up/3,
+                          P = Pt + 3/2 - up/3 )
> lung <- transform( lung, C = P - A )
> head( lung )

  sex         A        P        C D       Y up At   Pt   Ct
1   1 0.6666667 1944.167 1943.500 0 19546.2  1  0 1943 1942
2   1 0.3333333 1944.500 1944.167 0 20796.5  0  0 1943 1943
3   1 0.6666667 1945.167 1944.500 0 20681.3  1  0 1944 1943
4   1 0.3333333 1945.500 1945.167 0 22478.5  0  0 1944 1944
5   1 0.6666667 1946.167 1945.500 0 22369.2  1  0 1945 1944
6   1 0.3333333 1946.500 1946.167 0 23885.0  0  0 1945 1945
```

A bit of care is required with the `transform` function; each of the assignments is made in the original dataframe given as the first argument, hence it is not possible compute the correct `C`

3. We can make an overview of the rates if we can produce a table of the rates in a suitable form. This can be done by grouping on the fly and tabulating by sex too:

```
> lrate <- with( subset( lung, A>40 & A<90 ),
+               tapply( D, list(sex,
+                               floor(A/5)*5+2.5,
+                               floor((P-1943)/5)*5+1943+2.5),
+                           sum ) /
+               tapply( Y, list(sex,
+                               floor(A/5)*5+2.5,
+                               floor((P-1943)/5)*5+1943+2.5),
+                           sum ) * 10^5 )
```

With this three-way table we can plot the rates for males and females in one go, using the same scale for the axes among men and women; as seen in the figure 3.30:

```
> par( mfrow=c(2,4), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> rateplot( lrate[1,,], col="blue", ylim=range(lrate,na.rm=T) )
> rateplot( lrate[2,,], col="red", ylim=range(lrate,na.rm=T) )
```

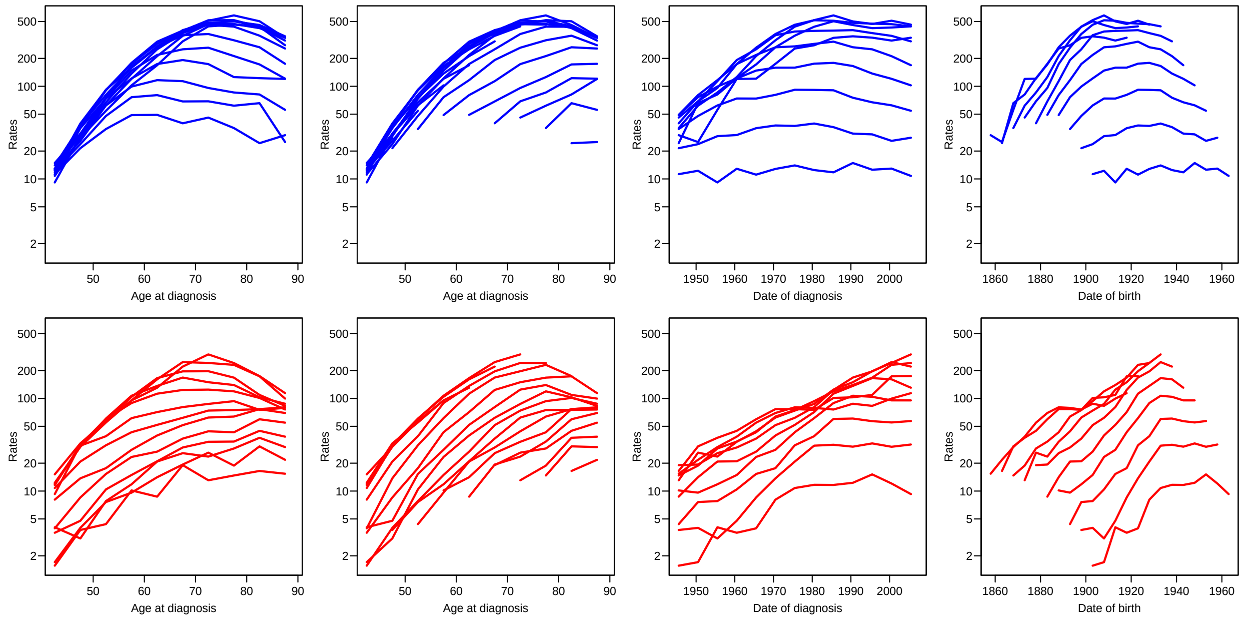

Figure 3.30:

4. The models are easily fitted separately using the `subset` function on the dataframe:

```
> apc.m <- apc.fit( subset(lung,sex==1 & A>40), npar=c(8,8,15), ref.c=1930, scale=10^5 )

[1] "ML of APC-model Poisson with log(Y) offset : ( ACP ):\n"

Analysis of deviance for Age-Period-Cohort model

                  Resid. Df Resid. Dev  Df Deviance   P(>|Chi|)
Age                    6091    23484.6
Age-drift              6090    16697.6   1   6787.0         0.0
Age-Cohort             6076     8239.8  14   8457.8         0.0
Age-Period-Cohort      6069     7451.5   7    788.3 6.178e-166
Age-Period             6083    10719.6 -14  -3268.0         0.0
Age-drift              6090    16697.6  -7  -5978.1         0.0

> apc.f <- apc.fit( subset(lung,sex==2 & A>40), npar=c(8,8,15), ref.c=1930, scale=10^5 )

[1] "ML of APC-model Poisson with log(Y) offset : ( ACP ):\n"

Analysis of deviance for Age-Period-Cohort model

                  Resid. Df Resid. Dev  Df Deviance   P(>|Chi|)
Age                    6091    24291.8
Age-drift              6090     8458.4   1  15833.4         0.0
Age-Cohort             6076     7535.0  14    923.3 4.305e-188
Age-Period-Cohort      6069     7045.8   7    489.2 1.641e-101
Age-Period             6083     7953.5 -14   -907.7 9.582e-185
Age-drift              6090     8458.4  -7   -504.9 7.198e-105
```

We can plot the the results separately and then judging from the displays find out what display is required for a sensible common plot

```
> apc.plot( apc.m, col="blue" )

cp.offset    RR.fac
 1753.333   100.000

> apc.plot( apc.f, col="red" )

cp.offset    RR.fac
 1753.333   100.000
```

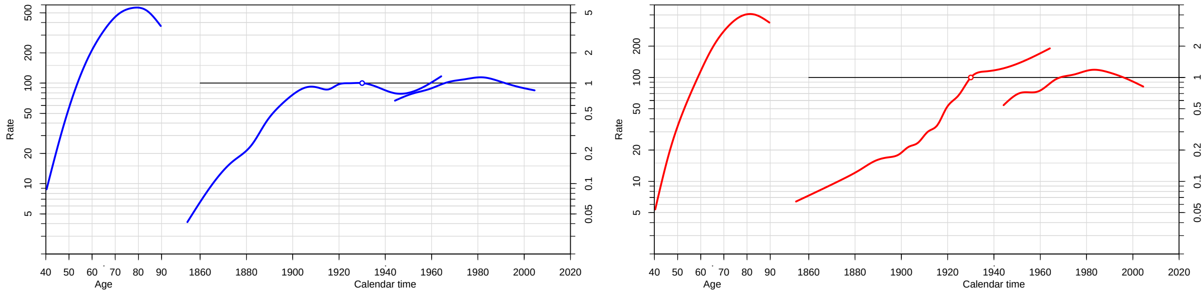

Figure 3.31: *initial sketch plots for the male and the female rates of ling cancer incidence in Denmark.*

Now we can set up a

```
> par( las=1, mar=c(4,3,1,4), mgp=c(3,1,0)/1.6 )
> fp <- apc.frame( a.lab = seq(40,90,20),
+                 cp.lab = seq(1880,2000,20),
+                  r.lab = c(6,c(1,2,5)*10,c(1,2,5)*100),
+ #               rr.lab = r.lab / rr.ref,
+                 rr.ref = 200,
+                  a.tic = seq(35,90,5),
+                 cp.tic = seq(1855,2005,5),
+                  r.tic = c(5:9,1:9*10,1:6*100),
+ #               rr.tic = r.tic / rr.ref,
+                tic.fac = 1.3,
+                  a.txt = "Age",
+                 cp.txt = "Calendar time",
+                  r.txt = "Lung cancer rate per 100,000 person-years",
+                 rr.txt = "Rate ratio",
+               ref.line = TRUE,
+                    gap = 13,
+               col.grid = gray(0.85),
+                  sides = c(1,2,4) )
> apc.lines( apc.m, col="blue", ci=T, frame.par=fp )
> apc.lines( apc.f, col="red" , ci=T, frame.par=fp )
```

5. The ratios of the rates also follows an age-period-cohort model:

$$\log\big(\lambda_M(a.p)/\lambda_F(a,p)\big) = \log\big(\lambda_M(a.p)\big) - \log\big(\lambda_F(a,p)\big) = \big(f_M(a) - f_F(a)\big) + \big(g_M(p) - g_F(p)\big) + \big(h_M(c) - h_F($$

so for the rate-ratios we have excatly the same identification problems, but we can for a start just compute the ratois of the effects with confidence intervals.

To this end is is easier to devise a small function that takes two sets of estimated rates/RRs with c.i.s and returns the ratio with c.i.s:

```
> rr <-
+ function( one, two )
+ {
+ one[,-1] <- log(one[,-1] )
+ two[,-1] <- log(two[,-1] )
+ sd.dif <- sqrt( ((one[,4]-one[,3])/3.92)^2 +
+                 ((two[,4]-two[,3])/3.92)^2 )
+ rat <- one
+ rat[,-1] <- exp( cbind( one[,2]-two[,2], sd.dif ) %*%
+                   rbind( c(1,1,1), 1.96*c(0,-1,1) ) )
+ rat
+ }
> rr.Age <- rr( apc.m$Age, apc.f$Age )
> rr.Per <- rr( apc.m$Per, apc.f$Per )
> rr.Coh <- rr( apc.m$Coh, apc.f$Coh )
```

In order to plot these in an apc-frame, we can just fake an apc-object, and in order to get a reasonable apc-frame we compute the ranges of the RRs:

```
> apc.mf <- apc.m
> apc.mf$Age <- rr.Age
> apc.mf$Per <- rr.Per
> apc.mf$Coh <- rr.Coh
> ( RRr <- range( rbind(rr.Age[,-1],
+                       rr.Per[,-1],
+                       rr.Coh[,-1]) ) )

[1] 0.2169004 4.3770837
```

So we devise a frame which stretces fom 0.2 to 5:

```
> par( las=1, mar=c(4,3,1,2), mgp=c(3,1,0)/1.6 )
> fp <- apc.frame( a.lab = seq(40,90,20),
+                 cp.lab = seq(1880,2000,20),
+                  r.lab = c(0.2,0.5,1,2,5),
+                 rr.ref = 1,
+                  a.tic = seq(35,90,5),
+                 cp.tic = seq(1855,2005,5),
+                  r.tic = c(2:9/10,1:5),
+                tic.fac = 1.3,
+                  a.txt = "Age",
+                 cp.txt = "Calendar time",
+                  r.txt = "M/F Rate ratio of lung cancer",
+                 rr.txt = "",
+               ref.line = TRUE,
+                    gap = 13,
+               col.grid = gray(0.85),
+                  sides = c(1,2,4) )
> abline( h=1 )
> apc.lines( apc.mf, col="black", ci=T, frame.par=fp )
```

6. In order to explicitly fix the knots we just use those from the male apc object:

```
> A.kn <- apc.m$Knots$Age
> nk.A <- length(A.kn)
> MA <- ns( lung$A, knots=A.kn[-c(1,nk.A)], Bo=A.kn[c(1,nk.A)], intercept=TRUE )
> P.kn <- apc.m$Knots$Per
> nk.P <- length(P.kn)
> MP <- ns( lung$P, knots=P.kn[-c(1,nk.P)], Bo=P.kn[c(1,nk.P)], intercept=TRUE )
> MP <- detrend( MP, lung$P )
> C.kn <- apc.m$Knots$Coh
```

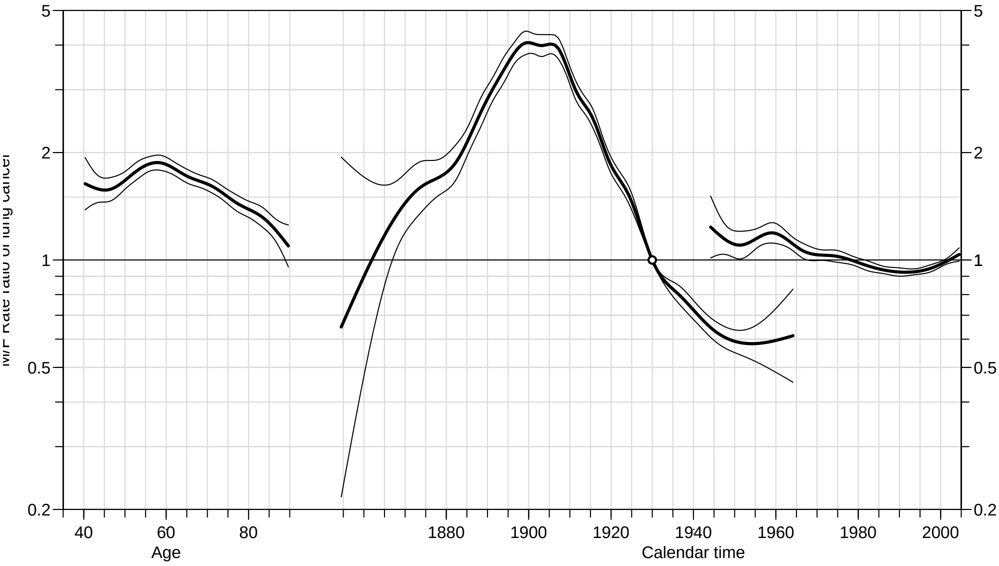

Figure 3.32: *M/F rate-ratio of lung cancer in Denmark.*

```
> nk.C <- length(C.kn)
> MC <- ns( lung$C, knots=C.kn[-c(1,nk.C)], Bo=C.kn[c(1,nk.C)], intercept=TRUE )
> MC <- detrend( MC, lung$C )
```

With these matrices we can now fit the models we want; the model with sex-interaction on all three variables and the one where we assume identical 2nd order period-effects:

```
> lung$sex <- factor(lung$sex,labels=c("M","F"))
> m.int <- glm( D ~ -1 + MA:sex + MP:sex + MC:sex + I(C-1950):sex +
+                     offset( log(Y) ), family=poisson, data=lung )
```

7. We can check if any of the second-order terms are identical between males and females by removing the interaction with sex. This will however only work for the period and the cohort effect, because the intercept and linear effect of age is included with the age-effect and removing the interation there would be tantamount to testing wheter the absolute levels and the (first order) shape were the same.

   So we start by checking wheter the period and age-effects have the same second-order properties (i.e. same shape):

```
> m.per <- update( m.int, . ~ . - MP:sex + MP )
> m.coh <- update( m.int, . ~ . - MC:sex + MC )
> anova( m.coh, m.int, m.per, test="Chisq" )

Analysis of Deviance Table

Model 1: D ~ MC + MA:sex + sex:MP + sex:I(C - 1950) + offset(log(Y)) -
    1
Model 2: D ~ -1 + MA:sex + MP:sex + MC:sex + I(C - 1950):sex + offset(log(Y))
Model 3: D ~ MP + MA:sex + sex:MC + sex:I(C - 1950) + offset(log(Y)) -
    1
  Resid. Df Resid. Dev    Df Deviance P(>|Chi|)
1     21912    19298.0
2     21898    17701.6    14   1596.4       0.0
3     21905    17741.1    -7    -39.5 1.551e-06
```

Alhough both effects are significant there is a much smaller deviance for the period effect, so we can assume that the period-effects have the same shape.

As goes for the age-effect we can test the same hypothesis, but we want to test a slightly stronger hypothesis, namely that the actual slope with age is the same too, so when we update the model we include the main effect of sex, but *not* the interction vith sex and age; or rather we make succesice tests for this:

```
> m.age <- update( m.int, . ~ . - MA:sex + MA + sex + sex:A )
> m.aln <- update( m.age, . ~ . - sex:A )
> anova( m.int, m.age, m.aln, test="Chisq" )

Analysis of Deviance Table

Model 1: D ~ -1 + MA:sex + MP:sex + MC:sex + I(C - 1950):sex + offset(log(Y))
Model 2: D ~ MA + sex + sex:MP + sex:MC + sex:I(C - 1950) + sex:A + offset(log(Y)) -
    1
Model 3: D ~ MA + sex + sex:MP + sex:MC + sex:I(C - 1950) + offset(log(Y)) -
    1
  Resid. Df Resid. Dev    Df Deviance P(>|Chi|)
1     21898    17701.6
2     21905    17827.6    -7   -126.0 4.202e-24
3     21906    17979.9    -1   -152.3 5.430e-35
```

We see that there quite strong evidence against the hypothesis that the age-effects have the same shape and even stronger that they should have the same “slopes”, i.e. first-order shapes too.

8. Thus it seems that a relevant description of the relationship of lung cancer rates between males and females in Denmark is that they follow an age-cohort model. This model is alreafy fitted, but in order to facilitate extractio of the parameters we refit it with a parametrization of the linear cohort effect that gives the difference of these, so it is easier to use a contrast matrix to get it out:

```
> m.RR <- glm( D ~ -1 + MA + MP + MC + I(C-1950) +
+                    MA:sex + MC:sex + I(C-1950):sex +
+                    offset( log(Y) ), family=poisson, data=lung )
> pr.RR <- predict( m.RR, type="terms", se.fit=TRUE )
> str( pr.RR )

List of 3
 $ fit           : num [1:21960, 1:7] -18.5 -18.6 -18.5 -18.6 -18.5 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:21960] "1" "2" "3" "4" ...
  .. ..$ : chr [1:7] "MA" "MP" "MC" "I(C - 1950)" ...
  ..- attr(*, "constant")= num 0
 $ se.fit        : num [1:21960, 1:7] 0.193 0.194 0.193 0.194 0.193 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:21960] "1" "2" "3" "4" ...
  .. ..$ : chr [1:7] "MA" "MP" "MC" "I(C - 1950)" ...
 $ residual.scale: num 1
```

This allows us to reconstruct the age and the period effects by taking the unique rows of the cohort and age-design matrices and multiply on the parameters of the interaction terms in order to get the log-RRs:

```
> # Unique ages and cohort
> au <- match( sort(unique(lung$A)), lung$A)
> cu <- match( sort(unique(lung$C)), lung$C)
> # Corresponding subsets of the design matrices
> A.ctr <- MA[au,]
> C.ctr <- cbind( MC[cu,], (lung$C-1950)[cu] )
> # Paramter names
> parnam <- names( coef(m.RR) )
> # Have we found the age-paramters we want?
> a.par <- intersect( grep("MA",parnam), grep("sexM",parnam) )
> parnam[a.par]

[1] "MA1:sexM" "MA2:sexM" "MA3:sexM" "MA4:sexM" "MA5:sexM" "MA6:sexM" "MA7:sexM"
[8] "MA8:sexM" "MA9:sexM"

> # Have we found the acohort-paramters we want?
> c.par <- c( grep("MC",parnam), grep("I",parnam) )
> c.par <- intersect( c.par, grep("sex",parnam) )
> parnam[c.par]

 [1] "MC1:sexF"          "MC2:sexF"          "MC3:sexF"          "MC4:sexF"
 [5] "MC5:sexF"          "MC6:sexF"          "MC7:sexF"          "MC8:sexF"
 [9] "MC9:sexF"          "MC10:sexF"         "MC11:sexF"         "MC12:sexF"
[13] "MC13:sexF"         "MC14:sexF"         "I(C - 1950):sexF"

> # Then we can extract effects, the parametrization for the cohort
> # effect is for F/M, hence we use -C.ctr
> A.eff <- ci.lin( m.RR, subset=a.par, ctr.mat= A.ctr, Exp=TRUE )[,5:7]
> C.eff <- ci.lin( m.RR, subset=c.par, ctr.mat=-C.ctr, Exp=TRUE )[,5:7]
```

These effects can now be plotted side by side:

```
> par( mfrow=c(1,2), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> matplot( lung$A[au], A.eff,
+          log="y", ylim=c(0.5,5),
+          type="l", lty=1, col="black", lwd=c(3,1,1) )
> matplot( lung$C[cu], C.eff,
+          log="y", ylim=c(0.5,5),
+          type="l", lty=1, col="black", lwd=c(3,1,1) )
```

Now these effects could also be superposed on those from the separate APC-models:

```
> par( las=1, mar=c(4,3,1,2), mgp=c(3,1,0)/1.6 )
> fp <- apc.frame( a.lab = seq(40,90,20),
+                 cp.lab = seq(1880,2000,20),
+                  r.lab = c(0.5,1,2,5),
+                 rr.ref = 1,
+                  a.tic = seq(35,90,5),
+                 cp.tic = seq(1855,2005,5),
+                  r.tic = c(4:9/10,1:6),
+                tic.fac = 1.3,
+                  a.txt = "Age",
+                 cp.txt = "Calendar time",
+                  r.txt = "M/F Rate ratio of lung cancer",
+                 rr.txt = "",
+               ref.line = TRUE,
+                    gap = 13,
+               col.grid = gray(0.85),
+                  sides = c(1,2,4) )
> abline( h=1 )
> apc.lines( apc.mf, col="black", ci=F, frame.par=fp, lwd=2 )
> matlines( lung$A[au], A.eff, lwd=c(3,1,1), lty=1, col="blue" )
> matlines( lung$C[cu]-fp[1], C.eff, lwd=c(3,1,1), lty=1, col="blue" )
```

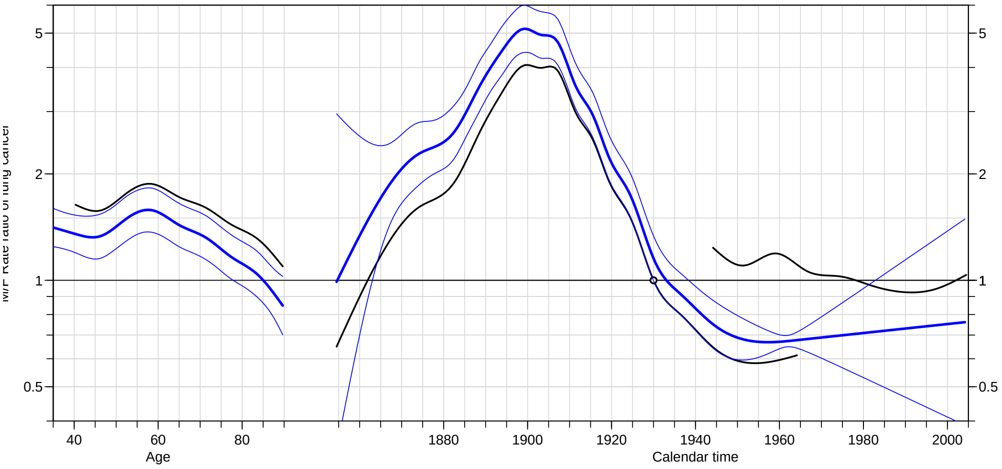

Figure 3.33: *Comparing the M/F rate-ratio between the simple approach and the approach using an explicit model.*

3.11 Prediction of breast cancer rates

1. First we read the data and take an overview:

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

'data.frame':        10980 obs. of  5 variables:
 $ A: int  0 0 0 0 0 0 0 0 0 0 ...
 $ P: int  1943 1943 1944 1944 1945 1945 1946 1946 1947 1947 ...
 $ C: int  1942 1943 1943 1944 1944 1945 1945 1946 1946 1947 ...
 $ D: int  0 0 0 0 0 0 0 0 0 0 ...
 $ Y: num  18649 19947 19854 21265 21236 ...

> summary( breast )

       A              P              C              D                Y
 Min.   : 0.0   Min.   :1943   Min.   :1853   Min.   : 0.00   Min.   :  385.2
 1st Qu.:22.0   1st Qu.:1958   1st Qu.:1905   1st Qu.: 0.00   1st Qu.:11059.5
 Median :44.5   Median :1973   Median :1928   Median : 9.00   Median :14538.3
 Mean   :44.5   Mean   :1973   Mean   :1928   Mean   :12.11   Mean   :13555.2
 3rd Qu.:67.0   3rd Qu.:1988   3rd Qu.:1951   3rd Qu.:21.00   3rd Qu.:17767.2
 Max.   :89.0   Max.   :2003   Max.   :2003   Max.   :69.00   Max.   :22549.0
```

2. We now replace A, P and C with the correct triangle means; recall that the upper triangles are characterized by the cohort being from the previous year, i.e. that $p - a - c = 1$.

```
> breast <- transform( breast, up = P-A-C )
> breast <- transform( breast, A = A+1/3+up/3,
+                              P = P+2/3-up/3,
+                              C = C+1/3+up/3 )
> with( breast, summary( P-A-C ) )

     Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
2.274e-13 2.274e-13 2.274e-13 2.274e-13 2.274e-13 2.274e-13

> head( breast )
```

```
          A        P        C D        Y up
1 0.6666667 1943.333 1942.667 0 18648.83  1
2 0.3333333 1943.667 1943.333 0 19946.50  0
3 0.6666667 1944.333 1943.667 0 19853.67  1
4 0.3333333 1944.667 1944.333 0 21265.00  0
5 0.6666667 1945.333 1944.667 0 21235.67  1
6 0.3333333 1945.667 1945.333 0 22407.00  0
```

3. In order to use `ratetab` we must produce a matrix classified by age and period in suitable intervals. This can be done choosing a tabulation interval length and then using this in producing the tables. This approach enables a simple way of experimenting with the length. Figure 3.34 shows the results.

```
> par( mfrow=c(2,2), mar=c(3,3,0,0), oma=c(0,0,1,1), mgp=c(3,1,0)/1.6 )
> ti <- 6
> with( subset( breast, A>30 ),
+       rateplot( tapply( D, list(floor(A/ti)*5+ti/2,
+                                 floor((P-1943)/ti)*5+1943+ti/2), sum ) /
+                 tapply( Y, list(floor(A/ti)*ti+ti/2,
+                                 floor((P-1943)/ti)*ti+1943+ti/2), sum ) * 10^5,
+                 col=heat.colors(12) ) )
```

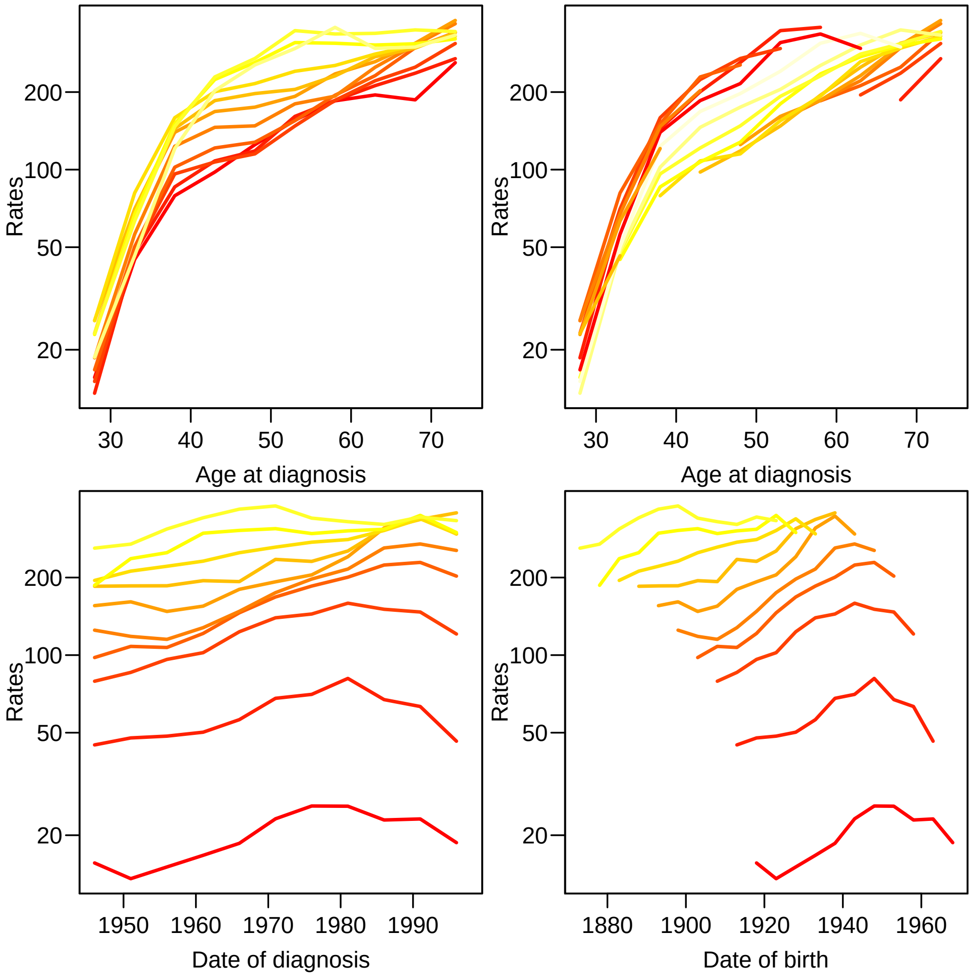

Figure 3.34: *Danish breast cancer rates in 6-year age and period intervals.*

4. We use `apc.fit` to fit a model with age, period and cohort effects as natural splines (the default), and the `apc.plot` to plot the estimated effects:

```
> m1 <- apc.fit( subset( breast, A>30 ), npar=c(10,7,15), ref.c=1920, scale=10^5 )

[1] "ML of APC-model Poisson with log(Y) offset : ( ACP ):\n"

Analysis of deviance for Age-Period-Cohort model

                  Resid. Df Resid. Dev  Df Deviance  P(>|Chi|)
Age                    7309    16264.9
Age-drift              7308    10198.5   1   6066.3        0.0
Age-Cohort             7294     9119.7  14   1078.8 1.915e-221
Age-Period-Cohort      7288     9018.1   6    101.7  1.121e-19
Age-Period             7302    10092.2 -14  -1074.1 1.918e-220
Age-drift              7308    10198.5  -6   -106.3  1.191e-20

> apc.plot( m1 )

cp.offset    RR.fac
     1750       100
```

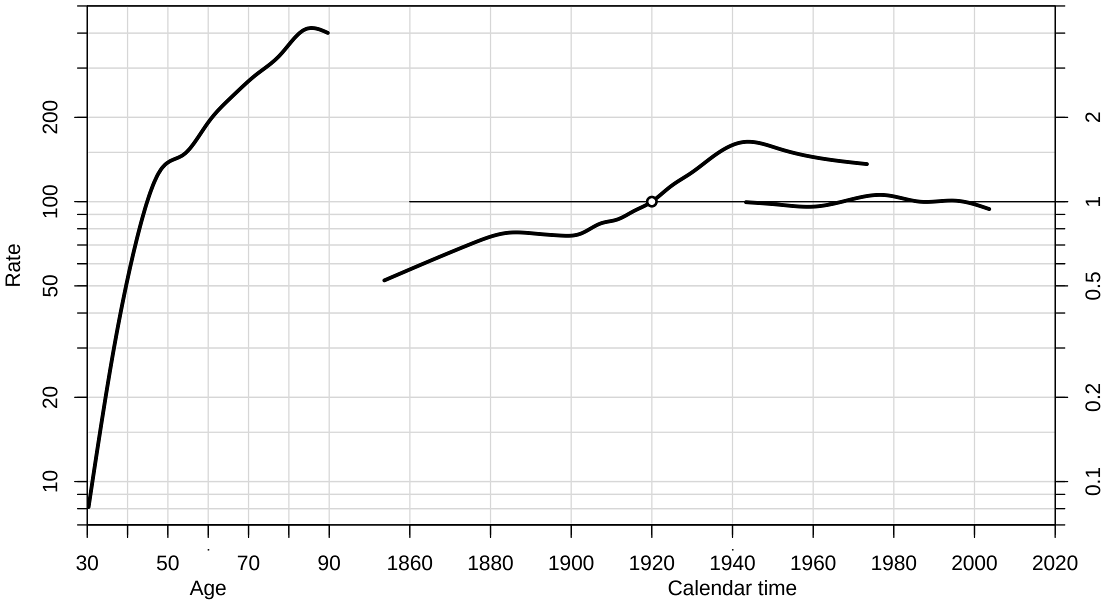

Figure 3.35: *Estimates of age- period- and cohort effects plotted the default way — crap!*

The plot (figure 3.35) is rather crappy, so we fine-tune the details by defining them explicit in `apc.frame`. This piece of code is made by copying the definition of all parameters from the help page and successively filling them in with suitable values:

```
> par( las=1, mar=c(3,4,1,4), mgp=c(3,1,0)/1.5 )
> fp <-  apc.frame( a.lab = seq(30,90,10),
+                  cp.lab = seq(1860,2005,20),
+                   r.lab = c(c(1,2,5)*10,c(1,2,5)*100),
+ #                 rr.lab = r.lab / rr.ref,
+                  rr.ref = 100,
+                   a.tic = seq(30,90,5),
+                  cp.tic = seq(1855,2005,5),
+                   r.tic = c(9,1:9*10,1:5*100),
+ #                 rr.tic = r.tic / rr.ref,
+                 tic.fac = 1.3,
+                   a.txt = "Age",
+                  cp.txt = "Calendar time",
+                   r.txt = "Rate per 100,000 person-years",
+                  rr.txt = "Rate ratio",
+                     gap = 8,
+                col.grid = gray(0.85),
+                   sides = c(1,2,4) )
> apc.lines( m1, frame.par=fp, ci=T, col="red" )
```

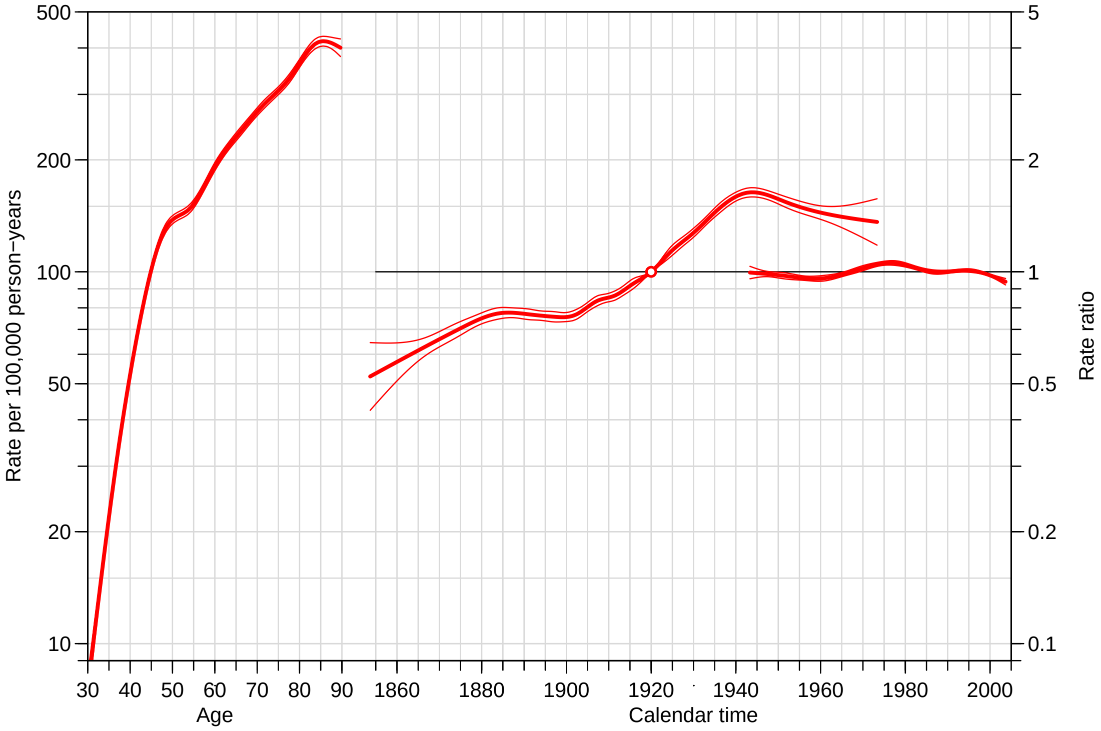

Figure 3.36: *Estimates of age- period- and cohort effects plotted after fine tuning the display using* `apc.frame`

5. First we define the prediction points and the anchor points on the period scale:

```
> P.pt <- 2000 + 0:20
> P.rf <- 2000 - c(30,0)
```

Then we compute the estimated period effect on the log-RR scale at the anchor points, and use these values for creating the prediction at 2020 (`P.pt`).

```
> Pp <- approx( m1$Per[,1], log(m1$Per[,2]), P.rf )$y
> P.eff <- Pp[2] + (Pp[2]-Pp[1])/diff(P.rf)*(P.pt-P.rf[2])
```

The same thing is done on the cohort scale:

```
> C.pt <- 1970 + 0:20
> C.rf <- 1970 - c(30,0)
> Cp <- approx( m1$Coh[,1], log(m1$Coh[,2]), C.rf )$y
> C.eff <- Cp[2] + (Cp[2]-Cp[1])/diff(C.rf)*(C.pt-C.rf[2])
```

Finally, these are added to the plot of the effects, after we have re-drawn the frame with a calendar-time axis extending to 2020 (remember that the `P.eff` and the `C.eff` are log-RRs, and hence we need to take the exp before plotting):

```
> par( las=1, mar=c(3,4,1,4), mgp=c(3,1,0)/1.5 )
> fp <-  apc.frame( a.lab = seq(30,90,10),
+                  cp.lab = seq(1860,2020,20),
+                   r.lab = c(c(1,2,5)*10,c(1,2,5)*100),
+ #                rr.lab = r.lab / rr.ref,
+                  rr.ref = 100,
+                   a.tic = seq(30,90,5),
+                  cp.tic = seq(1855,2020,5),
+                   r.tic = c(9,1:9*10,1:5*100),
+ #                rr.tic = r.tic / rr.ref,
+                 tic.fac = 1.3,
+                   a.txt = "Age",
+                  cp.txt = "Calendar time",
+                   r.txt = "Rate per 100,000 person-years",
+                  rr.txt = "Rate ratio",
+                     gap = 8,
+                col.grid = gray(0.85),
+                   sides = c(1,2,4) )
> apc.lines( m1, frame.par=fp, ci=T, col="red", lwd=c(3,1,1) )
> lines( P.pt-fp[1], exp(P.eff)*fp[2], col=gray(0.0), lty="11", lwd=2 )
> lines( C.pt-fp[1], exp(C.eff)*fp[2], col=gray(0.0), lty="11", lwd=2 )
```

6. The fitted model gives an age-effect, a period effect and a cohort effect; the `apc` object contains representations of these three effects as matrices with the age-values and the estimated effects (with c.i.s) at these values and similarly for the period and cohort effects.

   Prediction of the future rates will be based on extrapolations of the period and the cohort effects. These must be linear in the sense that a linear function of the underlying scale affects the prediction linearly.

   Therefore we can make these extrapolations using the estimated effects, by simply applying an appropriate linear function to the estimated values.

   In this case we use an extrapolation through the period point 2000, and a point 30 years prior to this, and a cohort point 1970 and a point 30 year prior to this.

   Cross-sectional rates: The first task is the prediction of cross-sectional age-specific rates in 2020.

   First we extract the estimated age-specific rates, and define the prediction point and the anchor points:

```
> A.pt <- m1$Age[,1]
> P.pt <- 2020
> P.rf <- 2000 - c(30,0)
```

   The period effect only need one point as we are predicting the cross-sectional rates in 2020. Then we compute the estimated period effect on the log-RR scale at the anchor points, and use these values for creating the prediction at 2020 (`P.pt`)

```
> Pp <- approx( m1$Per[,1], log(m1$Per[,2]), P.rf )$y
> P.eff <- Pp[2] + (Pp[2]-Pp[1])/diff(P.rf)*(P.pt-P.rf[2])
```

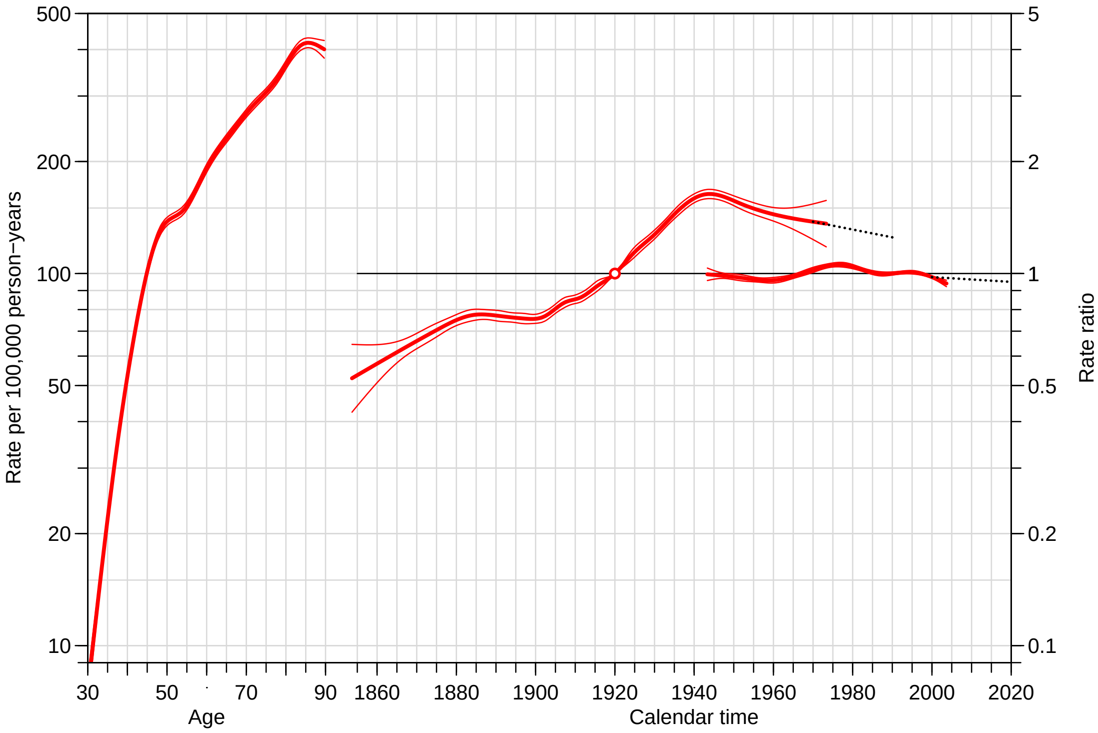

Figure 3.37: *Estimates of age- period- and cohort effects with the linear extension of the period and cohort effects used for prediction of future rates.*

For the cohort effect we need to compute it at all cohorts represented in 2020. First we compute the cohorts needed, set up a vector for the effects and then the reference points:

```
> C.pt <- P.pt - A.pt
> C.rf <- 1970 - c(30,0)
> C.eff <- numeric( length(C.pt) )
```

Then we can fill in the estimated cohort effects by interpolation for those cohorts that are before 1970:

```
> C.eff[C.pt<C.rf[2]] <- approx( m1$Coh[,1], log(m1$Coh[,2]), C.pt[C.pt<C.rf[2]] )$y
```

Subsequently we get the log-RRs for the two anchor points and use these for prediction of the cohorts after 1970:

```
> Cp <- approx( m1$Coh[,1], log(m1$Coh[,2]), C.rf )$y
> C.eff[C.pt>=C.rf[2]] <- Cp[2] + (Cp[2]-Cp[1])/diff(C.rf)*(C.pt[C.pt>C.rf[2]]-C.rf[2])
```

Finally, we can assemble the effects contributing to each of the ages represented, to give the predicted age-specific rates in 2020:

```
> A.per.2020 <- exp( log(m1$Age[,2]) + P.eff + C.eff )
```

Longitudinal rates: We can now apply a similar machinery to predict the age-specific rates for the 1950 cohort. The difference is now that the cohort effect is the same for all the points, whereas the period effects differ.

```
> # Cohort point needed --- simple because the cohort is inside the data already
> C.pt <- 1960
> C.eff <- approx( m1$Coh[,1], log(m1$Coh[,2]), C.pt )$y
> # Period points needed
> P.pt <- C.pt + A.pt
> P.rf <- 2000 - c(30,0)
> # Where to put the period effects
> P.eff <- numeric( length(P.pt) )
> P.eff[P.pt<P.rf[2]] <- approx( m1$Per[,1], log(m1$Per[,2]), P.pt[P.pt<P.rf[2]] )$y
> # Nowe we use the points from the interpolation
> Pp <- approx( m1$Per[,1], log(m1$Per[,2]), P.rf )$y
> P.eff[P.pt>=P.rf[2]] <- Pp[2] + (Pp[2]-Pp[1])/diff(P.rf)*(P.pt[P.pt>=P.rf[2]]-P.rf[2])
> # Note that the prediction of the log RRs are made based on the estimated RRs
> # that refer to the predicted age-specific rates.
> A.coh.1960 <- exp( log(m1$Age[,2]) + P.eff + C.eff )
```

Finally, we can plot the two predictions and the age-effect from the model, see figure 3.38

```
> matplot( A.pt, cbind( m1$Age[,2], A.coh.1960, A.per.2020 ),
+          type="l", lty=1, lwd=2, col=c("red","blue","black"),
+          log="y", xlab="Age", ylab="Predicted rates per 100,000" )
> abline( v=seq(30,90, 5), h=outer(1:9,1:3,function(x,y) x*10^y), col=gray(0.9) )
> abline( v=seq(30,90,10), h=outer(c(1,2,5),1:3,function(x,y) x*10^y), col=gray(0.8) )
> matlines( A.pt, cbind( m1$Age[,2], A.coh.1960, A.per.2020 ),
+           type="l", lty=1, lwd=5, col=c("red","blue","black") )
> box()
```

It is clear from the plot in figure 3.38 that the prediction of the cohort rates in the 1960 cohort are approximately proportional to the estimated age-effect. They are actually not, but the prediction of the period effects are almost constant, so the disturbance from the period effect over the lifespan of the 1960 cohort is minimal, and not visually detectable in the graph.

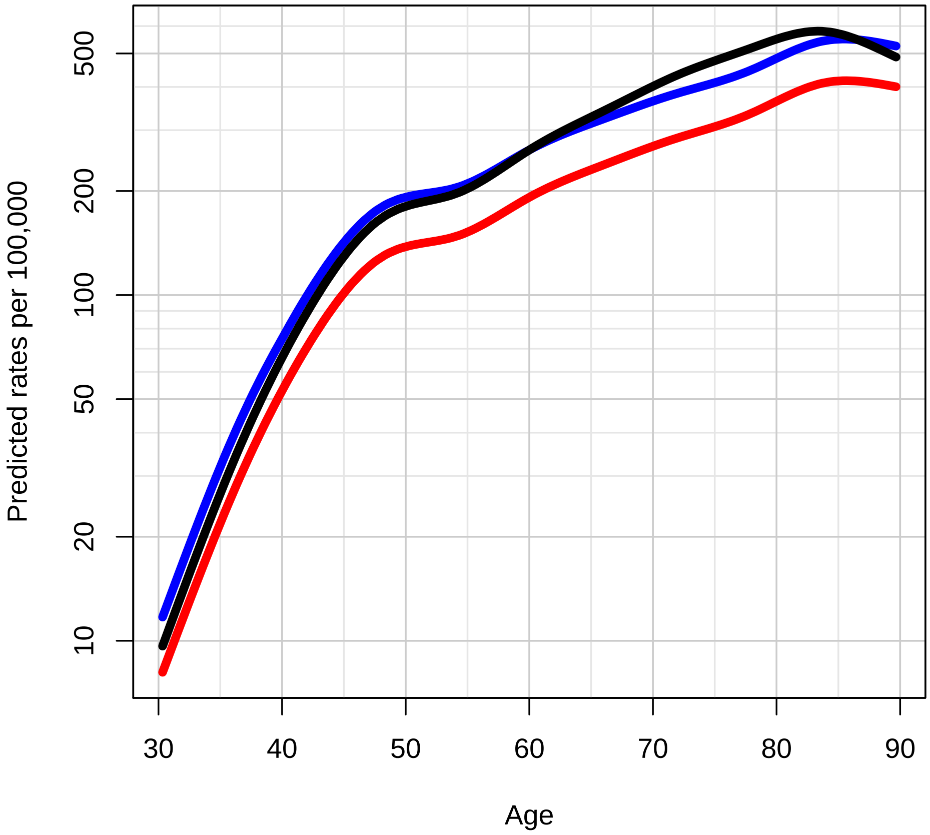

Figure 3.38: *Predicted age-specific breast cancer rates at 2020 (black) and in the 1950 cohort (blue) and the estimated age-effects.*

Chapter 4

The Epi manual (1.0.11)

apc.fit *Fit an Age-Period-Cohort model to tabular data.*

Description

Fits the classical five models to tabulated rate data (cases, person-years) classified by two of age, period, cohort: Age, Age-drift, Age-Period, Age-Cohort and Age-period. There are no assumptions about the age, period or cohort classes being of the same length, or that tabulation should be only by two of the variables. Only requires that mean age and period for each tabulation unit is given.

Usage

```
apc.fit( data,
            A,
            P,
            D,
            Y,
        ref.c,
        ref.p,
          dist = c("poisson","binomial"),
         model = c("ns","bs","ls","factor"),
       dr.extr = c("weighted","Holford"),
          parm = c("ACP","APC","AdCP","AdPC","Ad-P-C","Ad-C-P","AC-P","AP-C"),
          npar = c( A=5, P=5, C=5 ),
         scale = 1,
         alpha = 0.05,
     print.AOV = TRUE )
```

Arguments

`data` Data frame with (at least) variables, `A` (age), `P` (period), `D` (cases, deaths) and `Y` (person-years). Cohort (date of birth) is computed as `P-A`. If thsi argument is given the arguments `A`, `P`, `D` and `Y` are ignored.

`A` Age; numerical vector with mean age at diagnosis for each unit.

`P` Period; numerical vector with mean date of diagnosis for each unit.

`D` Cases, deaths; numerical vector.

`Y` Person-years; numerical vector. Also used as denominator for binomial data, see the `dist` argument.

`ref.c` Reference cohort, numerical. Defaults to median date of birth among cases. If used with `parm="AdCP"` or `parm="AdPC"`, the resdiual cohort effects will be 1 at `ref.c` `ref.p` Reference period, numerical. Defaults to median date of diagnosis among cases.

`dist` Distribution (or more precisely: Likelihood) used for modelling. if a binomial model us ised, `Y` is assuemd to be the denominator; `"binomial"` gives a binomial model with logit link.

`model` Type of model fitted:

- `ns` fits a model with natural splines for each of the terms, with `npar` parameters for the terms.
- `bs` fits a model with B-splines for each of the terms, with `npar` parameters for the terms.
- `ls` fits a model with linear splines.
- `factor` fits a factor model with one parameter per value of `A`, `P` and `C`. `npar` is ignored in this case.

`dr.extr` Character. How the drift parameter should be extracted from the age-period-cohort model. `"weighted"` (default) lets the weighted average (by marginal no. cases, `D`) of the estimated period and cohort effects have 0 slope. `"Holford"` uses the naive average over all values for the estimated effects, disregarding the no. cases.

`parm` Character. Indicates the parametrization of the effects. The first four refer to the ML-fit of the Age-Period-Cohort model, the last four give Age-effects from a smaller model and residuals relative to this. If one of the latter is chosen, the argument `dr.extr` is ignored. Possible values for `parm` are:

- `"ACP"`: ML-estimates. Age-effects as rates for the reference cohort. Cohort effects as RR relative to the reference cohort. Period effects constrained to be 0 on average with 0 slope.
- `"APC"`: ML-estimates. Age-effects as rates for the reference period. Period effects as RR relative to the reference period. Cohort effects constrained to be 0 on average with 0 slope.
- `"AdCP"`: ML-estimates. Age-effects as rates for the reference cohort. Cohort and period effects constrained to be 0 on average with 0 slope. These effects do not multiply to the fitted rates, the drift is missing and needs to be included to produce the fitted values.
- `"AdPC"`: ML-estimates. Age-effects as rates for the reference period. Cohort and period effects constrained to be 0 on average with 0 slope. These effects do not multiply to the fitted rates, the drift is missing and needs to be included to produce the fitted values.
- `"Ad-C-P"`: Age effects are rates for the reference cohort in the Age-drift model (cohort drift). Cohort effects are from the model with cohort alone, using log(fitted values) from the Age-drift model as offset. Period effects are from the model with period alone using log(fitted values) from the cohort model as offset.
- `"Ad-P-C"`: Age effects are rates for the reference period in the Age-drift model (period drift). Period effects are from the model with period alone, using log(fitted values) from the Age-drift model as offset. Cohort effects are from the model with cohort alone using log(fitted values) from the period model as offset.
- `"AC-P"`: Age effects are rates for the reference cohort in the Age-Cohort model, cohort effects are RR relative to the reference cohort. Period effects are from the model with period alone, using log(fitted values) from the Age-Cohort model as offset.
- `"AP-C"`: Age effects are rates for the reference period in the Age-Period model, period effects are RR relative to the reference period. Cohort effects are from the model with cohort alone, using log(fitted values) from the Age-Period model as offset.

`npar` The number of parameters to use for each of the terms in the model. It can be a list of three numerical vectors, in which case these taken as the knots for the age, period and cohort effect, the first and last element in each vector are used as the boundary knots.

`alpha` The significance level. Estimates are given with (1-`alpha`) confidence limits.

`scale` numeric(1), factor multiplied to the rate estimates before output.

`print.AOV` Should the analysis of deviance table for the models be printed?

Value

An object of class "apc" (recognized by `apc.lines` and `apc.plot`) — a list with components:

`Age` Matrix with 4 colums: `A.pt` with the ages (equals `unique(A)`) and three columns giving the estimated rates with c.i.s.

`Per` Matrix with 4 colums: `P.pt` with the dates of diagnosis (equals `unique(P)`) and three columns giving the estimated RRs with c.i.s.

`Coh` Matrix with 4 colums: `C.pt` with the dates of birth (equals `unique(P-A)`) and three columns giving the estimated RRs with c.i.s.

`Drift` A 3 column matrix with drift-estimates and c.i.s: The first row is the ML-estimate of the drift (as defined by `drift`), the second row is the estimate from the Age-drift model. For the sequential parametrizations, only the latter is given.

`Ref` Numerical vector of length 2 with reference period and cohort. If ref.p or ref.c was not supplied the corresponding element is NA.

`AOV` Analysis of deviance table comparing the five classical models.

`Type` Character string explaining the model and the parametrization.

`Knots` If `model` is one of `"ns"` or `"bs"`, a list with three components: `Age`, `Per`, `Coh`, each one a vector of knots. The max and the min are the boundary knots.

Author(s)

Bendix Carstensen, `http://www.biostat.ku.dk/~bxc`

References

The considerations behind the parametrizations used in this function are given in details in a preprint from Department of Biostatistics in Copenhagen: `http://www.pubhealth.ku.dk/bs/publikationer/rr-06-1.pdf`, later published as: B. Carstensen: Age-period-cohort models for the Lexis diagram. Statistics in Medicine, 10; 26(15):3018-45, 2007.

See Also

`apc.frame`, `apc.lines`, `apc.plot`.

Examples

```
library( Epi )
data(lungDK)

# Taylor a dataframe that meets the requirements
exd <- lungDK[,c("Ax","Px","D","Y")]
names(exd)[1:2] <- c("A","P")

# Two different ways of parametrizing the APC-model, ML
ex.H <- apc.fit( exd, npar=7, model="ns", dr.extr="Holford",  parm="ACP", scale=10^5 )
ex.W <- apc.fit( exd, npar=7, model="ns", dr.extr="weighted", parm="ACP", scale=10^5 )

# Sequential fit, first AC, then P given AC.
ex.S <- apc.fit( exd, npar=7, model="ns", parm="AC-P", scale=10^5 )

# Show the estimated drifts
ex.H[["Drift"]]
ex.W[["Drift"]]
ex.S[["Drift"]]

# Plot the effects
fp <- apc.plot( ex.H )
apc.lines( ex.W, frame.par=fp, col="red" )
apc.lines( ex.S, frame.par=fp, col="blue" )
```

apc.frame	*Produce an empty frame for display of parameter-estimates from Age-Period-Cohort-models.*

Description

A plot is generated where both the age-scale and the cohort/period scale is on the x-axis. The left vertical axis will be a logarithmic rate scale referring to age-effects and the right a logarithmic rate-ratio scale of the same relative extent as the left referring to the cohort and period effects (rate ratios).

Only an empty plot frame is generated. Curves or points must be added with `points`, `lines` or the special utility function `apc.lines`.

Usage

```
apc.frame( a.lab,
          cp.lab,
           r.lab,
          rr.lab = r.lab / rr.ref,
          rr.ref = r.lab[length(r.lab)/2],
           a.tic = a.lab,
          cp.tic = cp.lab,
           r.tic = r.lab,
          rr.tic = r.tic / rr.ref,
         tic.fac = 1.3,
           a.txt = "Age",
          cp.txt = "Calendar time",
           r.txt = "Rate per 100,000 person-years",
          rr.txt = "Rate ratio",
        ref.line = TRUE,
             gap = diff(range(c(a.lab, a.tic)))/3,
        col.grid = gray(0.85),
           sides = c(1,2,4) )
```

Arguments

`a.lab`	Numerical vector of labels for the age-axis.
`cp.lab`	Numerical vector of labels for the cohort-period axis.
`r.lab`	Numerical vector of labels for the rate-axis (left vertical)
`rr.lab`	Numerical vector of labels for the RR-axis (right vertical)
`rr.ref`	At what level of the rate scale is the RR=1 to be.
`a.tic`	Location of additional tick marks on the age-scale
`cp.tic`	Location of additional tick marks on the cohort-period-scale
`r.tic`	Location of additional tick marks on the rate-scale
`rr.tic`	Location of additional tick marks on the RR-axis.
`tic.fac`	Factor with which to diminish intermediate tick marks
`a.txt`	Text for the age-axis (left part of horizontal axis).
`cp.txt`	Text for the cohort/period axis (right part of horizontal axis).
`r.txt`	Text for the rate axis (left vertical axis).
`rr.txt`	Text for the rate-ratio axis (right vertical axis)
`ref.line`	Logical. Should a reference line at RR=1 be drawn at the calendar time part of the plot?
`gap`	Gap between the age-scale and the cohort-period scale
`col.grid`	Colour of the grid put in the plot.
`sides`	Numerical vector indicating on which sides axes should be drawn and annotated. This option is aimed for multi-panel displays where axes only are put on the outer plots.

Details

The function produces an empty plot frame for display of results from an age-period-cohort model, with age-specific rates in the left side of the frame and cohort and period rate-ratio parameters in the right side of the frame. There is a gap of `gap` between the age-axis and the calendar time axis, vertical grid lines at `c(a.lab,a.tic,cp.lab,cp.tic)`, and horizontal grid lines at `c(r.lab,r.tic)`.

The function returns a numerical vector of length 2, with names `c("cp.offset","RR.fac")`. The y-axis for the plot will be a rate scale for the age-effects, and the x-axis will be the age-scale. The cohort and period effects are plotted by subtracting the first element (named `"cp.offset"`) of the returned result form the cohort/period, and multiplying the rate-ratios by the second element of the returned result (named `"RR.fac"`).

Value

A numerical vector of length two, with names `c("cp.offset","RR.fac")`. The first is the offset for the cohort period-axis, the second the multiplication factor for the rate-ratio scale.

Side-effect: A plot with axes and grid lines but no points or curves.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://www.pubhealth.ku.dk/~bxc/

References

http://www.pubhealth.ku.dk/~bxc/APC/notes.pdf

See Also

`apc.lines`,`apc.lines`

Examples

```
par( mar=c(4,4,1,4) )
res <-
apc.frame( a.lab=seq(30,90,20), cp.lab=seq(1880,2000,30), r.lab=c(1,2,5,10,20,50),
           a.tic=seq(30,90,10), cp.tic=seq(1880,2000,10), r.tic=c(1:10,1:5*10),
           gap=27 )
res
# What are the axes actually?
par(c("usr","xlog","ylog"))
# How to plot in the age-part: a point at (50,10)
points( 50, 10, pch=16, cex=2, col="blue" )
# How to plot in the cohort-period-part: a point at (1960,0.3)
points( 1960-res[1], 0.3*res[2], pch=16, cex=2, col="red" )
```

apc.lines *Plot APC-estimates in an APC-frame.*

Description

When an APC-frame has been produced by `apc.frame`, this function draws a set of estimates from an APC-fit in the frame. An optional drift parameter can be added to the period parameters and subtracted from the cohort and age parameters.

Usage

```
apc.lines( A, P, C,
       scale = c("log","ln","rates","inc","RR"),
   frame.par = NULL,
       drift = 0,
          c0 = median( C[,1] ),
          a0 = median( A[,1] ),
          p0 = c0 + a0,
          ci = rep( FALSE, 3 ),
         lwd = c(3,1,1),
         lty = 1,
         col = "black",
        type = "l",
       knots = FALSE,
         ... )
```

Arguments

A	Age effects. A 4-column matrix with columns age, age-specific rates, lower and upper c.i. If A is of class apc (see apc.fit, P, C, c0, a0 and p0 are ignored, and the estimates from there plotted.

P	Period effects. Rate-ratios. Same form as for the age-effects.

C	Cohort effects. Rate-ratios. Same form as for the age-effects.

scale	Are effects given on a log-scale? Character variable, one of "log", "ln", "rates", "inc", "RR". If "log" or "ln" it is assumed that effects are log(rates) and log(RRs) otherwise the actual effects are assumed given in A, P and C. If A is of class apc, it is assumed to be "rates".

frame.par	2-element vector with the cohort-period offset and RR multiplicator. This will typically be the result from the call of apc.frame. See this for details.

drift	The drift parameter to be added to the period effect. If scale="log" this is assumed to be on the log-scale, otherwise it is assumed to be a multiplicative factor per unit of the first columns of A, P and C

c0	The cohort where the drift is assumed to be 0; the subtracted drift effect is drift*(C[,1]-c0).

a0	The age where the drift is assumed to be 0.

p0	The period where the drift is assumed to be 0.

ci	Should confidence interval be drawn. Logical or character. If character, any occurrence of "a" or "A" produces confidence intervals for the age-effect. Similarly for period and cohort.

lwd	Line widths for estimates, lower and upper confidence limits.

lty	Linetypes for the three effects.

col	Colours for the three effects.

type	What type of lines / points should be used.

knots	Should knots from the model be shown?

...	Further parameters to be transmitted to matlines used for plotting the three sets of curves.

Details

The drawing of three effects in an APC-frame is a rather trivial task, and the main purpose of the utility is to provide a function that easily adds the functionality of adding a drift so that several sets of lines can be easily produced in the same frame.

Value

A list of three matrices with the effects plotted is returned invisibly.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://www.pubhealth.ku.dk/~bxc

See Also

\code{apc.frame}, \code{apc.frame}, \code{apc.frame} apc.plot *Plot the estimates from a fitted Age-Period-Cohort model*

Description

This function plots the estimates created by apc.fit in a single graph. It just calls apc.frame after computing some sensible values of the parameters, and subsequently plots the estimates using apc.lines.

Usage

```
apc.plot(obj, r.txt = "Rate", ...)
```

Arguments

obj	An object of class apc.
r.txt	The text to put on the vertical rate axis.
...	Additional arguments passed on to apc.lines.

Value

A numerical vector of length two, with names c("cp.offset","RR.fac"). The first is the offset for the cohort period-axis, the second the multiplication factor for the rate-ratio scale. Therefore, if you want to plot at (x,y) in the right panel, use (x-res["cp.offset"],y/res["RR.fac"])=(x-res[1],y/res[2]). This vector should be supplied for the parameter frame.par to apc.lines if more sets of estimates is plotted in the same graph.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://www.pubhealth.ku.dk/~bxc

See Also

\code{apc.lines} ,\code{apc.lines}, \code{apc.lines}

Examples

```
data( lungDK )
attach( lungDK )
apc1 <- apc.fit( A=Ax, P=Px, D=D, Y=Y/10^5 )
fp <- apc.plot( apc1 )
apc.lines( apc1, frame.par=fp, drift=1.01, col="red" )
for( i in 1:11 )
  apc.lines( apc1, frame.par=fp, drift=1+(i-6)/100, col=rainbow(12)[i] )
```

bdendo *A case-control study of endometrial cancer*

Description

The bdendo data frame has 315 rows and 13 columns. These data concern a study in which each case of endometrial cancer was matched with 4 controls. Matching was by date of birth (within one year), marital status, and residence.

Format

This data frame contains the following columns:

`set`: Case-control set: a numeric vector

`d`: Case or control: a numeric vector (1=case, 0=control)

`gall`: Gall bladder disease: a factor with levels `No` `Yes`.

`hyp`: Hypertension: a factor with levels `No` `Yes`.

`ob`: Obesity: a factor with levels `No` `Yes`.

`est`: A factor with levels `No` `Yes`.

`dur`: Duration of conjugated oestrogen therapy: an ordered factor with levels $0 < 1 < 2 < 3 < 4$.

`non`: Use of non oestrogen drugs: a factor with levels `No` `Yes`.

`duration`: Months of oestrogen therapy: a numeric vector.

`age`: A numeric vector.

`cest`: Conjugated oestrogen dose: an ordered factor with levels $0 < 1 < 2 < 3$.

`agegrp`: A factor with levels `55-59` `60-64` `65-69` `70-74` `75-79` `80-84`

`age3`: a factor with levels `<64` `65-74` `75+`

Source

Breslow NE, and Day N, Statistical Methods in Cancer Research. Volume I: The Analysis of Case-Control Studies. IARC Scientific Publications, IARC:Lyon, 1980.

Examples

```
data(bdendo)
```

`bdendo11` *A 1:1 subset of the endometrial cancer case-control study*

Description

The `bdendo11` data frame has 126 rows and 13 columns. This is a subset of the dataset `bdendo` in which each case was matched with a single control.

Source

Breslow NE, and Day N, Statistical Methods in Cancer Research. Volume I: The Analysis of Case-Control Studies. IARC Scientific Publications, IARC:Lyon, 1980.

Examples

```
data(bdendo11)
```

`births` *Births in a London Hospital*

Description

Data from 500 singleton births in a London Hospital

Usage

```
data(births)
```

Format

A data frame with 500 observations on the following 8 variables.

`id`:	Identity number for mother and baby.
`bweight`:	Birth weight of baby.
`lowbw`:	Indicator for birth weight less than 2500 g.
`gestwks`:	Gestation period.
`preterm`:	Indicator for gestation period less than 37 weeks.
`matage`:	Maternal age.
`hyp`:	Indicator for maternal hypertension.
`sex`:	Sex of baby: 1:Male, 2:Female.

Source

Anonymous

References

Michael Hills and Bianca De Stavola (2002). A Short Introduction to Stata 8 for Biostatistics, Timberlake Consultants Ltd `http://www.timberlake.co.uk`

Examples

```
data(births)
```

`blcaIT` *Bladder cancer mortality in Italian males*

Description

Number of deaths from bladder cancer and person-years in the Italian male population 1955–1979, in ages 25–79.

Format

A data frame with 55 observations on the following 4 variables:

`age`:	Age at death. Left endpoint of age class
`period`:	Period of death. Left endpoint of period
`D`:	Number of deaths
`Y`:	Number of person-years.

Examples

```
data(blcaIT)
```

`brv` *Bereavement in an elderly cohort*

Description

The `brv` data frame has 399 rows and 11 columns. The data concern the possible effect of marital bereavement on subsequent mortality. They arose from a survey of the physical and mental health of a cohort of 75-year-olds in one large general practice. These data concern mortality up to 1 January, 1990 (although further follow-up has now taken place).

Subjects included all lived with a living spouse when they entered the study. There are three distinct groups of such subjects: (1) those in which both members of the couple were over 75 and therefore included in the cohort, (2) those whose spouse was below 75 (and was not, therefore, part of the main cohort study), and (3) those living in larger households (that is, not just with their spouse).

Format

This data frame contains the following columns:

`id`: subject identifier, a numeric vector
`couple`: couple identifier, a numeric vector
`dob`: date of birth, a date
`doe`: date of entry into follow-up study, a date
`dox`: date of exit from follow-up study, a date
`dosp`: date of death of spouse, a date (if the spouse was still alive at the end of follow-up, this was coded to January 1, 200
`fail`: status at end of follow-up, a numeric vector (0=alive,1=dead)
`group`: see Description, a numeric vector
`disab`: disability score, a numeric vector
`health`: perceived health status score, a numeric vector
`sex`: a factor with levels `Male` `Female`

Source

Jagger C, and Sutton CJ, Death after Marital Bereavement. Statistics in Medicine, 10:395-404, 1991. (Data supplied by Carol Jagger).

Examples

```
data(brv)
```

`cal.yr` *Functions to convert character, factor and various date objects into a number, and vice versa.*

Description

Dates are converted to a numerical value, giving the calendar year as a fractional number. 1 January 1970 is converted to 1970.0, and other dates are converted by assuming that years are all 365.25 days long, so inaccuracies may arise, for example, 1 Jan 2000 is converted to 1999.999. Differences between converted values will be 1/365.25 of the difference between corresponding `Date` objects.

Usage

```
cal.yr( x, format="%Y-%m-%d", wh=NULL )
as.Date.cal.yr( x, ... )
as.Date.numeric( x, ..., unit="d" )
```

Arguments

`x` A factor or character vector, representing a date in format `format`, or an object of class `Date`, `POSIXlt`, `POSIXct`, `date`, `dates` or `chron` (the latter two requires the `chron` package). If `x` is a data frame, all variables in the data-frame which are of one the classes mentioned are converted to class `cal.yr`. See arguemt `wh`, though.

`format` Format of the date values if `x` is factor or character. If this argument is supplied and `x` is a datafame, all character variables are converted to class `cal.yr`. Factors in the dataframe will be ignored.

`wh` Indices of the variables to convert if `x` is a data frame. Can be either a numerical or character vector.

`unit` Which units are the date measured in, `"y"` for years, `"d"` for days.

`...` Arguments passed on from other methods.

Value

cal.yr returns a numerical vector of the same length as x, of class c("cal.yr","numeric"). If x is a data frame a dataframe with some of the columns converted to class "cal.yr" is returned.

as.Date.cal.yr and as.Date.numeric return Date objects.

Author(s)

Bendix Carstensen, Steno Diabetes Center & Dept. of Biostatistics, University of Copenhagen, ⟨bxc@steno.dk⟩, http://www.pubhealth.ku.dk/~bxc

See Also

DateTimeClasses, Date

Examples

```
# Charcter vector of dates:
birth <- c("14/07/1852","01/04/1954","10/06/1987","16/05/1990",
           "01/01/1996","01/01/1997","01/01/1998","01/01/1999")
# Proper conversion to class "Date":
birth.dat <- as.Date( birth, format="%d/%m/%Y" )
# Converson of character to class "cal.yr"
bt.yr <- cal.yr( birth, format="%d/%m/%Y" )
# Back to class "Date":
bt.dat <- as.Date( bt.yr )
# Numerical calculation of days since 1.1.1970:
days <- Days <- (bt.yr-1970)*365.25
# Blunt assignment of class:
class( Days ) <- "Date"
# Then data.frame() to get readable output of results:
data.frame( birth, birth.dat, bt.yr, bt.dat, days, Days, round(Days) )
```

---

ccwc *Generate a nested case-control study*

Description

Given the basic outcome variables for a cohort study: the time of entry to the cohort, the time of exit and the reason for exit ("failure" or "censoring"), this function computes risk sets and generates a matched case-control study in which each case is compared with a set of controls randomly sampled from the appropriate risk set. Other variables may be matched when selecting controls.

Usage

```
ccwc(entry=0, exit, fail, origin=0, controls=1, match=list(), include=list(), data=NULL, silent=F)
```

Arguments

entry	Time of entry to follow-up
exit	Time of exit from follow-up
fail	Status on exit (1=Fail, 0=Censored)
origin	Origin of analysis time scale
controls	The number of controls to be selected for each case
match	List of categorical variables on which to match cases and controls
include	List of other variables to be carried across into the case-control study
data	Data frame in which to look for input variables
silent	If False, echos a . to the screen for each case-control set created; otherwise produces no output.

Value

The case-control study, as a dataframe containing:

Set	case-control set number
Map	row number of record in input dataframe
Time	failure time of the case in this set
Fail	failure status (1=case, 0=control)

These are followed by the matching variables, and finally by the variables in the `include` list

Author(s)

David Clayton

References

Clayton and Hills, Statistical Models in Epidemiology, Oxford University Press, Oxford:1993.

See Also

Lexis

Examples

```
#
# For the diet and heart dataset, create a nested case-control study
# using the age scale and matching on job
#
data(diet)
dietcc <- ccwc(doe, dox, chd, origin=dob, controls=2, data=diet,
               include=energy, match=job)
```

`ci.cum` *Compute cumulative sum of estimates.*

Description

Computes the cumulative sum of parameter functions and the standard error of it. Optionally the exponential is applied to the parameter functions before it is cumulated.

Usage

```
ci.cum( obj,
    ctr.mat = NULL,
     subset = NULL,
       intl = 1,
      alpha = 0.05,
        Exp = TRUE )
```

Arguments

obj	A model object.
ctr.mat	Contrast matrix defining the parameter functions from the parameters of the model.
subset	Subset of the parameters of the model to which `ctr.mat` should be applied.
intl	Interval length for the cumulation. Either a constant or a numerical vector of length `nrow(ctr.mat)`.
alpha	Significance level used when computing confidence limits.
Exp	Should the parameter function be exponentiated before it is cumulated?

Details

The purpose of this function is to compute cumulative rate based on a model for the rates. If the model is a multiplicative model for the rates, the purpose of `ctr.mat` is to return a vector of rates or log-rates when applied to the coefficients of the model. If log-rates are returned, the they should be exponentiated before cumulated, and the variances computed accordingly. Since log-linear models are the most common the `Exp` parameter defaults to TRUE.

Value

A matrix with 4 columns: Estimate, lower and upper c.i. and standard error.

Author(s)

Bendix Carstensen, http://www.pubhealth.ku.dk/~bxc

See Also

See also `ci.lin`

Examples

```
# Packages required for this example
library( splines )
library( survival )
data( lung )
par( mfrow=c(1,2) )

# Plot the Kaplan-meier-estimator
plot( survfit( Surv( time, status==2 ) ~ 1, data=lung ) )

# Declare data as Lexis
lungL <- Lexis( exit=list("tfd"=time),
                exit.status=(status==2)*1, data=lung )
summary( lungL )

# Cut the follow-up every 10 days
sL <- splitLexis( lungL, "tfd", breaks=seq(0,1100,10) )
str( sL )
summary( sL )

# Fit a Poisson model with a natural spline for the effect of time.
# Extract the variables needed
D <- status(sL, "exit")
Y <- dur(sL)
tB <- timeBand( sL, "tfd", "left" )
MM <- ns( tB, knots=c(50,100,200,400,700), intercept=TRUE )
mp <- glm( D ~ MM - 1 + offset(log(Y)),
           family=poisson, eps=10^-8, maxit=25 )

# Contrast matrix to extract effects, i.e. matrix to multiply with the
# coefficients to produce the log-rates: unique rows of MM, in time order.
T.pt <- sort( unique( tB ) )
T.wh <- match( T.pt, tB )
Lambda <- ci.cum( mp, ctr.mat=MM[T.wh,], intl=diff(c(0,T.pt)) )

# Put the estimated survival function on top of the KM-estimator
matlines( c(0,T.pt[-1]), exp(-Lambda[,1:3]), lwd=c(3,1,1), lty=1, col="Red" )

# Extract and plot the fitted intensity function
lambda <- ci.lin( mp, ctr.mat=MM[T.wh,], Exp=TRUE )
```

```
matplot( T.pt, lambda[,5:7]*10^3, type="l", lwd=c(3,1,1), col="black", lty=1,
         log="y", ylim=c(0.2,20) )
```

ci.lin	*Compute linear functions of parameters with s.e.*

Description

For a given model object the function computes a linear function of the parameters and the corresponding standard errors, p-values and confidence intervals.

Usage

```
ci.lin( obj,
    ctr.mat = NULL,
     subset = NULL,
      diffs = FALSE,
       fnam = !diffs,
       vcov = FALSE,
      alpha = 0.05,
        Exp = FALSE )
```

Arguments

`obj`	A model object (of class `lm`, `glm`, `lme`, `coxph` or `polr`).
`ctr.mat`	Contrast matrix to be multiplied to the parameter vector, i.e. the desired linear function of the parameters.
`subset`	The subset of the parameters to be used. If given as a character vector, the elements are in turn matched against the parameter names (using `grep`) to find the subset. Repeat parameters may result from using a character vector. This is considered a facility.
`diffs`	If TRUE, all differences between parameters in the subset are computed. `ctr.mat` is ignored. If `obj` inherits from `lm`, and `subset` is given as a string `subset` is used to search among the factors in the model and differences of all factor levels for the first match are shown. If `subset` does not match any of the factors in the model, all pairwise differences between parameters matching are returned.
`fnam`	Should the common part of the parameter names be included with the annotation of contrasts? Ignored if `diffs==T`. If a sting is supplied this will be prefixed to the labels.
`vcov`	Should the covariance matrix of the set of parameters be returned? If this is set, `Exp` is ignored.
`alpha`	Significance level for the confidence intervals.
`Exp`	If `TRUE` columns 5:6 are replaced with exp(columns 1,5,6).

Value

A matrix with number of rows and rownames as `ctr.mat`. The columns are Estimate, Std.Err, z, P, 2.5% and 97.5%. If `vcov=TRUE` a list with components `est`, the desired functional of the parameters and `vcov`, the variance covariance matrix of this, is returned but not printed. If `Exp==TRUE` the confidence intervals for the parameters are replaced with three columns: exp(estimate,c.i.).

Author(s)

Bendix Carstensen, `http://www.pubhealth.ku.dk/~bxc`

Examples

```
# Bogus data:
f <- factor( sample( letters[1:5], 200, replace=TRUE ) )
g <- factor( sample( letters[1:3], 200, replace=TRUE ) )
x <- rnorm( 200 )
y <- 7 + as.integer( f ) * 3 + 2 * x + 1.7 * rnorm( 200 )

# Fit a simple model:
mm <- lm( y ~ x + f + g )
ci.lin( mm )
ci.lin( mm, subset=3:6, diff=TRUE, fnam=FALSE )
ci.lin( mm, subset=3:6, diff=TRUE, fnam=TRUE )
ci.lin( mm, subset="f", diff=TRUE, fnam="f levels:" )
print( ci.lin( mm, subset="g", diff=TRUE, fnam="gee!:", vcov=TRUE ) )

# Use character defined subset to get ALL contrasts:
ci.lin( mm, subset="f", diff=TRUE )
```

`ci.pd` *Compute confidence limits for a difference of two independent proportions.*

Description

The usual formula for the c.i. of at difference of proportions is inaccurate. Newcombe has compared 11 methods and method 10 in his paper looks like a winner. It is implemented here.

Usage

```
ci.pd(aa, bb=NULL, cc=NULL, dd=NULL,
     method = "Nc",
      alpha = 0.05, conf.level=0.95,
     digits = 3,
      print = TRUE,
detail.labs = FALSE )
```

Arguments

`aa`	Numeric vector of successes in sample 1. Can also be a matrix or array (see details).
`bb`	Successes in sample 2.
`cc`	Failures in sample 1.
`dd`	Failures in sample 2.
`method`	Method to use for calculation of confidence interval, see ”Details”.
`alpha`	Significance level
`conf.level`	Confidence level
`print`	Should an account of the two by two table be printed.
`digits`	How many digits should the result be rounded to if printed.
`detail.labs`	Should the computing of probability differences be reported in the labels.

Details

Implements method 10 from Newcombe(1998) (method=”Nc”) or from Agresti & Caffo(2000) (method=”AC”).

`aa`, `bb`, `cc` and `dd` can be vectors. If `aa` is a matrix, the elements `[1:2,1:2]` are used, with successes `aa[,1:2]`. If `aa` is a three-way table or array, the elements `aa[1:2,1:2,]` are used.

Value

A matrix with three columns: probability difference, lower and upper limit. The number of rows equals the length of the vectors `aa`, `bb`, `cc` and `dd` or, if `aa` is a 3-way matrix, `dim(aa)[3]`.

Author(s)

Bendix Carstensen, Esa Laara. http://www.biostat.ku.dk/~bxc

References

RG Newcombe: Interval estimation for the difference between independent proportions. Comparison of eleven methods. Statistics in Medicine, 17, pp. 873-890, 1998.

A Agresti & B Caffo: Simple and effective confidence intervals for proportions and differences of proportions result from adding two successes and two failures. The American Statistician, 54(4), pp. 280-288, 2000.

See Also

`twoby2`, `binom.test`

Examples

```
( a <- matrix( sample( 10:40, 4 ), 2, 2 ) )
ci.pd( a )
twoby2( t(a) )
prop.test( t(a) )
( A <- array( sample( 10:40, 20 ), dim=c(2,2,5) ) )
ci.pd( A )
ci.pd( A, detail.labs=TRUE, digits=3 )
```

`contr.cum` *Contrast matrices*

Description

Return a matrix of contrasts for factor coding.

Usage

```
contr.cum(n)
contr.diff(n)
contr.2nd(n)
contr.orth(n)
```

Arguments

`n` A vector of levels for a factor, or the number of levels.

Details

These functions are used for creating contrast matrices for use in fitting regression models. The columns of the resulting matrices contain contrasts which can be used for coding a factor with `n` levels.

`contr.cum` gives a coding corresponding to successive differences between factor levels.

`contr.diff` gives a coding that correspond to the cumulative sum of the value for each level. This is not meaningful in a model where the intercept is included, therefore `n` columns ia always returned.

`contr.2nd` gives contrasts corresponding to 2nd order differences between factor levels. Returns a matrix with `n-2` columns.

`contr.orth` gives a matrix with `n-2` columns, which are mutually orthogonal and orthogonal to the matrix `cbind(1,1:n)`

Value

A matrix with `n` rows and `k` columns, with `k=n` for `contr.diff` `k=n-1` for `contr.cum` `k=n-2` for `contr.2nd` and `contr.orth`.

Author(s)

Bendix Carstensen

See Also

`contr.treatment`

Examples

```
contr.cum(6)
contr.2nd(6)
contr.diff(6)
contr.orth(6)
```

`cutLexis` *Cut follow-up at a specified date for each person.*

Description

Follow-up intervals in a Lexis object are divided into two sub-intervals: one before and one after an intermediate event. The intermediate event may denote a change of state, in which case the entry and exit status variables in the split Lexis object are modified.

Usage

```
cutLexis(data, cut, timescale = 1, new.state, progressive=FALSE,
         precursor.states=NULL, count=FALSE)
countLexis(data, cut, timescale = 1)
```

Arguments

`data` A `Lexis` object.

`cut` A numeric vector with the times of the intermediate event. If a time is missing (`NA`) then the event is assumed to occur at time `Inf`. `cut` can also be a dataframe, see details.

`timescale` The timescale that `cut` refers to.

`new.state` an optional vector, to be used when the cut point denotes a change of state. This may be a single value, which is applied to all rows of `data`, or a vector with a separate value for each row

`progressive` a logical flag that determines the changes to exit status. See Details below

`precursor.states`
an optional vector of states to be considered as ”less severe” than `new.state`. See Details below

`count` logical indicating wheter the `countLexis` options should be used. Specifying `count=TRUE` amounts to calling `countLexis`, and the arguments `new.state`, `progressive` and `precursor.states` will be ignored.

Details

The cutLexis function allows a number of different ways of specifying the cutpoints and of modifying the status variable.

If the cut argument is a dataframe it must have columns lex.id, cut and new.state. The values of lex.id must be unique. In this case it is assumed that each row represents a cutpoint (on the timescale indicated innthe arument timescale). This cutpoint will be applied to all records in data with the corresponding lex.id. This makes it possible to apply cutLexis to a split Lexis object.

If the new.state argument is omitted, then the subject is assumed to remain in the entry state. In this case, if an interval is split, the entry status is carried forward to the cut point.

If a new.state argument is supplied then, by default, the status variable is only modified at the time of the cut point. However, it is often useful to modify the status variable after the cutpoint when an important event occurs. There are three distinct ways of doing this.

If the progressive=TRUE argument is given, then a "progressive" model is assumed, in which the status can either remain the same or increase during follow-up, but never decrease. In this case, if new.state=X, then any exit status with a value less than X is replaced with X. This argument may only be used if the status variable is numeric or an ordered factor. The Lexis object must already be progressive, so that there are no rows for which the exit status is less than the entry status. If lex.Cst and lex.Xst are factors they must be ordered factors if progressive=TRUE is given.

As an alternative to the progressive argument, an explicit vector of precursor states, that are considered less severe than the new state, may be given. If new.state=X and precursor.states=c(Y,Z) then any exit status of Y or Z in the second interval is replaced with X and all other values for the exit status are retained.

The countLexis function is a variant of cutLexis when the cutpoint marks a recurrent event, and the status variable is used to count the number of events that have occurred. Times given in cut represent times of new events. Splitting with countLexis augments the status variable by 1. If the entry status is X and the exit status is Y before splitting, then after splitting the entry status is X, X+1 for the first and second intervals, respectively, and the exit status is X+1, Y+1 respectively.

Value

A Lexis object, for which each follow-up interval containing the cut point is split into two rows: one before and one after the cut point.

Note

The cutLexis function superficially resembles the splitLexis function. However, the splitLexis function splits on a vector of common cut-points for all rows of the Lexis object, whereas the cutLexis function splits on a single time point, which may be distinct for each row, and additionally modifies the status variables.

Author(s)

Bendix Carstensen, Steno Diabetes Center, ⟨bxc@steno.dk⟩, Martyn Plummer, IARC, ⟨plummer@iarc.fr⟩.

See Also

splitLexis, Lexis, summary.Lexis

Examples

```
# A small artificial example
xx <- Lexis( entry=list(age=c(17,24,33,29),per=c(1920,1933,1930,1929)),
             duration=c(23,57,12,15), exit.status=c(1,2,1,2) )
xx
cut <- c(33,47,29,50)
cutLexis(xx, cut, new.state=3, precursor=1)
cutLexis(xx, cut, new.state=3, precursor=2)
cutLexis(xx, cut, new.state=3, precursor=1:2)
# The same as the last example
cutLexis(xx, cut, new.state=3)
```

```
# The same example with a factor status variable
yy <- Lexis(entry = list(age=c(17,24,33,29),per=c(1920,1933,1930,1929)),
            duration = c(23,57,12,15),
            entry.status = factor(rep("alpha",4),
            levels=c("alpha","beta","gamma")),
            exit.status = factor(c("alpha","beta","alpha","beta"),
            levels=c("alpha","beta","gamma")))

cutLexis(yy,c(33,47,29,50),precursor="alpha",new.state="gamma")
cutLexis(yy,c(33,47,29,50),precursor=c("alpha","beta"),new.state="aleph")

## Using a dataframe as cut argument
rl <- data.frame( lex.id=1:3, cut=c(19,53,26), timescale="age", new.state=3 )
rl
cutLexis( xx, rl )
cutLexis( xx, rl, precursor=1 )
cutLexis( xx, rl, precursor=0:2 )

## It is immaterial in what order splitting and cutting is done
xs <- splitLexis( xx, breaks=seq(0,100,10), time.scale="age" )
xs
xsC <- cutLexis(xs, rl, precursor=0 )

xC <- cutLexis( xx, rl, pre=0 )
xC
xCs <- splitLexis( xC, breaks=seq(0,100,10), time.scale="age" )

xCs==xsC
xCs
```

`detrend` *Projection of a model matrix on to the orthogonal complement of a trend.*

Description

The columns of the model matrix `M` is projected on the orthogonal complement to the matrix `(1,t)`. Orthogonality is defined w.r.t. an inner product defined by the weights `weight`.

Usage

```
detrend( M, t, weight = rep(1, nrow(M)) )
```

Arguments

`M`	A model matrix.
`t`	The trend defining a subspace. A numerical vector of length `nrow(M)`
`weight`	Weights defining the inner product of vectors `x` and `y` as `sum(x*w*y)`. A numerical vector of length `nrow(M)`, defaults to a vector of 1s.

Details

The functions is intended to be used in parametrization of age-period-cohort models.

Value

A full-rank matrix with columns orthogonal to `(1,t)`.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://www.pubhealth.ku.dk/~bxc, with help from Peter Dalgaard.

See Also

projection.ip

diet *Diet and heart data*

Description

The diet data frame has 337 rows and 14 columns. The data concern a subsample of subjects drawn from larger cohort studies of the incidence of coronary heart disease (CHD). These subjects had all completed a 7-day weighed dietary survey while taking part in validation studies of dietary questionnaire methods. Upon the closure of the MRC Social Medicine Unit, from where these studies were directed, it was found that 46 CHD events had occurred in this group, thus allowing a serendipitous study of the relationship between diet and the incidence of CHD.

Format

This data frame contains the following columns:

- `id`: subject identifier, a numeric vector.
- `doe`: date of entry into follow-up study, a `Date` variable.
- `dox`: date of exit from the follow-up study, a `Date` variable.
- `dob`: date of birth, a `Date` variable.
- `y`: - number of years at risk, a numeric vector.
- `fail`: status on exit, a numeric vector (codes 1, 3, 11, and 13 represent CHD events)
- `job`: occupation, a factor with levels `Driver` `Conductor` `Bank worker`
- `month`: month of dietary survey, a numeric vector
- `energy`: total energy intake (KCal per day/100), a numeric vector
- `height`: (cm), a numeric vector
- `weight`: (kg), a numeric vector
- `fat`: fat intake (g/day), a numeric vector
- `fibre`: dietary fibre intake (g/day), a numeric vector
- `energy.grp`: high daily energy intake, a factor with levels `<=2750 KCal` `>2750 KCal`
- `chd`: CHD event, a numeric vector (1=CHD event, 0=no event)

Source

The data are described and used extensively by Clayton and Hills, Statistical Models in Epidemiology, Oxford University Press, Oxford:1993. They were rescued from destruction by David Clayton and reentered from paper printouts.

Examples

```
data(diet)
# Illustrate the follow-up in a Lexis diagram
Lexis.diagram( age=c(30,75), date=c(1965,1990),
               entry.date=cal.yr(doe), exit.date=cal.yr(dox), birth.date=cal.yr(dob),
               fail=(fail>0), pch.fail=c(NA,16), col.fail=c(NA,"red"), cex.fail=1.0,
               data=diet )
```

DMconv *Conversion to diabetes*

Description

Data from a randomized intervention study (”Addition”) where persons with prediabetic conditions are followed up for conversion to diabetes (DM). Conversion dates are interval censored. Original data are not published yet, so id-numbers have been changed and all dates have been randomly perturbed.

Usage

```
data(DMconv)
```

Format

A data frame with 1519 observations on the following 6 variables.

`id` Person identifier

`doe` Date of entry, i.e. first visit.

`dlw` Date last seen well, i.e. last visit without DM.

`dfi` Date first seen ill, i.e. first visit with DM.

`gtol` Glucose tolerance. Factor with levels: 1=”IFG” (impaired fasting glucose), 2=”IGT” (impaired glucose tolerance).

`grp` Randomization. Factor with levels: 1=”Intervention”, 2=”Control”.

Source

Signe Saetre Rasmussen, Steno Diabetes Center. The Addition Study.

Examples

```
data(DMconv)
str(DMconv)
head(DMconv)
```

effx.match *Function to calculate effects for individually matched case-control studies*

Description

The function calculates the effects of an exposure on a response, possibly stratified by a stratifying variable, and/or controlled for one or more confounding variables.

Usage

```
effx.match(response,
exposure,
match,
strata=NULL,
control=NULL,
base=1,
digits=3,
alpha=0.05,
data=NULL)
```

Arguments

`response`	The `response` variable - must be numeric
`exposure`	The `exposure` variable can be numeric or a factor
`match`	The variable which identifies the matched sets
`strata`	The `strata` stratifying variable - must be a factor
`control`	The `control` variable(s). These are passed as a list if there are more than one of them.
`base`	Baseline for the effects of a categorical exposure, default 1
`digits`	Number of significant digits for the effects, default 3
`alpha`	1 - confidence level
`data`	`data` refers to the data used to evaluate the function

Details

Effects are calculated odds ratios. The function is a wrapper for clogit, from the survival package. The k-1 effects for a categorical exposure with k levels are relative to a baseline which, by default, is the first level. The effect of a metric (quantitative) exposure is calculated per unit of exposure. The exposure variable can be numeric or a factor, but if it is an ordered factor the order will be ignored.

Value

`comp1`	Effects of exposure
`comp2`	Tests of significance

Author(s)

Michael Hills

References

www.mhills.pwp.blueyonder.co.uk

Examples

```
library(Epi)
library(survival)
data(bdendo)

# d is the case-control variable, set is the matching variable.
# The variable est is a factor and refers to estrogen use (no,yes)
# The variable hyp is a factor with 2 levels and refers to hypertension (no, yes)
# effect of est on the odds of being a case
effx.match(d,exposure=est,match=set,data=bdendo)
# effect of est on the odds of being a case, stratified by hyp
effx.match(d,exposure=est,match=set,strata=hyp,data=bdendo)
# effect of est on the odds of being a case, controlled for hyp
effx.match(d,exposure=est,match=set,control=hyp,data=bdendo)
```

`effx` *Function to calculate effects*

Description

The function calculates the effects of an exposure on a response, possibly stratified by a stratifying variable, and/or controlled for one or more confounding variables.

Usage

```
effx( response, type = "metric",
                  fup = NULL,
            exposure,
               strata = NULL,
              control = NULL,
              weights = NULL,
                alpha = 0.05,
                 base = 1,
               digits = 3,
                 data = NULL )
```

Arguments

`response`	The `response` variable - must be numeric
`type`	The type of response`type` - must be one of "metric", "binary", "failure", or "count"
`fup`	The `fup` variable contains the follow-up time for a failure response
`exposure`	The `exposure` variable can be numeric or a factor
`strata`	The `strata` stratifying variable - must be a factor
`control`	The `control` variable(s) - these are passed as a list if there are more than one.
`weights`	Frequency weights for binary response only
`base`	Baseline for the effects of a categorical exposure, default 1
`digits`	Number of significant digits for the effects, default 3
`alpha`	1 - confidence level
`data`	`data` refers to the data used to evaluate the function

Details

The function is a wrapper for glm. Effects are calculated as differences in means for a metric response, odds ratios for a binary response, and rate ratios for a failure or count response.

The k-1 effects for a categorical exposure with k levels are relative to a baseline which, by default, is the first level. The effect of a metric (quantitative) exposure is calculated per unit of exposure.

The exposure variable can be numeric or a factor, but if it is an ordered factor the order will be ignored.

Value

`comp1`	Effects of exposure
`comp2`	Tests of significance

Author(s)

Michael Hills

References

www.mhills.pwp.blueyonder.co.uk

Examples

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

# bweight is the birth weight of the baby in gms, and is a metric
# response (the default)
# effect of hypertension on birth weight
effx(bweight,exposure=hyp,data=births)
# effect of hypertension on birth weight stratified by sex
effx(bweight,exposure=hyp,strata=sex,data=births)
# effect of hypertension on birth weight controlled for sex
effx(bweight,exposure=hyp,control=sex,data=births)
# effect of gestation time on birth weight
effx(bweight,exposure=gestwks,data=births)
# effect of gestation time on birth weight stratified by sex
effx(bweight,exposure=gestwks,strata=sex,data=births)
# effect of gestation time on birth weight controlled for sex
effx(bweight,exposure=gestwks,control=sex,data=births)

# lowbw is a binary response coded 1 for low birth weight and 0 otherwise
# effect of hypertension on low birth weight
effx(lowbw,type="binary",exposure=hyp,data=births)
# etc.
```

ewrates *Rates of lung and nasal cancer mortality, and total mortality.*

Description

England and Wales mortality rates from lung cancer, nasal cancer, and all causes 1936 - 1980. The 1936 rates are repeated as 1931 rates in order to accomodate follow up for the nickel study.

Usage

```
data(ewrates)
```

Format

A data frame with 150 observations on the following 5 variables:

id:	Subject identifier (numeric)
year	Calendar period, 1931: 1931–35, 1936: 1936–40, . . .
age	Age class: 10: 10–14, 15:15–19, . . .
lung	Lung cancer mortality rate per 1,000,000 py.
nasal	Nasal cancer mortality rate per 1,000,000 py.
other	All cause mortality rate per 1,000,000 py.

Source

From Breslow and Day, Vol II, Appendix IX.

Examples

```
data(ewrates)
str(ewrates)
```

expand.data *Function to expand data for regression analysis of interval censored data.*

Description

This is a utility function.

The original records with `first.well`, `last.well` and `first.ill` are expanded to multiple records; several for each interval where the person is known to be well and one where the person is known to fail. At the same time columns for the covariates needed to estimate rates and the response variable are generated.

Usage

```
expand.data(fu, formula, breaks, data)
```

Arguments

`fu`	A 3-column matrix with `first.well`, `last.well` and `first.ill` in each row.
`formula`	Model fromula, used to derive the model matrix.
`breaks`	Defines the intervals in which the baseline rate is assumed constant. All follow-up before the first and after the last break is discarded.
`data`	Datafrem in which `fu` and `formula` is interpreted.

Value

Returns a list with three components

`rates.frame`	Dataframe of covariates for estimation of the baseline rates — one per interval defined by `breaks`.
`cov.frame`	Dataframe for estimation of the covariate effects. A data-framed version of the designmatrix from `formula`.
`y`	Response vector.

Author(s)

Martyn Plummer, ⟨plummer@iarc.fr⟩

References

B Carstensen: Regression models for interval censored survival data: application to HIV infection in Danish homosexual men. Statistics in Medicine, 15(20):2177-2189, 1996.

See Also

`Icens` `fit.mult` `fit.add`

`fit.add` *Fit an addive excess risk model to interval censored data.*

Description

Utility function.

The model fitted assumes a piecewise constant intensity for the baseline, and that the covariates act additively on the rate scale.

Usage

```
  fit.add( y, rates.frame, cov.frame, start )
```

Arguments

`y`	Binary vector of outcomes
`rates.frame`	Dataframe expanded from the original data by `expand.data`, cooresponding to covariates for the rate parameters.
`cov.frame`	do., but covariates corresponding to the `formula` argument of `Icens`
`start`	Starting values for the rate parameters. If not supplied, then starting values are generated.

Value

A list with one component:

`rates` A glm object from a binomial model with log-link function.

Author(s)

Martyn Plummer, ⟨plummer@iarc.fr⟩

References

B Carstensen: Regression models for interval censored survival data: application to HIV infection in Danish homosexual men. Statistics in Medicine, 15(20):2177-2189, 1996.

CP Farrington: Interval censored survival data: a generalized linear modelling approach. Statistics in Medicine, 15(3):283-292, 1996.

See Also

`Icens` `fit.mult`

Examples

```
  data( HIV.dk )
```

`fit.baseline` *Fit a piecewise contsnt intesity model for interval censored data.*

Description

Utility function

Fits a binomial model with logaritmic link, with `y` as outcome and covariates in `rates.frame` to estimate rates in the inttervals between `breaks`.

Usage

```
fit.baseline( y, rates.frame, start )
```

Arguments

`y` Binary vector of outcomes

`rates.frame` Dataframe expanded from the original data by `expand.data`

`start` Starting values for the rate parameters. If not supplied, then starting values are generated.

Value

A `glm` object, with binomial error and logaritmic link.

Author(s)

Martyn Plummer, ⟨plummer@iarc.fr⟩

See Also

`fit.add` `fit.mult`

`fit.mult` *Fits a multiplicative relative risk model to interval censored data.*

Description

Utility function.

The model fitted assumes a piecewise constant baseline rate in intervals specified by the argument `breaks`, and a multiplicative relative risk function.

Usage

```
fit.mult( y, rates.frame, cov.frame, start )
```

Arguments

`y`	Binary vector of outcomes
`rates.frame`	Dataframe expanded from the original data by `expand.data`, cooresponding to covariates for the rate parameters.
`cov.frame`	do., but covariates corresponding to the `formula` argument of `Icens`
`start`	Starting values for the rate parameters. If not supplied, then starting values are generated.

Details

The model is fitted by alternating between two generalized linear models where one estimates the underlying rates in the intervals, and the other estimates the log-relative risks.

Value

A list with three components:

`rates`	A glm object from a binomial model with log-link, estimating the baseline rates.
`cov`	A glm object from a binomial model with complementary log-log link, estimating the log-rate-ratios
`niter`	Nuber of iterations, a scalar

Author(s)

Martyn Plummer, ⟨plummer@iarc.fr⟩, Bendix Carstensen, ⟨bxc@steno.dk⟩

References

B Carstensen: Regression models for interval censored survival data: application to HIV infection in Danish homosexual men. Statistics in Medicine, 15(20):2177-2189, 1996.

CP Farrington: Interval censored survival data: a generalized linear modelling approach. Statistics in Medicine, 15(3):283-292, 1996.

See Also

`Icens` `fit.add`

Examples

```
data( HIV.dk )
```

float *Calculate floated variances*

Description

Given a fitted model object, the `float()` function calculates floating variances (a.k.a. quasi-variances) for a given factor in the model.

Usage

```
float(object, factor, iter.max=50)
```

Arguments

`object`	a fitted model object
`factor`	character string giving the name of the factor of interest. If this is not given, the first factor in the model is used.
`iter.max`	Maximum number of iterations for EM algorithm

Details

The `float()` function implements the ”floating absolute risk” proposal of Easton, Peto and Babiker(1992). This is an alternative way of presenting parameter estimates for factors in regression models, which avoids some of the difficulties of treatment contrasts. It was originally designed for epidemiological studies of relative risk, but the idea is widely applicable.

Treatment contrasts are not orthogonal. Consequently, the variances of treatment contrast estimates may be inflated by a poor choice of reference level, and the correlations between them may also be high. The `float()` function associates each level of the factor with a ”floating” variance (or quasi-variance), including the reference level. Floating variances are not real variances, but they can be used to calculate the variance error of contrast by treating each level as independent.

Plummer (2003) showed that floating variances can be derived from a covariance structure model applied to the variance-covariance matrix of the contrast estimates. This model can be fitted by minimizing the Kullback-Leibler information divergence between the true distribution of the parameter estimates and the simplified distribution given by the covariance structure model. Fitting is done using the EM algorithm.

In order to check the goodness-of-fit of the floating variance model, the `float()` function compares the standard errors predicted by the model with the standard errors derived from the true variance-covariance matrix of the parameter contrasts. The maximum and minimum ratios between true and model-based standard errors are calculated over all possible contrasts. These should be within 5 percent, or the use of the floating variances may lead to invalid confidence intervals.

Value

An object of class `floated`. This is a list with the following components

`coef`	A vector of coefficients. These are the same as the treatment contrasts but the reference level is present with coefficient 0.
`var`	A vector of floating (or quasi-) variances
`limits`	The bounds on the accuracy of standard errors over all possible contrasts

Note

Menezes(1999) and Firth and Menezes (2004) take a slightly different approach to this problem, using a pseudo-likelihood approach to fit the quasi-variance model. Their work is implemented in the package qvcalc.

Author(s)

Martyn Plummer

References

Easton DF, Peto J and Babiker GAG (1991) Floating absolute risk: An alternative to relative risk in survival and case control analysis avoiding an arbitrary reference group. *Statistics in Medicine*, **10**, 1025-1035.

Firth D and Mezezes RX (2004) Quasi-variances. *Biometrika* **91**, 65-80.

Menezes RX(1999) More useful standard errors for group and factor effects in generalized linear models. *D.Phil. Thesis*, Department of Statistics, University of Oxford.

Plummer M (2003) Improved estimates of floating absolute risk, *Statistics in Medicine*, **23**, 93-104.

See Also

`ftrend`, `qvcalc`

`ftrend` *Fit a floating trend to a factor in generalized linear model*

Description

Fits a "floating trend" model to the given factor in a glm in a generalized linear model by centering covariates.

Usage

```
ftrend(object, ...)
```

Arguments

`object`	fitted `lm` or `glm` object. The model must not have an intercept term
`...`	arguments to the `nlm` function

Details

`ftrend()` calculates "floating trend" estimates for factors in generalized linear models. This is an alternative to treatment contrasts suggested by Greenland et al. (1999). If a regression model is fitted with no intercept term, then contrasts are not used for the first factor in the model. Instead, there is one parameter for each level of this factor. However, the interpretation of these parameters, and their variance-covariance matrix, depends on the numerical coding used for the covariates. If an arbitrary constant is added to the covariate values, then the variance matrix is changed.

The `ftrend()` function takes the fitted model and works out an optimal constant to add to the covariate values so that the covariance matrix is approximately diagonal. The parameter estimates can then be treated as approximately independent, thus simplifying their presentation. This is particularly useful for graphical display of dose-response relationships (hence the name).

Greenland et al. (1999) originally suggested centring the covariates so that their weighted mean, using the fitted weights from the model, is zero. This heuristic criterion is improved upon by `ftrend()` which uses the same minimum information divergence criterion as used by Plummer (2003) for floating variance calculations. `ftrend()` calls `nlm()` to do the minimization and will pass optional arguments to control it.

Value

A list with the following components

`coef`	coefficients for model with adjusted covariates.
`vcov`	Variance-covariance matrix of adjusted coefficients.

Note

The "floating trend" method is an alternative to the "floating absolute risk" method, which is implemented in the function `float()`.

Author(s)

Martyn Plummer

References

Greenland S, Michels KB, Robins JM, Poole C and Willet WC (1999) Presenting statistical uncertainty in trends and dose-response relations, *American Journal of Epidemiology*, **149**, 1077-1086.

See Also

float

gmortDK *Population mortality rates for Denmark in 5-years age groups.*

Description

The gmortDK data frame has 418 rows and 21 columns.

Format

This data frame contains the following columns:

agr: Age group, 0:0–4, 5:5–9,..., 90:90+.
per: Calendar period, 38: 1938–42, 43: 1943–47, ..., 88:1988-92.
sex: Sex, 1: male, 2: female.
risk: Number of person-years in the Danish population.
dt: Number of deaths.
rt: Overall mortality rate in cases per 1000 person-years, i.e. rt=1000*dt/risk
Cause-specific mortality rates in cases per 1000 person-years:
r1: Infections
r2: Cancer.
r3: Tumors, benign, unspecific nature.
r4: Endocrine, metabolic.
r5: Blood.
r6: Nervous system, psychiatric.
r7: Cerebrovascular.
r8: Cardiac.
r9: Respiratory diseases, excl. cancer.
r10: Liver, excl. cancer.
r11: Digestive, other.
r12: Genitourinary.
r13: Ill-defined symptoms.
r14: All other, natural.
r15: Violent.

Source

Statistics Denmark, National board of health provided original data. Michael Andersson grouped the causes of death.

See Also

thoro, mortDK

Examples

```
data(gmortDK)
```

hivDK *hivDK: seroconversion in a cohort of Danish men*

Description

Data from a survey of HIV-positivity of a cohort of Danish men followed by regular tests from 1983 to 1989.

Usage

```
data(hivDK)
```

Format

A data frame with 297 observations on the following 7 variables.

`id` ID of the person

`entry` Date of entry to the study. Date variable.

`well` Date last seen seronegative. Date variable.

`ill` Date first seen seroconverted. Date variable.

`bth` Year of birth minus 1950.

`pyr` Annual number of sexual partners.

`us` Indicator of wheter the person has visited the USA.

Source

Mads Melbye, Statens Seruminstitut.

References

Becker N.G. and Melbye M.: Use of a log-linear model to compute the empirical survival curve from interval-censored data, with application to data on tests for HIV-positivity, Australian Journal of Statistics, 33, 125–133, 1990.

Melbye M., Biggar R.J., Ebbesen P., Sarngadharan M.G., Weiss S.H., Gallo R.C. and Blattner W.A.: Seroepidemiology of HTLV-III antibody in Danish homosexual men: prevalence, transmission and disease outcome. British Medical Journal, 289, 573–575, 1984.

Examples

```
data(hivDK)
str(hivDK)
```

Icens *Fits a regression model to interval censored data.*

Description

The models fitted assumes a piecewise constant baseline rate in intervals specified by the argument `breaks`, and for the covariates either a multiplicative relative risk function (default) or an additive excess risk function.

Usage

```
Icens( first.well, last.well, first.ill,
       formula, model.type=c("MRR","AER"), breaks,
       boot=FALSE, alpha=0.05, keep.sample=FALSE,
       data )
```

Arguments

`first.well` Time of entry to the study, i.e. the time first seen without event. Numerical vector.

`last.well` Time last seen without event. Numerical vector.

`first.ill` Time first seen with event. Numerical vector.

`formula` Model formula for the log relative risk.

`model.type` Which model should be fitted.

`breaks` Breakpoints between intervals in which the underlying timescale is assumed constant. Any observation outside the range of `breaks` is discarded.

`boot` Should bootstrap be performed to produce confidence intervals for parameters. If a number is given this will be the number of bootsrap samples. The default is 1000.

`alpha` 1 minus the confidence level.

`keep.sample` Should the bootstrap sample of the parameter values be returned?

`data` Data frame in which the times and formula are interpreted.

Details

The model is fitted by calling either `fit.mult` or `fit.add`.

Value

An object of class `"Icens"`: a list with three components:

`rates` A glm object from a binomial model with log-link, estimating the baseline rates, and the excess risk if `"AER"` is specfied.

`cov` A glm object from a binomial model with complementary log-log link, estimating the log-rate-ratios. Only if `"MRR"` is specfied.

`niter` Nuber of iterations, a scalar

`boot.ci` If `boot=TRUE`, a 3-column matrix with estimates and 1-`alpha` confidence intervals for the parameters in the model.

`sample` A matrix of the parameterestimates from the bootstrapping. Rows refer to parameters, columns to bootstrap samples.

Author(s)

Martyn Plummer, ⟨plummer@iarc.fr⟩, Bendix Carstensen, ⟨bxc@steno.dk⟩

References

B Carstensen: Regression models for interval censored survival data: application to HIV infection in Danish homosexual men. Statistics in Medicine, 15(20):2177-2189, 1996.

CP Farrington: Interval censored survival data: a generalized linear modelling approach. Statistics in Medicine, 15(3):283-292, 1996.

See Also

`fit.add` `fit.mult`

Examples

```
data( hivDK )
# Convert the dates to fractional years so that rates are
# expressed in cases per year
for( i in 2:4 ) hivDK[,i] <- cal.yr( hivDK[,i] )

m.RR <- Icens( entry, well, ill,
               model="MRR", formula=˜pyr+us, breaks=seq(1980,1990,5),
               data=hivDK)
# Currently the MRR model returns a list with 2 glm objects.
round( ci.lin( m.RR$rates ), 4 )
round( ci.lin( m.RR$cov, Exp=TRUE ), 4 )
# There is actually a print method:
print( m.RR )

m.ER <- Icens( entry, well, ill,
               model="AER", formula=˜pyr+us, breaks=seq(1980,1990,5),
               data=hivDK)
# There is actually a print method:
print( m.ER )
```

lep *An unmatched case-control study of leprosy incidence*

Description

The `lep` data frame has 1370 rows and 7 columns. This was an unmatched case-control study in which incident cases of leprosy in a region of N. Malawi were compared with population controls.

Format

This data frame contains the following columns:

`id`:	subject identifier: a numeric vector
`d`:	case/control status: a numeric vector (1=case, 0=control)
`age`:	a factor with levels `5-9` `10-14` `15-19` `20-24` `25-29` `30-44` `45+`
`sex`:	a factor with levels `male`, `female`
`bcg`:	presence of vaccine scar, a factor with levels `no` `yes`
`school`:	schooling, a factor with levels `none` `1-5yrs` `6-8yrs` `sec/tert`
`house`:	housing, a factor with levels `brick` `sunbrick` `wattle` `temp`

Source

The study is described in more detail in Clayton and Hills, Statistical Models in Epidemiology, Oxford University Press, Oxford:1993.

Examples

```
data(lep)
```

Lexis.diagram *Plot a Lexis diagram*

Description

Draws a Lexis diagram, optionally with life lines from a cohort, and with lifelines of a cohort if supplied. Intended for presentation purposes.

Usage

```
Lexis.diagram( age = c( 0, 60),
              alab = "Age",
              date = c( 1940, 2000 ),
              dlab = "Calendar time",
               int = 5,
           lab.int = 2*int,
          col.life = "black",
          lwd.life = 2,
          age.grid = TRUE,
         date.grid = TRUE,
          coh.grid = FALSE,
          col.grid = gray(0.7),
          lwd.grid = 1,
               las = 1,
        entry.date = NA,
         entry.age = NA,
         exit.date = NA,
          exit.age = NA,
         risk.time = NA,
        birth.date = NA,
              fail = NA,
          cex.fail = 1.1,
          pch.fail = c(NA,16),
          col.fail = rep( col.life, 2 ),
              data = NULL, ... )
```

Arguments

`age`	Numerical vector of length 2, giving the age-range for the diagram
`alab`	Label on the age-axis.
`date`	Numerical vector of length 2, giving the calendar time-range for the diagram
`dlab`	label on the calendar time axis.
`int`	The interval between grid lines in the diagram. If a vector of length two is given, the first value will be used for spacing of age-grid and the second for spacing of the date grid.
`lab.int`	The interval between labelling of the grids.
`col.life`	Colour of the life lines.
`lwd.life`	Width of the life lines.
`age.grid`	Should grid lines be drawn for age?
`date.grid`	Should grid lines be drawn for date?
`coh.grid`	Should grid lines be drawn for birth cohorts (diagonals)?
`col.grid`	Colour of the grid lines.
`lwd.grid`	Width of the grid lines.
`las`	How are the axis labels plotted?
`entry.date, entry.age, exit.date, exit.age, risk.time, birth.date`	Numerical vectors defining lifelines to be plotted in the diagram. At least three must be given to produce lines. Not all subsets of three will suffice, the given subset has to define life lines. If insufficient data is given, no life lines are produced.
`fail`	Logical of event status at exit for the persons whose life lines are plotted.
`pch.fail`	Symbols at the end of the life lines for censorings (`fail==0`) and failures (`fail != 0`).
`cex.fail`	Expansion of the status marks at the end of life lines.
`col.fail`	Character vector of length 2 giving the colour of the failure marks for censorings and failures respectively.
`data`	Dataframe in which to interpret the arguments.
`...`	Arguments to be passed on to the initial call to plot.

Details

The default unit for supplied variables are (calendar) years. If any of the variables `entry.date`, `exit.date` or `birth.date` are of class ”`Date`” or if any of the variables `entry.age`, `exit.age` or `risk.time` are of class ”`difftime`”, they will be converted to calendar years, and plotted correctly in the diagram. The returned dataframe will then have colums of classes ”`Date`” and ”`difftime`”.

Value

If sufficient information on lifelines is given, a data frame with one row per person and columns with entry ages and dates, birth date, risk time and status filled in.

Side effect: a plot of a Lexis diagram is produced with the life lines in it is produced. This will be the main reason for using the function. If the primary aim is to illustrate follow-up of a cohort, then it is better to represent the follow-up in a `Lexis` object, and use the generic `plot.Lexis` function.

Author(s)

Bendix Carstensen, `http://www.biostat.ku.dk/~bxc`

See Also

`Life.lines`, `Lexis.lines`

Examples

```
Lexis.diagram( entry.age = c(3,30,45),
               risk.time = c(25,5,14),
              birth.date = c(1970,1931,1925.7),
                    fail = c(TRUE,TRUE,FALSE) )
LL <- Lexis.diagram( entry.age = sample( 0:50, 17, replace=TRUE ),
                     risk.time = sample( 5:40, 17, r=TRUE),
                    birth.date = sample( 1910:1980, 17, r=TRUE ),
                          fail = sample( 0:1, 17, r=TRUE ),
                      cex.fail = 1.1,
                      lwd.life = 2 )
# Identify the persons' entry and exits
text( LL$exit.date, LL$exit.age, paste(1:nrow(LL)), col="red", font=2, adj=c(0,1) )
text( LL$entry.date, LL$entry.age, paste(1:nrow(LL)), col="blue", font=2, adj=c(1,0) )
data( nickel )
attach( nickel )
LL <- Lexis.diagram( age=c(10,100), date=c(1900,1990),
                     entry.age=age1st, exit.age=ageout, birth.date=dob,
                     fail=(icd %in% c(162,163)), lwd.life=1,
                     cex.fail=0.8, col.fail=c("green","red") )
abline( v=1934, col="blue" )
nickel[1:10,]
LL[1:10,]
```

`Lexis.lines` *Draw life lines in a Lexis diagram.*

Description

Add life lines to a Lexis diagram.

Usage

```
Lexis.lines( entry.date = NA,
              exit.date = NA,
             birth.date = NA,
              entry.age = NA,
               exit.age = NA,
              risk.time = NA,
               col.life = "black",
               lwd.life = 2,
                   fail = NA,
               cex.fail = 1,
               pch.fail = c(NA, 16),
               col.fail = col.life,
                   data = NULL )
```

Arguments

`entry.date, entry.age, exit.date, exit.age, risk.time, birth.date`
: Numerical vectors defining lifelines to be plotted in the diagram. At least three must be given to produce lines. Not all subsets of three will suffice, the given subset has to define life lines. If insufficient data is given, no life lines are produced.

`col.life` Colour of the life lines.

`lwd.life` Width of the life lines.

`fail` Logical of event status at exit for the persons whose life lines are plotted.

`cex.fail` The size of the status marks at the end of life lines.

`pch.fail` The status marks at the end of the life lines.

`col.fail` Colour of the marks for censorings and failures respectively.

`data` Data frame in which to interpret values.

Value

If sufficient information on lifelines is given, a data frame with one row per person and columns with entry ages and dates, birth date, risk time and status filled in.

Side effect: Life lines are added to an existing Lexis diagram. Lexis.lines adds life lines to an existing plot.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://www.biostat.ku.dk/~bxc

See Also

Lexis.diagram, Life.lines

Examples

```
Lexis.diagram( entry.age = c(3,30,45),
               risk.time = c(25,5,14),
              birth.date = c(1970,1931,1925.7),
                    fail = c(TRUE,TRUE,FALSE) )
Lexis.lines( entry.age = sample( 0:50, 100, replace=TRUE ),
             risk.time = sample( 5:40, 100, r=TRUE),
            birth.date = sample( 1910:1980, 100, r=TRUE ),
                  fail = sample(0:1,100,r=TRUE),
              cex.fail = 0.5,
              lwd.life = 1 )
```

`Lexis` *Create a Lexis object*

Description

Create an object of class `Lexis` to represent follow-up on multiple time scales.

Usage

```
Lexis(entry, exit, duration, entry.status = 0, exit.status = 0, id, data,
      merge=TRUE, states )
```

Arguments

`entry`	a named list of entry times. Each element of the list is a numeric variable representing the entry time on the named time scale. All time scales must have the same units (e.g. years).
`exit`	a named list of exit times.
`duration`	a numeric vector giving the duration of follow-up.
`entry.status`	a vector or a factor giving the status at entry
`exit.status`	a vector or factor giving status at exit. Any change in status during follow-up is assumed to take place exactly at the exit time.
`id`	a vector giving a unique identity value for each row of the Lexis object.
`data`	an optional data frame, list, or environment containing the variables. If not found in `data`, the variables are taken from the environment from which `Lexis` was called.
`merge`	a logical flag. If `TRUE` then the `data` argument will be coerced to a data frame and then merged with the resulting `Lexis` object.
`states`	A vector of labels for the states. If given, the state variables `lex.Cst` and `lex.Xst` are returned as factors with identical levels attributes.

Details

The analysis of long-term population-based follow-up studies typically requires multiple time scales to be taken into account, such as age, calender time, or time since an event. A `Lexis` object is a data frame with additional attributes that allows these multiple time dimensions of follow-up to be managed.

Separate variables for current end exit state allows representation of multistate data.

Lexis objects are named after the German demographer Wilhelm Lexis (1837-1914), who is credited with the invention of the ”Lexis diagram” for representing population dynamics simultaneously by age, period and cohort.

The `Lexis` function creates a minimal `Lexis` object with only those variables required to define the follow-up history in each row. Additional variables can be merged into the `Lexis` object using the `merge` method for `Lexis` objects. This is the default.

There are also `merge`, `subset` and `transform` methods for `Lexis` objects. They work as the corresponding methods for data-frames but ensures that the result is a `Lexis` object.

Value

An object of class `Lexis`. This is represented as a data frame with a column for each time scale, and additional columns with the following names:

`lex.id`	Identification of the inidvidual
`lex.dur`	Duration of follow-up
`lex.Cst`	Entry status (Current state), i.e. the state in which the follow up takes place.
`lex.Xst`	Exit status (eXit state), i.e. that state taken up after `dur` in `lex.Cst`.

If `merge=TRUE` then the `Lexis` object will also contain all variables from the `data` argument.

Note

Only two of the three arguments `entry`, `exit` and `duration` need to be given. If the third parameter is missing, it is imputed. If duration is given, it must be the same on all time scales.

`entry`, `exit` must be numeric, using `Date` variables will cause some of the utilites to crash. Transformation by `cal.yr` is recommended.

If only either `exit` or `duration` are supplied it is assumed that `entry` is 0. This is only meaningful (and therefore checked) if there is only one timescale.

If any of `entry.status` or `exit.status` are of mode character, they will both be converted to factors.

If `entry.status` is not given, then its class is automatically set to that of `exit.status`. If `exit.status` is factor, the value of `entry.status` is set to the first level. This may be highly undesirable, and therefore noted. For example, if `exit.status` is character the first level will be the first in the alphabetical ordering; slightly unfortunate if values are c("Well","Diseased"). If exit.status is logical, the value of entry.status set to FALSE.

If entry.status or exit.status are factors or character, the corresponding state variables in the returned Lexis object, lex.Cst and lex.Xst will be (unordered) factors with identical levels, namely the union of the levels of entry.status and exit.status.

Author(s)

Martyn Plummer

See Also

plot.Lexis, splitLexis, cutLexis, merge.Lexis, subset.Lexis, transform.Lexis, summary.Lexis, timeScales, timeBand, entry, exit, dur

Examples

```
# A small bogus cohort
xcoh <- structure( list( id = c("A", "B", "C"),
                      birth = c("14/07/1952", "01/04/1954", "10/06/1987"),
                      entry = c("04/08/1965", "08/09/1972", "23/12/1991"),
                       exit = c("27/06/1997", "23/05/1995", "24/07/1998"),
                       fail = c(1, 0, 1) ),
                     .Names = c("id", "birth", "entry", "exit", "fail"),
                  row.names = c("1", "2", "3"),
                      class = "data.frame" )

# Convert the character dates into numerical variables (fractional years)
xcoh$bt <- cal.yr( xcoh$birth, format="%d/%m/%Y" )
xcoh$en <- cal.yr( xcoh$entry, format="%d/%m/%Y" )
xcoh$ex <- cal.yr( xcoh$exit , format="%d/%m/%Y" )

# See how it looks
xcoh

# Define as Lexis object with timescales calendar time and age
Lcoh <- Lexis( entry = list( per=en ),
                exit = list( per=ex, age=ex-bt ),
         exit.status = fail,
                data = xcoh )
Lcoh

# Using character states may have undesired effects:
xcoh$Fail <- c("Dead","Well","Dead")
Lexis( entry = list( per=en ),
        exit = list( per=ex, age=ex-bt ),
 exit.status = Fail,
        data = xcoh )

# unless you order the levels correctly
( xcoh$Fail <- factor( xcoh$Fail, levels=c("Well","Dead") ) )
Lexis( entry = list( per=en ),
        exit = list( per=ex, age=ex-bt ),
 exit.status = Fail,
        data = xcoh )
```

Life.lines *Compute dates/ages for life lines in a Lexis diagram*

Description

Fills out the missing information for follow up of persons in a Lexis diagram if sufficient information is given.

Usage

```
Life.lines( entry.date = NA,
             exit.date = NA,
            birth.date = NA,
             entry.age = NA,
              exit.age = NA,
             risk.time = NA )
```

Arguments

`entry.date, exit.date,birth.date, entry.age, exit.age, risk.time`
: Vectors defining lifelines to be plotted in the diagram. At least three must be given to produce a result. Not all subsets of three will suffice, the given subset has to define life lines. If insufficient data is given, nothing is returned and a warning is given.

Value

Data frame with variables `entry.date`, `entry.age`, `exit.date`, `exit.age`, `risk.time`, `birth.date`, with all entries computed for each person. If any of `entry.date`, `exit.date` or `birth.date` are of class `Date` or if any of `entry.age`, `exit.age` or `risk.time` are of class `difftime` the date variables will be of class `Date` and the other three of class `difftime`.

See Also

`Lexis.diagram`, `Lexis.lines`

Examples

```
( Life.lines( entry.age = c(3,30,45),
              risk.time = c(25,5,14),
             birth.date = c(1970,1931,1925.7) ) )

# Draw a Lexis diagram
Lexis.diagram()

# Compute entry and exit age and date.
( LL <-  Life.lines( entry.age = c(3,30,45),
                     risk.time = c(25,5,14),
                    birth.date = c(1970,1931,1925.7) ) )
segments( LL[,1], LL[,2], LL[,3], LL[,4] ) # Plot the life lines.

# Compute entry and exit age and date, supplying a date variable
bd <- ( c(1970,1931,1925.7) - 1970 ) * 365.25
class( bd ) <- "Date"
( Life.lines( entry.age = c(3,30,45),
              risk.time = c(25,5,14),
             birth.date = bd ) )
```

lungDK *Male lung cancer incidence in Denmark*

Description

Male lung cancer cases and population riks time in Denmark, for the period 1943–1992 in ages 40–89.

Usage

```
data(lungDK)
```

Format

A data frame with 220 observations on the following 9 variables.

A5: Left end point of the age interval, a numeric vector.
P5: Left enpoint of the period interval, a numeric vector.
C5: Left enpoint of the birth cohort interval, a numeric vector.
up: Indicator of upper trianges of each age by period rectangle in the Lexis diagram. (`up=(P5-A5-C5)/5`).
Ax: The mean age of diagnois (at risk) in the triangle.
Px: The mean date of diagnosis (at risk) in the triangle.
Cx: The mean date of birth in the triangle, a numeric vector.
D: Number of diagnosed cases of male lung cancer.
Y: Risk time in the male population, person-years.

Details

Cases and person-years are tabulated by age and date of diagnosis (period) as well as date of birth (cohort) in 5-year classes. Each observation in the dataframe correponds to a triangle in a Lexis diagram. Triangles are classified by age and date of diagnosis, period of diagnosis and date of birth, all in 5-year groupings.

Source

The Danish Cancer Registry and Statistics Denmark.

References

For a more thorough exposition of statistical inference in the Lexis diagram, see:
http://staff.pubhealth.ku.dk/~bxc/APC/notes.pdf

Examples

```
data( lungDK )
# Draw a Lexis diagram and show the number of cases in it.
attach( lungDK )
Lexis.diagram( age=c(40,90), date=c(1943,1993), coh.grid=TRUE )
text( Px, Ax, paste( D ), cex=0.7 )
```

merge.data.frame *Merge data frame with a Lexis object*

Description

Merge two data frames, or a data frame with a `Lexis` object.

Usage

```
## S3 method for class 'data.frame':
merge(x, y, ...)
```

Arguments

`x, y`	data frames, or objects to be coerced into one
`...`	optional arguments for the merge method

Details

This version of `merge.default` masks the one in the `base`. It ensures that, if either `x` or `y` is a `Lexis` object, then `merge.Lexis` is called.

Value

A merged `Lexis` object or data frame.

Author(s)

Martyn Plummer

See Also

`Lexis`

`merge.Lexis` *Merge a Lexis object with a data frame*

Description

Merge additional variables from a data frame into a Lexis object.

Usage

```
## S3 method for class 'Lexis':
merge(x, y, id, by, ...)
```

Arguments

`x`	an object of class `Lexis`
`y`	a data frame
`id`	the name of the variable in `y` to use for matching against the variable `lex.id` in `x`.
`by`	if matching is not done by id, a vector of variable names common to both `x` and `y`
`...`	optional arguments to be passed to `merge.data.frame`

Details

A `Lexis` object can be considered as an augmented data frame in which some variables are time-dependent variables representing follow-up. The `Lexis` function produces a minimal object containing only these time-dependent variables. Additional variables may be added to a `Lexis` object using the `merge` method.

Value

A `Lexis` object with additional columns taken from the merged data frame.

Note

The variable given as the `by.y` argument must not contain any duplicate values in the data frame `y`.

Author(s)

Martyn Plummer

See Also

merge.data.frame, subset.Lexis

---

mh *Mantel-Haenszel analyses of cohort and case-control studies*

Description

This function carries out Mantel-Haenszel comparisons in tabulated data derived from both cohort and case-control studies.

Usage

```
mh(cases, denom, compare=1, levels=c(1, 2), by=NULL,
     cohort=!is.integer(denom), confidence=0.9)
```

Arguments

`cases`	the table of case frequencies (a multiway array).
`denom`	the denominator table. For cohort studies this should be a table of person-years observation, while for case-control studies it should be a table of control frequencies.
`compare`	the dimension of the table which defines the comparison groups (can be referred to either by number or by name). The default is the first dimension of the table.
`levels`	a vector identifying (either by number or by name) the two groups to be compared. The default is the first two levels of the selected dimension.
`by`	the dimensions not to be collapsed in the Mantel-Haenszel computations. Thus, this argument defines the structure of the resulting tables of estimates and tests.
`cohort`	an indicator whether the data derive from a cohort or a case-control study. If the denominator table is stored as an integer, a case-control study is assumed.
`confidence`	the approximate coverage probability for the confidence intervals to be computed.

Details

Multiway tables of data are accepted and any two levels of any dimension can be chosen as defining the comparison groups. The rate (odds) ratio estimates and the associated significance tests may be collapsed over all the remaining dimensions of the table, or over selected dimensions only, so that tables of estimates and tests are computed.

Value

A list giving tables of rate (odds) ratio estimates, their standard errors (on a log scale), lower and upper confidence limits, chi-squared tests (1 degree of freedom) and the corresponding p-values. The result list also includes numerator and denominator of the Mantel-Haenszel estimates (q, r), and score test statistics and score variance (u, v).

Side Effects

None

References

Clayton, D. and Hills, M. : Statistical Models in Epidemiology, Oxford University Press (1993).

See Also

Lexis

Examples

```
# If d and y are 3-way tables of cases and person-years
# observation formed by tabulation by two confounders
# (named "C1" and "C2") an exposure of interest ("E"),
# the following command will calculate an overall
# Mantel-Haenszel comparison of the first two exposure
# groups.
#
# Generate some bogus data
dnam <- list( E=c("low","medium","high"), C1=letters[1:2], C2=LETTERS[1:4] )
d <- array( sample( 2:80, 24),
            dimnames=dnam, dim=sapply( dnam, length ) )
y <- array( abs( rnorm( 24, 227, 50 ) ),
            dimnames=dnam, dim=sapply( dnam, length ) )
mh(d, y, compare="E")
#
# Or, if exposure levels named "low" and "high" are to be
# compared and these are not the first two levels of E :
#
mh(d, y, compare="E", levels=c("low", "high"))
#
# If we wish to carry out an analysis which controls for C1,
# but examines the results at each level of C2:
#
mh(d, y, compare="E", by="C2")
#
# It is also possible to look at rate ratios for every
# combination of C1 and C2 :
#
mh(d, y, compare="E", by=c("C1", "C2"))
#
# If dimensions and levels of the table are unnamed, they must
# be referred to by number.
#
```

mortDK *Population mortality rates for Denmark in 1-year age-classes.*

Description

The mortDK data frame has 1820 rows and 21 columns.

Format

This data frame contains the following columns:

age:	Age class, 0–89, 90:90+.
per:	Calendar period, 38: 1938–42, 43: 1943–47, ..., 88:1988-92.
sex:	Sex, 1: male, 2: female.
risk:	Number of person-years in the Danish population.
dt:	Number of deaths.
rt:	Overall mortality rate in cases per 1000 person-years, i.e. rt=1000*dt/risk
	Cause-specific mortality rates in cases per 1000 person-years:
r1:	Infections
r2:	Cancer.
r3:	Tumors, benign, unspecific nature.
r4:	Endocrine, metabolic.
r5:	Blood.
r6:	Nervous system, psychiatric.

`r7`:	Cerebrovascular.
`r8`:	Cardiac.
`r9`:	Respiratory diseases, excl. cancer.
`r10`:	Liver, excl. cancer.
`r11`:	Digestive, other.
`r12`:	Genitourinary.
`r13`:	Ill-defined symptoms.
`r14`:	All other, natural.
`r15`:	Violent.

Source

Statistics Denmark, National board of health provided original data. Michael Andersson grouped the causes of death.

See Also

`thoro`, `gmortDK`

Examples

```
data(mortDK)
```

`ncut` *Function to group a variable in intervals.*

Description

Cuts a continuous variable in intervals. As opposed to `cut` which returns a factor, `ncut` returns a numeric variable.

Usage

```
ncut(x, breaks, type="left" )
```

Arguments

`x`	A numerical vector.
`breaks`	Vector of breakpoints. `NA` will results for values below `min(x)` if `type="left"`, for values above `max(x)` if `type="right"` and for values outside `range(x)` if `type="mid"`
`type`	Character: one of `c("left","right","mid")`, indicating whether the left, right or midpoint of the intervals defined in breaks is returned.

Details

The function uses the base function `findInterval`.

Value

A numerical vector of the same length as `x`.

Author(s)

Bendix Carstensen, Steno Diabetes Center, ⟨bxc@steno.dk⟩, `http://www.biostat.ku.dk/~bxc/`, with essential input from Martyn Plummer, IARC.

See Also

`cut`, `findInterval`

Examples

```
br <- c(-2,0,1,2.5)
x <- c( rnorm( 10 ), br, -3, 3 )
cbind( x, l=ncut( x, breaks=br, type="l" ),
          m=ncut( x, breaks=br, type="m" ),
          r=ncut( x, breaks=br, type="r" ) )[order(x),]
x <- rnorm( 200 )
plot( x, ncut( x, breaks=br, type="l" ), pch=16, col="blue", ylim=range(x) )
abline( 0, 1 )
abline( v=br )
points( x, ncut( x, breaks=br, type="r" ), pch=16, col="red" )
points( x, ncut( x, breaks=br, type="m" ), pch=16, col="green" )
```

nice *Nice breakpoints*

Description

The function calls pretty for linear scale. For a log-scale nice are computed using a set of specified number in a decade.

Usage

```
nice(x, log = F, lpos = c(1, 2, 5), ...)
```

Arguments

x	Numerical vector to
log	Logical. Is the scale logartimic?
lpos	Numeric. Numbers between 1 and 10 giving the desired breakpoints in this interval.
...	Arguments passed on to pretty if log=FALSE

Value

A vector of breakpoints.

Author(s)

Bendix Carstensen, ⟨bxc@steno.dk⟩, http://www.biostat.ku.dk/~bxc

See Also

pretty

Examples

```
nice( exp( rnorm( 100 ) ), log=TRUE )
```

nickel *A Cohort of Nickel Smelters in South Wales*

Description

The nickel data frame has 679 rows and 7 columns. The data concern a cohort of nickel smelting workers in South Wales and are taken from Breslow and Day, Volume 2. For comparison purposes, England and Wales mortality rates (per 1,000,000 per annum) from lung cancer (ICDs 162 and 163), nasal cancer (ICD 160), and all causes, by age group and calendar period, are supplied in the dataset ewrates.

Format

This data frame contains the following columns:

id:	Subject identifier (numeric)
icd:	ICD cause of death if dead, 0 otherwise (numeric)
exposure:	Exposure index for workplace (numeric)
dob:	Date of birth (numeric)
age1st:	Age at first exposure (numeric)
agein:	Age at start of follow-up (numeric)
ageout:	Age at end of follow-up (numeric)

Source

Breslow NE, and Day N, Statistical Methods in Cancer Research. Volume II: The Design and Analysis of Cohort Studies. IARC Scientific Publications, IARC:Lyon, 1987.

Examples

```
data(nickel)
str(nickel)
```

pctab *Create percentages in a table*

Description

Computes percentages and a margin of totals along a given margin of a table.

Usage

```
pctab(TT, margin = length(dim(TT)), dec=1)
```

Arguments

TT	A table or array object
margin	Which margin should be the the total?
dec	How many decimals should be printed?

Value

A table, where all dimensions except the one specified margin has two extra levels named ”All” (where all entries are 100) and ”N”. The function prints the table with dec decimals.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://www.biostat.ku.dk/~bxc.

See Also

`addmargins`

Examples

```
Aye <- sample( c("Yes","Si","Oui"), 177, replace=TRUE )
Bee <- sample( c("Hum","Buzz"), 177, replace=TRUE )
Sea <- sample( c("White","Black","Red","Dead"), 177, replace=TRUE )
A <- table( Aye, Bee, Sea )
A
ftable( pctab( A ) )
ftable( pctab( addmargins( A, 1 ), 3 ) )
round( ftable( pctab( addmargins( A, 1 ), 3 ), row.vars=3 ), 1)
```

plot.Lexis *Lexis diagrams*

Description

The follow-up histories represented by a Lexis object can be plotted using one or two dimensions. The two dimensional plot is a Lexis diagram showing follow-up time simultaneously on two time scales.

Usage

```
## S3 method for class 'Lexis':
plot(x, time.scale = NULL, breaks="lightgray", ...)
## S3 method for class 'Lexis':
points(x, time.scale = NULL, ...)
## S3 method for class 'Lexis':
lines(x, time.scale = NULL, ...)
```

Arguments

`x` An object of class `Lexis`

`time.scale` A vector of length 1 or 2 giving the time scales to be plotted either by name or numerical order

`breaks` a string giving the colour of grid lines to be drawn when plotting a split Lexis object. Grid lines can be suppressed by supplying the value `NULL` to the `breaks` argument

`...` Further graphical parameters to be passed to the plotting methods.

Grids can be drawn (behind the life lines) using the following parameters in `plot`:

- `grid` If logical, a background grid is set up using the axis ticks. If a list, the first component is used as positions for the vertical lines and the last as positions for the horizontal. If a nunerical vector, grids on both axes are set up using the distance between the numbers.
- `col.grid="lightgray"` Color of the background grid.
- `lty.grid=2` Line type for the grid.
- `coh.grid=FALSE` Should a 45 degree grid be plotted?

Details

The plot method for `Lexis` objects traces “life lines” from the start to the end of follow-up. The `points` method plots points at the end of the life lines.

If `time.scale` is of length 1, the life lines are drawn horizontally, with the time scale on the X axis and the id value on the Y axis. If `time.scale` is of length 2, a Lexis diagram is produced, with diagonal life lines plotted against both time scales simultaneously.

If `lex` has been split along one of the time axes by a call to `splitLexis`, then vertical or horizontal grid lines are plotted (on top of the life lines) at the break points.

Author(s)

Martyn Plummer

See Also

Lexis, splitLexis

Examples

```
# A small bogus cohort
xcoh <- structure( list( id = c("A", "B", "C"),
                      birth = c("14/07/1952", "01/04/1954", "10/06/1987"),
                      entry = c("04/08/1965", "08/09/1972", "23/12/1991"),
                       exit = c("27/06/1997", "23/05/1995", "24/07/1998"),
                       fail = c(1, 0, 1) ),
                     .Names = c("id", "birth", "entry", "exit", "fail"),
                  row.names = c("1", "2", "3"),
                      class = "data.frame" )

# Convert the character dates into numerical variables (fractional years)
xcoh$bt <- cal.yr( xcoh$birth, format="%d/%m/%Y" )
xcoh$en <- cal.yr( xcoh$entry, format="%d/%m/%Y" )
xcoh$ex <- cal.yr( xcoh$exit , format="%d/%m/%Y" )

# See how it looks
xcoh

# Define as Lexis object with timescales calendar time and age
Lcoh <- Lexis( entry = list( per=en ),
                exit = list( per=ex, age=ex-bt ),
         exit.status = fail,
                data = xcoh )

# Default plot of follow-up
plot( Lcoh )
# With a grid and deaths as endpoints
plot( Lcoh, grid=0:10*10, col="black" )
points( Lcoh, pch=c(NA,16)[Lcoh$lex.Xst+1] )
# With a lot of bells and whistles:
plot( Lcoh, grid=0:20*5, col="black", xaxs="i", yaxs="i",
      xlim=c(1960,2010), ylim=c(0,50), lwd=3, las=1 )
points( Lcoh, pch=c(NA,16)[Lcoh$lex.Xst+1], col="red", cex=1.5 )
```

---

plotEst *Plot estimates with confidence limits*

Description

Plots parameter estimates with confidence intervals, annotated with parameter names. A dot is plotted at the estimate and a horizontal line extending from the lower to the upper limit is superimposed.

Usage

```
plotEst( ests,
            y = dim(ests)[1]:1,
          txt = rownames(ests),
       txtpos = y,
         ylim = range(y)-c(0.5,0),
         xlab = "",
         xtic = nice(ests[!is.na(ests)], log = xlog),
         xlim = range( xtic ),
         xlog = FALSE,
          pch = 16,
          cex = 1,
          lwd = 2,
          col = "black",
    col.lines = col,
   col.points = col,
         vref = NULL,
         grid = FALSE,
     col.grid = gray(0.9),
  restore.par = TRUE )

linesEst( ests, y = dim(ests)[1]:1, pch = 16, cex = 1, lwd = 2,
          col="black", col.lines=col, col.points=col )

pointsEst( ests, y = dim(ests)[1]:1, pch = 16, cex = 1, lwd = 2,
          col="black", col.lines=col, col.points=col )
```

Arguments

ests	Matrix with three columns: Estimate, lower limit, upper limit. If a model object is supplied, ci.lin is invoked for this object first.
y	Vertical position of the lines.
txt	Annotation of the estimates.
txtpos	Vertical position of the text. Defaults to y.
ylim	Extent of the vertical axis.
xlab	Annotation of the horizontal axis.
xtic	Location of tickmarks on the x-axis.
xlim	Extent of the x-axis.
xlog	Should the x-axis be logarithmic?
pch	What symbol should be used?
cex	Expansion of the symbol.
col	Colour of the points and lines.
col.lines	Colour of the lines.
col.points	Colour of the symbol.
lwd	Thickness of the lines.
vref	Where should vertical reference line(s) be drawn?
grid	If TRUE, vertical gridlines are drawn at the tickmarks. If a numerical vector is given vertical lines are drawn at grid.
col.grid	Colour of the vertical gridlines
restore.par	Should the graphics parameters be restored? If set to FALSE the coordinate system will still be available for additional plotting, and par("mai") will still have the very large value set in order to make room for the labelling of the estimates.

Details

plotEst make a news plot, whereas linesEst and pointsEst (identical functions) adds to an existing plot.

Value

NULL

Author(s)

Bendix Carstensen, ⟨bxc@steno.dk⟩, http://www.pubhealth.ku.dk/~bxc

See Also

ci.lin

Examples

```
# Bogus data and a linear model
f <- factor( sample( letters[1:5], 100, replace=TRUE ) )
x <- rnorm( 100 )
y <- 5 + 2 * as.integer( f ) + 0.8 * x + rnorm(100) * 2
m1 <- lm( y ~ f )

# Produce some confidence intervals for contrast to first level
( cf <- summary( m1 )$coef[2:5,1:2] %*% rbind( c(1,1,1), 1.96*(c(0,-1,1) ) ) )

# Plots with increasing amount of bells and whistles
par( mfcol=c(3,2), mar=c(3,3,2,1) )
plotEst( cf )
plotEst( cf, grid=TRUE )
plotEst( cf, grid=TRUE, cex=2, lwd=3 )
plotEst( cf, grid=TRUE, cex=2, col.points="red", col.lines="green" )
plotEst( cf, grid=TRUE, cex=2, col.points="red", col.lines="green",
          xlog=TRUE, xtic=c(1:8), xlim=c(0.8,6) )
rownames( cf )[1] <- "Contrast to fa:\n\n fb"
plotEst( cf, grid=TRUE, cex=2, col.points=rainbow(4), col.lines=rainbow(4), vref=1 )
```

plotevent *Plot Equivalence Classes*

Description

For interval censored data, segments of times between last.well and first.ill are plotted for each conversion in the data. It also plots the equivalence classes.

Usage

```
plotevent(last.well, first.ill, data)
```

Arguments

last.well	Time at which the individuals are last seen negative for the event
first.ill	Time at which the individuals are first seen positive for the event
data	Data with a transversal shape

Details

last.well and first.ill should be written as character in the function.

Value

Graph

Author(s)

Delphine Maucort-Boulch, Bendix Carstensen, Martyn Plummer

References

Carstensen B. Regression models for interval censored survival data: application to HIV infection in Danish homosexual men.Stat Med. 1996 Oct 30;15(20):2177-89.

Lindsey JC, Ryan LM. Tutorial in biostatistics methods for interval-censored data.Stat Med. 1998 Jan 30;17(2):219-38.

See Also

`Icens`

`projection.ip` *Projection of columns of a matrix.*

Description

Projects the columns of the matrix `M` on the space spanned by the columns of the matrix `X`, with respect to the inner product defined by `weight`: `<x|y>=sum(x*w*y)`.

Usage

```
projection.ip(X, M, orth = FALSE, weight = rep(1, nrow(X)))
```

Arguments

`X`	Matrix defining the space to project onto.
`M`	Matrix of columns to be projected. Must have the same number of rows as `X`.
`orth`	Should the projection be on the orthogonal complement to `span(X)`?
`weight`	Weights defining the inner product. Numerical vector of length `nrow(X)`.

Value

A matrix of full rank with columns in `span(X)`.

Author(s)

Bendix Carstensen, Steno Diabetes Center, `http://www.pubhealth.ku.dk/~bxc`, with help from Peter Dalgaard.

See Also

`detrend`

`rateplot` *Functions to plot rates from a table classified by age and calendar time (period)*

Description

Produces plots of rates versus age, connected within period or cohort (`Aplot`), rates versus period connected within age-groups (`Pplot`) and rates and rates versus date of birth cohort (`Cplot`). `rateplot` is a wrapper for these, allowing to produce the four classical displays with a single call.

Usage

```
rateplot( rates,
          which = c("ap","ac","pa","ca"),
            age = as.numeric( dimnames( rates )[[1]] ),
            per = as.numeric( dimnames( rates )[[2]] ),
           grid = FALSE,
         a.grid = grid,
         p.grid = grid,
         c.grid = grid,
          ygrid = grid,
       col.grid = gray( 0.9 ),
          a.lim = range( age, na.rm=TRUE ) + c(0,diff( range( age ) )/30),
          p.lim = range( per, na.rm=TRUE ) + c(0,diff( range( age ) )/30),
          c.lim = NULL,
           ylim = range( rates[rates>0], na.rm=TRUE ),
             at = NULL,
         labels = paste( at ),
          a.lab = "Age at diagnosis",
          p.lab = "Date of diagnosis",
          c.lab = "Date of birth",
           ylab = "Rates",
           type = "l",
            lwd = 2,
            lty = 1,
         log.ax = "y",
            las = 1,
            ann = FALSE,
          a.ann = ann,
          p.ann = ann,
          c.ann = ann,
          xannx = 1/20,
        cex.ann = 0.8,
         a.thin = seq( 1, length( age ), 2 ),
         p.thin = seq( 1, length( per ), 2 ),
         c.thin = seq( 2, length( age ) + length( per ) - 1, 2 ),
            col = par( "fg" ),
          a.col = col,
          p.col = col,
          c.col = col,
            ... )

Aplot( rates, age = as.numeric( dimnames( rates )[[1]] ),
       per = as.numeric( dimnames( rates )[[2]] ), grid = FALSE,
       a.grid = grid, ygrid = grid, col.grid = gray( 0.9 ),
       a.lim = range( age, na.rm=TRUE ), ylim = range( rates[rates>0], na.rm=TRUE ),
       at = NULL, labels = paste( at ), a.lab = names( dimnames( rates ) )[1],
       ylab = deparse( substitute( rates ) ), type = "l", lwd = 2, lty = 1,
       col = par( "fg" ), log.ax = "y", las = 1, c.col = col, p.col = col,
       c.ann = FALSE, p.ann = FALSE, xannx = 1/20, cex.ann = 0.8,
       c.thin = seq( 2, length( age ) + length( per ) - 1, 2 ),
       p.thin = seq( 1, length( per ), 2 ), p.lines = TRUE,
       c.lines = !p.lines, ... )

Pplot( rates, age = as.numeric( dimnames( rates )[[1]] ),
       per = as.numeric( dimnames( rates )[[2]] ), grid = FALSE,
       p.grid = grid, ygrid = grid, col.grid = gray( 0.9 ),
       p.lim = range( per, na.rm=TRUE ) + c(0,diff(range(per))/30),
       ylim = range( rates[rates>0], na.rm=TRUE ), p.lab = names( dimnames( rates ) )[2],
       ylab = deparse( substitute( rates ) ), at = NULL, labels = paste( at ),
       type = "l", lwd = 2, lty = 1, col = par( "fg" ), log.ax = "y",
       las = 1, ann = FALSE, cex.ann = 0.8, xannx = 1/20,
       a.thin = seq( 1, length( age ), 2 ), ... )

Cplot( rates, age = as.numeric( rownames( rates ) ),
       per = as.numeric( colnames( rates ) ), grid = FALSE,
       c.grid = grid, ygrid = grid, col.grid = gray( 0.9 ),
       c.lim = NULL, ylim = range( rates[rates>0], na.rm=TRUE ),
       at = NULL, labels = paste( at ), c.lab = names( dimnames( rates ) )[2],
       ylab = deparse( substitute( rates ) ), type = "l", lwd = 2, lty = 1,
       col = par( "fg" ), log.ax = "y", las = 1, xannx = 1/20, ann = FALSE,
       cex.ann = 0.8, a.thin = seq( 1, length( age ), 2 ), ...  )
```

Arguments

`rates`	A two-dimensional table (or array) with rates to be plotted. It is assumed that the first dimension is age and the second is period.
`which`	A character vector with elements from `c("ap","ac","apc","pa","ca")`, indication which plots should be produced. One plot per element is produced. The first letter indicates the x-axis of the plot, the remaining which groups should be connected, i.e. `"pa"` will plot rates versus period and connect age-classes, and `"apc"` will plot rates versus age, and connect both periods and cohorts.
`age`	Numerical vector giving the means of the age-classes. Defaults to the rownames of `rates` as numeric.
`per`	Numerical vector giving the means of the periods. Defaults to the columnnames of `rates` as numeric.
`grid`	Logical indicating whether a background grid should be drawn.
`a.grid`	Logical indicating whether a background grid on the age-axis should be drawn. If numerical it indicates the age-coordinates of the grid.
`p.grid`	do. for the period.
`c.grid`	do. for the cohort.
`ygrid`	do. for the rate-dimension.
`col.grid`	The colour of the grid.
`a.lim`	Range for the age-axis.
`p.lim`	Range for the period-axis.
`c.lim`	Range for the cohort-axis.
`ylim`	Range for the y-axis (rates).
`at`	Position of labels on the y-axis (rates).
`labels`	Labels to put on the y-axis (rates).
`a.lab`	Text on the age-axis. Defaults to ”Age”.
`p.lab`	Text on the period-axis. Defaults to ”Date of diagnosis”.
`c.lab`	Text on the cohort-axis. Defaults to ”Date of birth”.
`ylab`	Text on the rate-axis. Defaults to the name of the rate-table.
`type`	How should the curves be plotted. Defaults to `"l"`.
`lwd`	Width of the lines. Defaults to 2.
`lty`	Which type of lines should be used. Defaults to 1, a solid line.
`log.ax`	Character with letters from `"apcyr"`, indicating which axes should be logarithmic. `"y"` and `"r"` both refer to the rate scale. Defaults to `"y"`.
`las`	see `par`.
`ann`	Should the curves be annotated?
`a.ann`	Logical indicating whether age-curves should be annotated.

`p.ann`	do. for period-curves.
`c.ann`	do. for cohort-curves.
`xannx`	The fraction that the x-axis is expanded when curves are annotated.
`cex.ann`	Expansion factor for characters annotating curves.
`a.thin`	Vector of integers indicating which of the age-classes should be labelled.
`p.thin`	do. for the periods.
`c.thin`	do. for the cohorts.
`col`	Colours for the curves.
`a.col`	Colours for the age-curves.
`p.col`	do. for the period-curves.
`c.col`	do. for the cohort-curves.
`p.lines`	Should rates from the same period be connected?
`c.lines`	Should rates from the same cohort be connected?
`...`	Additional arguments pssed on to `matlines` when plotting the curves.

Details

Zero values of the rates are ignored. They are neiter in the plot nor in the calculation of the axis ranges.

Value

`NULL`. The function is used for its side-effect, the plot.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://www.pubhealth.ku.dk/~bxc/

See Also

`apc.frame`

Examples

```
data( blcaIT )
attach(blcaIT)

# Table of rates:
bl.rate <- tapply( D, list(age,period), sum ) /
           tapply( Y, list(age,period), sum )
bl.rate

# The four classical plots:
par( mfrow=c(2,2) )
rateplot( bl.rate*10^6 )

# The labels on the vertical axis could be nicer:
rateplot( bl.rate*10^6, at=10^(-1:3), labels=c(0.1,1,10,100,1000) )

# More bells an whistles
par( mfrow=c(1,3), mar=c(3,3,1,1), oma=c(0,3,0,0), mgp=c(3,1,0)/1.6 )
rateplot( bl.rate*10^6, ylab="", ann=TRUE, which=c("AC","PA","CA"),
                      at=10^(-1:3), labels=c(0.1,1,10,100,1000),
                      col=topo.colors(11), cex.ann=1.2 )
```

`Relevel` *Reorder and combine levels of a factor*

Description

The levels of a factor are re-ordered so that the levels specified by `ref` is first and the others are moved down. This is useful for `contr.treatment` contrasts which take the first level as the reference. Levels may also be combined.

Usage

```
Relevel(f, ref, first = TRUE, collapse="+" )
```

Arguments

`f`	An unordered factor
`ref`	The names or numbers of levels to be the first. If `ref` is a list, factor levels mentioned in each list element are combined. If the list is named the names are used as new factor levels.
`first`	Should the levels mentioned in ref come before those not?
`collapse`	String used when collapsing factor levels.

Value

An unordered factor.

Examples

```
ff <- factor( sample( letters[1:5], 100, replace=TRUE ) )
table( ff, Relevel( ff, list( AB=1:2, "Dee"=4, c(3,5) ) ) )
table( ff, rr=Relevel( ff, list( 5:4, Z=c("c","a") ), coll="-und-", first=FALSE ) )
```

`ROC` *Function to compute and draw ROC-curves.*

Description

Computes sensitivity, specificity and positive and negative predictive values for a test based on dichotomizing along the variable `test`, for prediction of `stat`. Alternatively a model formula may given, in which case the the linear predictor is the test variable and the response is taken as the true status variable. Plots curves of these and a ROC-curve.

Usage

```
ROC( test = NULL,
     stat = NULL,
     form = NULL,
     plot = c("sp", "ROC"),
       PS = is.null(test),
       PV = TRUE,
       MX = TRUE,
       MI = TRUE,
      AUC = TRUE,
     grid = seq(0,100,10),
 col.grid = gray( 0.9 ),
     cuts = NULL,
      lwd = 2,
     data = parent.frame(),
      ... )
```

Arguments

`test`	Numerical variable used for prediction.
`stat`	Logical variable of true status.
`form`	Formula used in a logistic regression. If this is given, `test` and `stat` are ignored. If not given then both `test` and `stat` must be supplied.
`plot`	Character variable. If ”sp”, the a plot of sensitivity, specificity and predictive values against test is produced, if ”ROC” a ROC-curve is plotted. Both may be given.
`PS`	logical, if TRUE the x-axis in the plot ”ps”-plot is the the predicted probability for `stat`==TRUE, otherwise it is the scale of `test` if this is given otherwise the scale of the linear predictor from the logistic regression.
`PV`	Should sensitivity, specificity and predictive values at the optimal cutpoint be given on the ROC plot?
`MX`	Should the “optimal cutpoint” (i.e. where sens+spec is maximal) be indicated on the ROC curve?
`MI`	Should model summary from the logistic regression model be printed in the plot?
`AUC`	Should the area under the curve (AUC) be printed in the ROC plot?
`grid`	Numeric or logical. If FALSE no background grid is drawn. Otherwise a grid is drawn on both axes at `grid` percent.
`col.grid`	Colour of the grid lines drawn.
`cuts`	Points on the test-scale to be annotated on the ROC-curve.
`lwd`	Thickness of the curves
`data`	Data frame in which to interpret the variables.
`...`	Additional arguments for the plotting of the ROC-curve. Passed on to `plot`

Value

A list with two components:

`res`	dataframe with variables sn, sp, pvp, pvn and fv. The latter is the unique values of test (for PS==FALSE) or linear predictor from the logistic regression
`lr`	glm object with the logistic regression result used for construction of the ROC curve

0, 1 or 2 plots are produced according to the setting of `plot`.

Author(s)

Bendix Carstensen, Steno Diabetes Center & University of Copenhagen, http://www.biostat.ku.dk/~bxc

Examples

```
x <- rnorm( 100 )
z <- rnorm( 100 )
w <- rnorm( 100 )
tigol <- function( x ) 1 - ( 1 + exp( x ) )^(-1)
y <- rbinom( 100, 1, tigol( 0.3 + 3*x + 5*z + 7*w ) )
ROC( form = y ~ x + z, plot="ROC" )
```

S.typh *Salmonella Typhimurium outbreak 1996 in Denmark.*

Description

Matched case-control study of food poisoning.

Format

A data frame with 136 observations on the following 15 variables:

`id`:	Person identification
`set`:	Matched set indicator
`case`:	Case-control status (1:case, 0:control
`age`:	Age of individual
`sex`:	Sex of individual (1:male, 2:female)
`abroad`:	Within the last two weeks visited abroad (1:yes, 0:no)
`beef`:	Within the last two weeks eaten beef
`pork`:	Within the last two weeks eaten pork
`veal`:	Within the last two weeks eaten veal
`poultry`:	Within the last two weeks eaten poultry
`liverp`:	Within the last two weeks eaten liverpaste
`veg`:	Within the last two weeks eaten vegetables
`fruit`:	Within the last two weeks eaten fruit
`egg`:	Within the last two weeks eaten eggs
`plant7`:	Within the last two weeks eaten meat from plant no. 7

Details

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 find the sources. Cases and two age-, sex- and residency-matched controls were telephone interviewed about their food intake during the last two weeks.

The participants were asked at which retailer(s) they had purchased meat. Retailers were independently of this linked to meat processing plants, and thus participants were linked to meat processing plants. This way persons could be linked to (amongst other) plant no 7.

Source

Tine Hald.

References

Molbak K and Hald T: Salmonella Typhimurium outbreak in late summer 1996. A Case-control study. (In Danish: Salmonella typhimurium udbrud paa Fyn sensommeren 1996. En case-kontrol undersogelse.) Ugeskrift for Laeger., 159(36):5372-7, 1997.

Examples

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

`splitLexis` *Split follow-up time in a Lexis object*

Description

The `splitLexis` function divides each row of a `Lexis` object into disjoint follow-up intervals according to the supplied break points.

Usage

```
splitLexis(lex, breaks, time.scale, tol=.Machine$double.eps^0.5)
```

Arguments

`lex`	an object of class `Lexis`
`breaks`	a vector of break points
`time.scale`	the name or number of the time scale to be split
`tol`	numeric value >= 0. Intervals shorter than this value are dropped

Value

An object of class Lexis with multiple rows for each row of the argument lex. Each row of the new Lexis object contains the part of the follow-up interval that falls inside one of the time bands defined by the break points.

The variables representing the various time scales, are appropriately updated in the new Lexis object. The entry and exit status variables are also updated according to the rule that the entry status is retained until the end of follow-up. All other variables are considered to represent variables that are constant in time, and so are replicated across all rows having the same id value.

Note

The splitLexis() function divides follow-up time into intervals using breakpoints that are common to all rows of the Lexis object. To split a Lexis object by break points that are unique to each row, use the cut.Lexis function.

Author(s)

Martyn Plummer

See Also

timeBand, cutLexis, summary.Lexis

Examples

```
# A small bogus cohort
xcoh <- structure( list( id = c("A", "B", "C"),
                      birth = c("14/07/1952", "01/04/1954", "10/06/1987"),
                      entry = c("04/08/1965", "08/09/1972", "23/12/1991"),
                       exit = c("27/06/1997", "23/05/1995", "24/07/1998"),
                       fail = c(1, 0, 1) ),
                     .Names = c("id", "birth", "entry", "exit", "fail"),
                  row.names = c("1", "2", "3"),
                      class = "data.frame" )

# Convert the character dates into numerical variables (fractional years)
xcoh$bt <- cal.yr( xcoh$birth, format="%d/%m/%Y" )
xcoh$en <- cal.yr( xcoh$entry, format="%d/%m/%Y" )
xcoh$ex <- cal.yr( xcoh$exit , format="%d/%m/%Y" )

# See how it looks
xcoh

# Define as Lexis object with timescales calendar time and age
Lcoh <- Lexis( entry = list( per=en ),
                exit = list( per=ex, age=ex-bt ),
         exit.status = fail,
                data = xcoh )

# Default plot of follow-up
plot( Lcoh )
# With a grid and deaths as endpoints
plot( Lcoh, grid=0:10*10, col="black" )
points( Lcoh, pch=c(NA,16)[Lcoh$lex.Xst+1] )
# With a lot of bells and whistles:
plot( Lcoh, grid=0:20*5, col="black", xaxs="i", yaxs="i",
      xlim=c(1960,2010), ylim=c(0,50), lwd=3, las=1 )
points( Lcoh, pch=c(NA,16)[Lcoh$lex.Xst+1], col="red", cex=1.5 )

# Split time along two time-axes
```

```
( x2 <- splitLexis( Lcoh, breaks = seq(1900,2000,5), time.scale="per") )
( x2 <- splitLexis( x2, breaks = seq(0,80,5), time.scale="age" ) )
str( x2 )

# Tabulate the cases and the person-years
summary( x2 )
tapply( status(x2,"exit")==1, list( timeBand(x2,"age","left"),
                                    timeBand(x2,"per","left") ), sum )
tapply( dur(x2),  list( timeBand(x2,"age","left"),
                        timeBand(x2,"per","left") ), sum )
```

`start.Lexis` *Time series methods for Lexis objects*

Description

Extract the entry time, exit time, status, or duration of follow-up from a `Lexis` object.

Usage

```
entry(x, time.scale = NULL)
exit(x, time.scale = NULL)
status(x, at="exit")
dur(x)
```

Arguments

`x`	an object of class `Lexis`.
`time.scale`	a string or integer indicating the time scale. If omitted, all times scales are used.
`at`	string indicating the time point(s) at which status is to be measured.

Value

The `entry` and `exit` functions return a vector of entry times and exit times, respectively, on the requested time scale. If multiple time scales are requested, then a matrix is returned.

The `status` function returns a vector giving the status at entry or exit and `dur` returns a vector with the lengths of the follow-up intervals.

Author(s)

Martyn Plummer

See Also

`Lexis`

`stattable.funs` *Special functions for use in stat.table*

Description

These functions may be used as `contents` arguments to the function `stat.table`. They are defined internally in `stat.table` and have no independent existence.

Usage

```
count(id)
ratio(d,y,scale=1, na.rm=TRUE)
percent(...)
```

Arguments

`id` numeric vector in which identical values identify the same individual.

`d, y` numeric vectors of equal length (`d` for Deaths, `y` for person-Years)

`scale` a scalar giving a value by which the ratio should be multiplied

`na.rm` a logical value indicating whether `NA` values should be stripped before computation proceeds.

`...` a list of variables taken from the `index` argument to `stat.table`

Value

When used as a `contents` argument to `stat.table`, these functions create the following tables:

`this-is-escaped-codenormal-bracket30bracket-normal`
If given without argument (`count()`) it returns a contingency table of counts. If given an `id` argument it returns a table of the number of different values of `id` in each cell, i.e. how many persons contribute in each cell.

`this-is-escaped-codenormal-bracket36bracket-normal`
returns a table of values `scale * sum(d)/sum(y)`

`this-is-escaped-codenormal-bracket40bracket-normal`
returns a table of percentages of the classifying variables. Variables that are in the `index` argument to `stat.table` but not in the call to `percent` are used to define strata, within which the percentages add up to 100.

Author(s)

Martyn Plummer

See Also

`stat.table`

`stat.table` *Tables of summary statistics*

Description

`stat.table` creates tabular summaries of the data, using a limited set of functions. A list of index variables is used to cross-classify summary statistics. It does NOT work inside `with()`!

Usage

```
stat.table(index, contents = count(), data, margins = FALSE)
## S3 method for class 'stat.table':
print(x, width=7, digits,...)
```

Arguments

`index` A factor, or list of factors, used for cross-classification. If the list is named, then the names will be used when printing the table. This feature can be used to give informative labels to the variables.

`contents` A function call, or list of function calls. Only a limited set of functions may be called (See Details below). If the list is named, then the names will be used when printing the table.

`data` an optional data frame containing the variables to be tabulated. If this is omitted, the variables will be searched for in the calling environment.

`margins` a logical scalar or vector indicating which marginal tables are to be calculated. If a vector, it should be the same length as the `index` argument: values corresponding to `TRUE` will be retained in marginal tables.

`x`	an object of class `stat.table`.
`width`	a scalar giving the minimum column width when printing.
`digits`	a scalar, or named vector, giving the number of digits to print after the decimal point. If a named vector is used, the names should correspond to one of the permitted functions (See Details below) and all results obtained with that function will be printed with the same precision.
`...`	further arguments passed to other print methods.

Details

This function is similar to `tapply`, with some enhancements: multiple summaries of multiple variables may be mixed in the same table; marginal tables may be calculated; columns and rows may be given informative labels; pretty printing may be controlled by the associated print method.

This function is not a replacement for `tapply` as it also has some limitations. The only functions that may be used in the `contents` argument are: `count`, `mean`, `weighted.mean`, `sum`, `quantile`, `median`, `IQR`, `max`, `min`, `ratio`, and `percent`.

The `count()` function, which is the default, simply creates a contingency table of counts. The other functions are applied to each cell created by combinations of the `index` variables.

Value

An object of class `stat.table`, which is a multi-dimensional array. A print method is available to create formatted one-way and two-way tables.

Note

The permitted functions in the contents list are defined inside `stat.table`. They have the same interface as the functions callable from the command line, except for two differences. If there is an argument `na.rm` then its default value is always `TRUE`. A second difference is that the `quantile` function can only produce a single quantile in each call.

Author(s)

Martyn Plummer

See Also

`table`, `tapply`, `mean`, `weighted.mean`, `sum`, `quantile`, `median`, `IQR`, `max`, `min`, `ratio`, `percent`, `count`

Examples

```
data(warpbreaks)
# A one-way table
stat.table(tension,list(count(),mean(breaks)),data=warpbreaks)
# The same table with informative labels
stat.table(index=list("Tension level"=tension),list(N=count(),
           "mean number of breaks"=mean(breaks)),data=warpbreaks)

# A two-way table
stat.table(index=list(tension,wool),mean(breaks),data=warpbreaks)
# The same table with margins over tension, but not wool
stat.table(index=list(tension,wool),mean(breaks),data=warpbreaks,
           margins=c(TRUE, FALSE))

# A table of column percentages
stat.table(list(tension,wool), percent(tension), data=warpbreaks)
# Cell percentages, with margins
stat.table(list(tension,wool),percent(tension,wool), margin=TRUE,
           data=warpbreaks)
```

```
# A table with multiple statistics
# Note how each statistic has its own default precision
a <- stat.table(index=list(wool,tension),
                contents=list(count(),mean(breaks),percent (wool)),
                data=warpbreaks)
print(a)
# Print the percentages rounded to the nearest integer
print(a, digits=c(percent=0))

# An Epidemiological example with follow-up time
data(nickel)
str(nickel)

# Make a grouped version of the exposure variable
nickel$egr <- cut( nickel$exposure, breaks=c(0, 0.5, 5, 10, 100), right=FALSE )
stat.table( egr, list( count(), percent(egr), mean( age1st ) ), data=nickel )

# Split the follow-up time by current age
nickel.ex <-
W.Lexis( entry=agein, exit=ageout, fail=icd %in% c(162,163),
         origin=0, breaks=seq(0,100,20),
         include=list( id, exposure, egr, age1st, icd ), data=nickel )
str( nickel.ex )

# Table of rates
stat.table( Time, list( n=count(), N=count(id), D=sum(Fail),
                        "Rate/1000"=ratio(Fail,Exit-Entry,1000) ),
            margin=1, data=nickel.ex )
# Two-way table of rates and no. persons contributing
stat.table( list(age=Time, Exposure=egr),
            list( N=count(id), D=sum(Fail), Y=sum((Exit-Entry)/1000),
                  Rate=ratio(Fail,Exit-Entry,1000) ),
            margin=TRUE, data=nickel.ex )
```

subset.Lexis *Subsetting Lexis objects*

Description

Return subsets of Lexis objects which meet conditions

Usage

```
## S3 method for class 'Lexis':
subset(x, ...)
```

Arguments

x	an object of class Lexis
...	additional arguments to be passed to subset.data.frame

Details

The subset method for Lexis objects works exactly as the method for data frames.

Value

A Lexis object with selected rows and columns.

Author(s)

Martyn Plummer

See Also

Lexis, merge.Lexis

---

summary.Lexis *Summarize transitions and risk time from a Lexis object*

Description

A two-way table of records and transitions classified by states (lex.Cst and lex.Xst), as well the risk time in each state.

Usage

```
## S3 method for class 'Lexis':
summary( object, simplify=TRUE, scale=1, ... )
## S3 method for class 'summary.Lexis':
print( x, ..., digits=2 )
```

Arguments

object	A Lexis object.
x	A summary.Lexis object.
simplify	Should rows with 0 follow-up time be dropped?
scale	Scaling factor for the rates. The calculated rates are multiplied by this number.
digits	How many digits should be used for printing?
...	Other parameters - ignored

Value

An object of class summary.Lexis, a list with two components, Transitions and Rates, each one a matrix with rows classified by states where persons spend time, and columns classified by stated to which persons transit. The Transitions contains number of transitions and has two extra columns of total number events and total risk time attached. The Rates contians the transitions rates.

Author(s)

Bendix Carstensen, ⟨bxc@steno.dk⟩

Examples

```
data( nickel )
# Lung cancer deaths and other deaths are coded 1 and 2
nic <- Lexis( data=nickel,
             entry=list(age=agein),
              exit=list(age=ageout,cal=ageout+dob,tfh=ageout-age1st),
       exit.status=factor( (icd > 0) + (icd %in% c(162,163)),
                           labels=c("Alive","Other","Lung") ) )
str( nic )
head( nic )
summary( nic )
```

tabplot *Graphical display of a 2-way contingency table*

Description

Entries in a table are plotted as rectangles proportional to the entry in the table. Width of rectangles are proportional to column totals, height proportional to entries within each column, hence areas are proportional to entries in the table.

Usage

```
tabplot( M,
       col,
    border = "black",
       lwd = 2,
   collabs = TRUE,
   rowlabs = NULL,
     equal = FALSE,
       las = 1,
      main = NULL,
  cex.main = 1.0,
     vaxis = FALSE )
```

Arguments

`M`	Two-way table
`col`	colors to use for coloring within each column. Defaults to a grayscale. May also be a function that takes an integer argument, as e.g. `rainbow()`.
`border`	color of borders around rectangles.
`lwd`	width of the lines around rectangles.
`collabs`	should colums be labelled.
`rowlabs`	character ”r” or ”l”: rows labelled on left or right side
`equal`	should colums be plotted of equal width? If yes a plot similar to that obtainable from barplot is the result.
`las`	how should labelling be rotated.
`main`	heading for the plot
`cex.main`	character expansion for the heading.
`vaxis`	should a vertical axis be drawn. If character it gives the side where it is drawn.

Details

The function offers a few more facilities for two-way tables than `mosaicplot`, but is restricted to two-way tables as input.

Value

NULL. The function is used for its sideeffects.

Author(s)

Bendix Carstensen, Steno Diabetes Center & Dept. of Biostatistics, University of Copenhagen ⟨bxc@steno.dk⟩, `http://www.pubhealth.ku.dk/~bxc`

See Also

See Also `barplot`, `plot.table`, `mosaicplot`

Examples

```
b <- sample( letters[1:4], 300, replace=TRUE, prob=c(3,1,2,4)/10 )
a <- rnorm( 300 ) - as.integer( factor( b ) ) / 8
tb <- table( cut( a, -3:2 ), b )
tabplot( tb )
tabplot( tb, rowlabs="right", col=heat.colors )

# Very similar plots
ptb <- sweep( tb, 2, apply( tb, 2, sum ), "/" )
par( mfrow=c(2,2) )
barplot( ptb, space=0 )
tabplot( tb, equal=TRUE, lwd=1 )
tabplot( tb, equal=TRUE, lwd=1, rowlabs="l" )
tabplot( tb, equal=FALSE, lwd=1, rowlabs="l" )
```

`thoro` *Thorotrast Study*

Description

The `thoro` data frame has 2470 rows and 14 columns. Each row represents one patient that have had cerebral angiography (X-ray of the brain) with an injected contrast medium, either Thorotrast or another one (the controls).

Format

This data frame contains the following columns:

`id`: Identification of person.
`sex`: Sex, 1: male / 2: female.
`birthdat`: Date of birth, `Date` variable.
`contrast`: Group, 1: Thorotrast / 2: Control.
`injecdat`: Date of contrast injection, `Date` variable.
`volume`: Injected volume of Thorotrast in ml. Control patients have a 0 in this variable.
`exitdat`: Date of exit from the study, `Date` variable.
`exitstat`: Status at exit, 1: dead / 2: alive, censored at closing of study, 20 February 1992 / 3: censored alive at some earlier
`cause`: Cause of death. See causes in the helpfile for `gmortDK`
`liverdat`: Date of liver cancer diagnosis, `Date` variable.
`liver`: Indicator of liver cancer diagnosis. Not all livercancers are histologically verified, hence `liver >= hepcc + chola`
`hepcc`: Hepatocellular carcinoma at `liverdat`.
`chola`: Cholangiocellular carcinoma at `liverdat`.
`hmang`: Haemangisarcoma carcinoma at `liverdat`.

Source

M Andersson, M Vyberg, J Visfeldt, B Carstensen & HH Storm: Primary liver tumours among Danish patients exposed to Thorotrast. Radiation Research, 137, pp. 262–273, 1994.

M Andersson, B Carstensen HH Storm: Mortality and cancer incidence after cerebral angiography. Radiation Research, 142, pp. 305–320, 1995.

See Also

`mortDK`, `gmortDK`

Examples

```
data(thoro)
str(thoro)
```

timeBand *Extract time band data from a split Lexis object*

Description

The break points of a Lexis object (created by a call to splitLexis) divide the follow-up intervals into time bands along a given time scale. The **breaks** function returns the break points, for a given time scale, and the **timeBand** classifies each row (=follow-up interval) into one of the time bands.

Usage

```
timeBand(lex, time.scale, type="integer")
breaks(lex, time.scale)
```

Arguments

lex	an object of class Lexis
time.scale	a character or integer vector of length 1 identifying the time scale of interest
type	a string that determines how the time bands are labelled. See Details below

Details

Time bands may be labelled in various ways according to the type argument. The permitted values of the type argument, and the corresponding return values are:

"integer" a numeric vector with integer codes starting from 0.

"factor" a factor (unordered) with labels "(left,right]"

"left" the left-hand limit of the time band

"middle" the midpoint of the time band

"right" the right-hand limit of the time band

Value

The **breaks** function returns a vector of break points for the Lexis object, or NULL if no break points have been defined by a call to splitLexis. The **timeBand** function returns a numeric vector or factor, depending on the value of the type argument.

Note

A newly created Lexis object has no break points defined. In this case, **breaks** will return NULL, and **timeBand** will a vector of zeros.

Author(s)

Martyn Plummer

See Also

Lexis

Examples

```
data(diet)
diet <- cal.yr(diet)
diet.lex <- Lexis(entry=list(period=doe),
                  exit=list(period=dox, age=dox-dob),
            exit.status=chd,
                  data=diet)
```

```
diet.split <- splitLexis(diet.lex, breaks=seq(40,70,5), "age" )
age.left <- timeBand(diet.split, "age", "left")
table(age.left)
age.fact <- timeBand(diet.split, "age", "factor")
table(age.fact)
age.mid <- timeBand(diet.split, "age", "mid")
table(age.mid)
```

timeScales *The time scales of a Lexis object*

Description

Function to get the names of the time scales of a Lexis object.

Usage

```
timeScales(x)
```

Arguments

x	an object of class Lexis

Value

A character vector containing the names of the variables in x that represent the time scales

Author(s)

Martyn Plummer

See Also

Lexis, splitLexis

transform.Lexis *Transform a Lexis objects*

Description

Transform a Lexis object

Usage

```
## S3 method for class 'Lexis':
transform(`_data`, ...)
```

Arguments

_data	an object of class Lexis.
...	additional arguments to be passed to transform.data.frame.

Details

The transform method for Lexis objects works exactly as the method for data frames.

Value

A transformed Lexis object.

Author(s)

Martyn Plummer

See Also

Lexis, merge.Lexis, subset.Lexis

---

twoby2 *Analysis of a two by two table*

Description

Computes the usual measures of association in a 2 by 2 table with confidence intervals. Also produces asymtotic and exact tests. Assumes that comparison of probability of the first column level between levels of the row variable is of interest. Output requires that the input matrix has meaningful row and column labels.

Usage

```
twoby2(exposure, outcome,
       alpha = 0.05, print = TRUE, dec = 4,
       conf.level = 1-alpha, F.lim = 10000)
```

Arguments

`exposure`	If a table the analysis is based on the first two rows and first two columns of this. If a variable, this variable is tabulated against
`outcome`	as the second variable
`alpha`	Significance level
`print`	Should the results be printed?
`dec`	Number of decimals in the printout.
`conf.level`	`1-alpha`
`F.lim`	If the table total exceeds `F.lim`, Fisher's exact test is not computed

Value

A list with elements:

`table`	The analysed 2 x 2 table augmented with probabilities and confidence intervals. The confidence intervals for the probabilities are computed using the normal approximation to the log-odds. Confidence intervals for the difference of proportions are computed using method 10 from Newcombe, Stat.Med. 1998, 17, pp.873 ff.
`measures`	A table of Odds-ratios and relative risk with confidence intervals.
`p.value`	Exact p-value for the null hypothesis of OR=1

Author(s)

Mark Myatt. Modified by Bendix Carstensen.

Examples

```
Treat <- sample(c("A","B"), 50, rep=TRUE )
Resp <- c("Yes","No")[1+rbinom(50,1,0.3+0.2*(Treat=="A"))]
twoby2( Treat, Resp )
twoby2( table( Treat, Resp )[,2:1] ) # Comparison the other way round
```

W.Lexis *Split follow-up time in cohort studies.*

Description

This is an old version of time-splitting. Consult the new version, Lexis.

For cohort input data the follow-up time is chopped in pieces along several time scales, and a dataframe of follow-up intervals is returned. Entry and exit times are assumed to be in the same timescale (the input time scale).

Usage

```
W.Lexis( entry = 0,
          exit,
          fail,
        origin = 0,
         scale = 1,
        breaks,
       include = NULL,
          data = NULL )
```

Arguments

`entry`	Date of entry on the input timescale. Numerical variable.
`exit`	Date of exit on the input timescale. Numerical variable.
`fail`	Failure indicator.
`origin`	Origin of the output timescale(s) on the input timescale. If for example the input timescale is calendar time and the output timescale is (current) age, the the origin is date of birth. If more than one timescale is used for splitting time this is a list. Elements of the list must be named and must have the same names as those in `scale` and `breaks`.
`scale`	Scale of the output timescale(s) relative to the input timescale. Elements of the list must be named and have the same names as those in `origin` and `breaks`.
`breaks`	Points on the output scale where the follow-up is cut. If more than one timescale is used for splitting time this is a list. Elements of the list must be named and must have the same names as those in `origin` and `scale`.
`include`	List of variables to carry unchanged from the original dataframe to the output dataframe.
`data`	Dataframe in which to interpret the arguments.

Details

The `data` is assumed to be a dataframe describing the follow-up of a cohort, giving entry and exit time (on the input timescale) for each individual as well as the exit status (failure status, `fail`). The purpose of the function is to split each individual's follow-up time along a number of timescales for example age, calendar time, time since entry etc. Any follow-up time before the first break on any timescale or after the last break on any of these timescales (the output timescales) is discarded.

NOTE: If a person has his/her exit before the first break or his entry after the last break on any of the timescales the function will crash.

Value

A dataframe with one row per follow-up interval. The data frame includes all variables given in the argument `include`, replicated across all intervals from the same individual. In addition, the dataframe has the following variables:

`Expand`	A numerical vector with values in `1:nrows(data)`, pointing at the rows of the input data frame that is expanded to the output intervals.

`Entry` Date of entry for each interval. On the input time scale.

`Exit` Date of exit for each interval. On the input time scale.

`Fail` Exit status for each interval. Coded 0 for censoring, for the last follow-up interval for each person it takes the value of `fail`.

`Time` If `origin`, `scale` or `breaks` were given as vectors this gives the left endpoints of the intervals on the output scale.

If `origin`, `scale` or `breaks`, were given as lists, there is no variable `Time` in the dataframe, instead variables with the same names as the list elements of these will be in the dataframe. The variables have values corresponding to the left endpoints of the intervals on the respective output time scales.

Author(s)

David Clayton, approx. 2000. Small modifications by Bendix Carstensen.

References

This function has approximately the same functionality as `stsplit` in Stata and the SAS-macro `%Lexis` (`http://www.biostat.ku.dk/~bxc/Lexis/Lexis.sas`). It has been attempted to keep argument names similar between the three functions.

See Also

`Lexis`,`Lexis.diagram`

Examples

```
# A small bogus cohort
#
xcoh <- structure( list( id = c("A", "B", "C"),
                      birth = c("14/07/1952", "01/04/1954", "10/06/1987"),
                      entry = c("04/08/1965", "08/09/1972", "23/12/1991"),
                       exit = c("27/06/1997", "23/05/1995", "24/07/1998"),
                       fail = c(1, 0, 1) ),
                     .Names = c("id", "birth", "entry", "exit", "fail"),
                  row.names = c("1", "2", "3"),
                      class = "data.frame" )

# Convert the character dates into numerical variables (fractional years)
#
xcoh$bt <- cal.yr( xcoh$birth, format="%d/%m/%Y" )
xcoh$en <- cal.yr( xcoh$entry, format="%d/%m/%Y" )
xcoh$ex <- cal.yr( xcoh$exit , format="%d/%m/%Y" )

# See how it looks
#
xcoh

# Split time along one time-axis
#
W.Lexis( entry = en,
          exit = ex,
          fail = fail,
         scale = 1,
        origin = bt,
        breaks = seq( 5, 40, 5 ),
       include = list( bt, en, ex, id ),
          data = xcoh )

# Split time along two time-axes
```

```
#
( x2 <-
W.Lexis( entry = en,
          exit = ex,
          fail = fail,
         scale = 1,
        origin = list( per=0,                 age=bt          ),
        breaks = list( per=seq(1900,2000,10), age=seq(0,80,5) ),
       include = list( bt, en, ex, id ),
          data = xcoh ) )

# Tabulate the cases and the person-years
#
tapply( x2$Fail, list( x2$age, x2$per ), sum )
tapply( x2$Exit - x2$Entry, list( x2$age, x2$per ), sum )
```