

Lee Carter models —und verwandte Gebiete

SDCC / Max Plack Insitute of Demographic Research, Rostock

April 2018

<http://BendixCarstensen.com/APC>

2

Compiled Sunday 29th April, 2018, 20:35

from: /home/bendix/teach/APC/00/LeeCarter/LCa.tex

Bendix Carstensen Steno Diabetes Center Copenhagen, Gentofte, Denmark
& Department of Biostatistics, University of Copenhagen
bendix.carstensen@regionh.dk b@bxc.dk
www.BendixCarstensen.com

Contents

1	Lee-Carter models and the APC	2
1.1	Lee-Carter model	2
1.2	The generalized Lee-Carter model	3
1.3	Danish Lung cancer data	3
1.4	Lee-Carter model with <code>demography</code>	6
1.5	Lee-Carter model with <code>ilc</code>	7
1.6	The parameter constraints	8
1.7	The APC-model with <code>ilc</code>	11
1.8	Iterative fitting of extended Lee-Carter spline models	17
1.8.1	Rescaling effects to compare with <code>ilc</code> and <code>demography</code> estimates	21
1.9	The LCa function with smooth effects	26
	References	38
2	Practical examples	39
2.1	Lung cancer in Danish women	39
2.2	Mortality in Czech men	47

Preface

In May 2016 I taught a course on Age-Period-Cohort models at the Max Planck Institute for Demographic Research.

I chose to include a bit about the Lee-Carter model as well because both the Lee-Carter model as well as the Age-Period-Cohort models can be seen as extensions of the simple Age-Period model with a parametric interaction, albeit of different form.

The MPIDR was so kind to house me as visiting scholar for a week after the course, during which I wrote the first draft of the `LCa.fit` function for the *Epi* package and some rudimentary pieces of this note.

The aim is to publish an edited version of this note in JSS.

Bendix Carstensen
April 2018

Chapter 1

Lee-Carter models and the APC

1.1 Lee-Carter model

The Lee-Carter model for mortality rates observed in a Lexis diagram — that is classified by age (a or x) and date of observation (p or t):

$$\log(\lambda(a, p)) = f(a) + b(a) \times k(p)$$

- Formulated originally using f , b and k as step-functions with one parameter per age and period, respectively.
- Implicitly assumes a data lay out by age and period (A-sets)
- Relative scaling of $b(a)$ and $k(t)$ cannot be determined
- $k(t)$ only determined up to an affine transformation (that is, a constant can be added), because the constant multiplied by $b(a)$ can be absorbed in $f(a)$
- Lee-Carter model is an extension of the age-period model; if $b(a) == 1$ it *is* the age-period model.
- The extension is an age \times period interaction

– ...but not a traditional additional interaction term:

$$\log(\lambda(a, p)) = f(a) + b(a) \times k(p) = f(a) + k(p) + (b(a) - 1) \times k(p)$$

— the main effect and the interaction component of p are constrained to be identical.

– ...and neither a simple product of main effects:

$$\log(\lambda(a, p)) = f(a) + k(p) + f(a) \times k(p) = f(a) + (f(a) - 1) \times k(p)$$

— the age-component of the interaction term is not constrained to have the same shape as the main effect of age, as would be the case of a simple multiplicative interaction.

- Thus the Lee-Carter model is more general than just adding the product of the two main effects, but less general than adding a product of two (new) main effects.

1.2 The generalized Lee-Carter model

The `ilc` package by Renshaw and Haberman [1] fits the Lee-Carter model using categorical parametrization — basically factor-type models. Incidentally, this type of parametrization assumes both categories of age and categories of period to be exchangeable. So the model does not exploit the ordering let alone the scaling of the categories of age and period.

The package contains the extension of the Lee-Carter model from the paper with an age-cohort term, similar to the age-period term:

$$\log(\lambda(a, p)) = f(a) + b_p(a) \times k_p(p) + b_c(a) \times k_c(p - a) \quad (1.1)$$

This model encompasses all models we discuss here, and depending on whether b_p and/or b_c are constrained to be 0 or 1 or not constrained at all we have a range of models, including the classical A, AP, AC and APC models, as shown in table 1.1.

The lower right corner of 6 models are named as in the vignette from the `ilc` package, and (in `tt` font) as the corresponding value of the `model` argument to the function `lca.rh`. Note however that the Lee-Carter models only appear as the constrained interactions with age; none of these models have the corresponding period or cohort main effect included.

Furthermore, if we impose the restriction $b_p(a) = b_c(a) = 1, \forall a$ on the model 1.1, we will have a classical Age-Period-Cohort model with the usual parametrization problems.

The reason that these problems do not exist in the Lee-Carter type models is that the chosen interactions do not include a separate main effect of period resp. cohort.

For a discussion and implementation of the smooth-effects Lee-Carter model we use Danish lung cancer data.

1.3 Danish Lung cancer data

First we read in the Danish lung cancer data:

```
> library( ilc )
> library( demography )
> library( Epi )
> library( splines )
```

Table 1.1: Models derivable from the M model by fixing one or two β s to 0 or 1. Notation as in the vignette for the `ilc` package. Note that models are nested in the sense that moving up or left in the table represent a simplification of the models.

		$b_c(a)$		
		0	1	free
$b_p(a)$	0	Age	Age+Coh	LCa(C) AC, <code>ac</code>
	1	Age+Per	Age+Per+Coh H_0 , <code>h0</code>	Age+Per+LCa(C) H_1 , <code>h1</code>
	free	LCa(P) LC, <code>lc</code>	Age+Coh+LCa(P) H_2 , <code>h2</code>	Age+LCa(P)+LCa(C) M, <code>m</code>

```

> clear()
> print( sessionInfo(), l=F )
R version 3.4.4 (2018-03-15)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 14.04.5 LTS

Matrix products: default
BLAS: /usr/lib/openblas-base/libopenblas.so.0
LAPACK: /usr/lib/lapack/liblapack.so.3.0

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

other attached packages:
[1] Epi_2.27      ilc_1.0      date_1.2-37  rainbow_3.4  pcaPP_1.9-72
[6] MASS_7.3-49  demography_1.20 forecast_8.1

loaded via a namespace (and not attached):
 [1] multicool_0.1-10  zoo_1.8-0      ks_1.10.7      etm_1.0.1
 [5] lattice_0.20-35  colorspace_1.3-2  stats4_3.4.4   mgcv_1.8-23
 [9] cobs_1.3-3       survival_2.41-3  rlang_0.1.1    hdrd_3.1
[13] TTR_0.23-2       plyr_1.8.4      quantmod_0.4-10  timeDate_3012.100
[17] sde_2.0.15       MatrixModels_0.4-1  munsell_0.4.3  gtable_0.2.0
[21] mvtnorm_1.0-6    codetools_0.2-15  misc3d_0.8-4   tseries_0.10-42
[25] SparseM_1.77     lmtest_0.9-35    quantreg_5.33  parallel_3.4.4
[29] curl_2.2         xts_0.10-0      Rcpp_0.12.12   KernSmooth_2.23-15
[33] scales_0.4.1     cmprsk_2.2-7     FNN_1.1        ftsa_4.8
[37] fracdiff_1.4-2   ggplot2_2.2.1    numDeriv_2016.8-1  grid_3.4.4
[41] quadprog_1.5-5   magrittr_1.5     rgl_0.93.996   lazyeval_0.2.0
[45] tibble_1.3.3     cluster_2.0.6    fda_2.4.4      Matrix_1.2-11
[49] data.table_1.10.4  nnet_7.3-12     nlme_3.1-131.1  compiler_3.4.4

> 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

```

For the sake of fitting with the `demography` and `ilc` packages we tabulate by age and period classes (A-sets):

```

> lap <- aggregate( lung[,c("D","Y")], lung[,c("sex","A","P")], FUN=sum )
> str( lap )
'data.frame':      10980 obs. of  5 variables:
 $ sex: int  1 2 1 2 1 2 1 2 1 2 ...
 $ A  : int  0 0 1 1 2 2 3 3 4 4 ...
 $ P  : int  1943 1943 1943 1943 1943 1943 1943 1943 1943 1943 ...
 $ D  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Y  : num  40343 38595 36716 35106 34008 ...

```

In order to set up `demogr` objects, we further need data in the form of an array classified by age and period.

```

> ltab <- xtabs( cbind(D,Y) ~ sex + A + P, data=lung )
> str( ltab )
xtabs [1:2, 1:90, 1:61, 1:2] 0 0 0 0 0 0 0 0 0 0 ...
- attr(*, "dimnames")=List of 4
..$ sex: chr [1:2] "1" "2"
..$ A : chr [1:90] "0" "1" "2" "3" ...
..$ P : chr [1:61] "1943" "1944" "1945" "1946" ...
..$ : chr [1:2] "D" "Y"
- attr(*, "call")= language xtabs(formula = cbind(D, Y) ~ sex + A + P, data = lung)
> lcM <- demogdata( data = as.matrix(ltab[1,40:90,,"D"]/ltab[1,40:90,,"Y"]),
+                 pop = as.matrix(ltab[1,40:90,,"Y"]),
+                 ages = as.numeric(dimnames(ltab)[[2]][40:90]),
+                 years = as.numeric(dimnames(ltab)[[3]]),
+                 type = "Lung cancer incidence",
+                 label = "Denmark",
+                 name = "male" )
> lcM
Lung cancer incidence data for Denmark
Series: male
Years: 1943 - 2003
Ages: 39 - 89
> lcF <- demogdata( data = as.matrix(ltab[2,40:90,,"D"]/ltab[2,40:90,,"Y"]),
+                 pop = as.matrix(ltab[2,40:90,,"Y"]),
+                 ages = as.numeric(dimnames(ltab)[[2]][40:90]),
+                 years = as.numeric(dimnames(ltab)[[3]]),
+                 type = "Lung cancer incidence",
+                 label = "Denmark",
+                 name = "female" )
> lcF
Lung cancer incidence data for Denmark
Series: female
Years: 1943 - 2003
Ages: 39 - 89
> str( lcF )
List of 7
 $ year : num [1:61] 1943 1944 1945 1946 1947 ...
 $ age : num [1:51] 39 40 41 42 43 44 45 46 47 48 ...
 $ rate :List of 1
 ..$ female: table [1:51, 1:61] 0.00 3.43e-05 0.00 0.00 0.00 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : chr [1:51] "39" "40" "41" "42" ...
 .. .. ..$ : chr [1:61] "1943" "1944" "1945" "1946" ...
 $ pop :List of 1
 ..$ female: table [1:51, 1:61] 29556 29119 28633 28243 27764 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : chr [1:51] "39" "40" "41" "42" ...
 .. .. ..$ : chr [1:61] "1943" "1944" "1945" "1946" ...
 $ type : chr "Lung cancer incidence"
 $ label : chr "Denmark"
 $ lambda: num 1
 - attr(*, "class")= chr "demogdata"

```

These objects are now (almost) suitable as input to the `lca` and `lca.rh` functions in the `demography` package, and also usable as input to the `ilc` package.

1.4 Lee-Carter model with demography

Then we fit the Lee-Carter model for the lung cancer incidence rates; but funnily enough `lca` checks the value of the `type` argument even if it seems to be only used for annotation, so we make a workaround to avoid having an erroneously labeled object:

```
> mrt <- function(x) {x$type <- "mortality" ; x}
> dmg.lcM <- lca( mrt(lcM), interpolate=TRUE )
> dmg.lcF <- lca( mrt(lcF), interpolate=TRUE )
> par( mfrow=c(1,3) )
> matplot( dmg.lcM$age, exp(cbind(dmg.lcM$ax,dmg.lcF$ax))*1000,
+         log="y", ylab="Lung cancer incidence rates per 1000 PY",
+         xlab="Age", col=c("blue","red"), type="l", lty=1, lwd=4 )
> matplot( dmg.lcM$age, cbind(dmg.lcM$bx,dmg.lcF$bx),
+         ylab="Age specific period multiplier",
+         xlab="Age", col=c("blue","red"), type="l", lty=1, lwd=4 )
> matplot( dmg.lcM$year, cbind(dmg.lcM$kt,dmg.lcF$kt),
+         ylab="Time effect",
+         xlab="Date", col=c("blue","red"), type="l", lty=1, lwd=4 )
> abline(h=0)
```

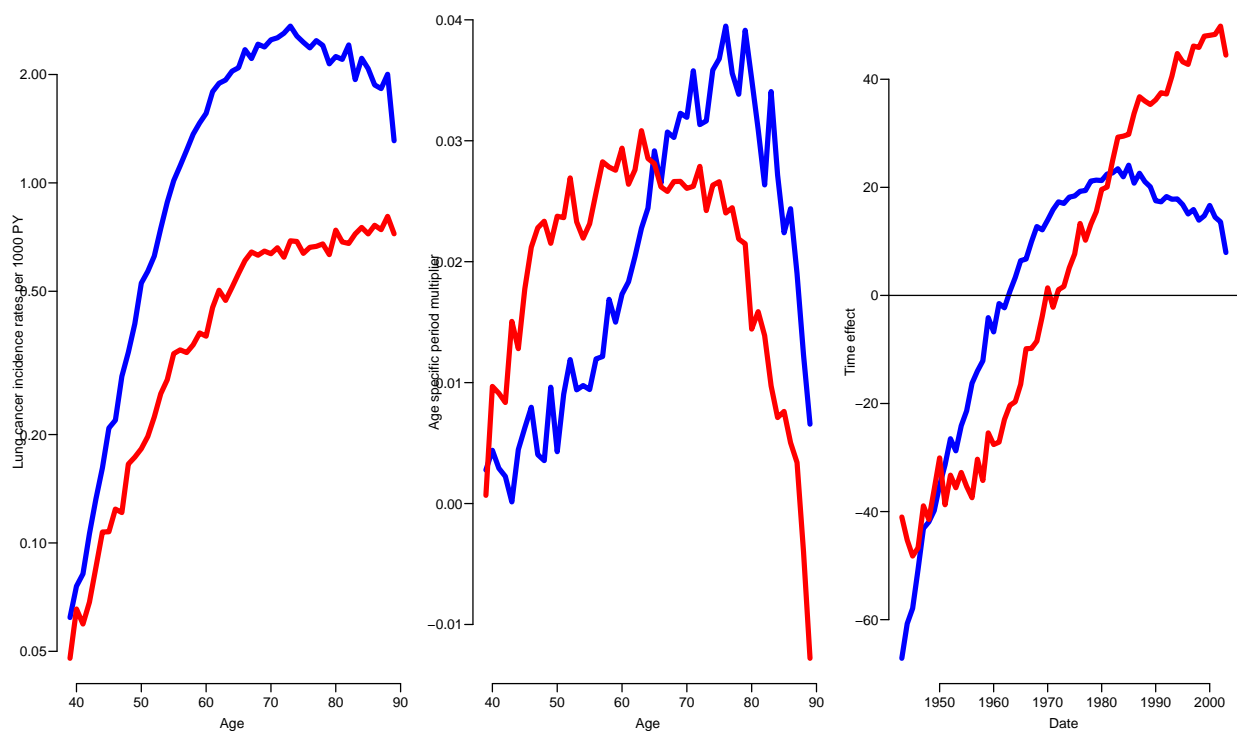


Figure 1.1: *Estimated effects in the Lee-Carter model for male (blue) and female (red) lung cancer in Denmark, using the demography package.* ../graph/LCa-demog-fit

Clearly the type of estimates shown in figure 1.1 calls for a smoother, that is, we want 3 continuous curves, particularly because we want to use the period effect for prediction. Instead of the random walk extrapolation we may just want to extend a fitted parametric curve in some way.

1.5 Lee-Carter model with ilc

We can fit the same model using maximum likelihood based on the Poisson likelihood:

```
> # This model takes forever to fit
> # m.lcM <- lca.rh( mrt(lcM), model="m" )
> # plot( m.lcM, lwd=5 )
> ilc.lcM <- lca.rh( mrt(lcM), model="lc", interpolate=TRUE, verbose=FALSE )
```

Original sample: Mortality data for Denmark

```
Series: male
Years: 1943 - 2003
Ages: 39 - 89
```

Applied sample: Mortality data for Denmark (Corrected: interpolate)

```
Series: male
Years: 1943 - 2003
Ages: 39 - 89
```

```
Fitting model: [ LC = a(x)+b1(x)*k(t) ]
- with poisson error structure and with deaths as weights -
```

Iterations finished in: 34 steps

```
> ilc.lcF <- lca.rh( mrt(lcF), model="lc", interpolate=TRUE, verbose=FALSE )
```

Original sample: Mortality data for Denmark

```
Series: female
Years: 1943 - 2003
Ages: 39 - 89
```

Applied sample: Mortality data for Denmark (Corrected: interpolate)

```
Series: female
Years: 1943 - 2003
Ages: 39 - 89
```

```
Fitting model: [ LC = a(x)+b1(x)*k(t) ]
- with poisson error structure and with deaths as weights -
```

Iterations finished in: 26 steps

```
> names( ilc.lcM )
```

```
[1] "label"      "age"        "year"       "male"       "ax"         "bx"         "kt"
[8] "df"         "residuals" "fitted"     "varprop"   "y"          "mdev"       "model"
[15] "adjust"    "type"       "call"       "conv.iter"  "bx0"        "itx"
```

```
> par( mfrow=c(1,1) )
> plot( ilc.lcM, lwd=5 )
```

We can extract and plot the fit from the `ilc::lca.rh` overlaid on the ones from the `demography::lca` function:

```
> par( mfcrow=c(1,3) )
> matplot( dmg.lcM$age, exp(cbind(dmg.lcM$ax,dmg.lcF$ax))*1000,
+         log="y", ylab="Lung cancer incidence rates per 1000 PY",
+         xlab="Age", col=c("blue","red"), type="l", lty=1, lwd=2 )
> matlines( ilc.lcM$age, exp(cbind(ilc.lcM$ax,ilc.lcF$ax))*1000,
+         col=c("blue","red"), lty=1, lwd=4 )
> matplot( dmg.lcM$age, cbind(dmg.lcM$bx,dmg.lcF$bx),
+         ylab="Age effect",
+         xlab="Age", col=c("blue","red"), type="l", lty=1, lwd=2 )
> matlines( ilc.lcM$age, cbind(ilc.lcM$bx,ilc.lcF$bx),
```

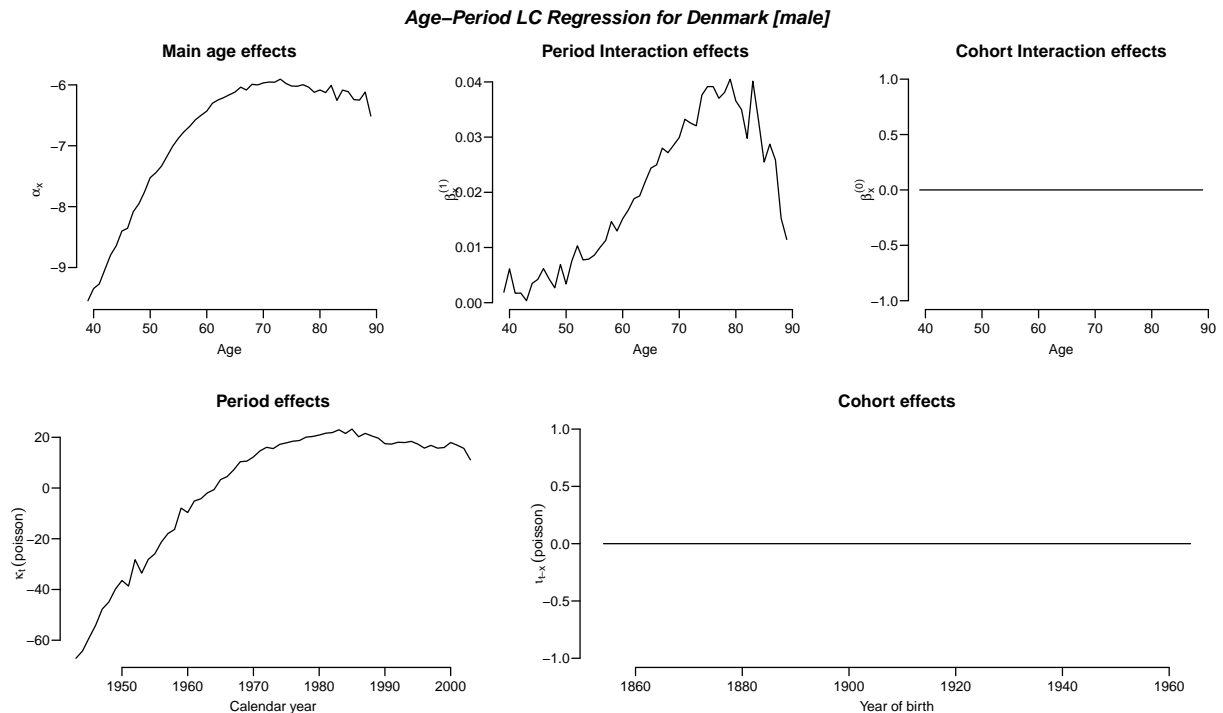


Figure 1.2: *Default plot of the estimates from the `ilc::lca.rh` function for the male lung cancer rates.*

`../graph/LCa-ilc-gr`

```
+           col=c("blue","red"), lty=1, lwd=4 )
> matplot( dmg.lcM$year, cbind(dmg.lcM$kt,dmg.lcF$kt),
+         ylab="Time effect",
+         xlab="Date", col=c("blue","red"), type="l", lty=1, lwd=2 )
> matlines( ilc.lcM$year, cbind(ilc.lcM$kt,ilc.lcF$kt),
+          col=c("blue","red"), lty=1, lwd=4 )
> abline(h=0)
```

We see that the two estimation approaches gives pretty much the same results for the standard Lee-Carter model.

1.6 The parameter constraints

The parameter constraints are basically sum constraints, $\sum b_x = 1$, $\sum k_t = 0$, at least as seen in the `ilc` package:

```
> sumP <- rbind(
+ with( dmg.lcM, c(sum(bx),sum(kt)) ),
+ with( dmg.lcF, c(sum(bx),sum(kt)) ),
+ with( ilc.lcM, c(sum(bx),sum(kt)) ),
+ with( ilc.lcF, c(sum(bx),sum(kt)) ) )
> rownames(sumP) <- paste(rep(c("dmg","ilc"),each=2),rep(c("M","F"),2))
> colnames(sumP) <- c("bx","kt")
> round( sumP, 5 )
      bx      kt
dmg M  1  86.49339
dmg F  1 177.79614
```

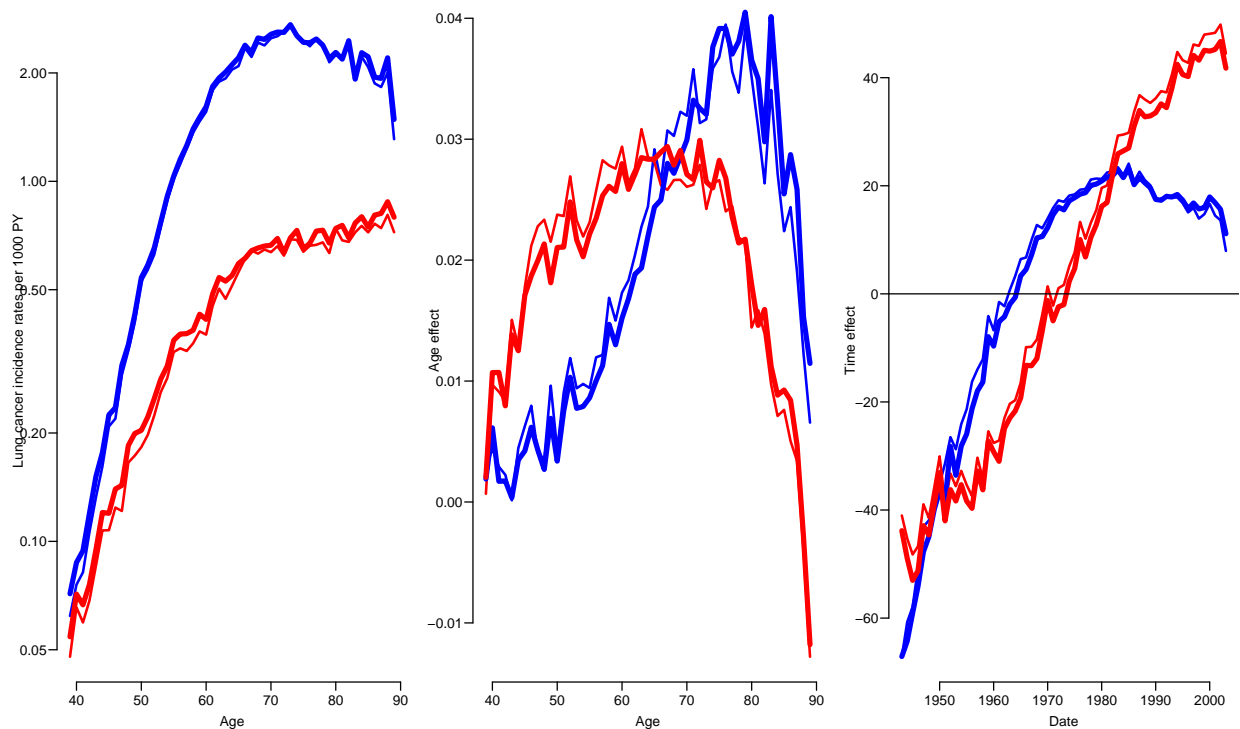


Figure 1.3: Parameters from the Lee-Carter model for Danish lung cancer incidence data (men: blue; women:red) by the `demography::lca` function (thin lines) and the `ilc::lca.rh` (thick lines).
`../graph/LCa-cmp-LC`

```
ilc M 1 0.00000
ilc F 1 0.00000
```

It is not quite clear how the estimates from the `demography` package are constrained.

However it might be more sensible — also for comparative purposes to rescale the effects so that the period effect is 0 at, say, 1980, and the age-specific multiplier is 1 at say age 60:

```
> # where are the reference paramters located
> wha <- which( dmg.lcF$age ==70 )
> why <- which( dmg.lcF$year==1980 )
> # so fish them out
> dMb0 <- dmg.lcM$bx[wha]
> dFb0 <- dmg.lcF$bx[wha]
> iMb0 <- ilc.lcM$bx[wha]
> iFb0 <- ilc.lcF$bx[wha]
> dMk0 <- dmg.lcM$kt[why]
> dFk0 <- dmg.lcF$kt[why]
> iMk0 <- ilc.lcM$kt[why]
> iFk0 <- ilc.lcF$kt[why]
> # and use for rescaling
> par( mfcol=c(1,3) )
> yl <- c(0.05,5)
> matplot( dmg.lcM$age, exp(cbind(dmg.lcM$ax+dmg.lcM$bx*dMk0,
+                               dmg.lcF$ax+dmg.lcF$bx*dFk0))*1000,
+         log="y", type="l", lty=1, lwd=1, ylim=yl,
+         ylab="Lung cancer incidence rates per 1000 PY",
+         xlab="Age", col=c("blue","red") )
```

```

> matlines( ilc.lcM$age, exp(cbind(ilc.lcM$ax+ilc.lcM$bx*iMk0,
+                               ilc.lcF$ax+ilc.lcF$bx*iFk0))*1000,
+          col=c("blue","red"), lty=1, lwd=4 )
> abline( v=70 )
> matplot( dmg.lcM$year, exp(cbind((dmg.lcM$kt-dMk0)*dMb0,
+                               (dmg.lcF$kt-dFk0)*dFb0)),
+         ylab="Time effect (RR)", ylim=y1, log="y",
+         xlab="Date", col=c("blue","red"), type="l", lty=1, lwd=1 )
> matlines( ilc.lcM$year, exp(cbind((ilc.lcM$kt-iMk0)*iMb0,
+                               (ilc.lcF$kt-iFk0)*iFb0)),
+         col=c("blue","red"), lty=1, lwd=4 )
> abline( h=1, v=1980 )
> matplot( dmg.lcM$age, cbind(dmg.lcM$bx/dMb0,dmg.lcF$bx/dFb0),
+         ylab="Age interaction (RR power)", ylim=c(-0.5,1.5),
+         xlab="Age", col=c("blue","red"), type="l", lty=1, lwd=1 )
> matlines( ilc.lcM$age, cbind(ilc.lcM$bx/iMb0,ilc.lcF$bx/iFb0),
+         col=c("blue","red"), lty=1, lwd=4 )
> abline( v=70, h=1 )

```

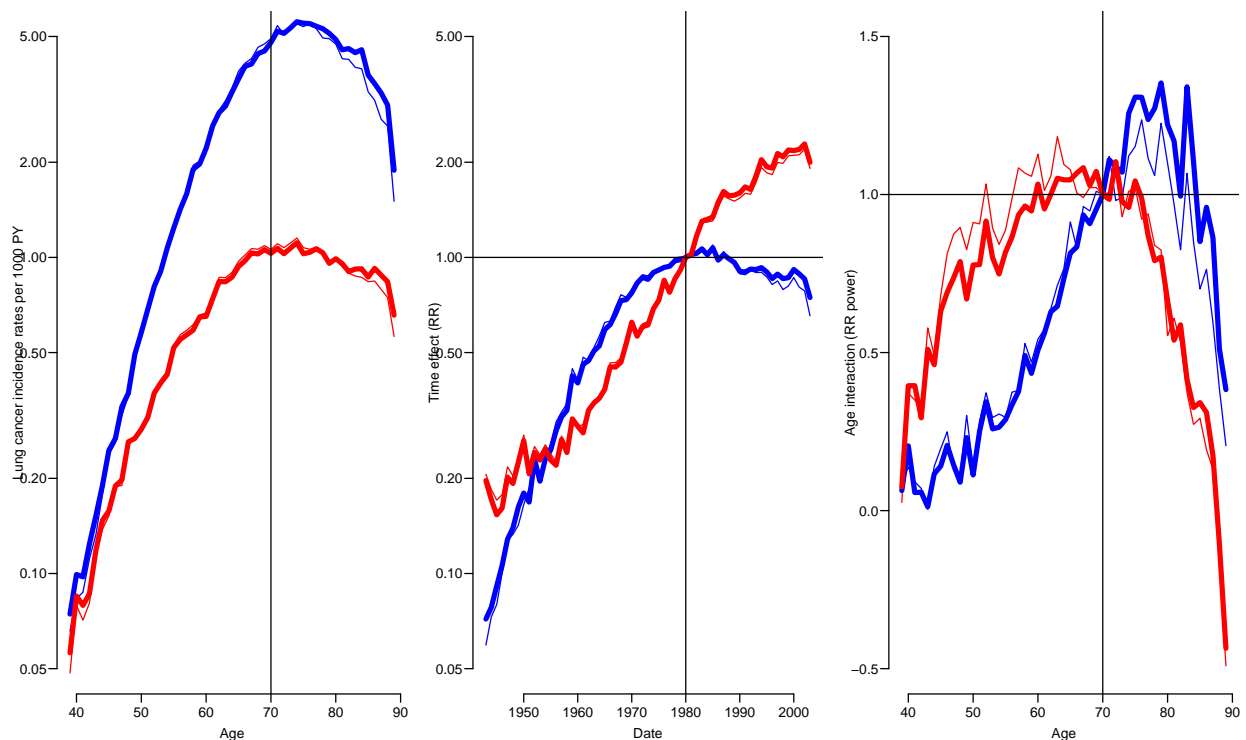


Figure 1.4: Rescaled parameters from the Lee-Carter model for Danish lung cancer incidence data (men: blue; women:red) by the `demography::lca` function (thin lines) and the `ilc::lca.rh` (thick lines). ../graph/LCa-cmp-LCr

In figure ?? we see the epidemiologically scaled parameters of the Lee-Carter model:

$$\log(\lambda(a, p)) = f(a) + b(a) \times k(p)$$

- The reference date is $p = 1980$ because we chose the constraint $k(1980) = 0$, so the predicted log-rates at 1980 are $f(a)$

- $k(p)$ is the log-RR for 70-year old relative to 1980, because we chose $k(1980) = 0$, and the reference age for this is 70 because we chose the constraint $b(70) = 1$. The rightmost graph shows the rate-ratio, that is $\exp(k(p))$.
- $b(p)$ is the component of the age-specific log-rates which is multiplied by the period effect on the log-scale. So this has the role of an exponentiation of the RR — the period-RR for a person ages a is $\exp(b(a)k(p)) = \exp(k(p))^{b(a)}$

1.7 The APC-model with *ilc*

We can also fit the APC-model with the *lca.rh*, by choosing the model to be *h0*:

```
> ilc.apcM <- lca.rh( mrt(lcM), model="h0", interpolate=TRUE, verbose=FALSE )
```

```
Original sample: Mortality data for Denmark
```

```
Series: male
Years: 1943 - 2003
Ages: 39 - 89
```

```
Applied sample: Mortality data for Denmark (Corrected: interpolate)
```

```
Series: male
Years: 1943 - 2003
Ages: 39 - 89
```

```
Fitting model: [ H(0) = a(x)+i(t-x)+k(t) ]
- with poisson error structure and with deaths as weights -
```

```
Iterations finished in: 1 steps
```

```
> ilc.apcF <- lca.rh( mrt(lcF), model="h0", interpolate=TRUE, verbose=FALSE )
```

```
Original sample: Mortality data for Denmark
```

```
Series: female
Years: 1943 - 2003
Ages: 39 - 89
```

```
Applied sample: Mortality data for Denmark (Corrected: interpolate)
```

```
Series: female
Years: 1943 - 2003
Ages: 39 - 89
```

```
Fitting model: [ H(0) = a(x)+i(t-x)+k(t) ]
- with poisson error structure and with deaths as weights -
```

```
Iterations finished in: 1 steps
```

We note the structure of the coefficients from the *ilc*-fit:

```
> ilc.apcM$ax
```

```

      39      40      41      42      43      44      45      46      47
-9.227887 -9.168335 -9.245999 -9.151405 -8.929130 -8.729993 -8.475459 -8.425608 -8.144999
      48      49      50      51      52      53      54      55      56
-7.991030 -7.808424 -7.548808 -7.473300 -7.375981 -7.199157 -7.031817 -6.893406 -6.797680
      57      58      59      60      61      62      63      64      65
-6.698461 -6.596265 -6.524344 -6.463077 -6.324566 -6.270765 -6.249951 -6.193521 -6.171192
      66      67      68      69      70      71      72      73      74
-6.055737 -6.112622 -6.020686 -6.038639 -5.992710 -5.979823 -5.950896 -5.905050 -5.968870
      75      76      77      78      79      80      81      82      83
-6.008455 -6.044352 -5.997987 -6.028564 -6.146490 -6.099577 -6.118032 -6.025449 -6.247306
      84      85      86      87      88      89
-6.110906 -6.177430 -6.281424 -6.191396 -5.997449 -6.305043
```

```
> ilc.apcM$kt
```

```
Time Series:
```

```
Start = 1943
```

```
End = 2003
```

```
Frequency = 1
```

1943	1944	1945	1946	1947	1948	1949	1950
0.0000000	0.08193739	0.10646979	0.20366736	0.31322239	0.30725354	0.31898291	0.37921522
1951	1952	1953	1954	1955	1956	1957	1958
0.42395299	0.49329971	0.42118695	0.48796292	0.51849647	0.60199991	0.62770249	0.64613538
1959	1960	1961	1962	1963	1964	1965	1966
0.80448866	0.72492341	0.82767549	0.79333176	0.84948750	0.89874734	0.96336626	0.96192243
1967	1968	1969	1970	1971	1972	1973	1974
1.03356969	1.09966126	1.08251537	1.12711954	1.17684700	1.21665702	1.21340723	1.24919021
1975	1976	1977	1978	1979	1980	1981	1982
1.26162659	1.29261715	1.30618217	1.36256659	1.37879973	1.38930612	1.43428062	1.45383677
1983	1984	1985	1986	1987	1988	1989	1990
1.48933593	1.46548054	1.53993640	1.46730707	1.53414003	1.51029759	1.50351470	1.45428528
1991	1992	1993	1994	1995	1996	1997	1998
1.46899342	1.51532482	1.52325188	1.54693335	1.54265398	1.52230154	1.56770266	1.54191575
1999	2000	2001	2002	2003			
1.58810844	1.66189411	1.63261850	1.63716592	1.52335749			

```
> ilc.apcM$itx
```

```
Time Series:
```

```
Start = 1854
```

```
End = 1964
```

```
Frequency = 1
```

1854	1855	1856	1857	1858	1859	1860	1861
0.0000000	0.0000000	0.0000000	-0.7857785	-0.9389982	-0.6745924	-0.9429274	-1.1295468
1862	1863	1864	1865	1866	1867	1868	1869
-1.1834753	-1.3654872	-1.4750560	-1.7051015	-1.6555361	-1.3565989	-1.5803838	-1.6411761
1870	1871	1872	1873	1874	1875	1876	1877
-1.4530241	-1.6076211	-1.6384319	-1.5358819	-1.5750117	-1.7936229	-1.5352488	-1.4260149
1878	1879	1880	1881	1882	1883	1884	1885
-1.4297348	-1.5526511	-1.5445816	-1.6336637	-1.3967068	-1.3700269	-1.3036928	-1.1911068
1886	1887	1888	1889	1890	1891	1892	1893
-1.1093342	-1.1475348	-1.1198157	-0.9929467	-0.9355475	-0.9144959	-0.9259696	-0.8832170
1894	1895	1896	1897	1898	1899	1900	1901
-0.7788794	-0.8056642	-0.7742002	-0.7084292	-0.7455432	-0.7175971	-0.6800120	-0.6865489
1902	1903	1904	1905	1906	1907	1908	1909
-0.6418411	-0.6381955	-0.6542311	-0.6631615	-0.6875211	-0.6590196	-0.7253572	-0.7232670
1910	1911	1912	1913	1914	1915	1916	1917
-0.7627860	-0.8266071	-0.8442115	-0.8894161	-0.9147037	-0.8698867	-0.9200493	-0.9117993
1918	1919	1920	1921	1922	1923	1924	1925
-0.9153020	-0.9213775	-0.8808207	-0.9157645	-0.8797183	-0.9759603	-0.9700132	-0.9851097
1926	1927	1928	1929	1930	1931	1932	1933
-1.0129443	-1.0660661	-1.0998287	-1.1446159	-1.1302849	-1.1318863	-1.1266522	-1.1879894
1934	1935	1936	1937	1938	1939	1940	1941
-1.2335260	-1.2654602	-1.3440108	-1.4604140	-1.5059575	-1.5548905	-1.5976113	-1.6024838
1942	1943	1944	1945	1946	1947	1948	1949
-1.5675117	-1.6066114	-1.5430802	-1.6465714	-1.6574457	-1.6636943	-1.6728852	-1.6923719
1950	1951	1952	1953	1954	1955	1956	1957
-1.5895166	-1.8970755	-1.6172577	-1.7099572	-1.4507323	-1.6097054	-1.5281791	-1.5287108
1958	1959	1960	1961	1962	1963	1964	
-1.8331760	-1.3740637	-1.8651192	-1.5480198	0.0000000	0.0000000	0.0000000	

So we have that the age-effect is essentially log-rates, the period effect *kt*, is constrained to 0 for the first level, and the cohort effect *itx* is constrained so that the outer 3 levels in

each end are 0 (this can be regulated by the `clip` argument to `lca.rh`.)

We can work out the deviances for the models fitted with `ilc`:

```
> names( ilc.apcM )
 [1] "label"      "age"        "year"       "male"       "ax"         "bx"         "kt"
 [8] "df"         "residuals" "fitted"     "varprop"    "y"          "mdev"       "model"
[15] "adjust"     "type"       "call"       "conv.iter"  "bx0"        "itx"

> devM <- with( ilc.apcM, cbind(df,mdev) )
> devF <- with( ilc.apcF, cbind(df,mdev) )
> dev <- cbind( devM[,1]*devM[,2], devM[,1],
+              devF[,1]*devF[,2], devF[,1] )
> rownames( dev ) <- c("Resi","Null")
> colnames( dev ) <- c( " Men-dev","df","Women-dev","df" )
> round( dev, 2 )

      Men-dev  df Women-dev  df
Resi  3368.45 2927  3467.58 2927
Null  4898.56 3009  4027.65 3009
```

Now the `demogr` objects `lcM` and `lcF` are based on a tabulation of the lung cancer data in `ltab`:

```
> str( ltab )
xtabs [1:2, 1:90, 1:61, 1:2] 0 0 0 0 0 0 0 0 0 0 0 ...
- attr(*, "dimnames")=List of 4
..$ sex: chr [1:2] "1" "2"
..$ A  : chr [1:90] "0" "1" "2" "3" ...
..$ P  : chr [1:61] "1943" "1944" "1945" "1946" ...
..$   : chr [1:2] "D" "Y"
- attr(*, "call")= language xtabs(formula = cbind(D, Y) ~ sex + A + P, data = lung)

> D <- data.frame(as.table(ltab[,40:90,,"D"]))
> Y <- data.frame(as.table(ltab[,40:90,,"Y"]))
> names(D)[4] <- "D"
> names(Y)[4] <- "Y"
> lc <- merge(D,Y)
> levels( lc$sex ) <- c("M","F")
> lc$C <- factor( as.numeric(as.character(lc$P)) -
+               as.numeric(as.character(lc$A)) )
> str( lc )

'data.frame':      6222 obs. of  6 variables:
 $ sex: Factor w/ 2 levels "M","F": 1 1 1 1 1 1 1 1 1 1 ...
 $ A  : Factor w/ 51 levels "39","40","41",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ P  : Factor w/ 61 levels "1943","1944",...: 1 2 3 4 5 6 7 8 9 10 ...
 $ D  : num  3 4 1 1 1 1 1 4 1 0 ...
 $ Y  : num  28488 29058 29406 29915 30701 ...
 $ C  : Factor w/ 111 levels "1854","1855",...: 51 52 53 54 55 56 57 58 59 60 ...
```

This should enable us to fit the APC-models that was also fitted with `ilc:lca.rh`. But first we do as they and alias the 6 outer parameters:

```
> nlevels(lc$C)
 [1] 111

> lc$Cr <- Relevel(lc$C,list("outer"=c(1:3,109:111)))
> nlevels(lc$A) +
+ nlevels(lc$P) +
+ nlevels(lc$Cr) - 2
```

```
[1] 216
> match( levels(lc$Cr), levels(lc$C) )
  [1] NA  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 [22] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
 [43] 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65
 [64] 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86
 [85] 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107
[106] 108

> glm.apcM <- glm( D ~ -1 + A + P + Cr,
+               offset = log(Y),
+               family = poisson,
+               data = subset( lc, sex=="M" ) )
> glm.apcF <- update( glm.apcM, data = subset( lc, sex=="F" ) )
> glm.apcMi <- update( glm.apcM, . ~ . +1 )
> glm.apcFi <- update( glm.apcF, . ~ . +1 )
```

The latter two models are just the same models with an intercept fitted in order to get the `null.deviance` as the deviance corresponding to a model with only an intercept, instead of a model with 0 intercept on the linear predictor scale (which is ludicrous).

We can therefore compare the deviances from these models with that from the function `ilc::lca.rh` function:

```
> gdev <- dev
> gdev[1,] <- c( glm.apcMi$dev,
+             glm.apcMi$df.res,
+             glm.apcFi$dev,
+             glm.apcFi$df.res )
> gdev[2,] <- c( glm.apcM>null.dev,
+             glm.apcMi$df.null,
+             glm.apcFi>null.dev,
+             glm.apcFi$df.null )
> round( rbind( dev, gdev ), 2 )
      Men-dev  df Women-dev  df
Resi    3368.45 2927   3467.58 2927
Null    4898.56 3009   4027.65 3009
Resi    3132.75 2895   3273.25 2895
Null 121165693.52 3110  40848.10 3110
```

It is not quite clear why we see these differences in the deviances.

We can then check graphically if it really is the same models we fitted:

```
> par( mfrow=c(1,3) )
> plot( as.numeric(gsub("A","",names(glm.apcM$coef)[1:51])),
+       glm.apcM$coef[1:51], type="l",lwd=3,
+       ylim=c(-12,-5) )
> with( ilc.apcM, lines( age, ax, type="l",lwd=3, col="blue" ) )
> plot( as.numeric(gsub("P","",names(glm.apcM$coef)[51+1:60])),
+       glm.apcM$coef[51+1:60], type="l",lwd=3,
+       ylim=c(0,2) )
> with( ilc.apcM, lines( year, kt, type="l",lwd=3, col="blue" ) )
> plot( as.numeric(gsub("Cr","",names(glm.apcM$coef)[-(1:111)])),
+       glm.apcM$coef[-(1:111)], type="l",lwd=3,
+       ylim=c(-12,-5)+9 )
> with( ilc.apcM, lines( as.numeric(names(itx)), itx, type="l",lwd=3, col="blue" ) )
```

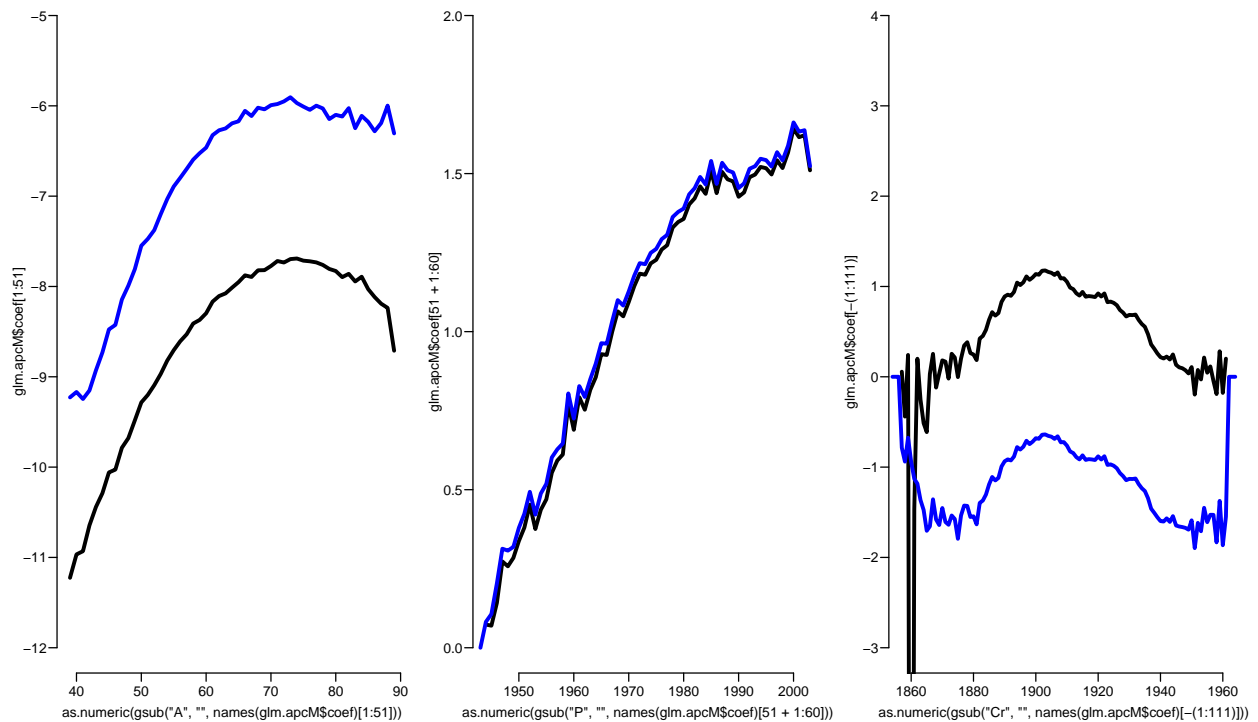



Figure 1.5: Comparison of *glm*-based parameters and *ilc*-based. Both sets have a priori constrained the three outer cohort parameters in both ends and the first period parameter to be 0.

../graph/LCa-apc-compM

```
> par( mfrow=c(1,3) )
> plot( as.numeric(gsub("A","",names(glm.apcF$coef)[1:51])),
+       glm.apcF$coef[1:51],
+       type="l",lwd=3, ylim=c(-30,0) )
> with( ilc.apcF, lines( age, ax, type="l",lwd=3, col="blue" ) )
> plot( as.numeric(gsub("P","",names(glm.apcF$coef)[51+1:60])),
+       glm.apcF$coef[51+1:60], type="l",lwd=3,
+       ylim=c(-15,15) )
> with( ilc.apcF, lines( year, kt, type="l",lwd=3, col="blue" ) )
> plot( as.numeric(gsub("Cr","",names(glm.apcF$coef)[-1:111])),
+       glm.apcF$coef[-1:111], type="l",lwd=3,
+       ylim=c(-10,20) )
> with( ilc.apcF, lines( as.numeric(names(itx)), itx, type="l",lwd=3, col="blue" ) )
```

We can see the differences in the average values of the parameters:

```
> # Men
> c(
+ mean( glm.apcM$coef[1:51]-ilc.apcM$ax ),
+ mean( glm.apcM$coef[51+1:60]-ilc.apcM$kt[-1] ),
+ mean( glm.apcM$coef[111+1:105]-ilc.apcM$itx[3+1:105] ) )
[1] -1.77820771 -0.03293757  1.56727869

> # Women
> c(
+ mean( glm.apcF$coef[1:51]-ilc.apcF$ax ),
+ mean( glm.apcF$coef[51+1:60]-ilc.apcF$kt[-1] ),
+ mean( glm.apcF$coef[111+1:105]-ilc.apcF$itx[3+1:105] ) )
```

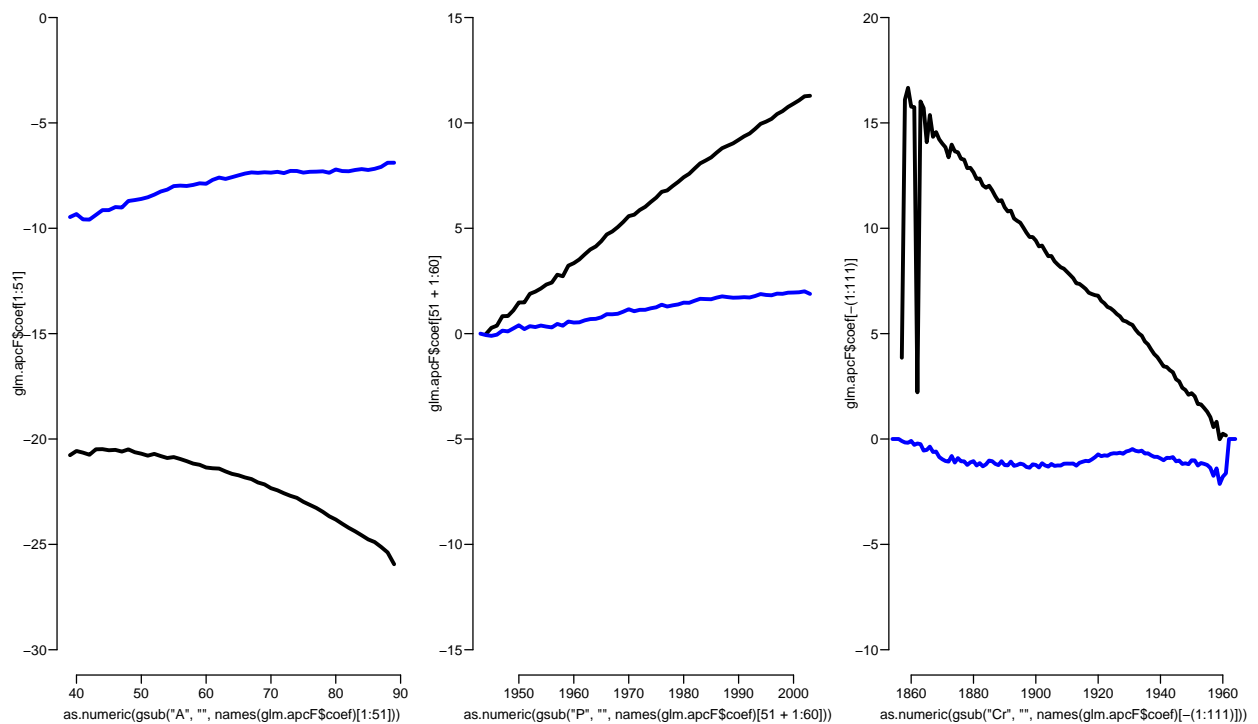


Figure 1.6: Comparison of *glm*-based parameters and *lc*-based for women. Both sets have a priori constrained the three outer cohort parameters in both ends and the first period parameter to be 0.

../graph/LCa-apc-compF

```
[1] -14.238895  4.886795  9.005737
> # Women - excluding wild paramters
> c(
+ mean( glm.apcF$coef[1:51]-ilc.apcF$ax)[5:45] ),
+ mean( glm.apcF$coef[51+1:60]-ilc.apcF$kt[-1])[5:54] ),
+ mean( glm.apcF$coef[111+1:105]-ilc.apcF$itx[3+1:105])[10:95] ) )
[1] -13.980974  4.749200  9.341182
```

The degrees of freedom:

```
> glm.df <- dev*0
> glm.df[,1] <- c(glm.apcMi$devi,glm.apcMi$null)
> glm.df[,2] <- c(glm.apcMi$df.r,glm.apcMi$df.n)
> glm.df[,3] <- c(glm.apcFi$devi,glm.apcFi$null)
> glm.df[,4] <- c(glm.apcFi$df.r,glm.apcFi$df.n)
> round( dev, 2 )
      Men-dev  df Women-dev  df
Resi  3368.45 2927  3467.58 2927
Null  4898.56 3009  4027.65 3009
> round( glm.df, 2 )
      Men-dev  df Women-dev  df
Resi  3132.75 2895  3273.25 2895
Null  95189.69 3110  40848.10 3110
```

Apparently the deviance calculations in *ilc* are a bit different from the standard ones — there are 3111 observations:

```
> with(lcM, prod( dim(rate$male) ) )
[1] 3111
> nrow(lc)
[1] 6222
```

and the number of parameters fitted is

```
> length(ilc.apcM$ax) +
+ length(ilc.apcM$kt) - 1 +
+ length(ilc.apcM$itx) - 6
[1] 216
```

which is exactly the no. of observations minus the `df.res` from the `glm` fitted model. The `ilc` estimates the same number of parameters, but produces 32 more d.f.

Thus the broad conclusion is that the the `ilc` produces roughly the same fitted values as do the traditional `glm` modeling, but possibly there is some unaccounted differences in the fitting.

1.8 Iterative fitting of extended Lee-Carter spline models

The general fitting algorithm does not (and indeed should not) rely on a particular tabulation of data; only **A** and **P** are needed as quantitative qualifiers for the response variable (**D**, **Y**) — basically, both the APC-model and the Lee-Carter model are extensions of the simple age-period model with an interaction term, just different choices of interactions.

First we transform to the correct means in the observation cells:

```
> lung$up <- with(lung,P-A-C)
> lung <- transform( lung, Ax = A+(1+up)/3,
+                   Px = P+(2-up)/3,
+                   sex = factor( sex, labels=c("M","F") ),
+                   Y = Y/1000 )
> Mlc <- subset( lung, sex=="M" & Ax>40 )
> Flc <- subset( lung, sex=="F" & Ax>40 )
```

Note that we also made separate subsets of male and female lung cancer

Before fitting we need a reference age and a reference date which we will use to center the components of the age-period interaction:

```
> a.ref <- 70
> p.ref <- 1980
```

Now fit the age-period model with natural splines; first defining the spline knots for the main age effect, the interaction age effect and the period effect:

```
> a.kn <- seq(10,85,,7)
> i.kn <- seq(20,80,,6)
> p.kn <- seq(1945,2004,,6)
> map <- glm( D ~ -1 + Ns( Ax, knots=a.kn, i=T ) +
+           Ns( Px, knots=p.kn, ref=p.ref),
+           offset = log(Y/100),
```

```

+           family = poisson,
+           data = Mlc )
> pap <- predict( map, type="terms" )
> str(pap)
num [1:6100, 1:2] 2.44 2.38 2.44 2.38 2.44 ...
- attr(*, "dimnames")=List of 2
..$ : chr [1:6100] "4881" "4882" "4883" "4884" ...
..$ : chr [1:2] "Ns(Ax, knots = a.kn, i = T)" "Ns(Px, knots = p.kn, ref = p.ref)"
- attr(*, "constant")= num 0
> colnames(pap) <- substr(colnames(pap),4,5)
> head( pap )
      Ax      Px
4881 2.441233 -1.847395
4882 2.383309 -1.822880
4883 2.441233 -1.773850
4884 2.383309 -1.749335
4885 2.441233 -1.700306
4886 2.383309 -1.675797

```

So now we have the predictions for the two terms in `pop` — the first column is the age-effect *including* the intercept, the other one is the period effect constrained to 0 at `p.ref`, both shown in figure 1.7 transformed to the rate- resp. RR scale.

Now we mimic the Lee-Carter extension of the AP-model by using the period-RR as fixed regression variable and compensate for the main effect of age by adding this to the offset.

```

> mx <- glm( D ~ -1 + Ns(Ax,knot=i.kn,i=T):pap[,2],
+           offset = pap[,1]+log(Y/100),
+           family = poisson,
+           data = Mlc )
> # slightly more elegant
> ma <- update( map, . ~ -1 + Ns(Ax,knot=i.kn,i=T):pap[,2],
+             offset = pap[,1]+log(Y/100) )
> round( c( map$deviance, ma$deviance, mx$deviance ), 3 )
[1] 10776.738 9260.281 9260.281

```

So we see that there actually *is* a decrease in the deviance by expanding the model *ad hoc*. The parameters obtained so far are only a first approximation to the ML-estimates but we extract the estimated age-interaction; this is simplest done by taking the prediction and dividing by the covariate. We also scale the age-component to be 1 at `a.ref`, and put it in the dataset:

```

> CA <- Ns( a.ref, knots=i.kn, i=T )
> Mlc$ba <- predict( ma, type="terms" ) / pap[,2] /
+   ci.lin( ma, ctr.mat=CA )[,1]

```

Before we plot the estimated effects we extract unique copies of them:

```

> aa <- sort(unique(Mlc$Ax))
> pp <- sort(unique(Mlc$Px))
> fa0 <- exp(pap[match(aa,Mlc$Ax),1])
> gp0 <- exp(pap[match(pp,Mlc$Px),2])
> ba1 <- Mlc$ba[match(aa,Mlc$Ax)]
> par( mfrow=c(1,3) )
> plot( aa, fa0, log="y", type="l", lwd=4,
+       ylab="Incidence rates per 100,000 PY" )

```

```

> plot( pp, gp0, log="y", type="l", lwd=4,
+       ylab=paste("Rate ratio relative to", p.ref) )
> abline( h=1, v=p.ref )
> plot( aa, ba1, type="l", lwd=4,
+       ylab="Relative log-RR" )
> abline( h=0:1, v=a.ref )

```

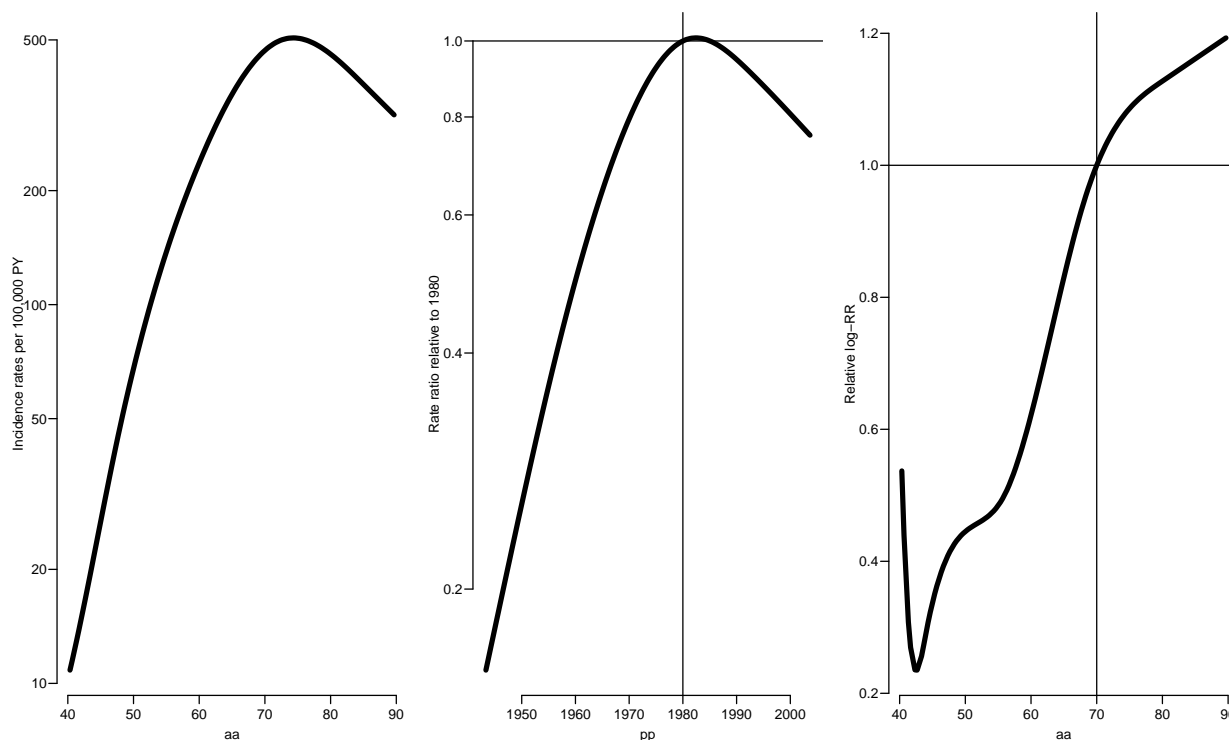


Figure 1.7: Age- and Period model estimates, and the first approximation to the age-interaction term.

../graph/LCa-AP-mod

Then we re-fit the age-period model with the scaled age interaction, extracted into `ba`, as covariate for the period:

```

> mapi <- glm( D ~ -1 + Ns( Ax, knot=a.kn, i=T )
+             + Ns( Px, knot=p.kn, ref=p.ref):ba,
+             offset = log(Y/100),
+             family = poisson,
+             data = Mlc )
> round( c( map$deviance, ma$deviance, mapi$deviance ), 2 )
[1] 10776.74 9260.28 8671.30

```

So we see we got a further reduction of the deviance.

Again, we cannot use the predict machinery because we do not want the predicted effects, but we can just as before remove the `ba` from the second term by division and get the estimated period effect. Note that since the (log-)period effect is 0 at `p.ref` this constraint is conserved by the division:

```

> papi <- predict( mapi, type="terms" ) / cbind(1,Mlc$ba)
> colnames(papi) <- substr(colnames(papi),4,5)
> head( papi )

```

```

      Ax      Px
4881 2.274268 -2.460429
4882 2.217565 -2.426770
4883 2.274268 -2.359454
4884 2.217565 -2.325795
4885 2.274268 -2.258480
4886 2.217565 -2.224831

```

From now on it get repetitive; we just repeat the two last steps again

```

> devs <- NArray( list( iter = 1:20,
+                       devi = c("AP","A") ) )
> for( i in 1:20 )
+ {
+ ma <- glm( D ~ -1 + Ns( Ax, knots=i.kn, i=T ):papi[,2],
+           offset = papi[,1]+log(Y/100),
+           family = poisson,
+           data = Mlc )
+ Mlc$ba <- predict( ma, type="terms" ) / papi[,2] / ci.lin( ma, ctr.mat=CA )[,1]
+ mapi <- glm( D ~ -1 + Ns( Ax, knot=a.kn, i=T )
+            + Ns( Px, knot=p.kn, ref=p.ref ):ba,
+            offset = log(Y/100),
+            family = poisson,
+            data = Mlc )
+ papi <- predict( mapi, type="terms" ) / cbind( 1, Mlc$ba )
+ devs[i,] <- c( ma$deviance, mapi$deviance )
+ cat( "Iteration ", i, round( devs[i,], 4 ), "\n" )
+ }
Iteration 1 8436.693 8354.576
Iteration 2 8324.386 8314.013
Iteration 3 8310.367 8309.116
Iteration 4 8308.682 8308.533
Iteration 5 8308.482 8308.464
Iteration 6 8308.458 8308.456
Iteration 7 8308.455 8308.455
Iteration 8 8308.455 8308.455
Iteration 9 8308.455 8308.455
Iteration 10 8308.455 8308.455
Iteration 11 8308.455 8308.455
Iteration 12 8308.455 8308.455
Iteration 13 8308.455 8308.455
Iteration 14 8308.455 8308.455
Iteration 15 8308.455 8308.455
Iteration 16 8308.455 8308.455
Iteration 17 8308.455 8308.455
Iteration 18 8308.455 8308.455
Iteration 19 8308.455 8308.455
Iteration 20 8308.455 8308.455
> round( apply(devs,2,diff), 4 )
      devi
      AP      A
2 -112.3077 -40.5629
3 -14.0192 -4.8973
4 -1.6844 -0.5823
5 -0.2004 -0.0691
6 -0.0238 -0.0082
7 -0.0028 -0.0010

```

```

8      -0.0003  -0.0001
9       0.0000   0.0000
10      0.0000   0.0000
11      0.0000   0.0000
12      0.0000   0.0000
13      0.0000   0.0000
14      0.0000   0.0000
15      0.0000   0.0000
16      0.0000   0.0000
17      0.0000   0.0000
18      0.0000   0.0000
19      0.0000   0.0000
20      0.0000   0.0000

```

After convergence we can extract the parameters as before and compare with the initial ones:

```

> fax <- exp(papi[match(aa,Mlc$Ax),1])
> gpx <- exp(papi[match(pp,Mlc$Px),2])
> bax <- Mlc$ba[match(aa,Mlc$Ax)]
> par( mfrow=c(1,3) )
> matplot( aa, cbind(fa0,fax), log="y", type="l", lwd=4, lty=1,
+         col=gray(c(0.7,0)), ylab="Incidence rates per 100,000 PY" )
> matplot( pp, cbind(gp0,gpx), log="y", type="l", lwd=4, lty=1,
+         col=gray(c(0.7,0)), ylab=paste("Rate ratio relative to", p.ref) )
> abline( h=1, v=p.ref )
> matplot( aa, cbind(ba1,bax), type="l", lwd=4, lty=1,
+         col=gray(c(0.7,0)), ylab="Relative log-RR" )
> abline( h=0:1, v=a.ref )

```

1.8.1 Rescaling effects to compare with ilc and demography estimates

We rescaled the effects of the estimates for men and women and we can now compare with the estimated effects from the smoothed Lee-Carter model, as implemented in the `LCa.fit` function in the `Epi` package. First we re-read the data and groom them to the relevant form:

```

> lung$up <- with( lung, P-A-C )
> lung <- transform( lung, Ax = A+(1+up)/3,
+                  Px = P+(2-up)/3,
+                  sex = factor( sex, labels=c("M","F") ),
+                  Y = Y/1000 )
> Mlc <- subset( lung, sex=="M" & Ax>40, select=c("Ax","Px","D","Y") )
> Flc <- subset( lung, sex=="F" & Ax>40, select=c("Ax","Px","D","Y") )
> names( Mlc )[1:2] <- names( Flc )[1:2] <- c("A","P")
> head( Mlc )

```

	A	P	D	Y
4881	40.66667	1943.333	1	0.0140535
4882	40.33333	1943.667	1	0.0140985
4883	40.66667	1944.333	0	0.0140668
4884	40.33333	1944.667	2	0.0143337
4885	40.66667	1945.333	0	0.0142978
4886	40.33333	1945.667	0	0.0145865

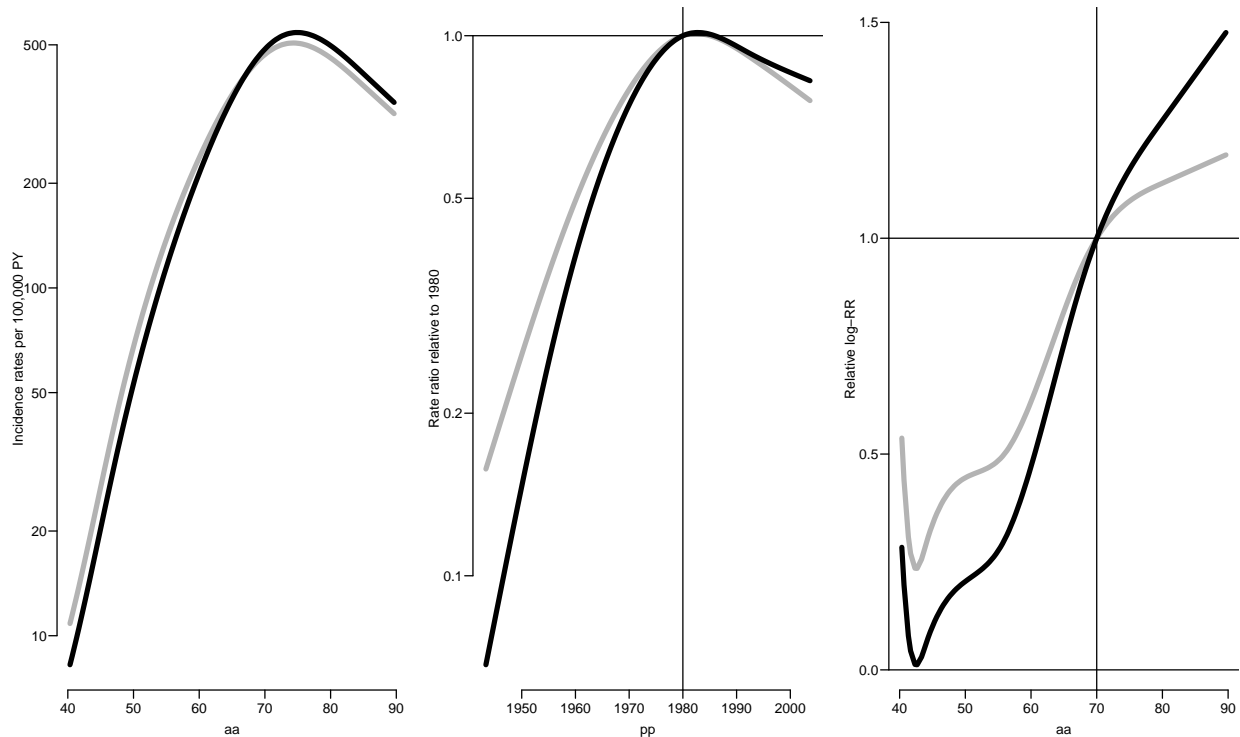


Figure 1.8: *Parameter estimates from the Lee-Carter model, using splines for the effects. The black lines are the final estimates; the gray the starting set of estimates...*/graph/LCa-LC-mod

```
> head( Flc )
      A      P D      Y
15861 40.66667 1943.333 0 0.0146313
15862 40.33333 1943.667 1 0.0144880
15863 40.66667 1944.333 0 0.0144577
15864 40.33333 1944.667 1 0.0150110
15865 40.66667 1945.333 0 0.0149128
15866 40.33333 1945.667 0 0.0149468

> system.time( LCa.Mlc <- LCa.fit( Mlc, a.ref=70, p.ref=1980, VC=FALSE ) )
LCa.fit convergence in 8 iterations, deviance: 8417.157 on 6084 d.f.
  user system elapsed
 8.979 10.976  5.840

> system.time( LCa.Flc <- LCa.fit( Flc, a.ref=70, p.ref=1980, VC=FALSE ) )
LCa.fit convergence in 5 iterations, deviance: 7657.562 on 6084 d.f.
  user system elapsed
 5.546  6.696  3.566
```

We can then superpose these effects on the corresponding estimates from `ilc` and `demography` packages:

```
> # pdf( "../graph/LCa-cmp-LCc.pdf", width=10 )
> par( mfcol=c(1,3),mar=c(3,3,1,1),mgp=c(3,1,0)/1.6,las=1,bty="n")
> matplot( dmg.lcM$age, exp(cbind(dmg.lcM$ax+dmg.lcM$bx*dMk0,
+                               dmg.lcF$ax+dmg.lcF$bx*dFk0))*1000,
+         log="y", ylab="Lung cancer incidence rates per 1000 PY",
+         xlab="Age", col=c("blue","red"), type="l", lty=1, lwd=1 )
> matlines( ilc.lcM$age, exp(cbind(ilc.lcM$ax+ilc.lcM$bx*iMk0,
```



```

+                               ilc.lcF$ax+ilc.lcF$bx*iFk0))*1000,
+                               col=c("blue","red"), lty=1, lwd=2 )
> matlines( as.numeric(rownames(LCa.Mlc$ax)), cbind(LCa.Mlc$ax[,1],LCa.Flc$ax[,1])/1000,
+           col=c("blue","red"), lty="11", lwd=4, lend="butt" )
> abline( v=70 )
> matplot( dmg.lcM$age, cbind(dmg.lcM$bx/dMb0,dmg.lcF$bx/dFb0),
+         ylab="RR Age effect (power)", ylim=c(-0.2,1.5),
+         xlab="Age", col=c("blue","red"), type="l", lty=1, lwd=1 )
> matlines( ilc.lcM$age, cbind(ilc.lcM$bx/iMb0,ilc.lcF$bx/iFb0),
+           col=c("blue","red"), lty=1, lwd=2 )
> matlines( as.numeric(rownames(LCa.Mlc$pi)), cbind(LCa.Mlc$pi[,1],LCa.Flc$pi[,1]),
+           col=c("blue","red"), lty="11", lwd=4, lend="butt" )
> abline( v=70, h=1 )
> matplot( dmg.lcM$year, exp(cbind((dmg.lcM$kt-dMk0)*dMb0,
+                                 (dmg.lcF$kt-dFk0)*dFb0)),
+         ylab="Time effect (RR)", log="y",
+         xlab="Date", col=c("blue","red"), type="l", lty=1, lwd=1 )
> matlines( ilc.lcM$year, exp(cbind((ilc.lcM$kt-iMk0)*iMb0,
+                                 (ilc.lcF$kt-iFk0)*iFb0)),
+           col=c("blue","red"), lty=1, lwd=2 )
> matlines( as.numeric(rownames(LCa.Mlc$kp)), cbind(LCa.Mlc$kp[,1],LCa.Flc$kp[,1]),
+           col=c("blue","red"), lty="11", lwd=4, lend="butt" )
> abline( h=1, v=1980 )
> # dev.off()

```

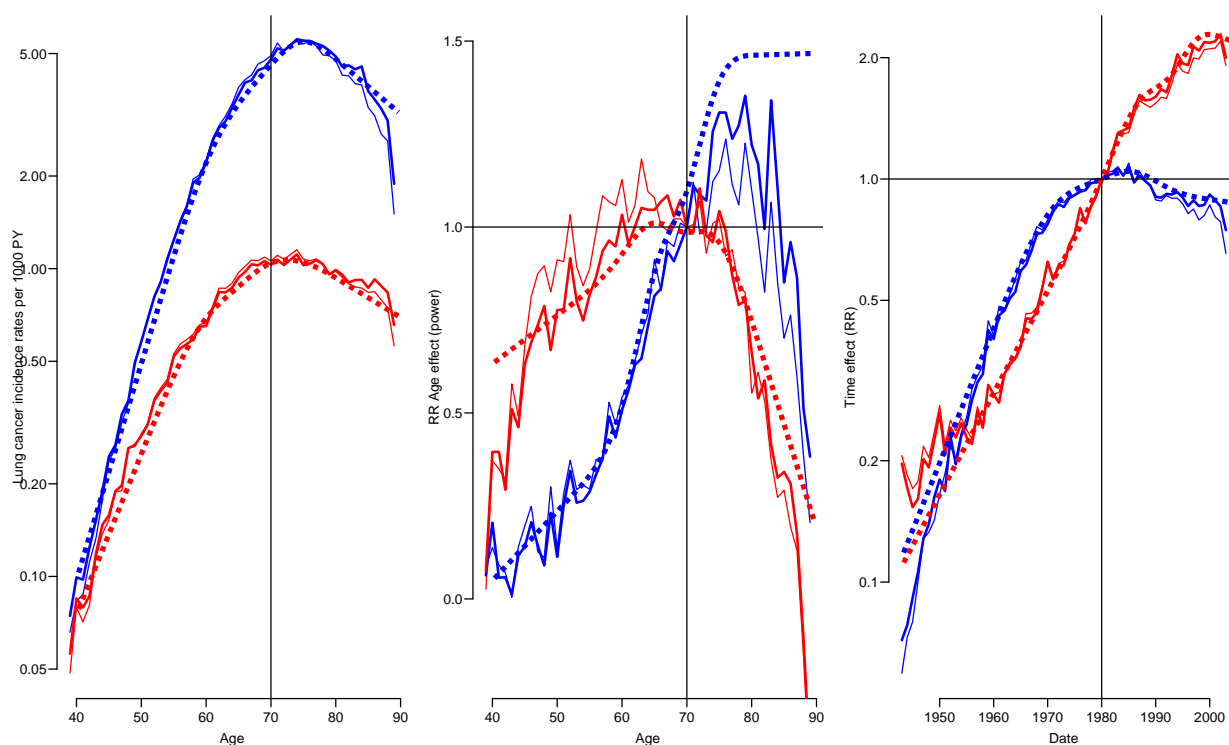


Figure 1.9: Rescaled parameters from the Lee-Carter model for Danish lung cancer incidence data (men: blue; women:red) by the `demography::lca` function (thin lines) and the `ilc::lca.rh` (thick lines). The smooth version from `Epi:LCa.fit` is superposed as a dotted line. We see that the two approaches gives pretty much the same results.../graph/LCa-cmp-LCc

The deviances from the two models can be compared to the deviance of the model
 HER

```
> system.time( LCa.Mlcc <- LCa.fit( Mlc, a.ref=70, p.ref=1980, VC=FALSE,
+                               model="ACa", maxit=500 ) )
```

```
LCa.fit convergence in 8 iterations, deviance: 7975.153 on 6084 d.f.
  user system elapsed
 8.549 10.228  5.294
```

```
> system.time( LCa.Flcc <- LCa.fit( Flc, a.ref=70, p.ref=1980, VC=FALSE,
+                               model="ACa", maxit=500 ) )
```

```
LCa.fit convergence in 19 iterations, deviance: 7241.101 on 6084 d.f.
  user system elapsed
21.090 25.281 13.202
```

```
> Mapc <- apc.fit( Mlc, npar=6 )
```

NOTE: npar is specified as:A P C

6 6 6

```
[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	Pr(>Chi)
Age	6094	23617.8			
Age-drift	6093	16827.5	1	6790.3	< 2.2e-16
Age-Cohort	6089	8450.5	4	8377.0	< 2.2e-16
Age-Period-Cohort	6085	7654.9	4	795.6	< 2.2e-16
Age-Period	6089	10872.6	-4	-3217.7	< 2.2e-16
Age-drift	6093	16827.5	-4	-5954.9	< 2.2e-16

No reference period given:

Reference period for age-effects is chosen as

the median date of birth for persons with event: 1914.333 .

```
> Fapc <- apc.fit( Flc, npar=6 )
```

NOTE: npar is specified as:A P C

6 6 6

```
[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	Pr(>Chi)
Age	6094	24355.6			
Age-drift	6093	8520.6	1	15835.0	< 2.2e-16
Age-Cohort	6089	7639.2	4	881.3	< 2.2e-16
Age-Period-Cohort	6085	7142.5	4	496.7	< 2.2e-16
Age-Period	6089	8003.9	-4	-861.4	< 2.2e-16
Age-drift	6093	8520.6	-4	-516.6	< 2.2e-16

No reference period given:

Reference period for age-effects is chosen as

the median date of birth for persons with event: 1922.333 .

```
> DF <- cbind(Mapc$Anova[3:5,2,drop=F],Fapc$Anova[3:5,2,drop=F])
```

```
> DF <- DF[c(1,1,2,3,3),]
```

```
> DF[2,] <- c( LCa.Mlcc$deviance, LCa.Flcc$deviance)
```

```
> DF[4,] <- c( LCa.Mlc $deviance, LCa.Flc $deviance)
```

```
> colnames( DF ) <- c("Men","Women")
```

```
> rownames( DF )[c(2,4)] <- c("Lee-Carter-Cohort","Lee-Carter-Period")
```

```
> round( DF, 1 )
```

	Men	Women
Age-Cohort	8450.5	7639.2
Lee-Carter-Cohort	7975.2	7241.1
Age-Period-Cohort	7654.9	7142.5
Lee-Carter-Period	8417.2	7657.6
Age-Period.1	10872.6	8003.9

1.9 The LCa function with smooth effects

This has all been wrapped in a function, `LCa.fit` in the `Epi` package: The arguments to the function is basically as for the `apc.fit` function, just requiring a dataset with the relevant variables, a specification of the knots (with suitable defaults).

```
> LCa.fit
```

```
function( data, A, P, D, Y,
  model = "APa",      # or one of "ACa", "APaC", "APCa" or "APaCa"
  a.ref,              # age reference for the interactions
  pi.ref = a.ref,     # age reference for the period interaction
  ci.ref = a.ref,     # age reference for the cohort interaction
  p.ref,              # period reference for the interaction
  c.ref,              # cohort reference for the interactions
  npar = c(a = 6,     # no. knots for main age-effect
            p = 6,     # no. knots for period-effect
            c = 6,     # no. knots for cohort-effect
            pi = 6,    # no. knots for age in the period interaction
            ci = 6),  # no. knots for age in the cohort interaction
  VC = TRUE,         # numerical calculation of the Hessian?
  alpha = 0.05,      # 1 minus confidence level
  eps = 1e-6,        # convergence criterion
  maxit = 100,       # max. no iterations
  quiet = TRUE )    # cut the crap
{
# "model" must have values in c(APa/ACa/APaC/APCa/APaCa)?
if( !(model %in% c("APa","ACa","APaC","APCa","APaCa")) )
  stop( "'model' must be one of 'APa','ACa','APaC','APCa','APaCa', but is', model,'\n' )

# Which main effects and interactions are in the model
intP <- as.logical(length(grep("Pa",model)))
intC <- as.logical(length(grep("Ca",model)))
mainP <- as.logical(length(grep("P" ,model))) # Also includes the age-period interaction
mainC <- as.logical(length(grep("C" ,model))) # Also includes the age-cohort product

# if a dataframe is supplied, fish out data and put in the function's environment
if( !missing(data) )
  {
  if (length(match(c("A", "P", "D", "Y"), names(data))) != 4)
    stop("Data frame ", deparse(substitute(data)),
         " has columns:\n", names(data),
         "\nmust have variables:\n", "A (age), P (period), D (cases) and Y (person-time)")
  data <- data[,c("A","P","D","Y")]
  data <- data[complete.cases(data),]
  A <- data$A
  P <- data$P
  D <- data$D
  Y <- data$Y
  } else { # if single vectors supplied, check they are all there
  nm <- c(missing(A),
          missing(P),
          missing(D),
          missing(Y))
  if (any(nm))
    stop("Variable", if (sum(nm) > 1)
         "s", paste(c(" A", " P", " D", " Y")[nm], collapse = ","),
         " missing from input")
}
```

```

# and that they have the same length
if( diff(range( lv <- c( length(A),
                      length(P),
                      length(D),
                      length(Y) ) )) != 0 )
  stop( "\nLengths of variables (", paste(paste(names(lv),
      lv, sep = ":"), collapse = ", ", ") are not the same." )
} # end of data acquisition

# code-simplifier for knot calculation
eqqnt <- function(n) round( (1:n-0.5)/n, 2 )
# Define knots - we compute also the knots not needed
if( is.list(npar) ) {
  # Check if names is a named list
  if( is.null(names(npar)) ) stop( "If npar= is a list, it must be a *named* list.\n" )
  a.kn <- if( length(npar$a )>1 ) npar$a else quantile( rep( A,D), probs=eqqnt(npar$a ) )
  p.kn <- if( length(npar$p )>1 ) npar$p else quantile( rep( P ,D), probs=eqqnt(npar$p ) )
  c.kn <- if( length(npar$c )>1 ) npar$c else quantile( rep( P-A,D), probs=eqqnt(npar$c ) )
  pi.kn <- if( length(npar$pi)>1 ) npar$pi else quantile( rep( A,D), probs=eqqnt(npar$pi) )
  ci.kn <- if( length(npar$ci)>1 ) npar$ci else quantile( rep( A,D), probs=eqqnt(npar$ci) )
}
else { # if npar is too short fill it up
  npar <- rep( npar, 5 )[1:5]
  # if not named, name it and notify
  if( is.null(names(npar)) ) names(npar) <- c("a","p","c","pi","ci")
  a.kn <- quantile( rep( A,D), probs=eqqnt(npar["a"] ) )
  p.kn <- quantile( rep( P ,D), probs=eqqnt(npar["p"] ) )
  c.kn <- quantile( rep( P-A,D), probs=eqqnt(npar["c"] ) )
  pi.kn <- quantile( rep( A,D), probs=eqqnt(npar["pi"]) )
  ci.kn <- quantile( rep( A,D), probs=eqqnt(npar["ci"]) )
}

# Reference points
if( missing( p.ref ) ) p.ref <- median( rep( P ,D) )
if( missing( c.ref ) ) c.ref <- median( rep( P-A,D) )
if( missing( pi.ref ) ) pi.ref <- median( rep( A,D) )
if( missing( ci.ref ) ) ci.ref <- median( rep( A,D) )

#####
# Here starts the actual modelling
commence <- Sys.time()

# Matrices to extract the age-interaction terms at reference points
Ap <- Ns( rep(pi.ref,length(A)), knots=pi.kn, intercept=TRUE )
Ac <- Ns( rep(ci.ref,length(A)), knots=ci.kn, intercept=TRUE )

# Current age-effects (in the iteration these will be term predictions)
ba <- matrix( 1, length(A), 2 ) # cbind( rep(1,length(A)), 1 )

# set to 0 if term is not in model at all
if( !mainP ) ba[,1] <- 0
if( !mainC ) ba[,2] <- 0
# Main effects model with (at least one) age-interaction
# --- at this stage it is either 0 or 1
mat <- glm( D ~ -1 + Ns( A, knots=a.kn, intercept=TRUE ) +
           Ns( P , knots=p.kn, ref=p.ref):ba[,1] +
           Ns( P-A, knots=c.kn, ref=c.ref):ba[,2],

```

```

        offset = log(Y),
        family = poisson )
oldmb <- oldmat <- mat$deviance

# Terms prediction --- three terms here.
# No need to divide by the ba at this point, it is either 1 or 0
pat <- predict( mat, type="terms" )

# iteration counter and continuation indicator
nit <- 0
one.more <- TRUE

# For simple formatting of the iteration output
fC <- function(x,d) formatC(x,format="f",digits=d)

# now, iterate till convergence
while( one.more )
{
  nit <- nit+1

# The estimated terms from the modeling of the APC-effects to be used
# as offsets
  Aoff <- pat[,1]
  Pint <- Poff <- pat[,2]
  Cint <- Coff <- pat[,3]
# P or C terms with main effects should be either in interaction or
# offset, so one of these should always be 0
  if( intP ) Poff <- Poff*0 else Pint <- Pint*0
  if( intC ) Coff <- Coff*0 else Cint <- Cint*0
# Iteration of the age-components of the interaction
  mb <- glm( D ~ -1 + Ns( A, knots=pi.kn, intercept=TRUE ):Pint +
            Ns( A, knots=ci.kn, intercept=TRUE ):Cint,
            offset = Aoff + Poff + Coff + log(Y),
            family = poisson )

# Get the age-interaction terms only, and if one is not needed set to 0
  ba <- predict( mb, type="terms" ) /
        cbind(Pint,Cint) /
        cbind( ci.lin( mb, subset="pi.kn", ctr.mat=Ap)[,1], # These are the values at the re
              ci.lin( mb, subset="ci.kn", ctr.mat=Ac)[,1] ) # point for A; we want the RRs a
  ba[is.na(ba)] <- 0

# If no interaction only main should be fitted; if no main effect, set to 0 using mainP/C
  if( !intP ) ba[,1] <- rep(1,length(A)) * mainP
  if( !intC ) ba[,2] <- rep(1,length(A)) * mainC
# apc model with assumed known interactions with age
  mat <- glm( D ~ -1 + Ns( A, knots=a.kn, intercept=TRUE ) +
            Ns( P , knots=p.kn, ref=p.ref):ba[,1] +
            Ns( P-A, knots=c.kn, ref=c.ref):ba[,2],
            offset = log(Y),
            family = poisson )

# extract age and period terms - removing the interactions
  pat <- predict( mat, type="terms" ) / cbind( 1, ba )
  pat[is.na(pat)] <- 0

# convergence? Check both that the two models give the same deviance

```

```

# and that the chnage in each is small
newmat <- mat$deviance
newmb <- mb$deviance
conv <- ( reldif <- max( (abs(newmat-newmb)/(newmat+newmb)/2),
                      (oldmat-newmat)/newmat,
                      (oldmb -newmb )/newmb ) ) < eps
one.more <- ( !conv & ( nit < maxit ) )
oldmat <- newmat
oldmb <- newmb
if( !quiet & nit==1 )
  cat( "    Deviances: model(AT) model(A) Rel. diff.\n" )
if( !quiet ) cat( "Iteration", formatC( nit, width=3, flag=" "), "",
                fC(mat$deviance,3),
                fC( mb$deviance,3),
                fC( reldif, 7 ), "\n" )
} # end of iteration loop

# Deviance and d.f - there is a "+1" because the intercept is in both models
# but not explicit, (both models fitted with "-1"), hence the df.null
# is the total no. observations
dev <- mb$deviance
df <- mat$df.null - ( mb$df.null- mb$df.res # no. parms in mb
                   + mat$df.null-mat$df.res # no. parms in mat
                   - 1 )                 # common intercept
if( conv ) cat( "LCA.fit convergence in ", nit,
               " iterations, deviance:", dev, "on", df, "d.f.\n")
if( !conv ) cat( "LCA.fit *not* converged in ", nit,
                " iterations:\ndeviceance (AT):", mat$deviance,
                ", deviance (B):" , mb$deviance, "\n",
                if( VC ) "...no variance-covariance computed.\n" )
fin <- Sys.time()
if( !quiet ) cat("...using", round(difftime(fin,commence,units="secs"),1), "seconds.\n")

# unique values of A, P and C in the dataset for reporting effects
a.pt <- sort(unique( A))
p.pt <- sort(unique(P ))
c.pt <- sort(unique(P-A))

# extract effects from final models after convergence
ax <- ci.exp( mat, subset= "a.kn", ctr.mat=Ns(a.pt,knots= a.kn,intercept=TRUE ) )
kp <- ci.exp( mat, subset= "p.kn", ctr.mat=Ns(p.pt,knots= p.kn,ref=p.ref ) )
kc <- ci.exp( mat, subset= "c.kn", ctr.mat=Ns(c.pt,knots= c.kn,ref=c.ref ) )
pi <- ci.exp( mb , subset="pi.kn", ctr.mat=Ns(a.pt,knots=pi.kn,intercept=TRUE), Exp=FALSE )
ci <- ci.exp( mb , subset="ci.kn", ctr.mat=Ns(a.pt,knots=ci.kn,intercept=TRUE), Exp=FALSE )

# Label the estimated effects
rownames( ax ) <-
rownames( pi ) <-
rownames( ci ) <- a.pt
rownames( kp ) <- p.pt
rownames( kc ) <- c.pt

# do we bother about the correct variance-covariance?
if( VC & conv ) # ...certainly not without convergence
{
  commence <- Sys.time()
  if( !quiet ) cat("...computing Hessian by numerical differentiation...\n")

```

```

# the number of parameters for each of the 5 effects
na <- length( grep( "a.kn", names(coef(mat)) ) )
np <- length( grep( "p.kn", names(coef(mat)) ) )
nc <- length( grep( "c.kn", names(coef(mat)) ) )
npi <- length( grep( "pi.kn", names(coef(mb )) ) )
nci <- length( grep( "ci.kn", names(coef(mb )) ) )

# get only the parameters for effects that are non-zero (the others
# are in the models but they are 0)
ml.cf <- c( coef(mat)[c(rep(TRUE,na),
                       rep(mainP,np),
                       rep(mainC,nc))],
           coef(mb)[c(rep(intP,npi),
                      rep(intC,nci))] )

# and some more snappy names for the parameters: first all names
all.nam <- c( paste("ax",1:na,sep=""),
             paste("kp",1:np,sep=""),
             paste("kc",1:nc,sep=""),
             paste("pi",1:npi,sep=""),
             paste("ci",1:nci,sep="") )

# ...then those actually present in the model
names( ml.cf ) <- all.nam[c(rep( TRUE,na),
                           rep(mainP,np),
                           rep(mainC,nc),
                           rep(intP,npi),
                           rep(intC,nci))]

# We need the variance-covariance of the estimates as the 2nd
# derivative of the log-likelihood, D*log(lambda) - lambda*Y,
# or for eta=log(lambda), D*eta - exp(eta)*Y,
# assuming the sequence of parameters is ax, kp, kc, pi, ci
# (first A, P, C from model mat, then Pa, Ca from model mb)
# Note that we cannot simplify this calculation because the model is
# non-linear in pi,kp resp. ci,kc

# Matrices to use in calculation of the terms of the model for each parms
MA <- Ns( A, knots= a.kn, intercept=TRUE )
Mp <- Ns( P , knots= p.kn, ref=p.ref )
Mc <- Ns( P-A, knots= c.kn, ref=c.ref )
Mpi <- Ns( A, knots=pi.kn, intercept=TRUE )
Mci <- Ns( A, knots=ci.kn, intercept=TRUE )

# Computing the log-likelihood for any set of parameters
llik <-
function( parms )
{
  ax <- MA %*% parms[ 1:na] ; nn <- na
  if( mainP ) { kp <- Mp %*% parms[nn+1:np] ; nn <- nn+np } else kp = rep(0,length(ax))
  if( mainC ) { kc <- Mc %*% parms[nn+1:nc] ; nn <- nn+nc } else kc = rep(0,length(ax))
  if( intP ) { pi <- Mpi %*% parms[nn+1:npi] ; nn <- nn+npi } else pi = rep(1,length(ax))
  if( intC ) { ci <- Mci %*% parms[nn+1:nci] ; nn <- nn+nci } else ci = rep(1,length(ax))
  eta <- ax + pi*kp + ci*kc
  sum( D*eta - exp(eta)*Y )
}

# Numerical calculation of the Hessian for llik

```



```

ivar <- -numDeriv::hessian( llik, ml.cf )

# Sometimes not quite positive definite, fix that after inverting the Hessian
vcov <- Matrix::nearPD( solve( ivar ) )
vcov <- as.matrix( vcov$mat )
fin <- Sys.time()
if( !quiet ) cat("...done - in", round(difftime(fin,commence,units="secs"),1), "seconds.\n")

# Since we now have the variances of the parameters for each of the
# effects we can compute corrected c.i.s for the effects
se.eff <-
function( sub, cM, Alpha=alpha )
{
wh <- grep( sub, names(ml.cf) )
res <- cbind(
      cM %*% ml.cf[wh],
      sqrt( diag( cM %*% vcov[wh,wh] %*% t(cM) ) ) ) %*% ci.mat(alpha=Alpha)
colnames(res)[1] <- paste( "Joint", colnames(res)[1] )
res
}

# Append the corrected c.i.s to the effect objects
ax <- cbind( ax, exp( se.eff( "ax", Ns(a.pt,knots= a.kn,intercept=TRUE) ) ) )
if( mainP ) kp <- cbind( kp, exp( se.eff( "kp", Ns(p.pt,knots= p.kn,ref=p.ref) ) ) )
if( mainC ) kc <- cbind( kc, exp( se.eff( "kc", Ns(c.pt,knots= c.kn,ref=c.ref) ) ) )
if( intP ) pi <- cbind( pi, se.eff( "pi", Ns(a.pt,knots=pi.kn,intercept=TRUE) ) )
if( intC ) ci <- cbind( ci, se.eff( "ci", Ns(a.pt,knots=ci.kn,intercept=TRUE) ) )
}

# Collect refs and knots in lists
klist <- list( a.kn=a.kn, pi.kn=pi.kn, p.kn=p.kn, ci.kn=ci.kn, c.kn=c.kn )
rlist <- list( pi.ref=pi.ref, p.ref=p.ref, ci.ref=ci.ref, c.ref=c.ref )

# Finally output object
res <- list( model = model,
            ax = ax,
            pi = if( intP ) pi else NULL,
            kp = if( mainP ) kp else NULL,
            ci = if( intC ) ci else NULL,
            kc = if( mainC ) kc else NULL,
            mod.at = mat,
            mod.b = mb,
            coef = if( VC & conv ) ml.cf else NULL,
            vcov = if( VC & conv ) vcov else NULL,
            knots = klist,
            refs = rlist,
            deviance = dev,
            df.residual = df,
            iter = nit )
# Remove redundant stuff before returning
res <- res[!sapply(res,is.null)]
class( res ) <- "LCa"
invisible( res )
}

```

A problem is that the standard errors are *conditional* on either $(f(a), g(p))$ or $b(a)$ being known, so these will be too small. You will note that the Hessian is derived numerically, because of the intractability of an analytic expression for the second derivative resulting

from the non-linearity of the model.

```
> system.time( LCa.Mlc0 <- LCa.fit( Mlc, VC=FALSE ) )
LCa.fit convergence in 8 iterations, deviance: 8417.158 on 6084 d.f.
  user system elapsed
 8.800  9.614  5.952
> system.time( LCa.Mlc <- LCa.fit( Mlc, quiet=FALSE ) )
  Deviances: model(AT) model(A) Rel. diff.
Iteration   1 8804.601 9401.055 0.2336112
Iteration   2 8469.482 8558.406 0.0984587
Iteration   3 8423.860 8435.773 0.0145372
Iteration   4 8418.008 8419.538 0.0019283
Iteration   5 8417.265 8417.460 0.0002468
Iteration   6 8417.171 8417.196 0.0000314
Iteration   7 8417.159 8417.162 0.0000040
Iteration   8 8417.157 8417.158 0.0000005
LCa.fit convergence in 8 iterations, deviance: 8417.158 on 6084 d.f.
...using 5.3 seconds.
...computing Hessian by numerical differentiation...
...done - in 1.3 seconds.
  user system elapsed
10.334 13.032  6.606
> class( LCa.Mlc )
[1] "LCa"
> mode( LCa.Mlc )
[1] "list"
> names( LCa.Mlc )
 [1] "model"      "ax"          "pi"          "kp"          "mod.at"      "mod.b"
 [7] "coef"       "vcov"        "knots"       "refs"        "deviance"    "df.residual"
[13] "iter"
```

There are also summary, plot and predict methods for LCa objects; and now that we have the LCa.Mlc fitted object we can demonstrate how they work.

Here is a simple plot of the estimated effects, using the plot function. Note that the confidence intervals are based on the numerically derived Hessian from LCa.Mlc, because the LCa object was created with VC=TRUE, and hence contains the list component vcov.

```
> plot.LCa
function( x, ... )
{
# terms in the model
mt <- model.terms( x )

# A small plot utility to exploit the structure of the effects
plc <-
function( x, ... ) matplot( as.numeric( rownames(x) ), x[,ncol(x)-2:0],
                           type="l", lty=1, lwd=c(3,1,1), ... )

plc( x$ax, col="black", xlab="Age", ylab="Age-specific rates", log="y" )
rug( x$knots$a.kn, lwd=2 )

if( mt$intP ){
plc( x$pi, col="black", xlab="Age", ylab="Relative period log-effect multiplier" )
abline(h=1,v=x$refs$pi.ref)
```

```

rug( x$knots$pi, lwd=2 )
}

if( mt$mainP ){
plc( x$kp, col="black", log="y", xlab="Date of follow-up (period)", ylab="Period effect (RR)" )
abline(h=1,v=x$refs$p.ref)
rug( x$knots$kp, lwd=2 )
}

if( mt$intC ){
plc( x$ci, col="black", xlab="Age", ylab="Relative cohort log-effect multiplier" )
abline(h=1,v=x$refs$ci.ref)
rug( x$knots$ci, lwd=2 )
}

if( mt$mainC ){
plc( x$kc, col="black", log="y", xlab="Date of birth (cohort)", ylab="Cohort effect (RR)" )
abline(h=1,v=x$refs$c.ref)
rug( x$knots$kc, lwd=2 )
}

}

> par( mfrow=c(1,3) ) ; plot( LCa.Mlc, rnam="Lung cancer incidence per 1000 PY" )

```

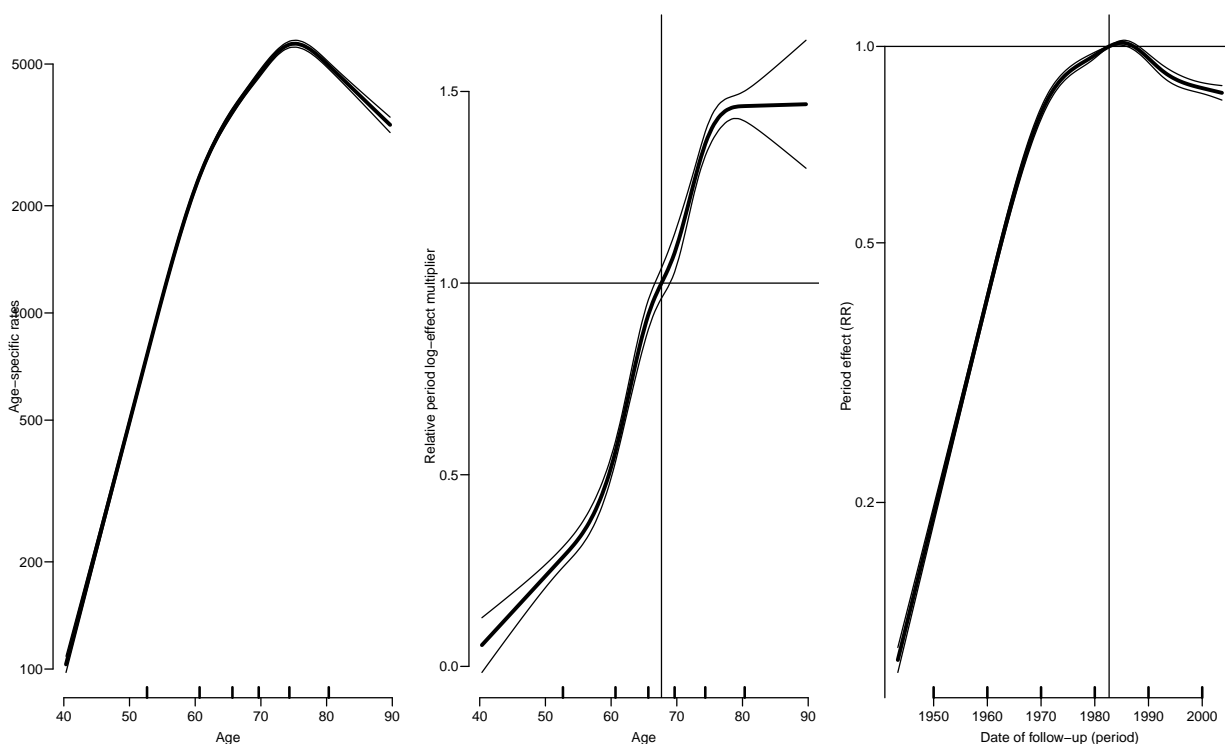


Figure 1.10: Predictions of rates and interaction effects for lung cancer incidence in Denmark. `../graph/LCa-plres`

The `predict` method return predictions from each of the two final conditional models (which should give the same predictions, but presumably different confidence intervals, both sets too narrow). If the Lee-Carter model is fitted with `LCa.fit` with argument

VC=TRUE (which is the default), the confidence intervals are based on parametric bootstrap of the parameters in the model using the Hessian derived by numerical differentiation.

```
> predict.LCa
```

```
function( object,
          newdata,
          alpha = 0.05,
          level = 1-alpha,
          sim = ( "vcov" %in% names(object) ),
          ... )
{
# What main effects and interactions are in the model
mt <- model.terms( object )

# is person-years supplied, otherwise use units as in the model
if( "Y" %in% names(newdata) ) Y <- newdata$Y else
  Y <- rep(1,nrow(newdata))

# Matrices to extract effects at newdata rows
Ma <- Ns( newdata$A, knots = object$knots$a.kn, intercept = TRUE)
Mp <- Ns( newdata$P, knots = object$knots$p.kn, ref=object$refs$p.ref )
Mc <- Ns( newdata$P-newdata$A, knots = object$knots$c.kn, ref=object$refs$c.ref )
Mpi <- Ns( newdata$A, knots = object$knots$pi.kn, intercept = TRUE)
Mci <- Ns( newdata$A, knots = object$knots$ci.kn, intercept = TRUE)

# Default terms values for models without interactions
kp <- kc <- rep( 0, nrow(newdata) )
pi <- ci <- rep( 1, nrow(newdata) )

# P, C and interaction term(s) if included in the model
if( mt$intP ) {
  pi <- ci.lin( object$mod.b , subset="pi.kn", ctr.mat=Mpi )[,1]
  kp <- ci.lin( object$mod.at, subset= "p.kn", ctr.mat=Mp )[,1]
}
if( mt$intC ) {
  ci <- ci.lin( object$mod.b , subset="ci.kn", ctr.mat=Mci )[,1]
  kc <- ci.lin( object$mod.at, subset= "c.kn", ctr.mat=Mc )[,1]
}

# First fitted values from mod.at
# Note that the model object mod.at always has the same number of
# parameters, for some of the models either period or cohort parameters
# are 0, hence not used.
pr0 <- ci.exp( object$mod.at, alpha=alpha, ctr.mat=cbind(Ma,Mp*pi,Mc*ci) )

# Then fitted values from mod.b
# But mod.b has an offset beyond log(Y), namely all the APC terms
lp.b <- ci.lin( object$mod.b , ctr.mat=cbind(Mpi*kp,Mci*kc) )[,1:2]
lp.b[,1] <- lp.b[,1] + ci.lin( object$mod.at, ctr.mat=cbind(Ma,Mp*(!mt$intP),Mc*(!mt$intC))
pr0 <- cbind( pr0, exp( lp.b %*% ci.mat(alpha=alpha) ) )

# label the estimates
colnames( pr0 )[c(1,4)] <- c("at|b Est.,"b|at Est.")

# The doings above gives confidence intervals based on the conditional
# models, so if we want proper intervals we should simulate instead,
# using the posterior distribuion of all parameters, albeit under the
# slightly fishy assumption that the joint posterior is normal...
```

```

if( sim ) # also renders TRUE if sim is numerical (and not 0)
{
if( is.logical(sim) & sim ) sim <- 1000
# Check that there is a vcov component of the model
if( !( "vcov" %in% names(object) ) )
  warning(
    "No variance-covariance in LCa object, only conditional c.i.s available.\n",
    "no simulation (parametric bootstrap) is done.\n" )
else {
# require( MASS )
# using the parametric bootstrap based on the parameters and the
# (numerically computed) Hessian
eta <- NArray( list( pt = 1:nrow(pr0),
                    it = 1:sim ) )
parms <- MASS::mvrnorm( n = sim,
                       mu = object$coef,
                       Sigma = object$vcov )

na <- ncol( Ma )
np <- ncol( Mp )
nc <- ncol( Mc )
npi <- ncol( Mpi )
nci <- ncol( Mci )
# Compute the linear predictor in each of the simulated samples
# period and cohort effects if not in the model
kp <- kc <- rep( 0, nrow(newdata) )
pi <- ci <- rep( 1, nrow(newdata) )
for( i in 1:sim ){
  ax <- Ma %*% parms[i, 1:na] ; nn <- na
if( mt$mainP ) { kp <- Mp %*% parms[i,nn+1:np] ; nn <- nn+np }
if( mt$mainC ) { kc <- Mc %*% parms[i,nn+1:nc] ; nn <- nn+nc }
if( mt$intP ) { pi <- Mpi %*% parms[i,nn+1:npi] ; nn <- nn+npi }
if( mt$intC ) { ci <- Mci %*% parms[i,nn+1:nci] }
eta[,i] <- ax + kp*pi + kc*ci
}
# predicted rates with bootstrap confidence limits
pr.sim <- exp( t( apply( eta, 1, quantile,
                      probs=c(0.5,alpha/2,1-alpha/2),
                      na.rm=TRUE ) ) )
colnames( pr.sim )[1] <- "Joint est."
return( pr.sim )
}
}
else return( pr0 )
}

> nd <- data.frame( A=rep(50:80,3), P=rep(c(1960,1970,1980),each=31) )
> # separate by NA to facilitate plotting of curves
> nd <- rbind( nd[1:31,], NA, nd[31+1:31,], NA, nd[31*2+1:31,] )
> system.time( pp0 <- predict( LCa.Mlc0, newdata=nd ) )
  user  system elapsed
0.148   0.013   0.149

> system.time( pp <- predict( LCa.Mlc , newdata=nd, sim=10000 ) )
  user  system elapsed
0.427   0.360   0.391

> head( pp0 )

```

```

      at|b Est.      2.5%    97.5% b|at Est.      2.5%    97.5%
[1,] 400.8798 393.9041 407.9789 400.8798 392.4780 409.4614
[2,] 463.7158 456.3712 471.1786 463.7158 454.6933 472.9173
[3,] 536.4011 528.5133 544.4065 536.4011 526.4180 546.5734
[4,] 620.4681 611.7157 629.3458 620.4681 608.9813 632.1716
[5,] 716.9005 706.8052 727.1400 716.9005 703.1999 730.8680
[6,] 825.0678 813.1018 837.2099 825.0678 808.4757 842.0005

```

```
> head( pp )
```

```

pt  Joint est.      2.5%    97.5%
 1   400.8571 392.0622 409.8693
 2   463.6064 454.4562 473.2922
 3   536.2538 526.1505 546.8542
 4   620.3032 608.7393 632.4766
 5   716.7920 702.8424 730.9943
 6   824.9664 808.0345 842.1151

```

We can inspect the difference between the curves from the conditional fits and the quantiles from the parametric bootstrap (simulation):

```

> par( mfrow=c(1,1) )
> matplot( nd$A, cbind(pp,pp0),
+         type="l", lwd=c(4,2,2),
+         lty=c("solid", "22", "44")[c(1,1,1,2,3,3,2,3,3)],
+         col=rep(c("red", "limegreen", "blue"), each=3), log="y",
+         ylab="DK male lung cancer incidence per 1000 PY, 1960, -70, -80",
+         xlab="Age" )

```

From figure 1.11 we see that the confidence limits for the two conditional models are narrower than the simulation-based limits. Not surprising.

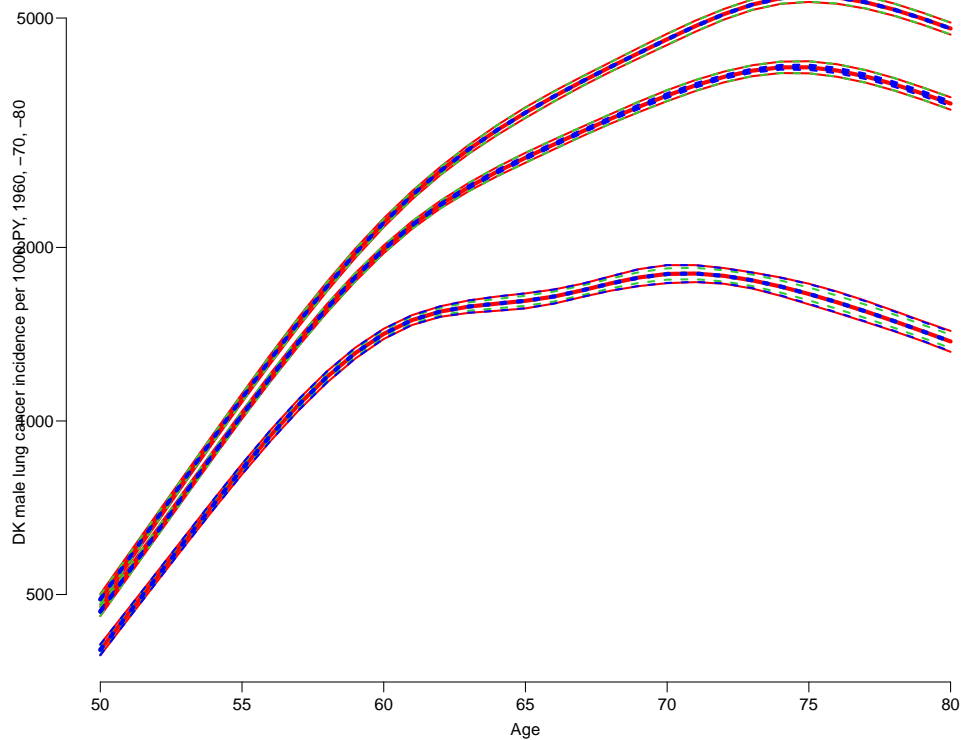


Figure 1.11: *Predicted incidence rates with 95% c.i. from a Lee-Carter model, with natural splines. Red lines are based on parametric simulation from the full model; broken green lines are from the model for main effects of age and period conditional on the age-interactions ($b_p(a)$ term); broken blue lines are from the model for the interaction term conditional on the main effects of age and period.*

../graph/LCa-pl-predict

References

- [1] Renshaw AR and Haberman S. A cohort-based extension to the lee-carter model for mortality reduction factors. *Insurance: Mathematics and Economics*, 2006.

Chapter 2

Practical examples

2.1 Lung cancer in Danish women

This exercise is parallel to the example on male lung cancer from the lectures. The point is to fit age-period-cohort models as well as Lee-Carter models and inspect their relative merits and different fits to data on female lung cancer in Denmark.

1. First we read the lung-cancer data and subset it to women only:

```
library( Epi )
lC <- read.table( "../data/lung-mf.txt", header=TRUE )
lF <- subset( lC, sex==2 )
head( lF )
```

	sex	A	P	C	Y	D	A5	P5	C5
5401	2	40.66667	1943.333	1902.667	14631.33	0	40	1943	1898
5402	2	40.33333	1943.667	1903.333	14488.00	1	40	1943	1903
5403	2	40.66667	1944.333	1903.667	14457.67	0	40	1943	1903
5404	2	40.33333	1944.667	1904.333	15011.00	1	40	1943	1903
5405	2	40.66667	1945.333	1904.667	14912.83	0	40	1943	1903
5406	2	40.33333	1945.667	1905.333	14946.83	0	40	1943	1903

2. In order to get a rough picture of data, we tabulate the data in 5-year classes by age and period (using rates per 1000):

```
t5 <- xtabs( cbind(D,Y=Y/1000) ~ A5 + P5, data=lF )
str( t5 )
```

```
xtabs [1:10, 1:11, 1:2] 15 23 28 53 44 67 35 29 16 5 ...
- attr(*, "dimnames")=List of 3
..$ A5: chr [1:10] "40" "45" "50" "55" ...
..$ P5: chr [1:11] "1943" "1948" "1953" "1958" ...
..$ : chr [1:2] "D" "Y"
- attr(*, "call")= language xtabs(formula = cbind(D, Y = Y/1000) ~ A5 + P5, data = lF)
```

```
r5 <- t5[,,"D"]/t5[,,"Y"]
```

```
par( mfrow=c(2,2),mar=c(3,3,1,1),mgp=c(3,1,0)/1.6,bty="n",las=1,oma=c(0,0,1,0) )
rateplot( r5*100, ylab="", col=heat.colors(20)[1:20], lwd=3 )
mtext("Female lung cancer rates per 100,000 PY in DK", outer=TRUE)
```

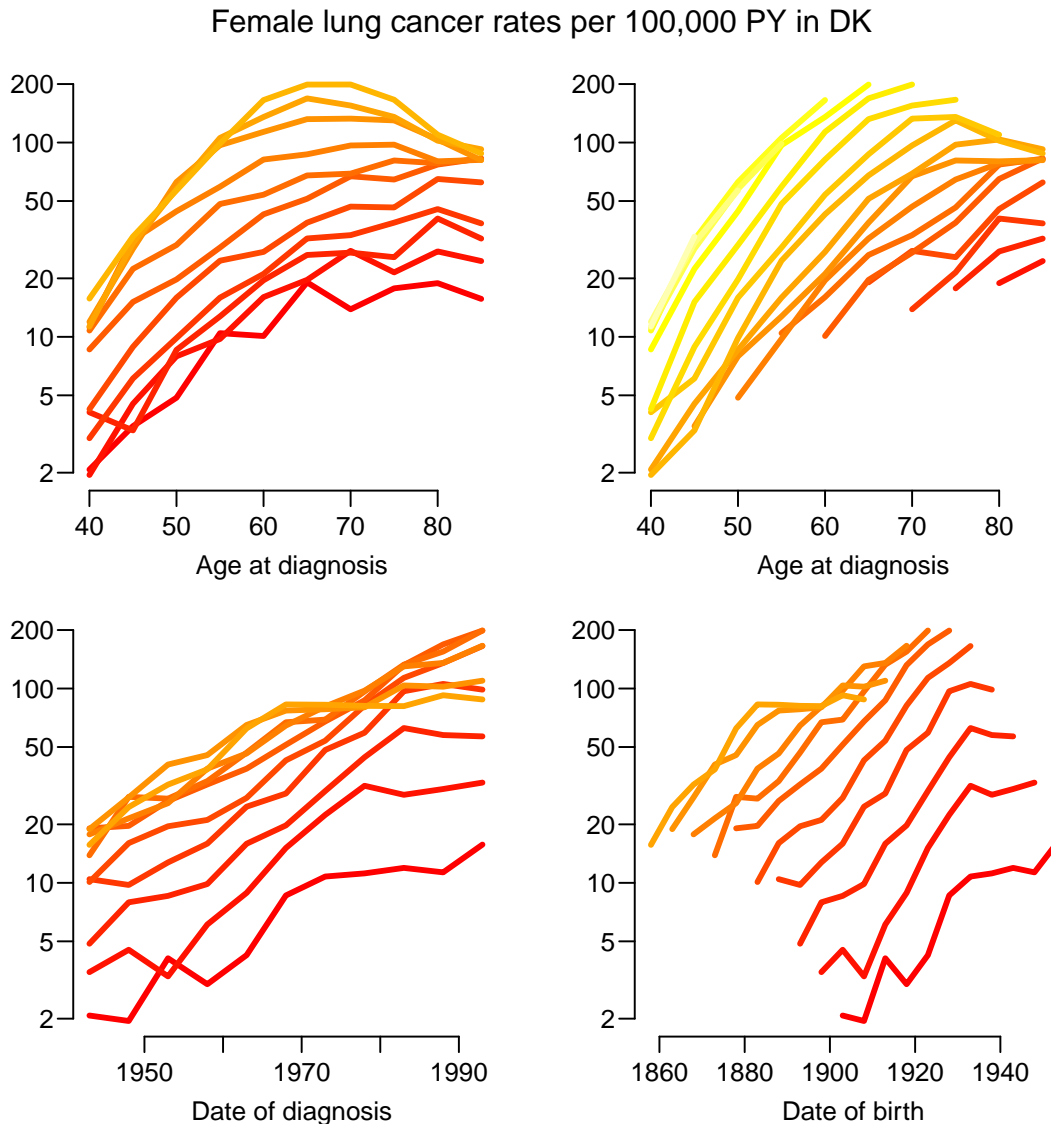


Figure 2.1: Lung cancer rates in Danish women, by 5-year classes. `../graph/DK-Flung-rates`

- When fitting APC-models and Lee-Carter models we will use natural splines for fitting, so we must devise knots on the age and time-scales for the splines. Since the information in the data on event rates is in the number of *cases*, we would like to place the n knots such that there is $1/n$ between each pair of successive knots and $1/2n$ below the first and above the last knot.

We then devise 7 knots (number taken out of thin air) for each term:

```
nk <- 7
( a.kn <- with( lF, quantile( rep( A,D), probs=(1:nk-0.5)/nk ) ) )
7.142857% 21.42857% 35.71429%      50% 64.28571% 78.57143% 92.85714%
49.66667  57.33333  62.33333  66.66667  70.66667  75.33333  81.66667

( p.kn <- with( lF, quantile( rep( P ,D), probs=(1:nk-0.5)/nk ) ) )
7.142857% 21.42857% 35.71429%      50% 64.28571% 78.57143% 92.85714%
1961.333  1973.667  1980.667  1985.333  1988.667  1992.333  1995.333
```

```
( c.kn <- with( lF, quantile( rep(P-A,D), probs=(1:nk-0.5)/nk ) ) )
7.142857% 21.42857% 35.71429%      50% 64.28571% 78.57143% 92.85714%
1891.333 1904.333 1911.667 1917.667 1922.667 1928.667 1937.333
```

4. The fitting of the APC-model and the sub-models is done by the function `apc.fit` — note that we are using a list as argument to `npar`, allowing us to explicitly allocate the knots:

```
APC <- apc.fit( lF, npar=list(A=a.kn,P=p.kn,C=c.kn),
               ref.p=1980, 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	Pr(>Chi)
Age	5393	17811.5			
Age-drift	5392	6605.7	1	11205.8	< 2.2e-16
Age-Cohort	5387	6259.3	5	346.4	< 2.2e-16
Age-Period-Cohort	5382	5974.6	5	284.7	< 2.2e-16
Age-Period	5387	6432.2	-5	-457.6	< 2.2e-16
Age-drift	5392	6605.7	-5	-173.5	< 2.2e-16

Because of the very large number of events, the non-linear effects of both period and cohort are strongly significant, the period effect a little less so, though.

5. We can then plot the estimated effects:

```
par( mfrow=c(1,1), mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
plot( APC, "Female lung cancer in Danmark per 100,000 PY", col="red" )

cp.offset      RR.fac
      1765         100
```

6. For comparison we fit the two Lee-Carter models:

```
system.time(
  LCP <- LCa.fit( lF, npar=list(a=a.kn,pi=a.kn,p=p.kn,c=c.kn,ci=a.kn),
                a.ref=60, p.ref=1980,
                model="APa", maxit=500, q=F ) )

Deviances: model(AT) model(A) Rel. diff.
Iteration  1  6114.314 6131.333 0.0519834
Iteration  2  6114.091 6114.095 0.0028193
Iteration  3  6114.091 6114.091 0.0000007
LCa.fit convergence in 3 iterations, deviance: 6114.091 on 5381 d.f.
...using 1.7 seconds.
...computing Hessian by numerical differentiation...
...done - in 1.2 seconds.
  user system elapsed
  4.734  6.471  2.947

system.time(
  LCC <- LCa.fit( lF, npar=list(a=a.kn,pi=a.kn,p=p.kn,c=c.kn,ci=a.kn),
                a.ref=60, c.ref=1930,
                model="ACa", maxit=500, q=F ) )
```

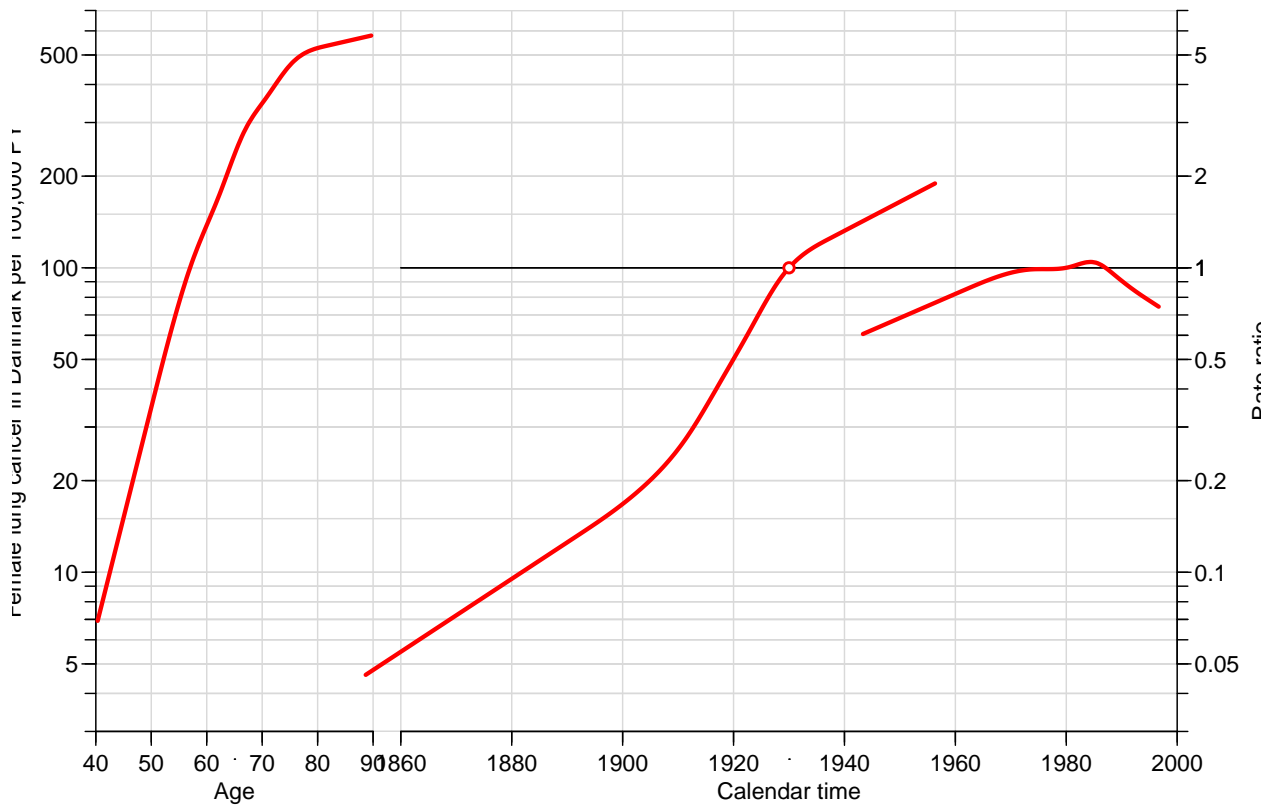


Figure 2.2: *Standard plot of APC-effects for female lung cancer in Denmark 1943–1997*
 ../graph/DK-Flung-plotAPC

```

Deviiances: model(AT) model(A) Rel. diff.
Iteration  1  6208.592 6230.136 0.0081655
Iteration  2  6175.221 6190.380 0.0064223
Iteration  3  6149.380 6161.459 0.0046938
Iteration  4  6127.963 6138.086 0.0038079
Iteration  5  6109.754 6118.399 0.0032177
Iteration  6  6094.142 6101.566 0.0027587
Iteration  7  6080.744 6087.118 0.0023736
Iteration  8  6069.278 6074.732 0.0020390
Iteration  9  6059.510 6064.153 0.0017445
Iteration 10  6051.233 6055.164 0.0014845
Iteration 11  6044.260 6047.569 0.0012559
Iteration 12  6038.421 6041.189 0.0010560
Iteration 13  6033.559 6035.861 0.0008826
Iteration 14  6029.532 6031.438 0.0007335
Iteration 15  6026.215 6027.784 0.0006062
Iteration 16  6023.496 6024.781 0.0004984
Iteration 17  6021.277 6022.325 0.0004078
Iteration 18  6019.473 6020.325 0.0003323
Iteration 19  6018.013 6018.702 0.0002696
Iteration 20  6016.834 6017.390 0.0002180
Iteration 21  6015.886 6016.333 0.0001757
Iteration 22  6015.125 6015.484 0.0001412
Iteration 23  6014.517 6014.803 0.0001131
Iteration 24  6014.030 6014.259 0.0000904
Iteration 25  6013.643 6013.825 0.0000722

```

```

Iteration 26 6013.335 6013.480 0.0000575
Iteration 27 6013.090 6013.205 0.0000457
Iteration 28 6012.896 6012.987 0.0000362
Iteration 29 6012.742 6012.814 0.0000287
Iteration 30 6012.620 6012.678 0.0000227
Iteration 31 6012.524 6012.570 0.0000180
Iteration 32 6012.448 6012.484 0.0000142
Iteration 33 6012.388 6012.417 0.0000112
Iteration 34 6012.341 6012.363 0.0000089
Iteration 35 6012.304 6012.321 0.0000070
Iteration 36 6012.275 6012.288 0.0000055
Iteration 37 6012.251 6012.262 0.0000043
Iteration 38 6012.233 6012.242 0.0000034
Iteration 39 6012.219 6012.226 0.0000027
Iteration 40 6012.208 6012.213 0.0000021
Iteration 41 6012.199 6012.203 0.0000017
Iteration 42 6012.192 6012.195 0.0000013
Iteration 43 6012.186 6012.189 0.0000010
Iteration 44 6012.182 6012.184 0.0000008
LCa.fit convergence in 44 iterations, deviance: 6012.184 on 5381 d.f.
...using 28.8 seconds.
...computing Hessian by numerical differentiation...
...done - in 2 seconds.
  user system elapsed
42.509 53.066 31.014

mod.cmp <- rbind( c( APC$Model$df.res, APC$Model$deviance ),
                 c( LCP$df, LCP$dev ),
                 c( LCC$df, LCC$dev ) )
mod.cmp <- cbind( mod.cmp, mod.cmp[,2]/mod.cmp[,1] )
rownames( mod.cmp ) <- c("APC", "LCaP", "LCaC")
colnames( mod.cmp ) <- c("d.f.", "deviance", "dev./d.f")
round( mod.cmp, 3 )

      d.f. deviance dev./d.f
APC 5382 5974.591 1.110
LCaP 5381 6114.091 1.136
LCaC 5381 6012.184 1.117

```

We see that the APC-model provides a better fit to data as judged by the deviance, but also that the cohort-version of the Lee-Carter model is much better than the period-version.

7. We can plot the estimated effects with the devised `plot` method for LCa objects:

```

par( mfrow=c(2,3) )
plot( LCP )
plot( LCC )

```

8. We may get a better view of the behaviour of the different models, we can plot the predicted rates over the time-span of the data frame at select ages, in this case 50, 60, 70 and 80. We put NAs between the age-classes in order to be able to plot rates in one go:

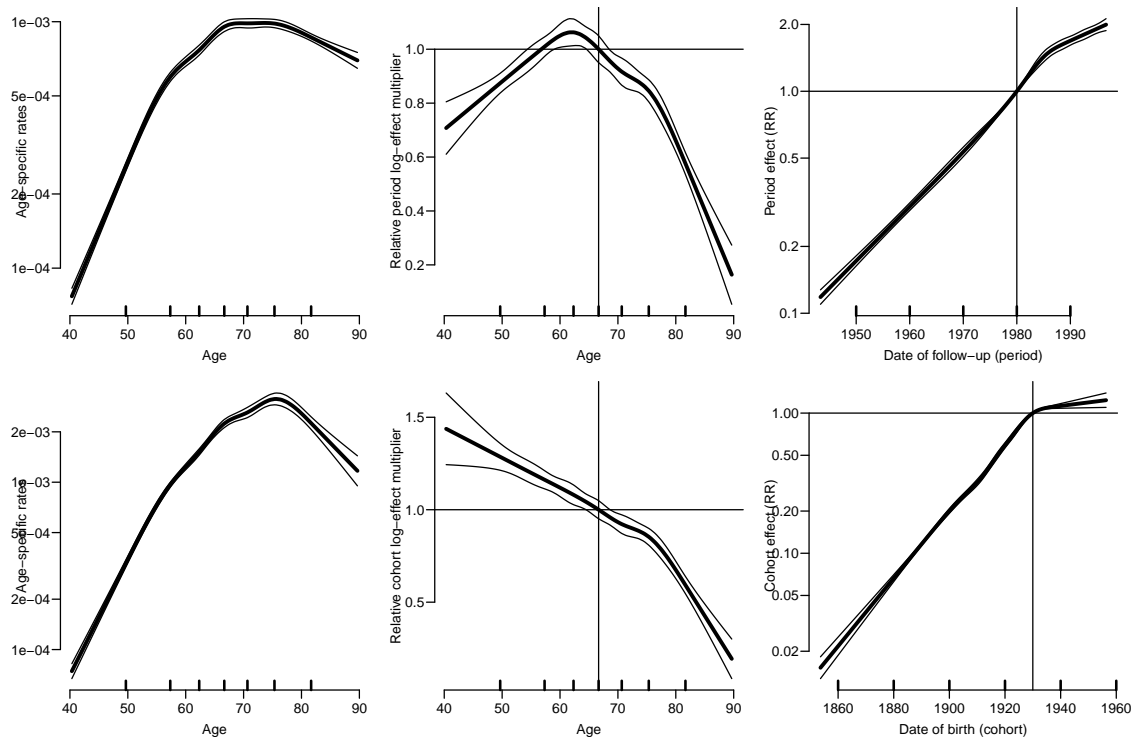


Figure 2.3: *Estimated effects from the Lee-Carter model with period-interaction, resp. cohort-interaction for female lung cancer in Denmark 1943–97.*

../graph/DK-Flung-LCplots

```
p.pt <- 1950:1997 ; np <- length(p.pt)
a.pt <- 5:8*10    ; na <- length(a.pt)
nd <- data.frame( A = rep(a.pt,each=np),
                  P = rep(p.pt,   na),
                  Y = 1000 )

# select the relevant ages
nd <- rbind( nd[ 1:np,], NA,
             nd[1*np+1:np,], NA,
             nd[2*np+1:np,], NA,
             nd[3*np+1:np,] )
```

The models fitted in the `apc.fit` are using specially designed matrices designed to give the desired parametrizations and are therefore not suitable for predictions, so we fit the models explicitly:

```
AP <- glm( D ~ Ns(A,knots=a.kn)+Ns(P,knots=p.kn),
           offset=log(Y), family=poisson, data=lF )
AC <- glm( D ~ Ns(A,knots=a.kn)+
           Ns(P-A,knots=c.kn),
           offset=log(Y), family=poisson, data=lF )
APC <- glm( D ~ Ns(A,knots=a.kn)+Ns(P,knots=p.kn)+Ns(P-A,knots=c.kn),
            offset=log(Y), family=poisson, data=lF )
```

With these models we can now produce the fitted rates under each of the models:

```
fAP <- ci.pred( AP , nd )
fAC <- ci.pred( AC , nd )
fAPC <- ci.pred( APC, nd )
fLCP <- predict( LCP, nd, sim=10000 )*1000
fLCC <- predict( LCC, nd, sim=10000 )*1000
```

And then we can show the age-specific rates both by period and cohort:

```
ppm <-
function( prd, mod )
{
matplot( nd$P-nd$A, prd, type="l", lty=1, lwd=c(3,1,1), col="black", log="y",
         ylim=c(0.02,2), xlim=1860+c(0,90), ylab="", xlab="Date of birth" )
text( 1860, 2, mod, adj=c(0,1) )
matplot( nd$P      , prd, type="l", lty=1, lwd=c(3,1,1), col="black", log="y",
         ylim=c(0.02,2), xlim=1920+c(0,90), ylab="", xlab="Date of event" )
text( 1920, 2, mod, adj=c(0,1) )
}
par( mfcol=c(2,5), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6, bty="n", las=1 )
ppm( fAP , "Age-Period" )
ppm( fLCP, "Lee-Carter-Period" )
ppm( fAPC, "Age-Period-Cohort" )
ppm( fLCC, "Lee-Carter Cohort" )
ppm( fAC , "Age-Cohort" )
```

We could also show age-specific rates at select dates or age-specific rates in select cohorts, which most conveniently are derived by redefining the `nd` prediction data frame.

```
# Age-specific rates by period
p.pt <- 1950+0:4*10 ; np <- length(p.pt)
a.pt <- 40:90      ; na <- length(a.pt)
nd <- data.frame( A = rep(a.pt,      np),
                  P = rep(p.pt,each=na),
                  Y = 1000 )
nd <- rbind( nd[ 1:na,], NA,
            nd[1*na+1:na,], NA,
            nd[2*na+1:na,], NA,
            nd[3*na+1:na,], NA,
            nd[4*na+1:na,] )
pAP <- ci.pred( AP , nd )
pAC <- ci.pred( AC , nd )
pAPC <- ci.pred( APC, nd )
pLCP <- predict( LCP, nd, sim=10000 )*1000
pLCC <- predict( LCC, nd, sim=10000 )*1000
# Age-specific rates by cohort
c.pt <- 1870+0:8*10 ; nc <- length(c.pt)
a.pt <- 40:90      ; na <- length(a.pt)
nc <- data.frame( A = rep(a.pt,      nc),
                  C = rep(c.pt,each=na),
                  Y = 1000 )
nc <- rbind( nc[ 1:na,], NA,
            nc[1*na+1:na,], NA,
            nc[2*na+1:na,], NA,
            nc[3*na+1:na,], NA,
            nc[4*na+1:na,], NA,
            nc[5*na+1:na,], NA,
            nc[6*na+1:na,], NA,
            nc[7*na+1:na,], NA,
            nc[8*na+1:na,] )
nc$P <- nc$C + nc$A
nc <- subset( nc, (P>1943 & P<2000) | is.na(A) )
```

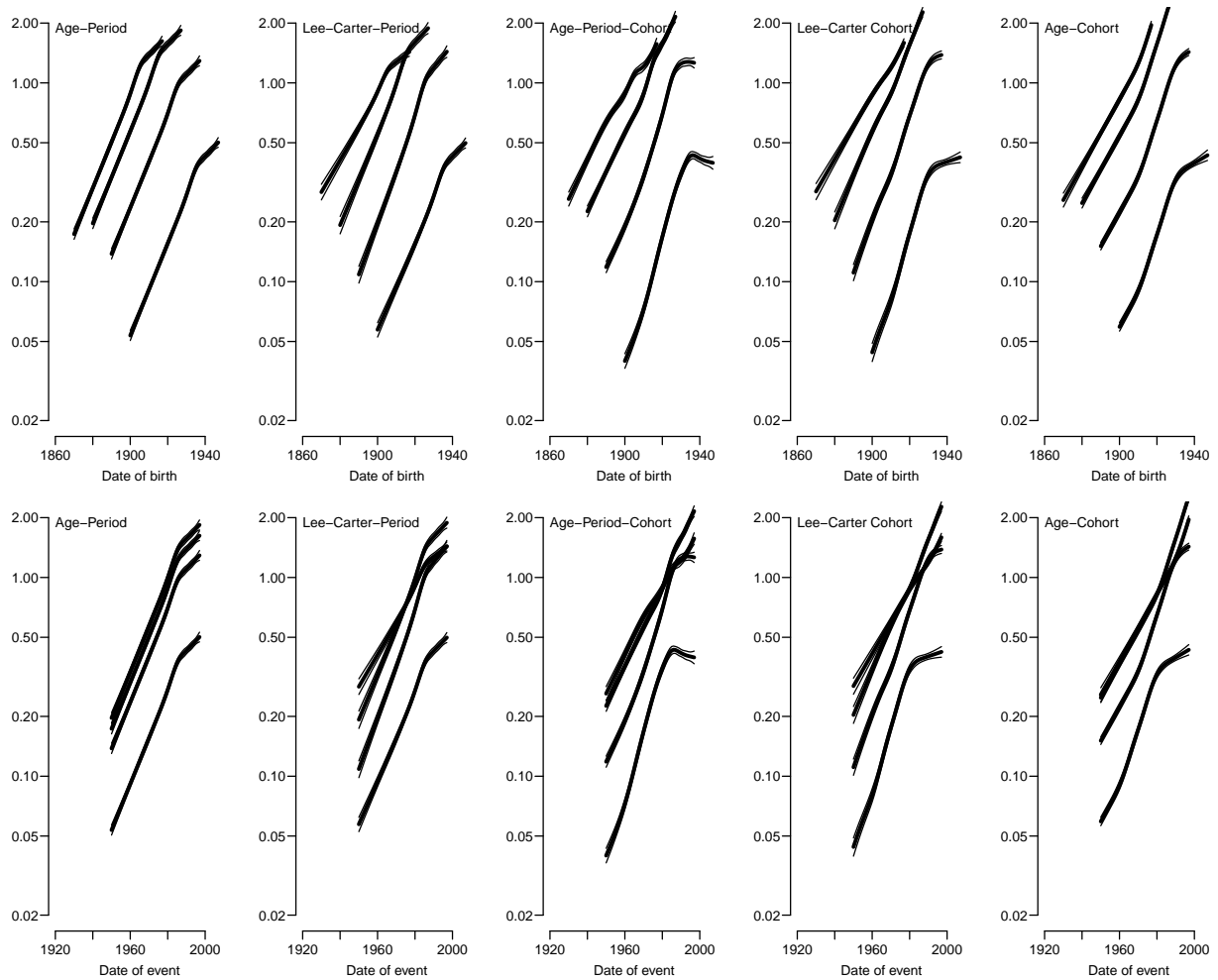


Figure 2.4: Comparison of predicted rates from different models, top panels are rates in ages 50, 60, 70 and 80 as they evolve by date of birth; bottom panels as they evolve by date of observation. The curves in upepr an lower panels are exactly the same, they are just placed horizontally differently.

../graph/DK-Flung-fL-cmpt

```

cAP <- ci.pred( AP , nc )
cAC <- ci.pred( AC , nc )
cAPC <- ci.pred( APC , nc )
cLCP <- predict( LCP , nc , sim=10000 )*1000
cLCC <- predict( LCC , nc , sim=10000 )*1000

```

```

ppm <-
function( prp , prc , mod )
{
matplot( nd$A , prp , type="l" , lty=1 , lwd=c(3,1,1) , col="black" , log="y" ,
        ylim=c(0.02,2) , xlim=c(30,90) , ylab="" , xlab="Age" )
text( 30 , 2 , mod , adj=c(0,1) )
matplot( nc$A , prc , type="l" , lty=1 , lwd=c(3,1,1) , col="black" , log="y" ,
        ylim=c(0.02,2) , xlim=c(30,90) , ylab="" , xlab="Age" )
text( 30 , 2 , mod , adj=c(0,1) )
}
par( mfc col=c(2,5) , mar=c(3,3,1,1) , mgp=c(3,1,0)/1.6 , bty="n" , las=1 )

```



```
ppm( pAP , cAP , "Age-Period")
ppm( pLCP, cLCP, "Lee-Carter-Period")
ppm( pAPC, cAPC, "Age-Period-Cohort")
ppm( pLCC, cLCC, "Lee-Carter Cohort")
ppm( pAC , cAC , "Age-Cohort")
```

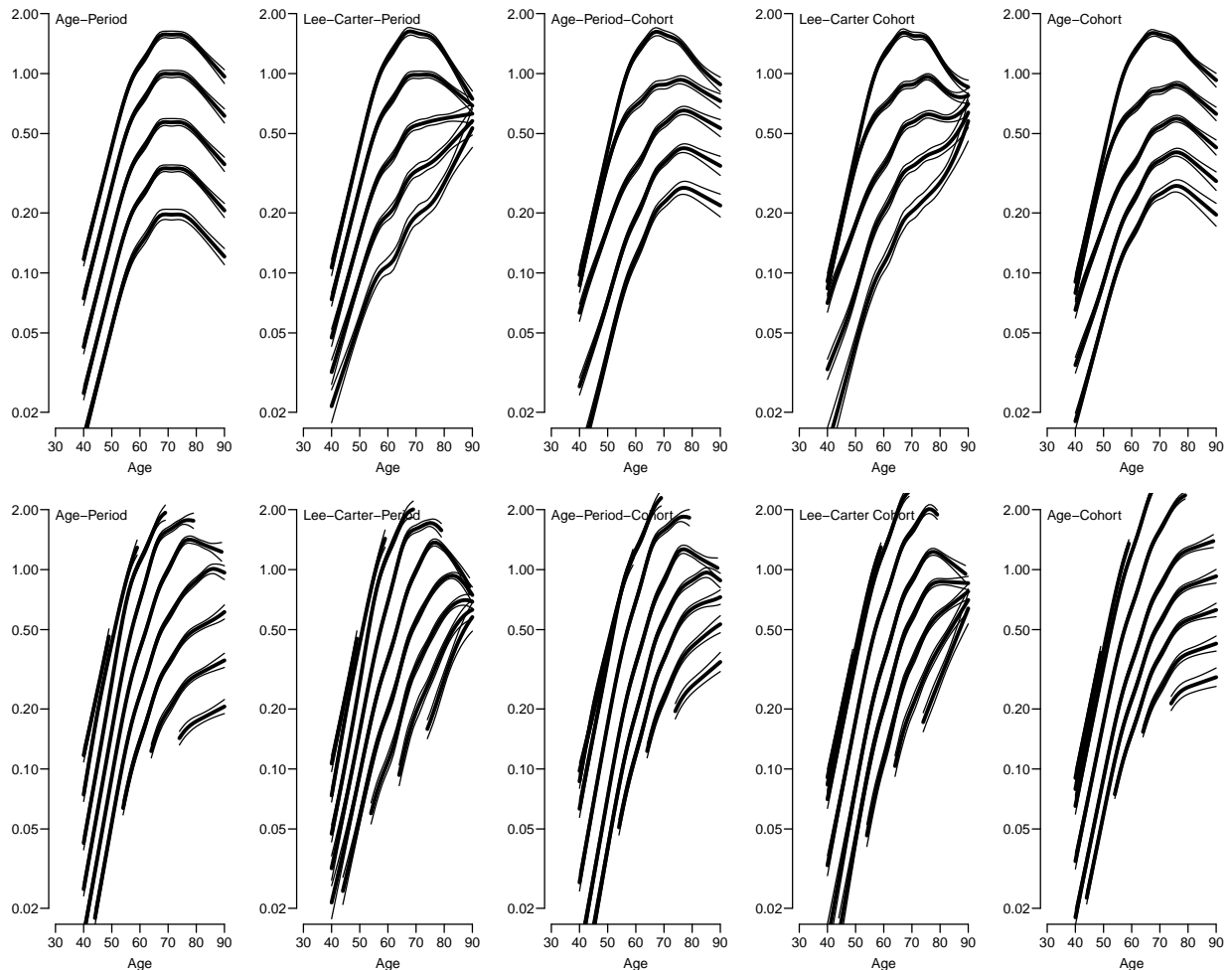


Figure 2.5: Comparison of predicted rates from different models, top panels are age-specific rates at dates 1950, 1960, ...1990; bottom panels are age-specific rates for dates of birth 1870, 1880, ...1950.

../graph/DK-Flung-fL-cmpa

2.2 Mortality in Czech men

The point of this exercise is to fit age-period-cohort models as well as Lee-Carter models and inspect their relative merits and different fits to data on male mortality in the Czech republic.

1. First we read the lung-cancer data and subset it to men only:

```
> library( Epi )
> clear()
```

```
> mM <- read.table( "../data/apc-CZ.txt", header=TRUE )
> str( mM )

'data.frame':      22400 obs. of  6 variables:
 $ sex: int  1 1 1 1 1 1 1 1 1 1 ...
 $ A  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ P  : int  1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 ...
 $ C  : int  1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 ...
 $ D  : int  1660 1604 1011 817 587 463 414 380 428 258 ...
 $ Y  : num  44851 45274 45479 44491 43083 ...
```

```
> mM <- subset( mM, sex==1 )
> str( mM )
```

```
'data.frame':      11200 obs. of  6 variables:
 $ sex: int  1 1 1 1 1 1 1 1 1 1 ...
 $ A  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ P  : int  1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 ...
 $ C  : int  1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 ...
 $ D  : int  1660 1604 1011 817 587 463 414 380 428 258 ...
 $ Y  : num  44851 45274 45479 44491 43083 ...
```

```
> mM$up <- with( mM, P-A-C )
> mM <- transform( mM, A = A+(1+up)/3,
+                 P = P+(2-up)/3 )[,c("A", "P", "D", "Y")]
> head( mM )
```

	A	P	D	Y
1	0.6666667	1950.333	1660	44850.54
2	0.6666667	1951.333	1604	45273.55
3	0.6666667	1952.333	1011	45478.96
4	0.6666667	1953.333	817	44490.95
5	0.6666667	1954.333	587	43082.91
6	0.6666667	1955.333	463	42079.85

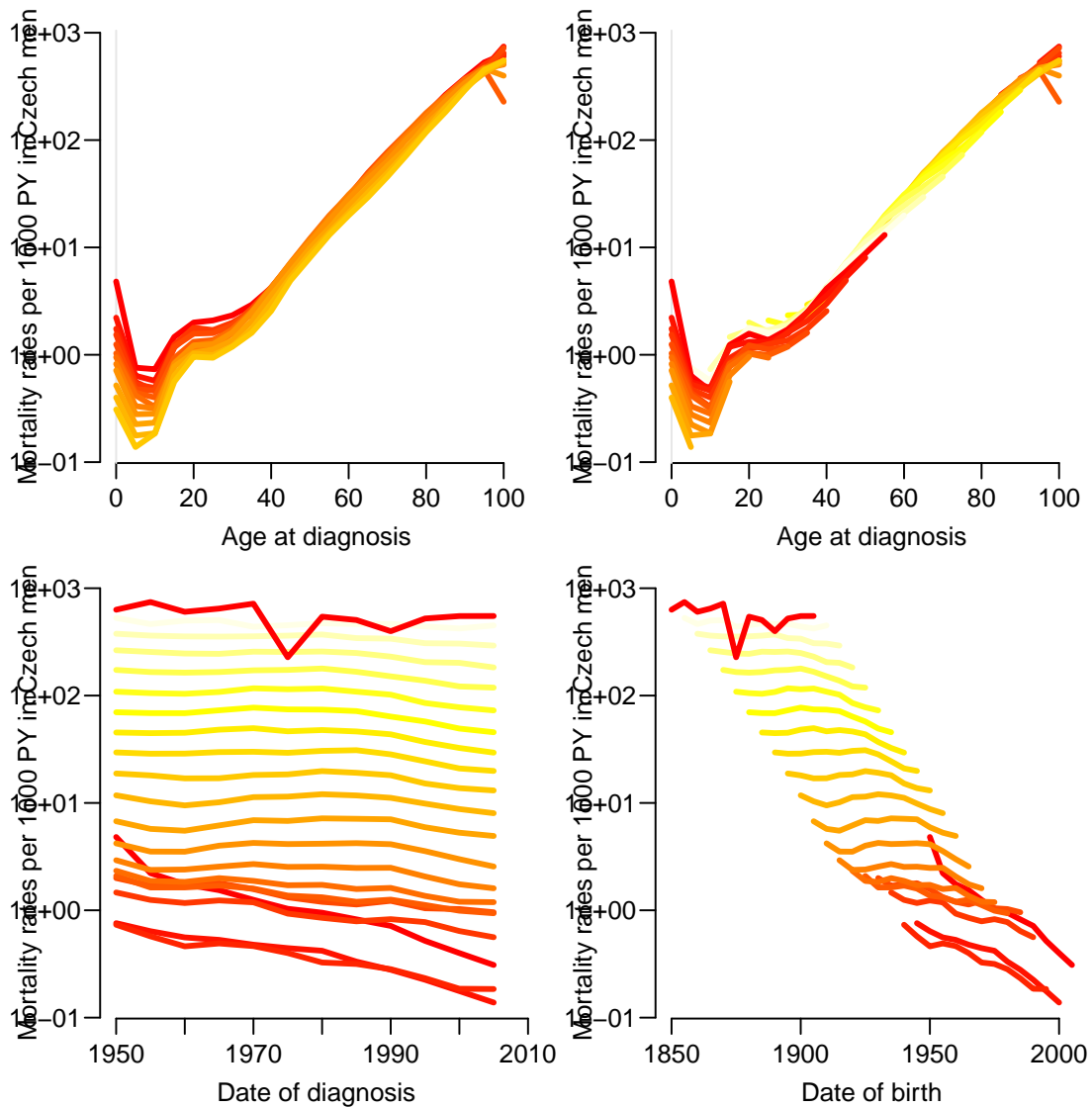
2. In order to get a rough picture of data, we tabulate the data in 5-year classes by age and period (using rates per 1000):

```
> t5 <- xtabs( cbind(D,Y=Y/10^3) ~ I(floor(A/5)*5) + I(floor(P/5)*5), data=mM )
> str( t5 )

xtabs [1:21, 1:12, 1:2] 9731 1536 1146 2145 3400 ...
- attr(*, "dimnames")=List of 3
..$ I(floor(A/5) * 5): chr [1:21] "0" "5" "10" "15" ...
..$ I(floor(P/5) * 5): chr [1:12] "1950" "1955" "1960" "1965" ...
..$ : chr [1:2] "D" "Y"
- attr(*, "call")= language xtabs(formula = cbind(D, Y = Y/10^3) ~ I(floor(A/5) * 5)

> r5 <- t5[,,"D"]/t5[,,"Y"]

> par( mfrow=c(2,2),mar=c(3,3,1,1),mgp=c(3,1,0)/1.6,bty="n",las=1 )
> rateplot( r5, ylab="Mortality rates per 1000 PY in Czech men",
+          col=heat.colors(20)[1:20], lwd=3 )
```

Figure 2.6: *Mortality rates in Czech men, by 5-year classes.*

../graph/CZM-rates

- When fitting APC-models and Lee-Carter models we will use natural splines for fitting, so we must devise knots on the age and time-scales for the splines. Since the information in the data on event rates is in the number of *cases*, we would like to place the n knots such that there is $1/n$ between each pair of successive knots and $1/2n$ below the first and above the last knot.

We then devise 8 knots (number taken out of thin air) for each term:

```
> nk <- 8
> ( a.kn <- with( mM, quantile( rep( A,D), probs=(1:nk-0.5)/nk ) ) )
  6.25%  18.75%  31.25%  43.75%  56.25%  68.75%  81.25%  93.75%
39.66667 55.66667 62.66667 67.66667 71.66667 75.66667 79.66667 85.66667
> ( p.kn <- with( mM, quantile( rep( P ,D), probs=(1:nk-0.5)/nk ) ) )
  6.25%  18.75%  31.25%  43.75%  56.25%  68.75%  81.25%  93.75%
1954.333 1962.333 1969.667 1976.333 1982.333 1988.667 1995.333 2002.333
```

```
> ( c.kn <- with( mM, quantile( rep(P-A,D), probs=(1:nk-0.5)/nk ) ) )
      6.25%  18.75%  31.25%  43.75%  56.25%  68.75%  81.25%  93.75%
1880.667 1893.667 1901.333 1907.333 1913.333 1921.667 1929.667 1946.667
```

4. The fitting of the APC-model and the sub-models is done by the function `apc.fit` — note that we are using a list as argument to `npar`, allowing us to explicitly allocate the knots:

```
> APC <- apc.fit( mM, npar=list(A=a.kn,P=p.kn,C=c.kn),
+               ref.c=1900, scale=10^3 )
[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	Pr(>Chi)
Age	11192	180855			
Age-drift	11191	154862	1	25993.1	< 2.2e-16
Age-Cohort	11185	146368	6	8494.5	< 2.2e-16
Age-Period-Cohort	11179	128263	6	18104.1	< 2.2e-16
Age-Period	11185	131744	-6	-3480.8	< 2.2e-16
Age-drift	11191	154862	-6	-23117.9	< 2.2e-16

Because of the very large number of events, the non-linear effects of both period and cohort are strongly significant, the period effect a little less so.

5. We can then plot the estimated effects:

```
> par( mfrow=c(1,1), mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
> plot( APC, "Mortality in Czech men, per 1000 PY", col="blue",
+       lwd=c(4,1,1), ci=T )
cp.offset  RR.fac
      1728      100
```

6. For comparison we fit the two Lee-Carter models:

```
> system.time(
+   LCP <- LCa.fit( mM, npar=list(a=a.kn,pi=a.kn,p=p.kn,c=c.kn,ci=a.kn),
+                 a.ref=60, p.ref=1980,
+                 model="APa", maxit=500, q=F ) )
Deviances: model(AT) model(A) Rel. diff.
Iteration  1 121509.927 125613.529 0.0842261
Iteration  2 118922.909 119366.032 0.0523390
Iteration  3 118833.145 118844.900 0.0043850
Iteration  4 118831.007 118831.297 0.0001145
Iteration  5 118830.951 118830.959 0.0000028
Iteration  6 118830.949 118830.949 0.0000001
LCa.fit convergence in 6 iterations, deviance: 118830.9 on 11178 d.f.
...using 8.9 seconds.
...computing Hessian by numerical differentiation...
...done - in 6.4 seconds.
user system elapsed
20.527 27.123 15.932
```

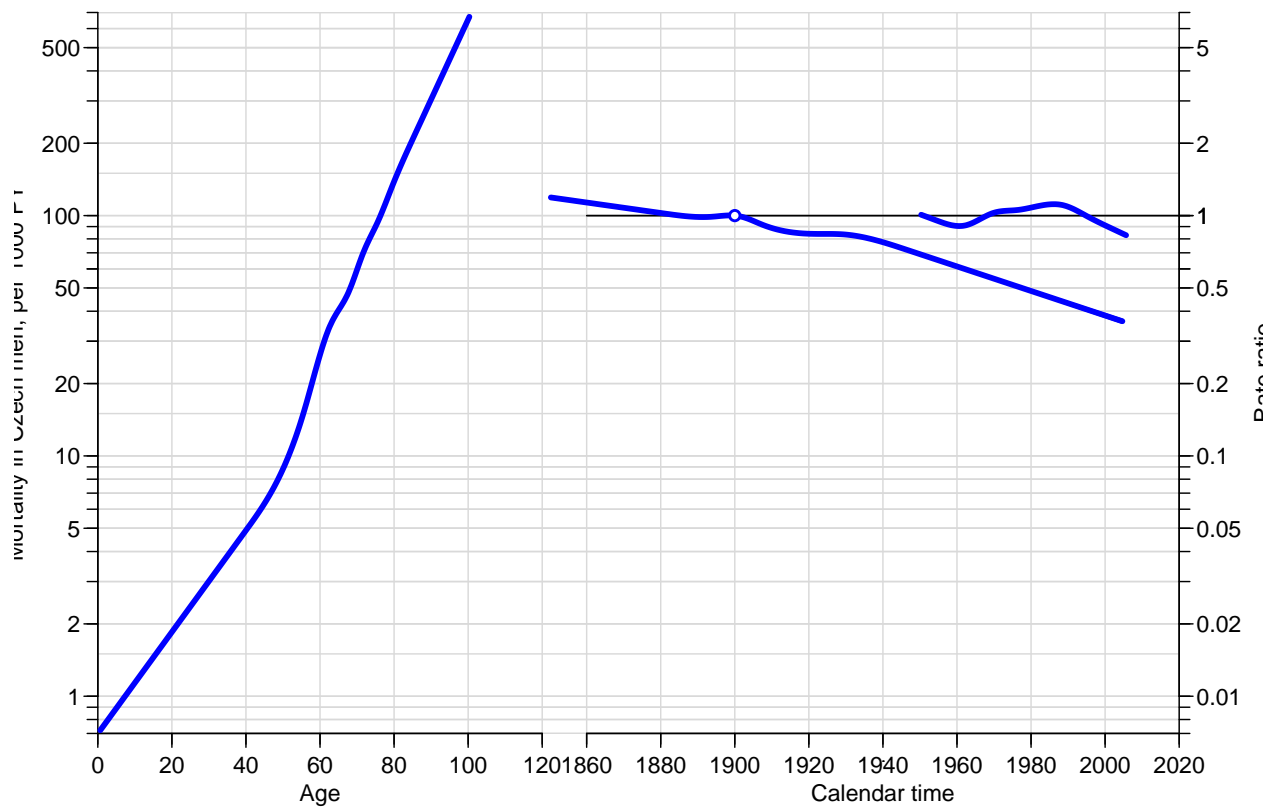


Figure 2.7: *Standard plot of APC-effects for mortality of Czech males 1950–2006.*

../graph/CZM-plotAPC

```
> system.time(
+   LCC <- LCa.fit( mM, npar=list(a=a.kn,pi=a.kn,p=p.kn,c=c.kn,ci=a.kn),
+                 a.ref=60, c.ref=1930,
+                 model="ACa", maxit=500, q=F ) )
```

```
Deviances: model(AT) model(A) Rel. diff.
Iteration  1  140038.359 142063.384 0.0451962
Iteration  2  138219.212 139000.658 0.0220339
Iteration  3  137048.502 137621.106 0.0100243
Iteration  4  136051.061 136553.470 0.0078184
Iteration  5  135151.370 135606.538 0.0069829
Iteration  6  134333.780 134747.127 0.0063780
Iteration  7  133591.997 133966.320 0.0058284
Iteration  8  132920.961 133258.845 0.0053090
Iteration  9  132315.633 132619.750 0.0048190
Iteration 10  131770.907 132043.980 0.0043604
Iteration 11  131281.704 131526.433 0.0039349
Iteration 12  130843.071 131062.072 0.0035431
Iteration 13  130450.256 130646.024 0.0031845
Iteration 14  130098.776 130273.650 0.0028584
Iteration 15  129784.447 129940.597 0.0025631
Iteration 16  129503.409 129642.826 0.0022969
Iteration 17  129252.125 129376.626 0.0020576
Iteration 18  129027.382 129138.609 0.0018431
Iteration 19  128826.274 128925.705 0.0016514
Iteration 20  128646.185 128735.147 0.0014802
```

Iteration	21	128484.773	128564.450	0.0013277
Iteration	22	128339.949	128411.392	0.0011919
Iteration	23	128209.852	128273.996	0.0010711
Iteration	24	128092.829	128150.501	0.0009637
Iteration	25	127987.418	128039.348	0.0008681
Iteration	26	127892.324	127939.158	0.0007831
Iteration	27	127806.404	127848.710	0.0007075
Iteration	28	127728.650	127766.928	0.0006401
Iteration	29	127658.169	127692.863	0.0005800
Iteration	30	127594.178	127625.675	0.0005264
Iteration	31	127535.982	127564.625	0.0004786
Iteration	32	127482.970	127509.061	0.0004358
Iteration	33	127434.601	127458.407	0.0003974
Iteration	34	127390.396	127412.154	0.0003630
Iteration	35	127349.934	127369.850	0.0003321
Iteration	36	127312.838	127331.098	0.0003043
Iteration	37	127278.775	127295.544	0.0002793
Iteration	38	127247.451	127262.873	0.0002567
Iteration	39	127218.601	127232.807	0.0002363
Iteration	40	127191.992	127205.096	0.0002178
Iteration	41	127167.414	127179.519	0.0002011
Iteration	42	127144.680	127155.878	0.0001859
Iteration	43	127123.622	127133.996	0.0001721
Iteration	44	127104.092	127113.715	0.0001596
Iteration	45	127085.953	127094.892	0.0001481
Iteration	46	127069.085	127077.399	0.0001377
Iteration	47	127053.380	127061.122	0.0001281
Iteration	48	127038.737	127045.956	0.0001194
Iteration	49	127025.070	127031.810	0.0001114
Iteration	50	127012.298	127018.597	0.0001040
Iteration	51	127000.348	127006.242	0.0000973
Iteration	52	126989.154	126994.677	0.0000911
Iteration	53	126978.657	126983.837	0.0000854
Iteration	54	126968.802	126973.666	0.0000801
Iteration	55	126959.541	126964.112	0.0000752
Iteration	56	126950.827	126955.129	0.0000708
Iteration	57	126942.621	126946.673	0.0000666
Iteration	58	126934.884	126938.705	0.0000628
Iteration	59	126927.583	126931.189	0.0000592
Iteration	60	126920.685	126924.093	0.0000559
Iteration	61	126914.164	126917.386	0.0000528
Iteration	62	126907.991	126911.041	0.0000500
Iteration	63	126902.143	126905.033	0.0000473
Iteration	64	126896.598	126899.339	0.0000449
Iteration	65	126891.335	126893.937	0.0000426
Iteration	66	126886.336	126888.808	0.0000404
Iteration	67	126881.583	126883.933	0.0000384
Iteration	68	126877.060	126879.297	0.0000365
Iteration	69	126872.752	126874.883	0.0000348
Iteration	70	126868.646	126870.678	0.0000331
Iteration	71	126864.729	126866.668	0.0000316
Iteration	72	126860.990	126862.841	0.0000302
Iteration	73	126857.417	126859.186	0.0000288
Iteration	74	126854.001	126855.693	0.0000275
Iteration	75	126850.732	126852.351	0.0000263
Iteration	76	126847.602	126849.152	0.0000252
Iteration	77	126844.602	126846.088	0.0000242

Iteration	78	126841.725	126843.150	0.0000232
Iteration	79	126838.965	126840.332	0.0000222
Iteration	80	126836.313	126837.627	0.0000213
Iteration	81	126833.765	126835.028	0.0000205
Iteration	82	126831.315	126832.530	0.0000197
Iteration	83	126828.958	126830.126	0.0000189
Iteration	84	126826.688	126827.813	0.0000182
Iteration	85	126824.501	126825.585	0.0000176
Iteration	86	126822.393	126823.438	0.0000169
Iteration	87	126820.359	126821.367	0.0000163
Iteration	88	126818.396	126819.369	0.0000158
Iteration	89	126816.500	126817.441	0.0000152
Iteration	90	126814.668	126815.577	0.0000147
Iteration	91	126812.897	126813.776	0.0000142
Iteration	92	126811.184	126812.034	0.0000137
Iteration	93	126809.526	126810.349	0.0000133
Iteration	94	126807.921	126808.717	0.0000129
Iteration	95	126806.365	126807.137	0.0000125
Iteration	96	126804.857	126805.606	0.0000121
Iteration	97	126803.395	126804.121	0.0000117
Iteration	98	126801.976	126802.681	0.0000114
Iteration	99	126800.599	126801.283	0.0000110
Iteration	100	126799.262	126799.926	0.0000107
Iteration	101	126797.963	126798.608	0.0000104
Iteration	102	126796.700	126797.327	0.0000101
Iteration	103	126795.472	126796.082	0.0000098
Iteration	104	126794.278	126794.871	0.0000096
Iteration	105	126793.115	126793.693	0.0000093
Iteration	106	126791.984	126792.546	0.0000090
Iteration	107	126790.882	126791.429	0.0000088
Iteration	108	126789.808	126790.342	0.0000086
Iteration	109	126788.762	126789.282	0.0000084
Iteration	110	126787.742	126788.249	0.0000081
Iteration	111	126786.748	126787.242	0.0000079
Iteration	112	126785.777	126786.260	0.0000077
Iteration	113	126784.830	126785.301	0.0000076
Iteration	114	126783.906	126784.366	0.0000074
Iteration	115	126783.004	126783.452	0.0000072
Iteration	116	126782.122	126782.560	0.0000070
Iteration	117	126781.261	126781.689	0.0000069
Iteration	118	126780.419	126780.838	0.0000067
Iteration	119	126779.596	126780.005	0.0000066
Iteration	120	126778.792	126779.192	0.0000064
Iteration	121	126778.005	126778.396	0.0000063
Iteration	122	126777.235	126777.618	0.0000061
Iteration	123	126776.482	126776.856	0.0000060
Iteration	124	126775.745	126776.111	0.0000059
Iteration	125	126775.023	126775.382	0.0000058
Iteration	126	126774.316	126774.668	0.0000056
Iteration	127	126773.624	126773.969	0.0000055
Iteration	128	126772.946	126773.284	0.0000054
Iteration	129	126772.282	126772.613	0.0000053
Iteration	130	126771.631	126771.955	0.0000052
Iteration	131	126770.993	126771.311	0.0000051
Iteration	132	126770.368	126770.679	0.0000050
Iteration	133	126769.755	126770.060	0.0000049
Iteration	134	126769.153	126769.453	0.0000048

Iteration	135	126768.563	126768.857	0.0000047
Iteration	136	126767.985	126768.273	0.0000046
Iteration	137	126767.417	126767.700	0.0000045
Iteration	138	126766.860	126767.138	0.0000044
Iteration	139	126766.314	126766.586	0.0000044
Iteration	140	126765.777	126766.044	0.0000043
Iteration	141	126765.250	126765.513	0.0000042
Iteration	142	126764.733	126764.991	0.0000041
Iteration	143	126764.225	126764.478	0.0000040
Iteration	144	126763.727	126763.975	0.0000040
Iteration	145	126763.237	126763.481	0.0000039
Iteration	146	126762.756	126762.995	0.0000038
Iteration	147	126762.283	126762.519	0.0000038
Iteration	148	126761.819	126762.050	0.0000037
Iteration	149	126761.363	126761.590	0.0000036
Iteration	150	126760.914	126761.138	0.0000036
Iteration	151	126760.474	126760.693	0.0000035
Iteration	152	126760.041	126760.256	0.0000034
Iteration	153	126759.615	126759.827	0.0000034
Iteration	154	126759.196	126759.405	0.0000033
Iteration	155	126758.785	126758.990	0.0000033
Iteration	156	126758.380	126758.582	0.0000032
Iteration	157	126757.983	126758.181	0.0000032
Iteration	158	126757.591	126757.786	0.0000031
Iteration	159	126757.207	126757.398	0.0000031
Iteration	160	126756.828	126757.017	0.0000030
Iteration	161	126756.456	126756.641	0.0000030
Iteration	162	126756.090	126756.272	0.0000029
Iteration	163	126755.730	126755.909	0.0000029
Iteration	164	126755.375	126755.552	0.0000028
Iteration	165	126755.027	126755.201	0.0000028
Iteration	166	126754.684	126754.855	0.0000027
Iteration	167	126754.346	126754.514	0.0000027
Iteration	168	126754.014	126754.180	0.0000026
Iteration	169	126753.687	126753.850	0.0000026
Iteration	170	126753.366	126753.526	0.0000026
Iteration	171	126753.049	126753.207	0.0000025
Iteration	172	126752.738	126752.893	0.0000025
Iteration	173	126752.431	126752.584	0.0000024
Iteration	174	126752.129	126752.279	0.0000024
Iteration	175	126751.832	126751.980	0.0000024
Iteration	176	126751.539	126751.685	0.0000023
Iteration	177	126751.251	126751.395	0.0000023
Iteration	178	126750.968	126751.109	0.0000023
Iteration	179	126750.688	126750.827	0.0000022
Iteration	180	126750.414	126750.550	0.0000022
Iteration	181	126750.143	126750.278	0.0000022
Iteration	182	126749.876	126750.009	0.0000021
Iteration	183	126749.614	126749.745	0.0000021
Iteration	184	126749.355	126749.484	0.0000021
Iteration	185	126749.101	126749.228	0.0000020
Iteration	186	126748.850	126748.975	0.0000020
Iteration	187	126748.603	126748.726	0.0000020
Iteration	188	126748.360	126748.481	0.0000019
Iteration	189	126748.121	126748.240	0.0000019
Iteration	190	126747.885	126748.003	0.0000019
Iteration	191	126747.653	126747.768	0.0000018


```

Iteration 192 126747.424 126747.538 0.0000018
Iteration 193 126747.199 126747.311 0.0000018
Iteration 194 126746.977 126747.087 0.0000018
Iteration 195 126746.758 126746.867 0.0000017
Iteration 196 126746.542 126746.650 0.0000017
Iteration 197 126746.330 126746.436 0.0000017
Iteration 198 126746.121 126746.225 0.0000017
Iteration 199 126745.915 126746.018 0.0000016
Iteration 200 126745.712 126745.813 0.0000016
Iteration 201 126745.512 126745.612 0.0000016
Iteration 202 126745.315 126745.413 0.0000016
Iteration 203 126745.121 126745.218 0.0000015
Iteration 204 126744.930 126745.025 0.0000015
Iteration 205 126744.742 126744.835 0.0000015
Iteration 206 126744.556 126744.648 0.0000015
Iteration 207 126744.373 126744.464 0.0000015
Iteration 208 126744.193 126744.283 0.0000014
Iteration 209 126744.015 126744.104 0.0000014
Iteration 210 126743.840 126743.927 0.0000014
Iteration 211 126743.668 126743.754 0.0000014
Iteration 212 126743.498 126743.582 0.0000014
Iteration 213 126743.330 126743.414 0.0000013
Iteration 214 126743.165 126743.247 0.0000013
Iteration 215 126743.003 126743.084 0.0000013
Iteration 216 126742.842 126742.922 0.0000013
Iteration 217 126742.684 126742.763 0.0000013
Iteration 218 126742.529 126742.606 0.0000012
Iteration 219 126742.375 126742.452 0.0000012
Iteration 220 126742.224 126742.299 0.0000012
Iteration 221 126742.075 126742.149 0.0000012
Iteration 222 126741.928 126742.001 0.0000012
Iteration 223 126741.783 126741.856 0.0000012
Iteration 224 126741.641 126741.712 0.0000011
Iteration 225 126741.500 126741.570 0.0000011
Iteration 226 126741.362 126741.431 0.0000011
Iteration 227 126741.225 126741.293 0.0000011
Iteration 228 126741.090 126741.157 0.0000011
Iteration 229 126740.958 126741.024 0.0000011
Iteration 230 126740.827 126740.892 0.0000010
Iteration 231 126740.698 126740.762 0.0000010
Iteration 232 126740.571 126740.634 0.0000010
Iteration 233 126740.446 126740.508 0.0000010
LCa.fit convergence in 233 iterations, deviance: 126740.5 on 11178 d.f.
...using 323.1 seconds.
...computing Hessian by numerical differentiation...
...done - in 3.8 seconds.
  user system elapsed
540.334 595.209 327.404

> mod.cmp <- rbind( c( APC$Model$df.res, APC$Model$deviance ),
+                 c( LCP$df, LCP$dev ),
+                 c( LCC$df, LCC$dev ) )
> mod.cmp <- cbind( mod.cmp, mod.cmp[,2]/mod.cmp[,1] )
> rownames( mod.cmp ) <- c("APC", "LcAP", "LcAC")
> colnames( mod.cmp ) <- c("d.f.", "deviance", "dev./d.f")
> round( mod.cmp, 3 )

```

	d.f.	deviance	dev./d.f
APC	11179	128263.4	11.474
LCaP	11178	118830.9	10.631
LCaC	11178	126740.5	11.338

We see that the best fit is provided by the Lee-Carter period model as judged by the deviance.

7. We can plot the estimated effects with the devised `plot` method for LCa objects:

```
> par( mfrow=c(2,3) )
> plot( LCP )
> plot( LCC )
```

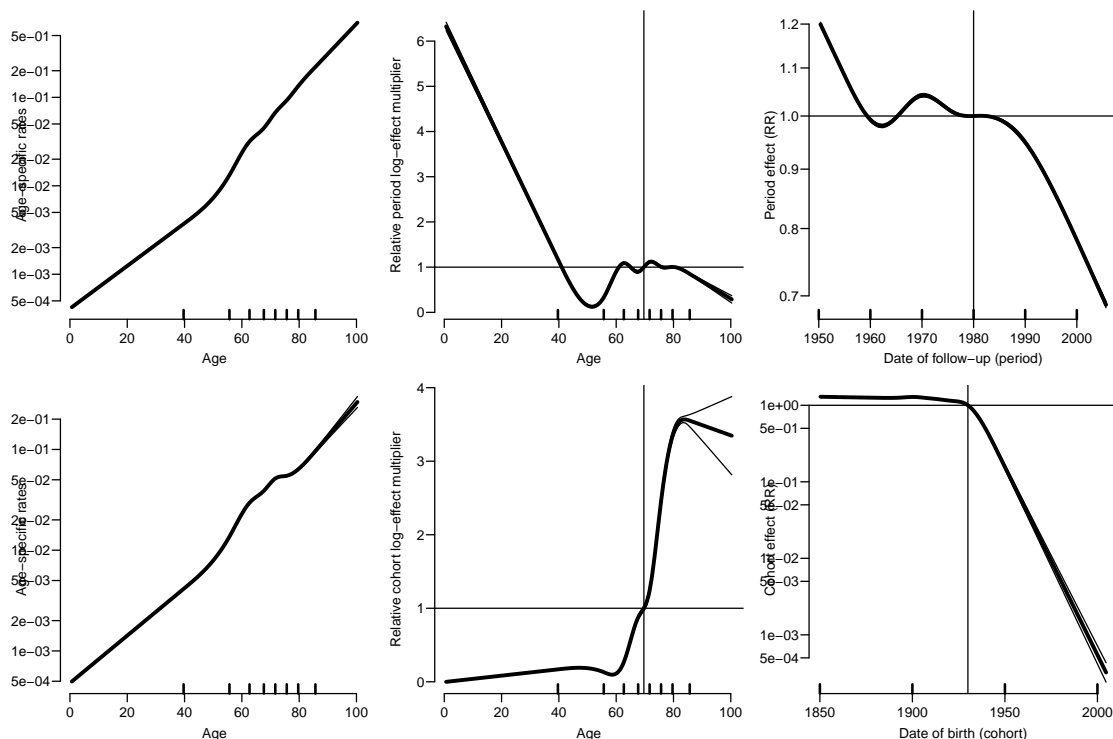


Figure 2.8: *Estimated effects from the Lee-Carter model with period-interaction (top) and cohort-interaction (bottom) for mortality among Czech men.* `../graph/CZM-LCplotP`

8. We may get a better view of the behaviour of the different models, we can plot the predicted rates over the time-span of the data frame at select ages, in this case 50 to 80. We put NAs between the age-classes in order to be able to plot rates in one go:

```
> p.pt <- 1950:2005 ; np <- length(p.pt)
> a.pt <- seq(50,80,5) ; na <- length(a.pt)
> nd <- data.frame( A = rep(a.pt,each=np),
+                  P = rep(p.pt, na),
+                  Y = 1000 )
> # Put NAs to make plots intelligible
> nd[(1:na)*np,] <- NA
```

The models fitted in the `apc.fit` are using specially designed matrices designed to give the desired parametrizations and are therefore not suitable for predictions, so we fit the models explicitly:

```
> AP <- glm( D ~ Ns(A,knots=a.kn)+Ns(P,knots=p.kn),
+           offset=log(Y), family=poisson, data=mM )
> AC <- glm( D ~ Ns(A,knots=a.kn)+           Ns(P-A,knots=c.kn),
+           offset=log(Y), family=poisson, data=mM )
> APC <- glm( D ~ Ns(A,knots=a.kn)+Ns(P,knots=p.kn)+Ns(P-A,knots=c.kn),
+            offset=log(Y), family=poisson, data=mM )
```

With these models we can now produce the fitted rates under each of the models:

```
> fAP <- ci.pred( AP , nd )
> fAC <- ci.pred( AC , nd )
> fAPC <- ci.pred( APC, nd )
> fLCP <- predict( LCP, nd, sim=10000 )*1000
> fLCC <- predict( LCC, nd, sim=10000 )*1000
```

And then we can show the age-specific rates both by period and cohort:

```
> ppm <-
+ function( prd, mod, yl=c(0.2,10)*20 )
+ {
+ matplot( nd$P-nd$A, prd, type="l",lty=c(1,2,2), lwd=c(2,1,1), col="black", log="y",
+         ylim=yl, xlim=1860+c(0,90), ylab="", xlab="Date of birth" )
+ text( par("usr")[1:2]*%*c(0.95,0.05), yl[2], mod, adj=c(0,1) )
+ matplot( nd$P      , prd, type="l",lty=c(1,2,2), lwd=c(2,1,1), col="black", log="y",
+         ylim=yl, xlim=1920+c(0,90), ylab="", xlab="Date of event" )
+ }
> par( mfc=c(2,5), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6, bty="n", las=1 )
> ppm( fAP , "Age-Period" )
> ppm( fLCP, "LCa-Period" )
> ppm( fAPC, "Age-Period-Cohort" )
> ppm( fLCC, "LCa Cohort" )
> ppm( fAC , "Age-Cohort" )
```

We could also show age-specific rates at select dates or age-specific rates in select cohorts, which most conveniently are derived by redefining the `nd` prediction data frame.

```
> # Age-specific rates by period
> p.pt <- 1950+0:8*5 ; np <- length(p.pt)
> a.pt <- 40:90      ; na <- length(a.pt)
> pp <- data.frame( A = rep(a.pt,      np),
+                 P = rep(p.pt,each=na),
+                 Y = 1000 )
> pp[(1:np)*na,] <- NA
> pAP <- ci.pred( AP , pp )
> pAC <- ci.pred( AC , pp )
> pAPC <- ci.pred( APC, pp )
> pLCP <- predict( LCP, pp, sim=10000 )*1000
> pLCC <- predict( LCC, pp, sim=10000 )*1000
> # Age-specific rates by cohort
> c.pt <- 1870+0:8*10 ; nc <- length(c.pt)
```

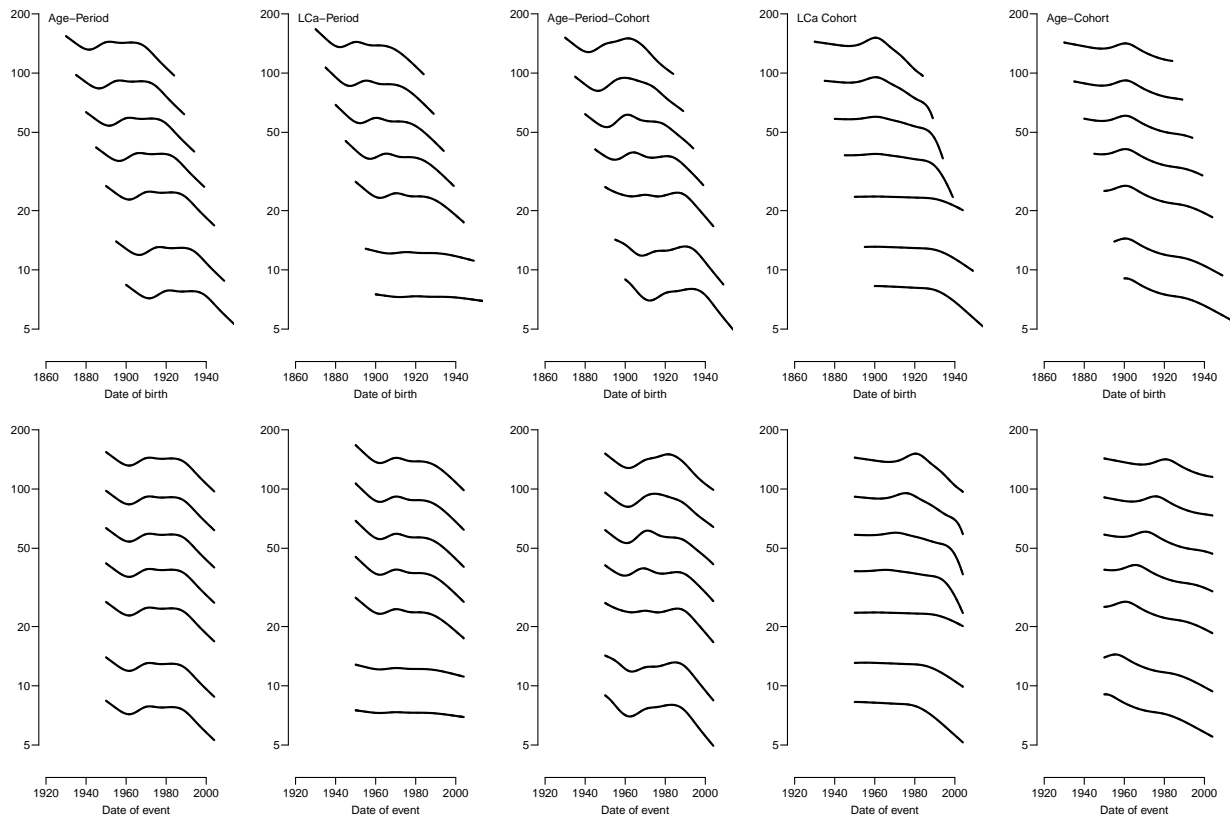


Figure 2.9: Comparison of predicted rates from different models, top panels are rates in ages 50, 55, ..., 80 as they evolve by date of birth; bottom panels as they evolve by date of observation.

../graph/CZM-fL-cmpt

```

> a.pt <- 40:91      ; na <- length(a.pt)
> pc <- data.frame( A = rep(a.pt,      nc),
+                  C = rep(c.pt,each=na),
+                  Y = 1000 )
> pc[(1:nc)*na,] <- NA
> pc$P <- pc$C + pc$A
> pc <- subset( pc, (P>1943 & P<2000) | is.na(P) )
> cAP <- ci.pred( AP , pc )
> cAC <- ci.pred( AC , pc )
> cAPC <- ci.pred( APC, pc )
> cLCP <- predict( LCP, pc, sim=10000 )*1000
> cLCC <- predict( LCC, pc, sim=10000 )*1000

> ppm <-
+ function( prp, prc, mod, yl=c(0.2,20)*20 )
+ {
+   matplot( pp$A, prp, type="l",lty=c(1,2,2), lwd=c(2,1,1), col="black", log="y",
+           ylim=yl, xlim=c(30,90), ylab="", xlab="Age" )
+   text( par("usr")[1:2]*%*c(0.95,0.05), yl[2], mod, adj=c(0,1) )
+   matplot( pc$A, prc, type="l",lty=c(1,2,2), lwd=c(2,1,1), col="black", log="y",
+           ylim=yl, xlim=c(30,90), ylab="", xlab="Age" )
+ }
> par( mfc=c(2,5), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6, bty="n", las=1 )
> ppm( pAP , cAP , "Age-Period" )

```

```

> ppm( pLCP, cLCP, "LCa-Period")
> ppm( pAPC, cAPC, "Age-Period-Cohort")
> ppm( pLCC, cLCC, "LCa Cohort")
> ppm( pAC , cAC , "Age-Cohort")

```

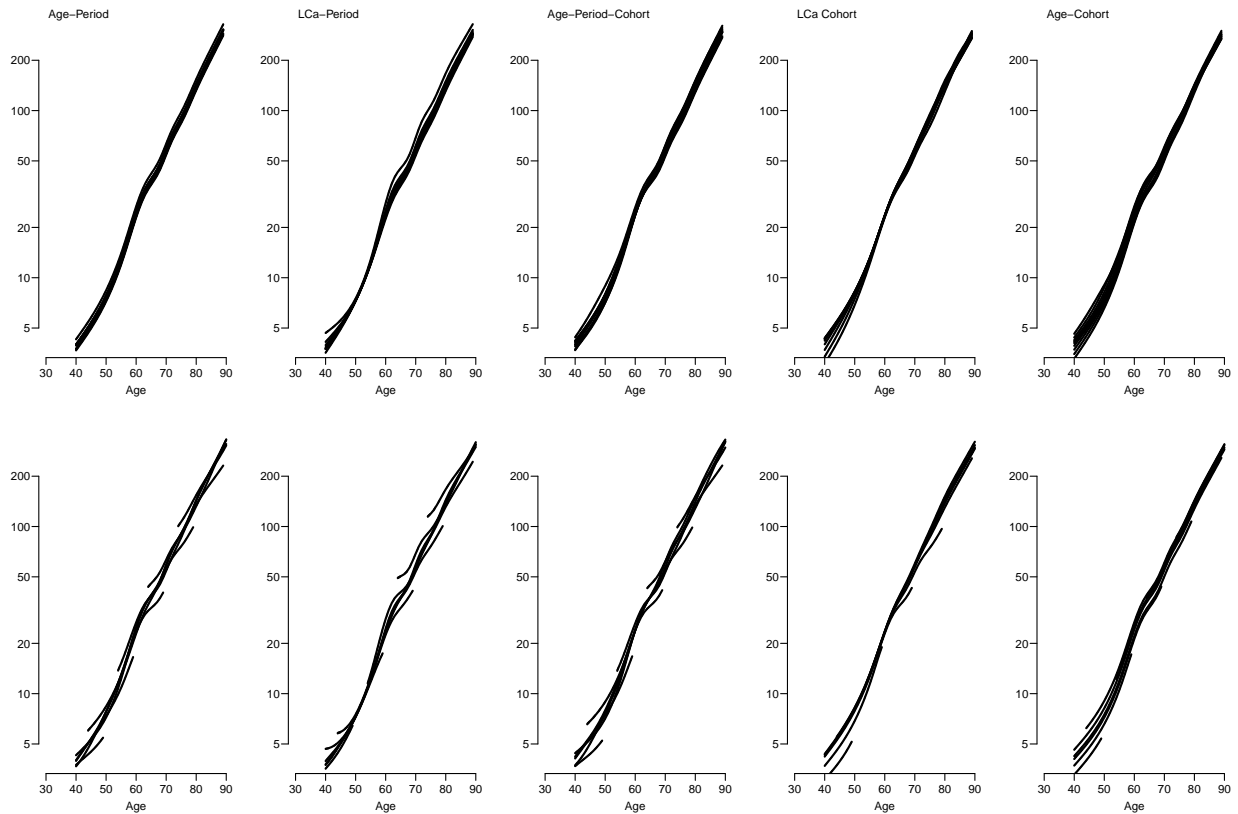


Figure 2.10: Comparison of predicted rates from different models, top panels are age-specific rates at dates 1950, 1955, ... 1990; bottom panels are age-specific rates for dates of birth 1870, 1880, ... 1950.

../graph/CZM-fL-cmpa