Advanced Life Insurance Mathematics

Computer lab on the Lee Carter model

Katrien Antonio, Bavo Campo and Sander Devriendt

Workshop mortality dynamics | March, 2020

Prologue

Introduction

Workshop

https://github.com/katrienantonio/mortality-dynamics

The workshop repo on GitHub, where we upload presentation, code, data sets, etc.

Us

https://katrienantonio.github.io/

katrien.antonio@kuleuven.be & bavo.decock@kuleuven.be & Sander Devriendt

(Katrien) Professor in insurance data science

(Bavo) PhD student in insurance data science

(Sander) former PhD student, now with NBB (Brussels, Belgium)

Checklist

☑ Do you have a fairly recent version of R?

```
version$version.string
```

☑ Do you have a fairly recent version of RStudio?

```
RStudio.Version()$version
## Requires an interactive session but should return something like "[1] ‘1.2.5001’"
```

☑ Have you installed the R packages listed in the software requirements?

Goals of the workshop

Goals of the workshop

This computer lab allows you to:

- visualize concepts discussed in ALIM classes
- improve intuition on these concepts
- translating the ALIM concepts to code
- spend time with R code provided.

You hereby focus on:

- the Lee Carter model
- forecasting the fitted period effects with a random walk with drift.

Outline of the workshop:

- Prologue
- Download and import mortality data
- Fit Lee Carter with iterative LS
- Fit Lee Carter by maximizing a Poisson likelihood
- Forecasting

Download and import mortality data

Life table from HMD

We download the life table for males in Belgium, as available on the HMD.

We store this table as a .txt and import the data in R.

```
library(readr)
Belgium_male_1×1 ← readr::read_delim(
                   file = "./data/Belgium_male_life_table_1×1.txt",
                   delim = " ", col_types = "nnnnnnnnnn",
                   col_names = FALSE, skip = 1)

names(Belgium_male_1×1) ← c("Year", "Age", "mx", "qx", "ax", "lx", "dx", "Lx", "Tx", "ex")
```

We use the {readr} package to control the column types when importing the data. We use the `col_types` argument.

Here, `n` refers to numeric; each of the 10 columns in the data is numeric.

`skip = 1` indicates that the first line (with the variable names) should be skipped when reading the input file.

(This table was downloaded from the HMD on March 30, 2020.)

Variables stored in the life table:

```
names(Belgium_male_1×1)
##  [1] "Year" "Age"  "mx"   "qx"   "ax"   "lx"   "dx"
```

Range of years:

```
min(Belgium_male_1×1$Year)
## [1] 1841
max(Belgium_male_1×1$Year)
## [1] 2018
```

Range of ages:

```
min(Belgium_male_1×1$Age)
## [1] 0
max(Belgium_male_1×1$Age)
## [1] 110
```

In the demos following next, we will only use the data from `start_year` on, e.g.

```
library(dplyr)
start_year ← 1950
end_year ← max(Belgium_male_1×1$Year)
Belgium_male ← filter(Belgium_male_1×1,
                              Year ⩾ start_year)

Belgium_male %>% select(Year, Age, mx,
                                qx, lx, dx, ex) %>%
  slice(1:4) %>% kable(format = 'html')
```

Year	Age	mx	qx	lx	dx	ex
1950	0	0.06180	0.05900	100000	5900	63.81
1950	1	0.00381	0.00380	94100	358	66.80
1950	2	0.00214	0.00214	93742	200	66.05
1950	3	0.00162	0.00162	93542	151	65.19

Fitting the Lee Carter model with an iterative least squares approach

The Lee Carter model with an iterative LS approach

We assume $\mu_{x,t} = m_{x,t}$, which holds under the assumption of a piecewise constant force of mortality.

In the Lee Carter model we propose the following parametric expression for $\log m_{x,t}$:

$$m_{x,t} = \exp\left(\beta_x^{(1)} + \beta_x^{(2)} \cdot \kappa_t^{(2)}\right).$$

We first demonstrate how to estimate the unknown parameters $\beta_x^{(1)}$, $\beta_x^{(2)}$ and $\kappa_t^{(2)}$ by minimizing:

$$\min_{x,t} \left(\log m_{x,t} - \beta_x^{(1)} - \beta_x^{(2)} \cdot \kappa_t^{(2)}\right)^2.$$

Thus, we calibrate the parameters by minimizing the squared differences between the observed death rates and the proposed Lee Carter expression.

The Lee Carter model with an iterative LS approach

We implement the following **iterative LS strategy**:

(1) $\hat{\beta}_x^{(1)} = \frac{1}{T} \quad _t \log m_{x,t}$, an analytic expression, with T the number of years used in the calibration period

(2) define $Z_{x,t} = \log m_{x,t} - \hat{\beta}_x^{(1)}$, then we iteratively solve for $\beta_x^{(2)}$ and $\kappa_t^{(2)}$.

In step (1):

- ignoring the $\beta_x^{(2)} \cdot \kappa_t^{(2)}$ we simply fit a linear model with age-specific intercept to the response, $\log m_{x,t}$

In step (2):

- fixing $\beta_x^{(2)}$ we fit a linear model to $Z_{x,t}$ where the numeric $\beta_x^{(2)}$ (for each record in $Z_{x,t}$) interacts with the factor year covariate, this gives us updated $\hat{\kappa}_t^{(2)}$
- fixing $\kappa_t^{(2)}$ we fit a linear model to $Z_{x,t}$ where the numeric $\kappa_t^{(2)}$ (for each record in $Z_{x,t}$) interacts with the factor age covariate, this gives us updated $\hat{\beta}_x^{(2)}$.

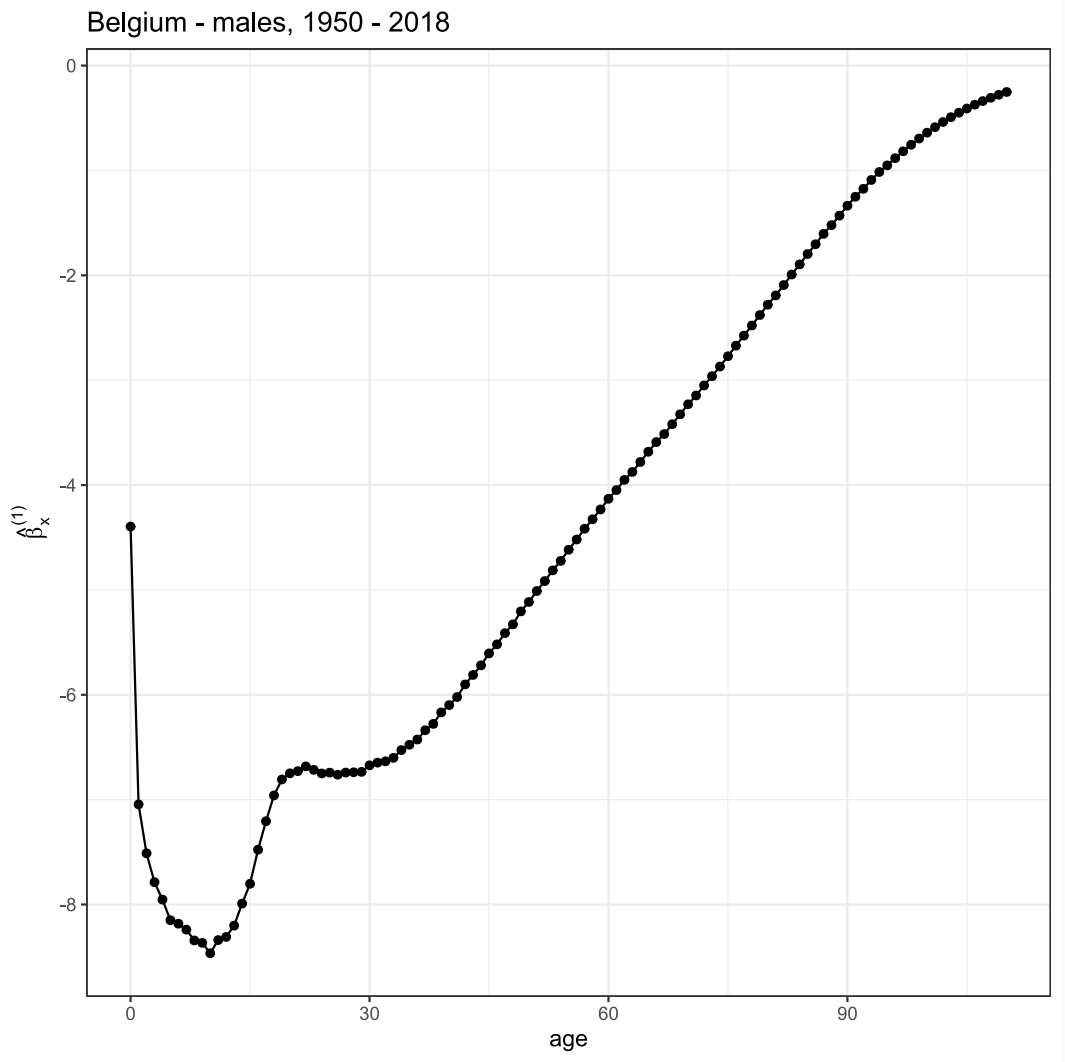
We fit $\hat{\beta}_x^{(1)}$ for each age in the data set.

We fit a linear model with age specific intercept.

This goes as follows:

```
X ← model.matrix(
          ~ as.factor(Belgium_male$Age)
          - 1)
dim(X)
## [1] 7659  111
y ← log(Belgium_male$mx)

alpha_est_expl ← solve(
          crossprod(X)) %*%
                   t(X) %*% y # DIY
alpha_est ← lm(
              y ~ -1 +
              as.factor(Belgium_male$Age))$coef
              # use lm(.) function
              # extract $coef from the fitted
              # lm(.) object
```

We plot the resulting parameter estimates on the right.



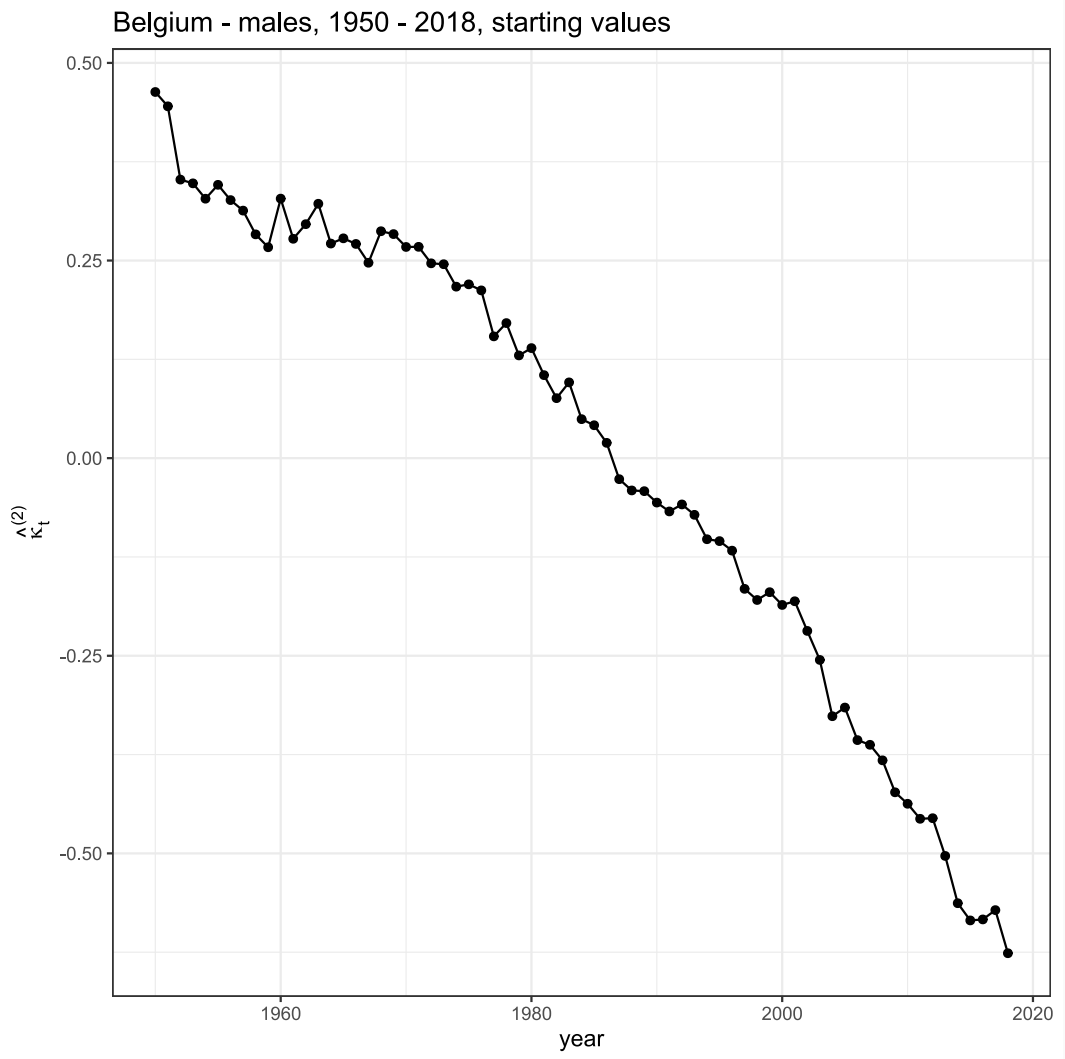
We **initialize** the estimation of the $\kappa_t^{(2)}$'s as follows.

First, we create the new response variable $Z_{x,t}$.

Keep the dimension and specification of `Belgium_male$mx` in mind when subtracting the $\hat{\beta}_x^{[1]}$!

```
Z ← log(Belgium_male$mx) -
     alpha_est[Belgium_male$Age -
                 min(Belgium_male$Age) + 1]

X ← model.matrix(~ as.factor(Belgium_male$Year) - 1)

kappa_est_expl ← solve(crossprod(X)) %*% t(X) %*% Z

kappa_est ← lm(
              Z ~ -1 +
              as.factor(Belgium_male$Year))$coef
```

We plot the resulting parameter estimates on the right.



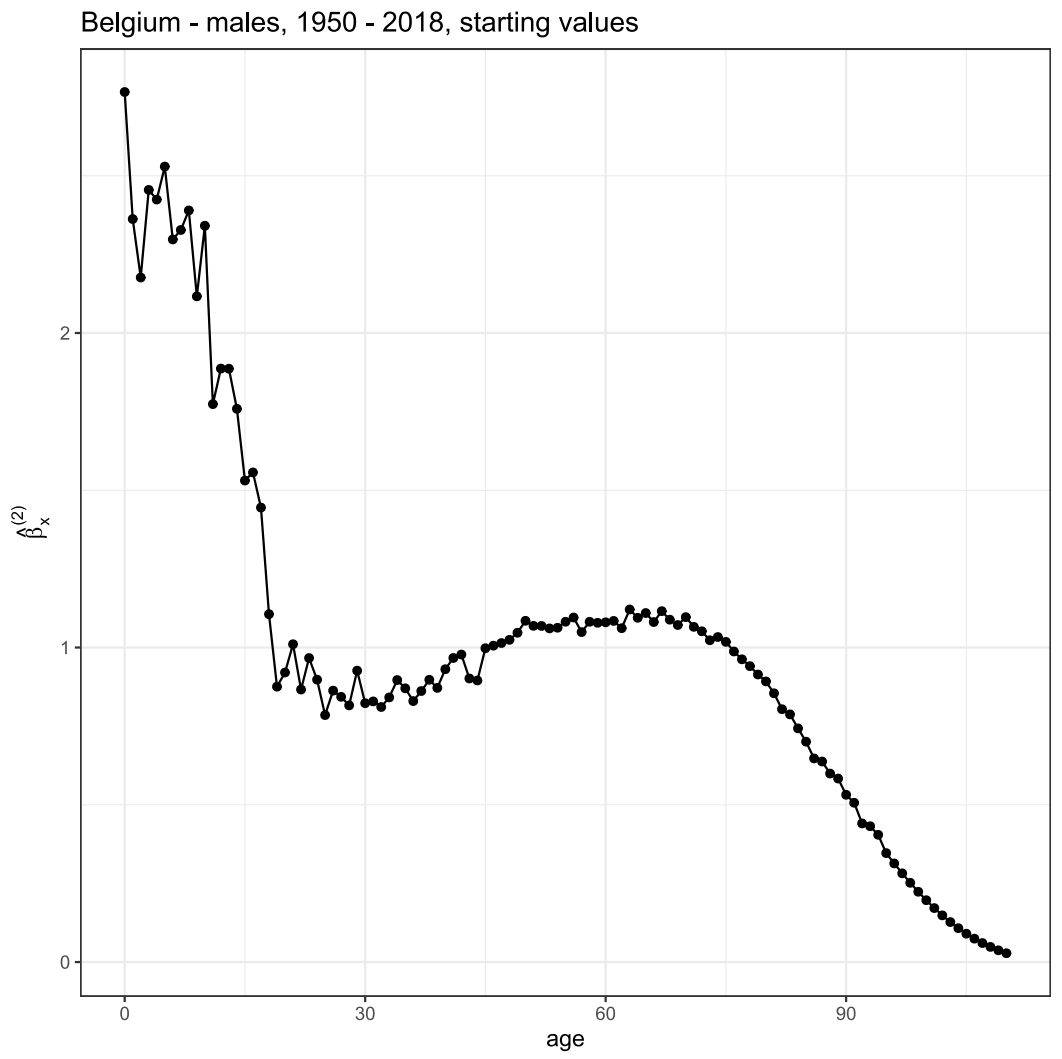
We **initialize** the estimation of the $\beta_x^{(2)}$'s as follows:

```
var_kappa ← kappa_est[Belgium_male$Year -
                        min(Belgium_male$Year) + 1]

X ← model.matrix(~
              as.factor(Belgium_male$Age):var_kappa
              - 1)

beta_est_expl  ← solve(crossprod(X)) %*% t(X) %*% Z

beta_est ← lm(
          Z ~ -1 +
          as.factor(Belgium_male$Age):var_kappa)$coef
```

Mind the interaction constructed`:

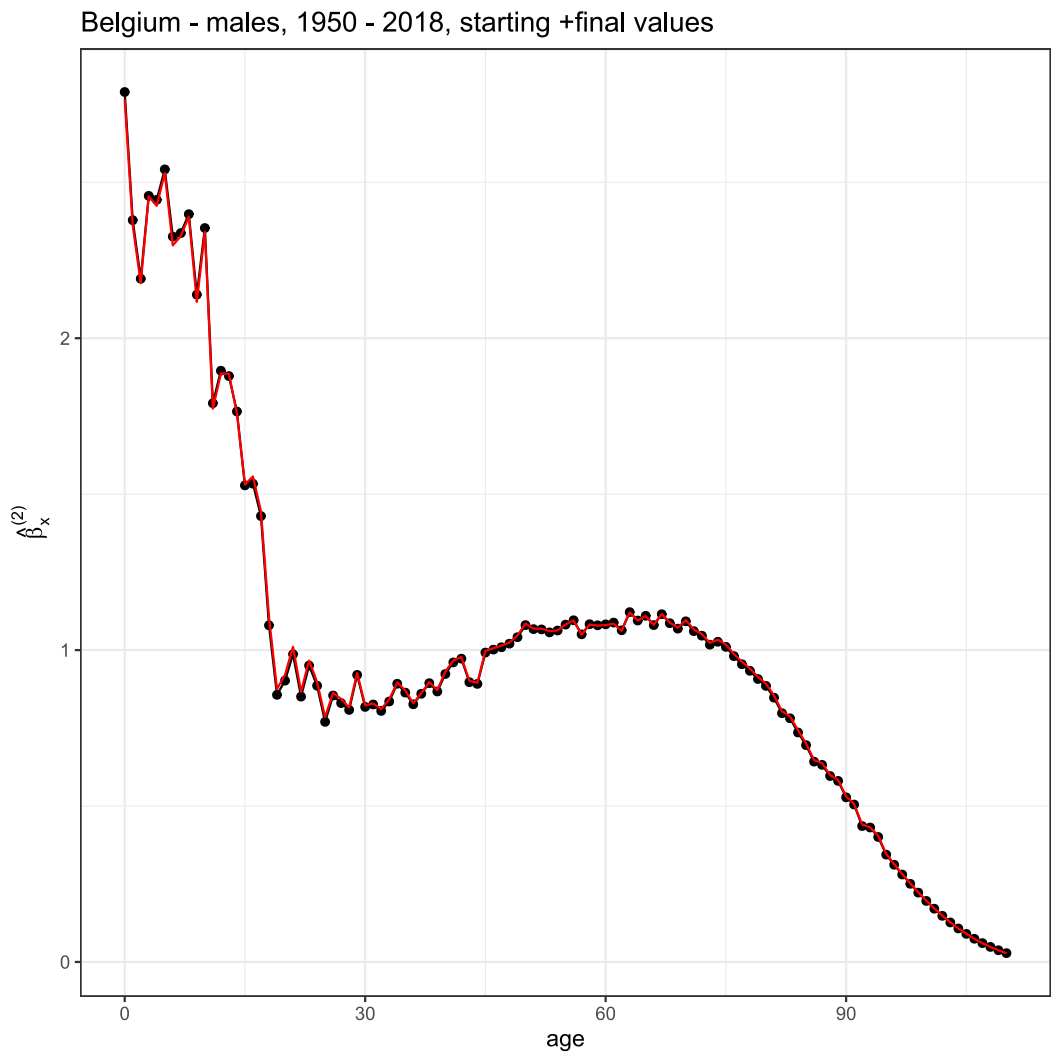
`var_kappa` is a numeric variable with the same length as `Z`

via `as.factor(Belgium_male$Age):var_kappa` we build the interaction between this variable and an age-specific intercept.

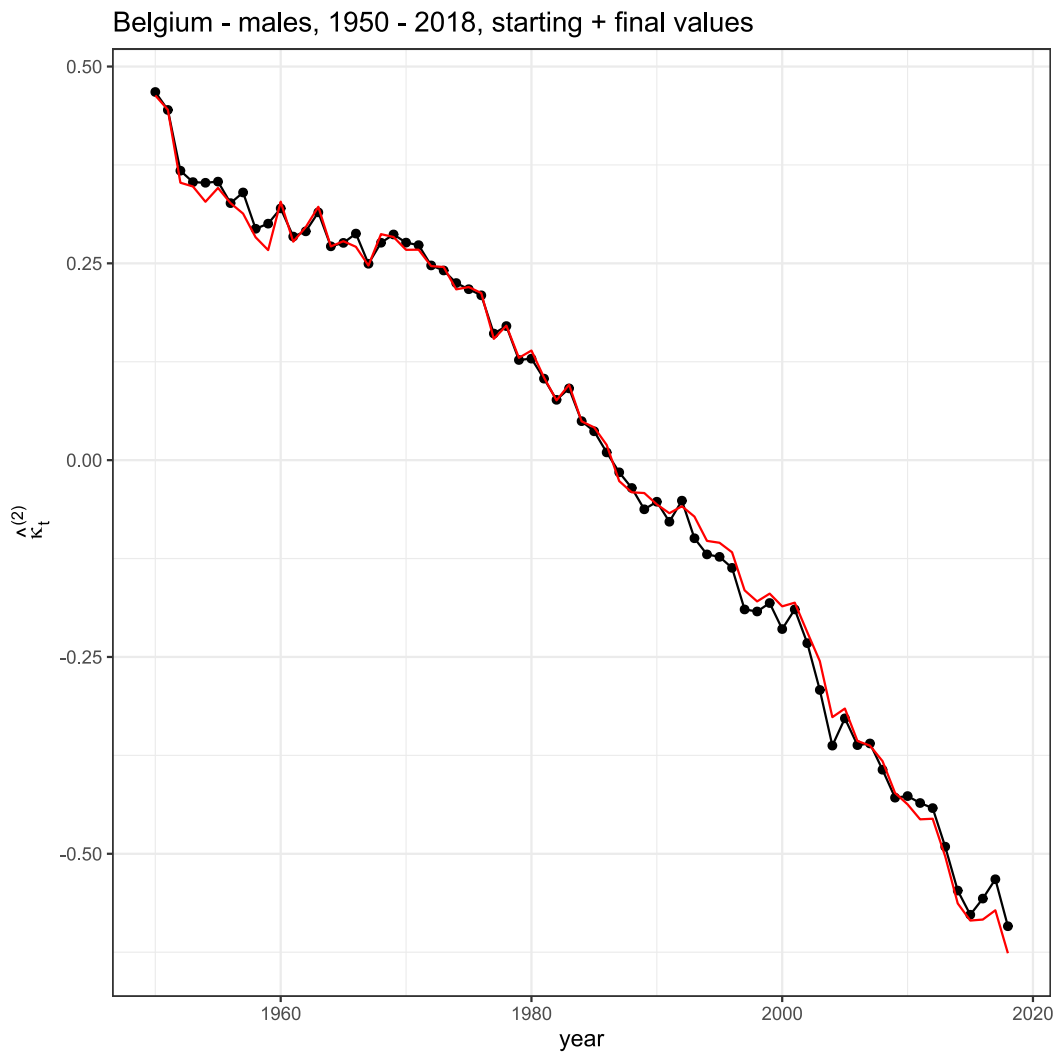


```
converged = F
iter      = 1

while(!converged){
  beta_est_old  = beta_est
  kappa_est_old = kappa_est

  # (2): estimate kappa's
  var_beta = beta_est[Belgium_male$Age - min(Belgium_male$Age) + 1]
  X        = model.matrix(~ as.factor(Belgium_male$Year):var_beta - 1)
  kappa_est = solve(crossprod(X)) %*% t(X) %*% Z

  # (3): estimate beta's
  var_kappa = kappa_est[Belgium_male$Year - min(Belgium_male$Year) + 1]
  X         = model.matrix(~ as.factor(Belgium_male$Age):var_kappa - 1)
  beta_est   = solve(crossprod(X)) %*% t(X) %*% Z

  # stopping criterion
  converged =
  max(abs(beta_est - beta_est_old) / abs(beta_est_old), abs(kappa_est - kappa_est_old) / abs(kappa_est_old)) < 1e-8
  iter = iter + 1
  if(iter %% 1e2 == 0)
    cat("\n\nIteration number", iter, "\n\n")
}
```

We plot the initial estimates and the parameter estimates obtained **after convergence**.



We plot the initial estimates and the parameter estimates obtained **after convergence**.



Finally, we make sure the parameter estimates satisfy the (usual) set of Lee Carter **parameter constraints**.

That is

$$\sum_t \kappa_t^{(2)} = 0$$

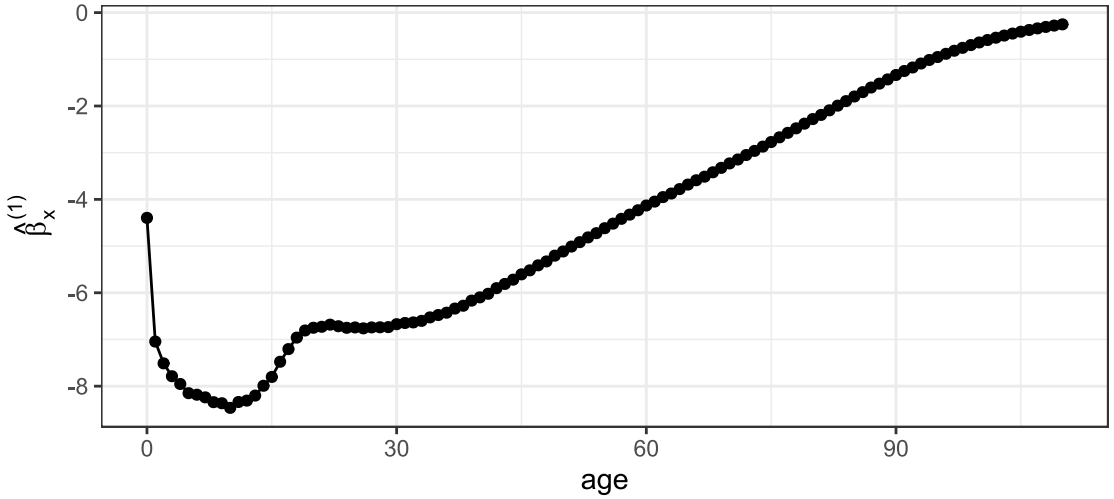
$$\sum_x \beta_x^{(2)} = 1.$$

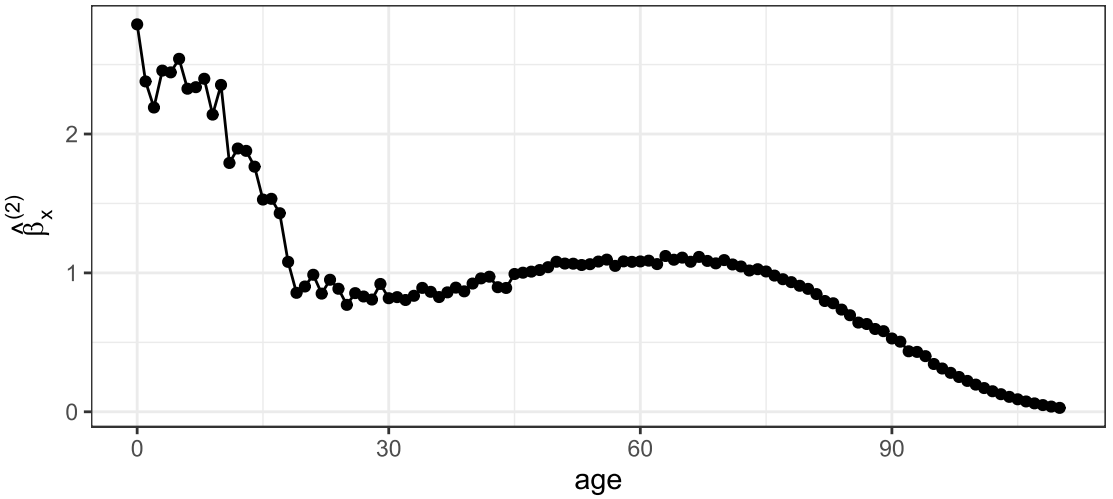
```
## apply constraints ##
beta_est_LS  = beta_est / sum(beta_est)
kappa_est_LS = (kappa_est - mean(kappa_est)) * sum(beta_est)
alpha_est_LS = alpha_est + beta_est * mean(kappa_est)
```

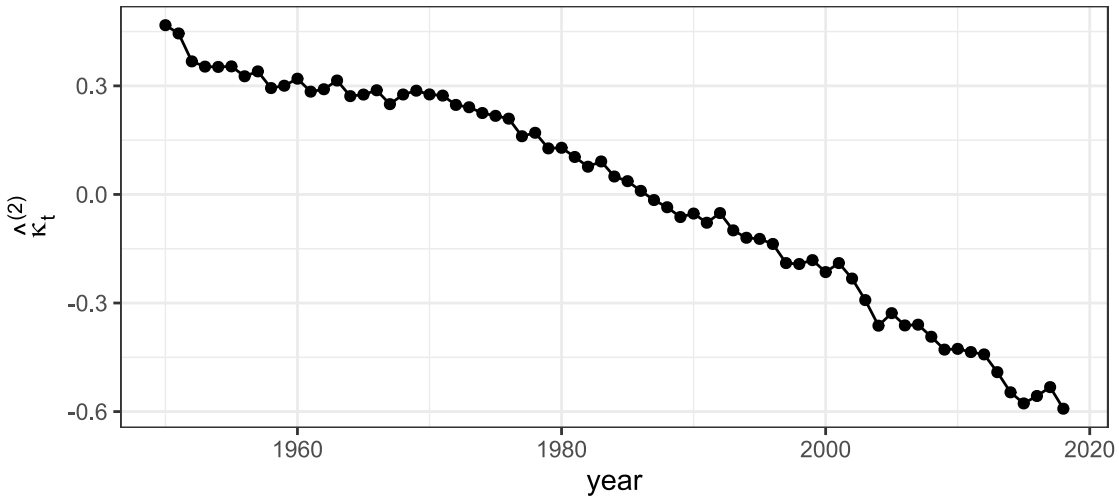
Final check

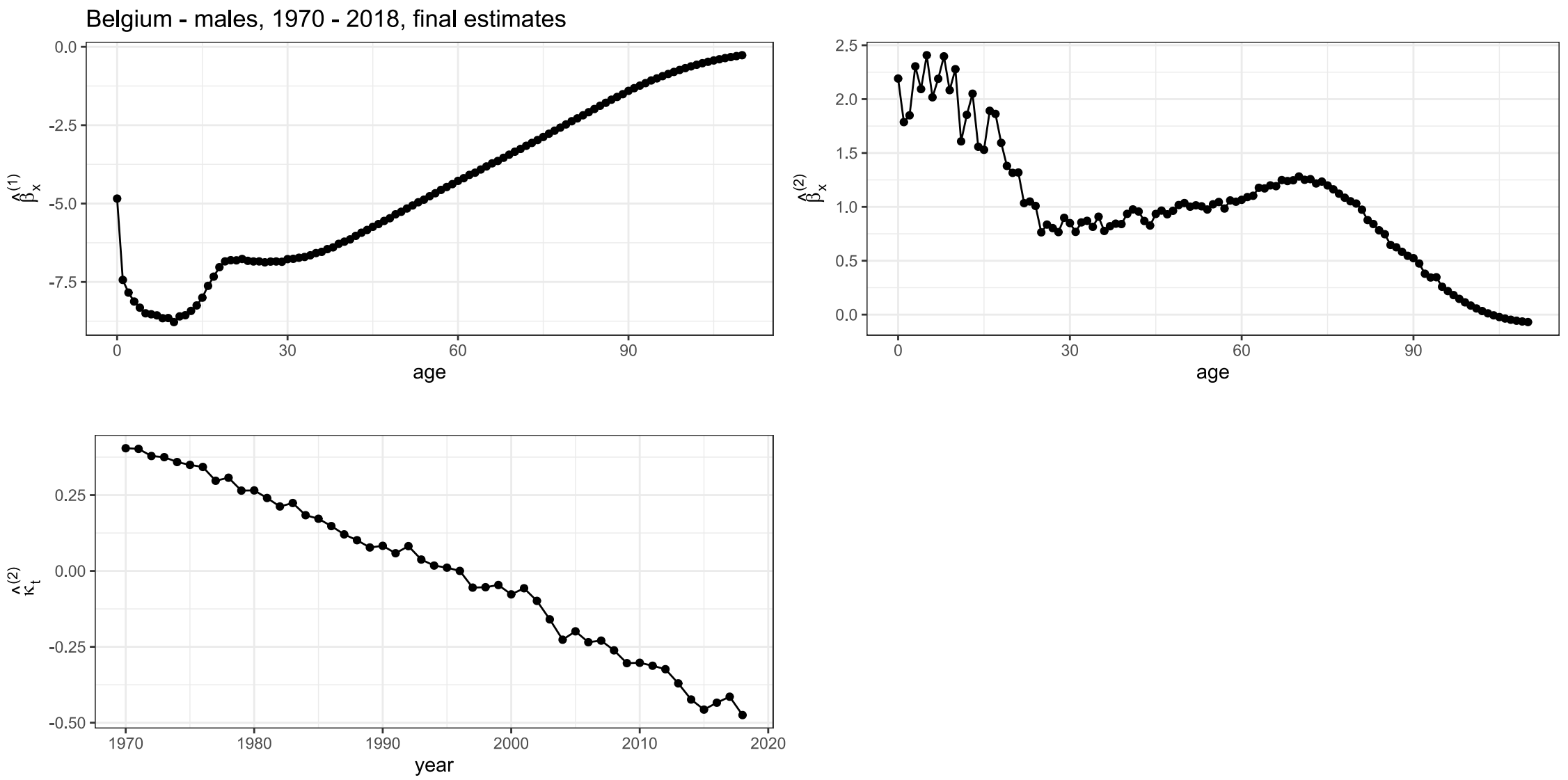
```
sum(beta_est_LS)
## [1] 1
sum(kappa_est_LS)
## [1] 4.107825e-14
```

Belgium - males, 1950 - 2018, final estimates








Belgium - males, 1970 - 2018, final estimates
$\hat{\beta}_x^{(1)}$
0.0
-2.5
-5.0
-7.5
0
30
60
90
age
$\hat{\beta}_x^{(2)}$
2.5
2.0
1.5
1.0
0.5
0.0
0
30
60
90
age
$\hat{\kappa}_t^{(2)}$
0.25
0.00
-0.25
-0.50
1970
1980
1990
2000
2010
2020
year

Fit the Lee Carter model by maximizing a Poisson likelihood

Maximizing a Poisson likelihood

We put focus on the following Poisson model for the deaths $D_{x,t}$

$$D_{x,t} \sim \text{POI}(e_{x,t} \cdot \mu_{x,t})$$
$$\mu_{x,t} = \exp\left(\beta_x^{(1)} + \beta_x^{(2)} \cdot \kappa_t^{(2)}\right).$$

The log-likelihood then follows:

$$L(\beta^{(1)}, \beta^{(2)}, \kappa^{(2)}) = \sum_x \sum_t d_{xt}(\beta_x^{(1)} + \beta_x^{(2)} \cdot \kappa_t^{(2)}) - e_{xt} \exp\left(\beta_x^{(1)} + \beta_x^{(2)} \cdot \kappa_t^{(2)}\right) + c.$$

Here, $d_{x,t}$ and $e_{x,t}$ are the observed deaths and exposures, respectively.

```r
library(demography)
country ← c("BEL", "Belgium")
user    ← "vuulenbak42@hotmail.com"
pw      ← "testEAA"
df      ← hmd.mx(country[1], user , pw , country[2])
years   ← 1970:max(df$year)
ages    ← 0:89
```

```r
df  ← demography::extract.years(
                    df, years = years)
df  ← demography::extract.ages(
                    df, ages = ages,
                    combine.upper = FALSE)

min(df$year)
## [1] 1970
max(df$year)
## [1] 2018
max(df$age)
## [1] 89
```

To construct the Poisson likelihood, we need observations on d_{xt} and e_{xt}.

We specify the calibration period as `years`.

We specify the age range as `ages`.

We use `extract.years` and `extract.ages` from the {demography} package to subset `df` accordingly.

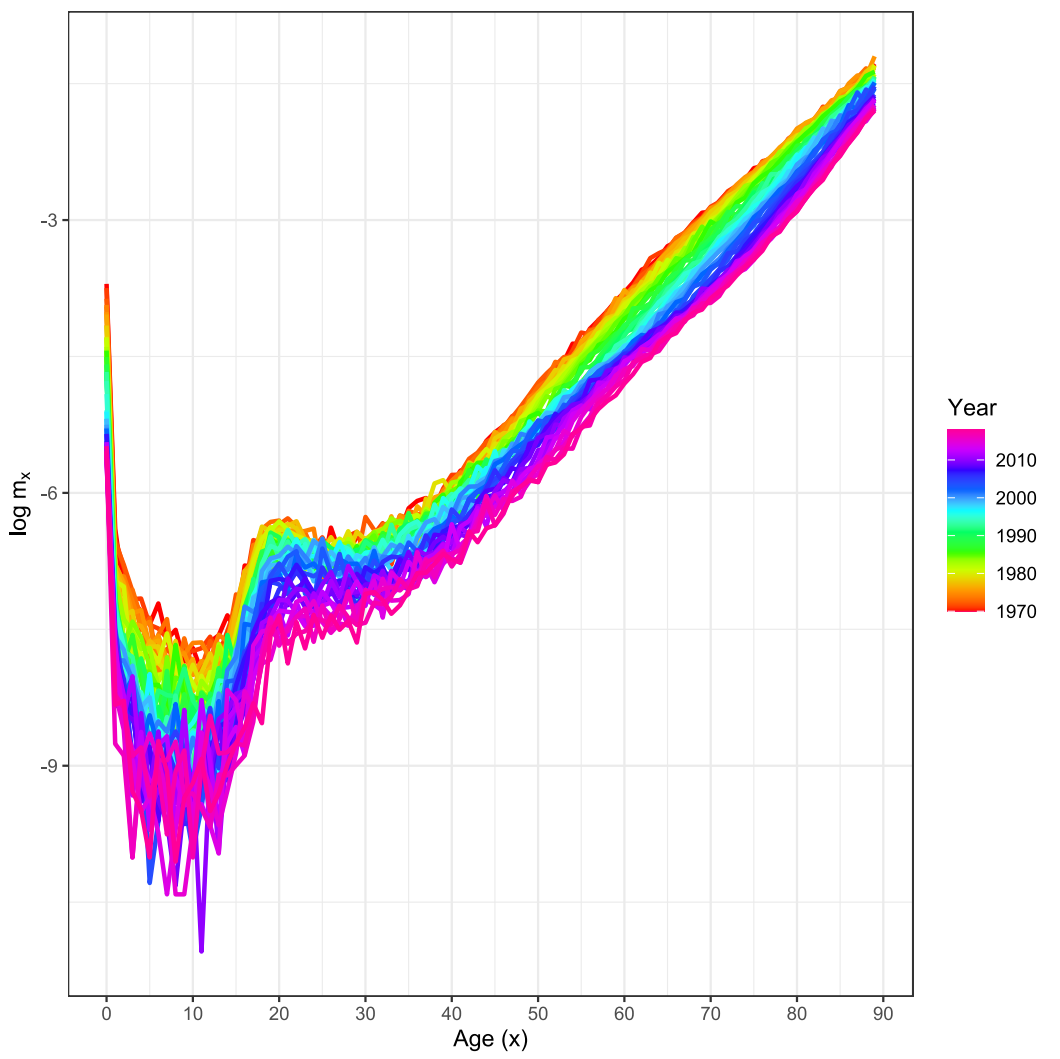
```
etx ← t(df$pop$male)
dim(etx)
## [1] 49 90

dtx ← etx * t(df$rate$male)
dim(dtx)
## [1] 49 90
```

We extract the exposures and put these in `etx`.

Here, years are in the rows and ages in the columns of the list. `t(.)` is for matrix transpose.

The deaths are not directly stored in `df`, but follow from $m_{x,t} \cdot e_{x,t}$.

```r
df ← expand.grid(Year = years, Age = ages)
      # age 0 across all years, age 1 ...
rates ← dtx/etx
      # years in rows, ages in columns
df$rates ← log(as.vector(rates))
      # age 0 across all years, age 1 ...
names(df) ← c("Year", "Age", "logRate")

p ← ggplot(df,
          aes(x = Age, y = logRate, group = Year)) +
     geom_line(aes(colour = Year),
               size = 1, linetype = 1) +
     scale_colour_gradientn(colours = rainbow(10)) +
     scale_x_continuous(
                      breaks =
                      seq(ages[1], tail(ages, 1) + 1,
                      10)) +
     theme_bw() + ylab(expression("log" ~ m[x])) +
     xlab("Age (x)")
p
```



In the `ALIM_computer_lab_2.R` we offer two implementations to optimize the Poisson likelihood with univariate Newton-Raphson steps.

(1) Own written routine `LCNRopt ← function(dxt, ext, eps = 1e-4, maxiter = 1e4)`

(2) The `fit701(ages, years, Etx, Dtx, matrix(1, length(years), length(ages)))` function from the `fitModels.R` script.

This function is written by prof. Andrew Cairns, see LifeMetrics software.

`fit701(.)` calibrates Lee Carter, `fit702(.)` fits the Renshaw Haberman model, `fit703(.)` the Age-Period-Cohort (APC) model, `fit705(.)` the CBD model, `fit706(.)` the CBD model with cohort effect, and so on.

In both implementations, the following types of iterative steps are crucial:

$$\hat{\kappa}_t^{(k+1)} = \hat{\kappa}_t^{(k)} - \frac{\sum_x d_{xt} - e_{xt} \exp\left(\hat{\alpha}_x^{(k+1)} + \hat{\beta}_x^{(k)} \hat{\kappa}_t^{(k)}\right) \hat{\beta}_x^{(k)}}{-\sum_x e_{xt} \exp\left(\hat{\alpha}_x^{(k+1)} + \hat{\beta}_x^{(k)} \hat{\kappa}_t^{(k)}\right) \left(\hat{\beta}_x^{(k)}\right)^2}.$$

We use parametrization $\mu_{x,t} = \exp\left(\alpha_x + \beta_x \cdot \kappa_t\right)$ for ease of notation.

Similar expressions are given in the lecture sheets for the other parameters.

For example, in `fit701(.)` the function `llmax2D(.)` does the NR step for the period effect

```
thetat = b2*ev*exp(b1+b3*g3)
s1     = sum(dv*b2*wv)
f0     = sum((exp(k20*b2)*thetat)*wv)-s1
df0    = sum((exp(k20*b2)*b2*thetat)*wv)
k21    = k20-f0/df0
```

```
source('./scripts/fitModels.R')

LCfit701 ← fit701(ages, years, etx, dtx,
                  matrix(1, length(years),
                         length(ages)))

names(LCfit701)
```

```
## -21530.77 → -20501.14  -›–20314.4
## -20294.5 → -20294.31  -›–20291.36
## -20290.4 → -20290.39  -›–20290.1
## -20290.01 → -20290.01  -›–20289.98
## -20289.97 → -20289.97  -›–20289.96
## -20289.96 → -20289.96  -›–20289.96
## -20289.96 → -20289.96  -›–20289.96
## -20289.96 → -20289.96  -›–20289.96
```

```
##  [1] "beta1"       "beta2"       "beta3"       "kapp
##  [6] "x"           "y"           "cy"          "wa"
## [11] "mhat"        "ll"          "BIC"         "npar
```

We call `fit701(.)` from the `fitModels.R` script.

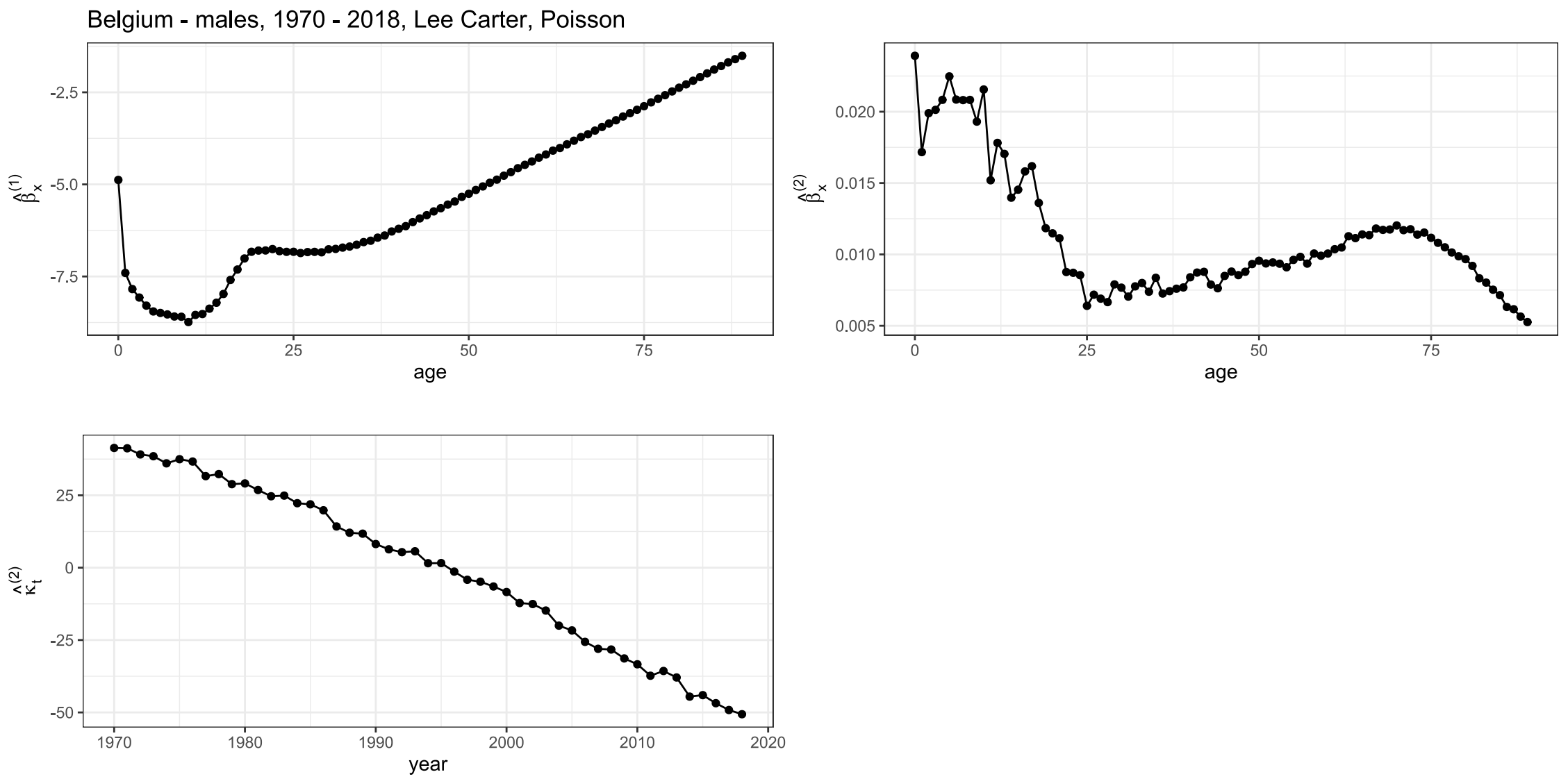
We store the results in the object `LCfit701`.

Useful attributes in this object are:

- `beta1` the estimates for $\beta_x^{(1)}$ (or α_x)
- `beta2` the estimates for $\beta_x^{(2)}$ (or β_x)
- `kappa2` the estimates for $\kappa_t^{(2)}$ (or κ_t)
- the BIC
- `epsilon` the residuals
- `mhat` the fitted $\hat{m}_{x,t}$.

The residuals are Pearson residuals for a Poisson setting

```
epsilon = (dtx-etx * mhat)/sqrt(etx * mhat)
```


Belgium - males, 1970 - 2018, Lee Carter, Poisson
$\hat{\beta}_x^{(1)}$
-2.5
-5.0
-7.5
0
25
50
75
age
$\hat{\beta}_x^{(2)}$
0.020
0.015
0.010
0.005
0
25
50
75
age
$\hat{\kappa}_t^{(2)}$
25
0
-25
-50
1970
1980
1990
2000
2010
2020
year

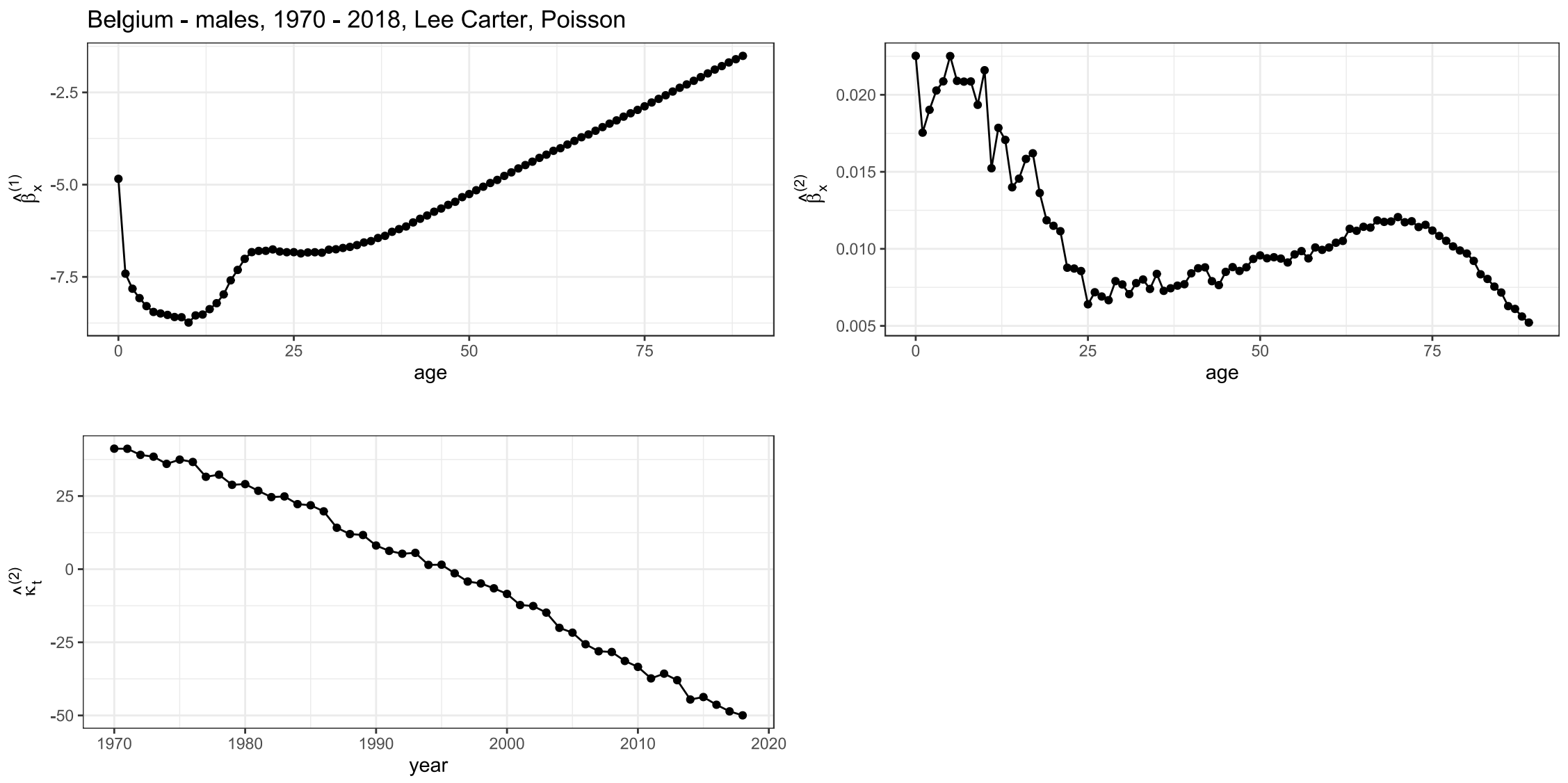
If you find the steps in `fit701(.)` too overwhelming you can take a look at our DIY routine `LCNRopt(.)`

```
LeeCarterNR ← LCNRopt(dxt   = t(Dtx),
                      ext   = t(Etx),
                      eps = 1e-4, maxiter = 2e3)
```

where you recognize similar univariate NR steps to update the parameter estimates

```
K0i = K1i
K1i = K0i -
(sum( (dxti -  exti * exp(B1 + B2 * K0i)) * B2 ))  /
          - (sum(exti * exp(B1 + B2 * K0i) * B2^2))
```

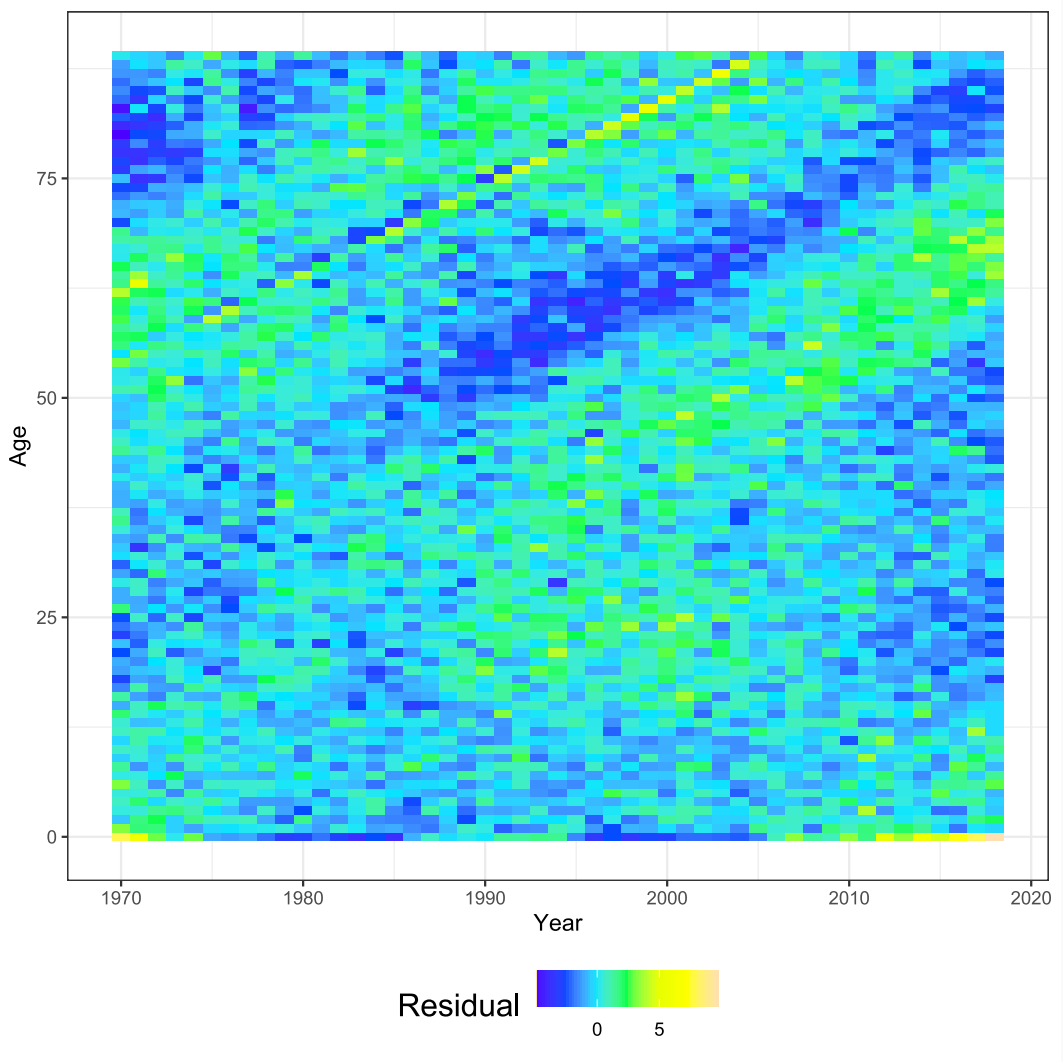
Let's have a look at the parameter estimates obtained with this routine!


Belgium - males, 1970 - 2018, Lee Carter, Poisson
$\hat{\beta}_x^{(1)}$
-2.5
-5.0
-7.5
0
25
50
75
age
$\hat{\beta}_x^{(2)}$
0.020
0.015
0.010
0.005
0
25
50
75
age
$\hat{\kappa}_t^{(2)}$
25
0
-25
-50
1970
1980
1990
2000
2010
2020
year

We construct a {ggplot2} heatmap of the residuals produced by `fit701(.)`.

```
grid ← expand.grid(period = years, age = ages)
grid$res ← as.vector(LCfit701$epsilon)
names(grid) ← c("Year", "Age", "Residual")

p ← ggplot(grid, aes(x = Year, y = Age)) +
     geom_tile(aes(fill = Residual)) +
     scale_fill_gradientn(colours =  topo.colors(7)) +
     theme_bw() +
     theme(legend.position = "bottom",
           legend.title = element_text(size = 15))
p
```

```
head(grid)
##   Year Age   Residual
## 1 1970   0  7.9057034
## 2 1971   0  5.9636049
## 3 1972   0  2.9845978
## 4 1973   0  0.2741458
## 5 1974   0  2.9748547
## 6 1975   0 -0.9675698
```

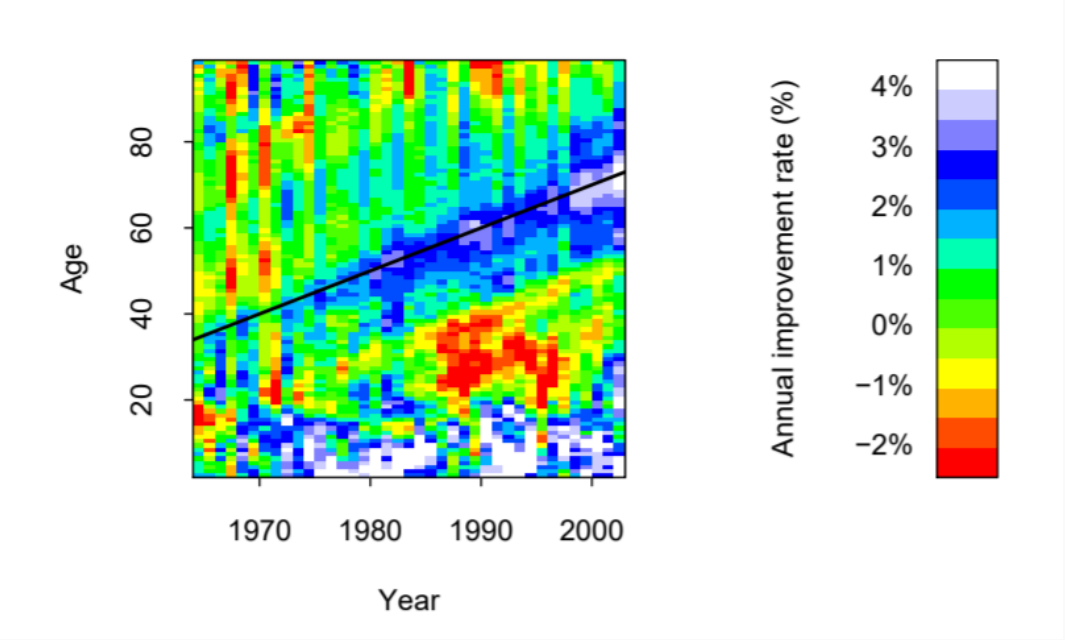


For a more in depth discussion of **cohort effects**, we refer to Cairns et al. (2009) on A quantitative comparison of stochastic mortality models using data from England and Wales and the United States.

In the longer version of this paper (available online) the authors discuss the plot shown on the right (Figure 3 on page 7).

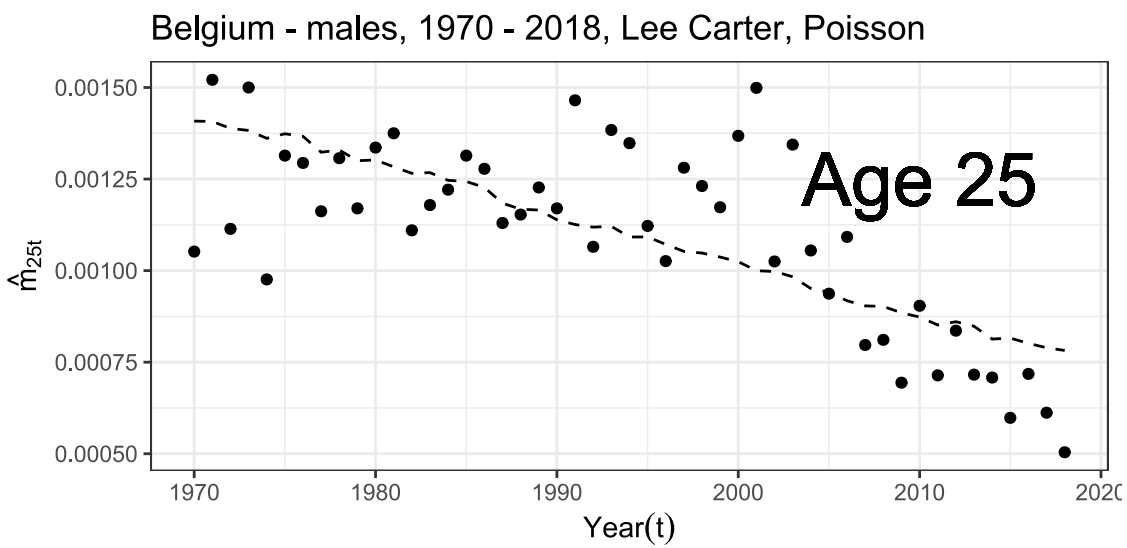
This is often used as the rationale for including cohort effects in mortality forecasting models.

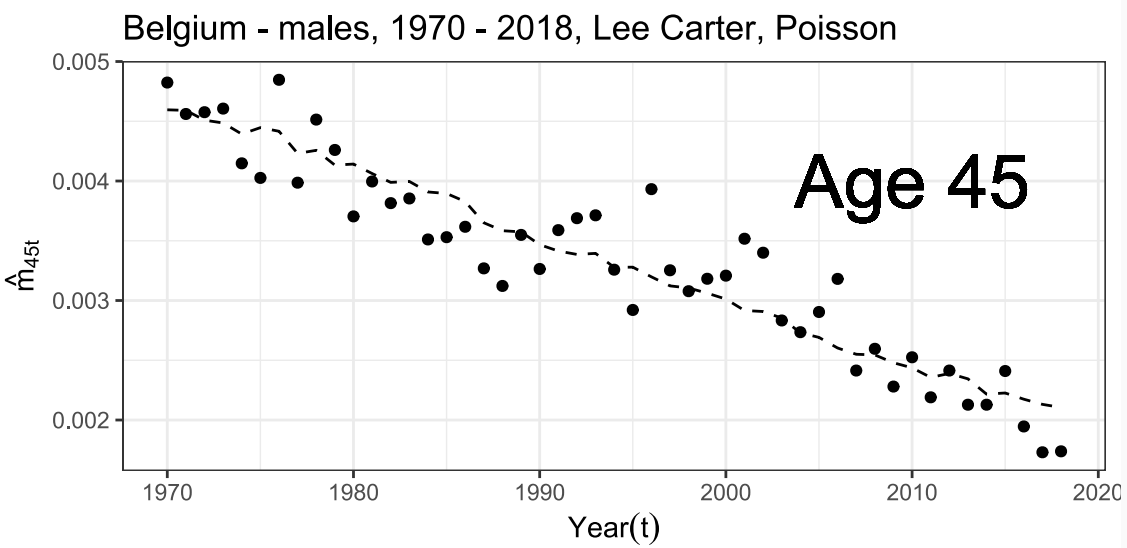
However, calibrating and forecasting such cohort effects comes with a lot of difficulties (as discussed in class)!

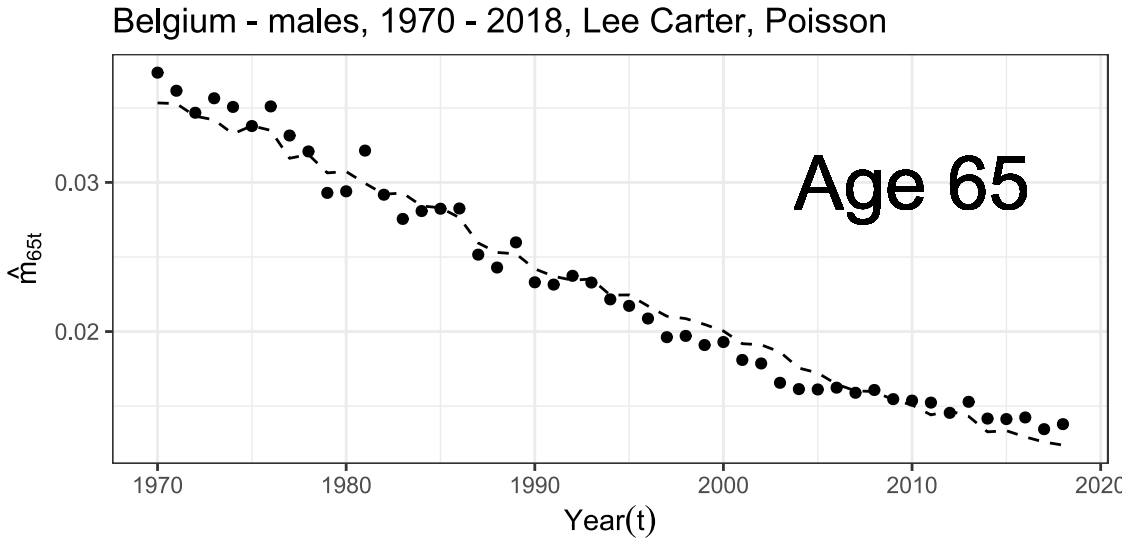


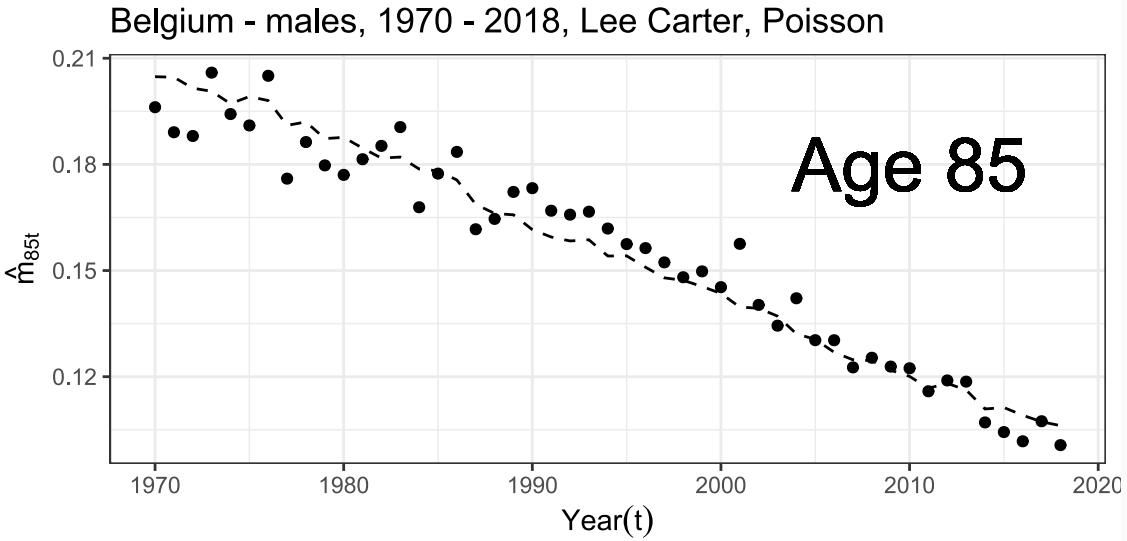
This Figure shows **improvement rates** in mortality for England & Wales by calendar year and age relative to mortality rates at the same age in the previous year. Red cells imply that mortality is deteriorating; green small rates of improvement, and blue and white strong rates of improvement.

The black diagonal line follows the **progress of the 1930 cohort**.


Belgium - males, 1970 - 2018, Lee Carter, Poisson
Age 25
$\hat{m}_{25t}$
0.00150
0.00125
0.00100
0.00075
0.00050
1970
1980
1990
2000
2010
Year(t)


Belgium - males, 1970 - 2018, Lee Carter, Poisson
Age 45
$\hat{m}_{45t}$
0.005
0.004
0.003
0.002
1970
1980
1990
2000
2010
Year(t)


Belgium - males, 1970 - 2018, Lee Carter, Poisson
Age 65
$\hat{m}_{65t}$
0.03
0.02
1970
1980
1990
2000
2010
Year(t)


Belgium - males, 1970 - 2018, Lee Carter, Poisson
Age 85
$\hat{m}_{85t}$
0.21
0.18
0.15
0.12
1970
1980
1990
2000
2010
Year(t)

Forecasting using a random walk with drift

We use the ARIMA toolbox to find a suitable time series model for the fitted parameter estimates $\hat{\kappa}_t^{(2)}$.

We use the random walk with drift, or ARIMA(0,1,0), for the $\hat{\kappa}_t^{(2)}$ in Lee Carter.

This implies

$$\begin{aligned} \hat{\kappa}_t^{(2)} &= \hat{\kappa}_{t-1}^{(2)} + \theta + \epsilon_t \\ \epsilon_t &\sim N(0, \sigma^2), \end{aligned}$$

where θ is the drift parameter and ϵ_t the i.i.d. error terms.

We have to estimate the unknown parameters θ and σ^2.

```r
library(forecast)

time_series ← Arima(LCfit701$kappa2,
                    order = c(0, 1, 0),
                    include.drift = TRUE)

time_series
## Series: LCfit701$kappa2
## ARIMA(0,1,0) with drift
##
## Coefficients:
##         drift
##       -1.9160
## s.e.   0.2615
##
## sigma^2 estimated as 3.353:  log likelihood=-96.64
## AIC=197.29   AICc=197.55   BIC=201.03
```

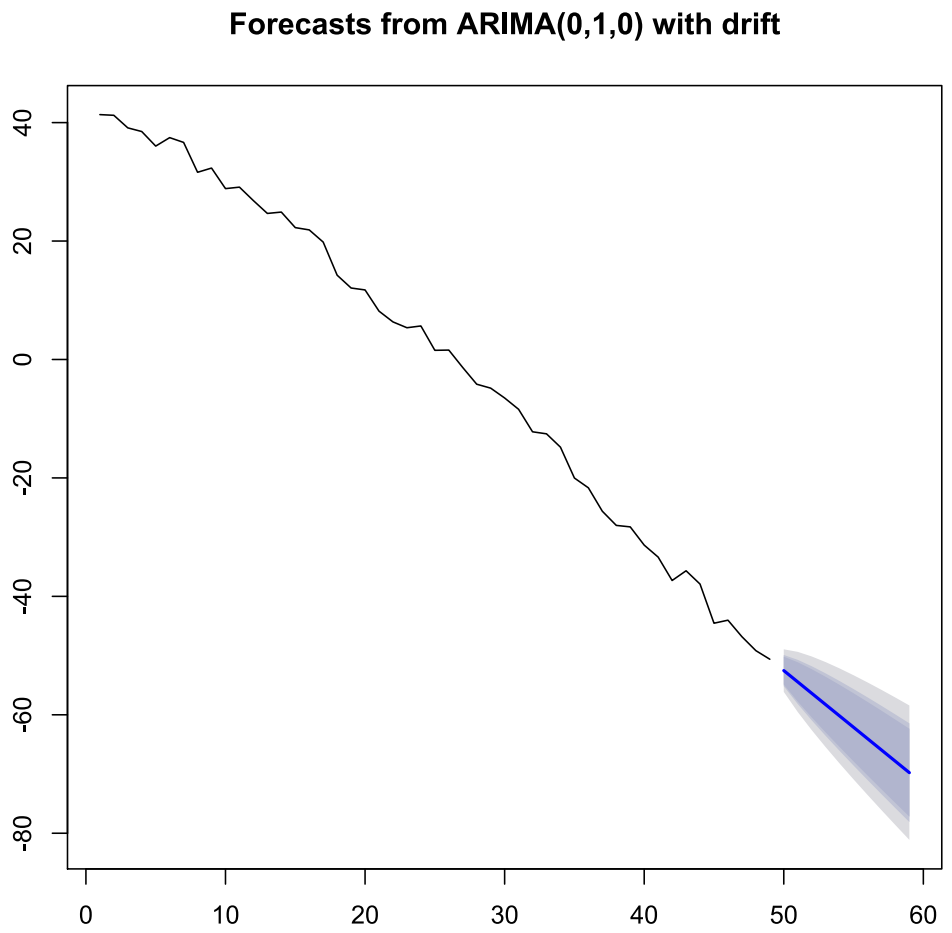
We use the {forecast} package to fit the RW with drift time series model to the calibrated $\hat{\kappa}_t^{(2)}$ parameters.

We use the `Arima(.)` function, with order `c(0,1,0)` to specify the random walk.

More general, in the `Arima(.)` function the components `(p, d, q)` are the AR order, the degree of differencing and the MA order.

We put `include.drift = TRUE` to include the drift term.

```
plot(forecast(time_series, level = c(80, 85, 95)))
```



We use the `forecast` function, together with the `time_series` object.

We put `level = c(80, 85, 95)` to specify the confidence level for the prediction intervals.

The function `plot` produces a plot of the forecasts and prediction intervals.

Projecting the force of mortality and mortality rates

In the LifeMetrics code the script `simModels.R` includes the function `sim2001(.)` to project the fitted Lee Carter model.

```
sim2001 ← function(xx, yy, beta1v, beta2v, kappa2v, mtxLastYear,
                   nsim = 100, tmax = 20, nyears = 0, x0 = 65, fixStartPoint=FALSE)
```

Here:

- `xx` refers to the ages used in the fitting procedure
- `yy` are the calendar years used in the fitting procedure
- `beta1v, beta2v, kappa2v` the parameters fitted for the Lee Carter model
- `nsim = 100` the number of paths generated and `tmax = 20` the forecasting horizon
- `nyears` the number of years used to calibrate the RW with drift model.

```r
source('./scripts/simModels.R')

sim_LC = sim2001(
  xx = LCfit701$x,
  yy = LCfit701$y,
  beta1v = LCfit701$beta1,
  beta2v = LCfit701$beta2,
  kappa2v = LCfit701$kappa2,
  nsim = 10000,
  tmax = 50,
  nyears = length(years)
)

names(sim_LC)
## [1] "y"   "xx"  "xv"  "mu2" "v22" "dda" "pa"  "tpa"
```

We call the `sim2001` function from the `simModels.R` script to generate scenarios for the random walk with draft.

We store the results in the object `sim_LC`.

```
sim_LC$y
##  [1] 2018 2019 2020 2021 2022 2023 2024 2025 2026 20
## [16] 2033 2034 2035 2036 2037 2038 2039 2040 2041 20
## [31] 2048 2049 2050 2051 2052 2053 2054 2055 2056 20
## [46] 2063 2064 2065 2066 2067

dim(sim_LC$dda)
## [1]     1    50 10001

sim_LC$dda[ , 1:50, 1]
##  [1]  -50.61649  -52.53247  -54.44844  -56.36441  -5
##  [7]  -62.11232  -64.02829  -65.94426  -67.86023  -6
## [13]  -73.60815  -75.52412  -77.44009  -79.35606  -8
## [19]  -85.10397  -87.01994  -88.93591  -90.85188  -9
## [25]  -96.59980  -98.51577 -100.43174 -102.34771 -10
## [31] -108.09562 -110.01159 -111.92756 -113.84353 -11
## [37] -119.59145 -121.50742 -123.42339 -125.33936 -12
## [43] -131.08727 -133.00324 -134.91921 -136.83518 -13
## [49] -142.58310 -144.49907

LCfit701$kappa2[length(years)]
## [1] -50.61649

dim(sim_LC$qaa)
## [1]    90    50 10001
```

`sim_LC$y` reveals that `sim_LC` stores results/simulations for years 2018 (including) until 2067.

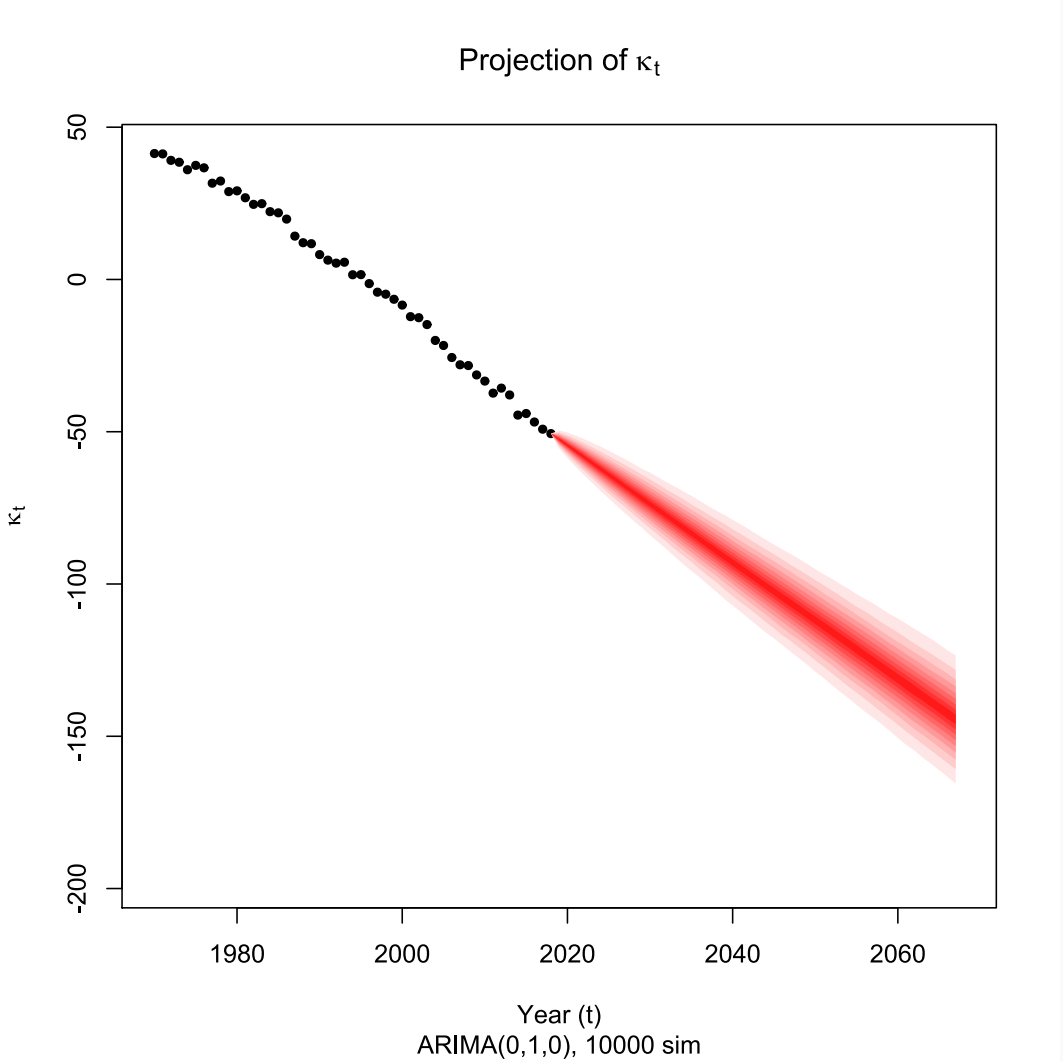
`sim_LC$dda` is a list of dimensions (1, 50, 10001) where 50 refers to the length of `y`.

`sim_LC$dda[ , , 1]` stores the best estimate path (with zero noise terms), then 10 000 paths are generated are stored. Each path starts from the most recent $\hat{\kappa}_t^{(2)}$.

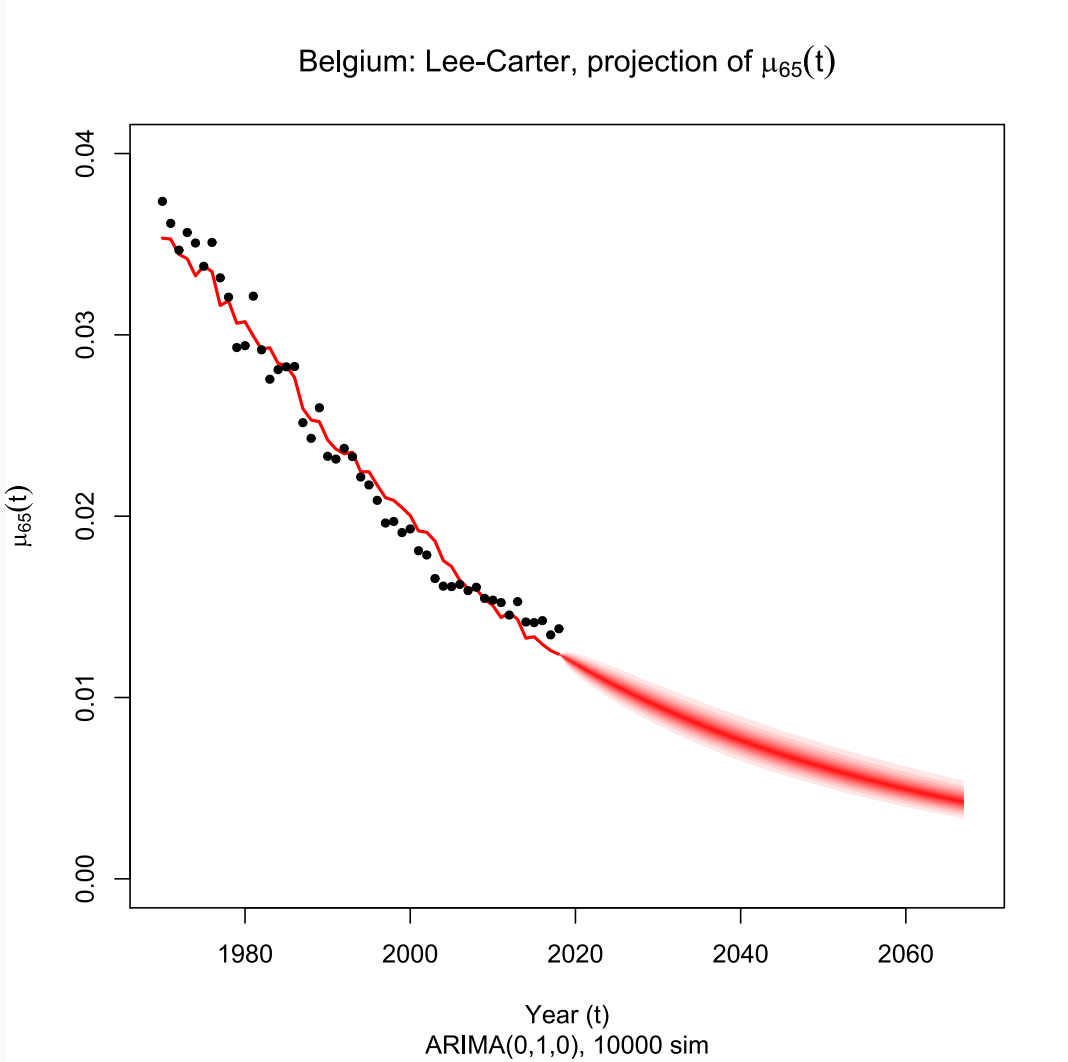
`sim_LC$qaa` is a list of dimensions (90, 50, 10001) where 90 is the number of ages used in the calibration data set. This stores the $q_{x,t}$ for each age x, the projected time horizon (2018 - 2067, in our case), the best estimate and 10 000 generated scenarios.

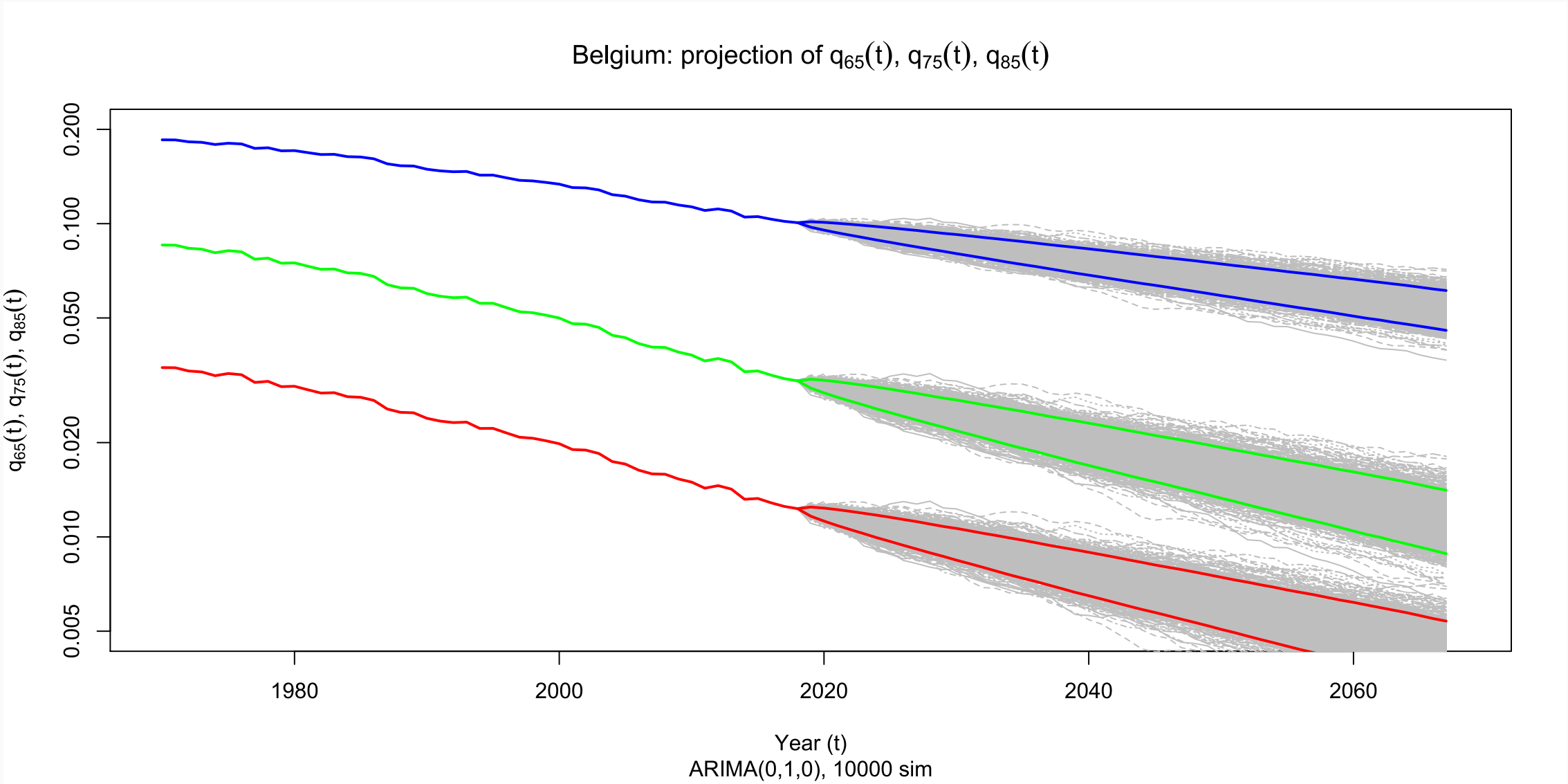
```
# time series kappa_t
plot(
  LCfit701$y,
  LCfit701$kappa2,
  pch = 20,
  xlim = c(1970, 2067),
  ylim = range(c(
    range(LCfit701$kappa2), range(sim_LC$dda[, , ])
  )),
  main = bquote(paste("Projection of ", kappa[t])),
  sub = "ARIMA(0,1,0), 10000 sim",
  xlab = "Year (t)",
  ylab = bquote(kappa[t])
)

# fan chart
fan(sim_LC$y, sim_LC$dda[, , ], color = "red")
```

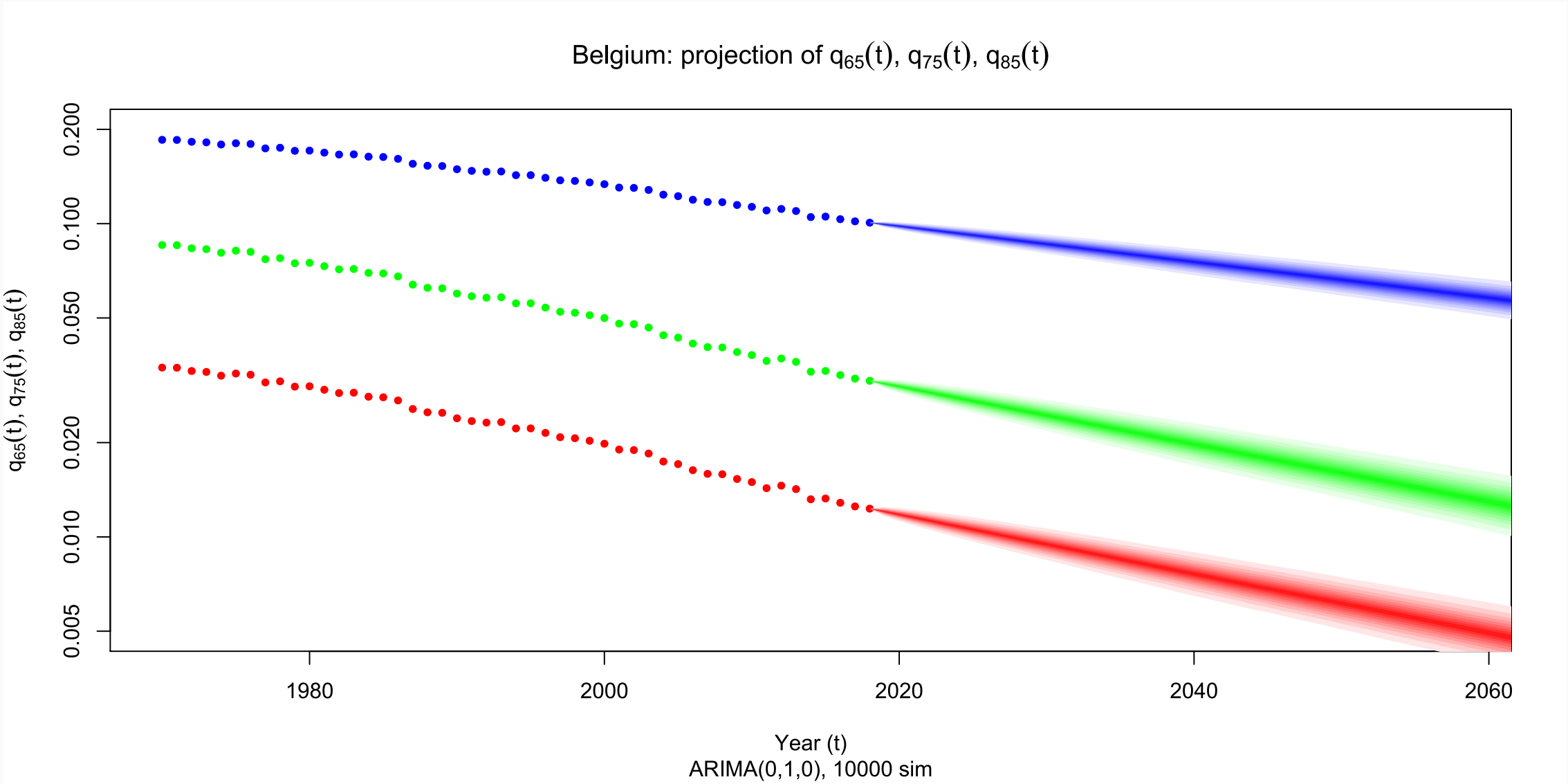
We use the `fan(.)` function from the `simModels.R` script. The outer limits are the 5% and 95% quantiles for a given year or age. The bands in between are 5% quantiles. The darkest band in the middle encompasses the 45% to 55% quantile range.



```r
age = 65
minage = min(ages)

plot(
  LCfit701$y,
  exp(LCfit701$beta1[age - minage + 1] +
      LCfit701$beta2[age - minage + 1] *
      LCfit701$kappa2),
  lwd = 2,
  col = "red",
  type = "l",
  ylim = c(0, 0.04),
  main = bquote(paste(
    "Belgium: Lee-Carter, projection of ", mu[65](t)
  )),
  xlim = c(1970, 2068),
  sub = "ARIMA(0,1,0), 10000 sim",
  xlab = "Year (t)",
  ylab = bquote(paste(mu[65](t)))
)
fan(sim_LC$y,
    exp(LCfit701$beta1[age - minage + 1] +
        LCfit701$beta2[age - minage + 1] *
          sim_LC$dda[, , ]), col = "red")
points(LCfit701$y, rates[, age - minage + 1],
       col = "black", pch = 20)
```




Belgium: projection of $q_{65}(t)$, $q_{75}(t)$, $q_{85}(t)$
$q_{65}(t)$, $q_{75}(t)$, $q_{85}(t)$
0.200
0.100
0.050
0.020
0.010
0.005
1980
2000
2020
2040
2060
Year (t)
ARIMA(0,1,0), 10000 sim

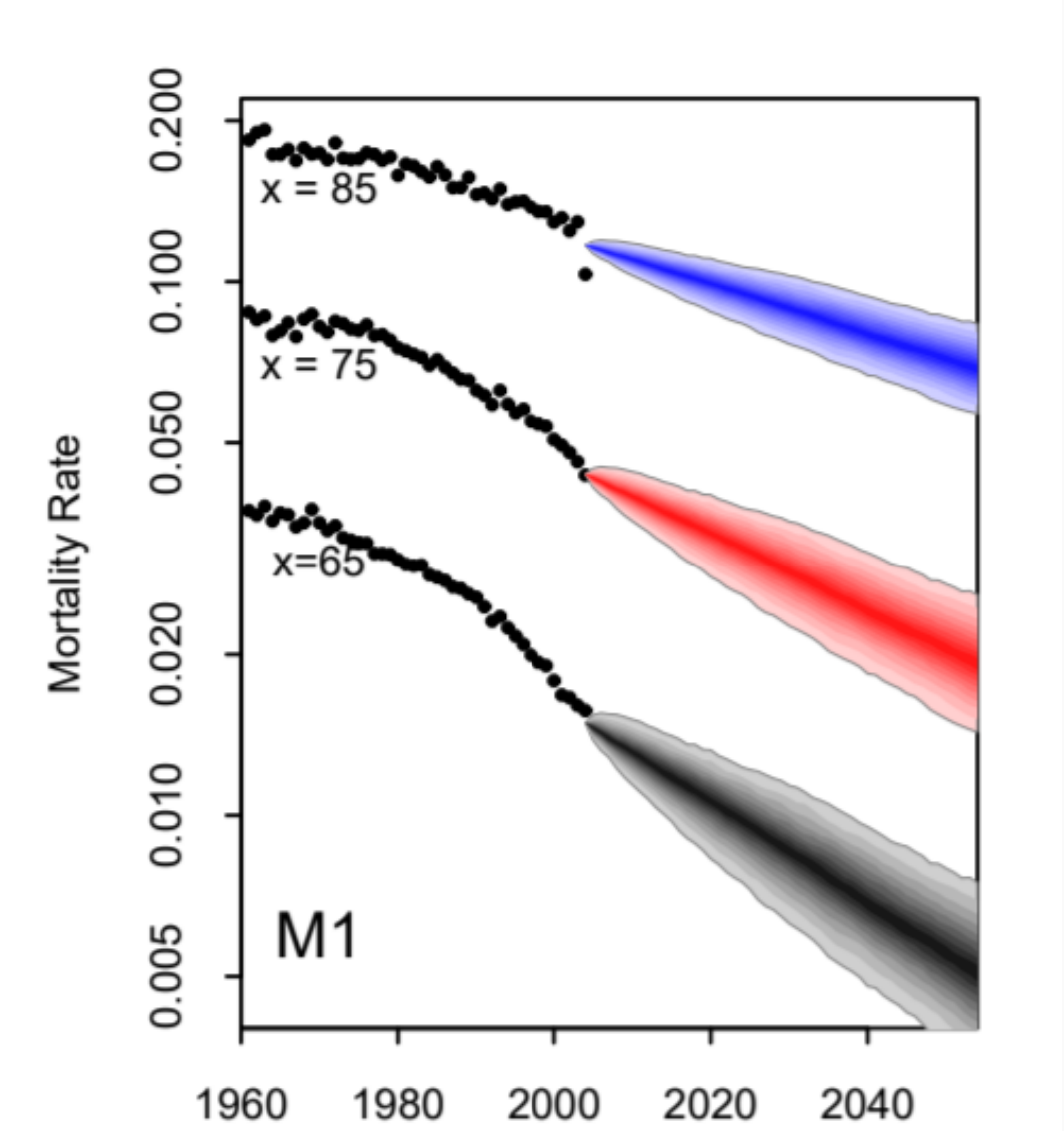

Belgium: projection of $q_{65}(t)$, $q_{75}(t)$, $q_{85}(t)$
$q_{65}(t)$, $q_{75}(t)$, $q_{85}(t)$
0.200
0.100
0.050
0.020
0.010
0.005
1980
2000
2020
2040
2060
Year (t)
ARIMA(0,1,0), 10000 sim

For a more in depth discussion of the fan charts produced by a variety of LifeMetrics models, we refer to Cairns et al. (2011, IME) on Mortality density forecasts: an analysis of six stochastic mortality models.

Quoting from page 13 in this paper:

With M1, the **age-85 fans are narrower than the age-65 fans**. (...) because it (=M1) has a single stochastic period effect, $\kappa_t^{(2)}$. For M1, the widths of the fans are proportional to the age effect, $\beta_x^{(2)}$, and this is unlikely to satisfy the criterion of biological reasonableness.

Historical improvements have been lower at higher ages, forcing $\beta_x^{(2)}$ to be lower at higher ages, so causing the fans at higher ages to be narrower, rather than wider.



Thanks!

Slides created with the R package xaringan.

Course material available via

https://github.com/katrienantonio/mortality-dynamics