Advanced Life Insurance Mathematics

Computer lab on demogRaphics and mortality laws

Katrien Antonio, Bavo Campo and Sander Devriendt

Workshop mortality dynamics | March, 2022

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
- spend time with R code provided.

You hereby focus on:

- parametric mortality laws
- simulating future lifetimes from calibrated mortality laws.

Outline of the workshop:

- Prologue
- Download and import mortality data
- Parametric mortality laws
  - Calibrating Makeham and simulating lifetimes
  - Calibrating Heligman & Pollard and simulating lifetimes.

Download and import mortality data

Download mortality data with {demography}

We do a **live** download from the Human Mortality Database with the {demography} package in R.

Using the package for the first time, you will first have to install it

```
install.packages("demography")
```

You load the package

```
library(demography)
```

```
? hmd.mx
User = "vuulenbak42@hotmail.com"
pw   = "testEAA"
Df   = hmd.mx("BEL", User , pw , "Belgium")
```

`hmd.mx` reads the "Mx" (1 x 1) data from the HMD.

We specify the country code (`BEL`), the user name and password and the character string with the name of the country from which the data are taken.

To get smooth access to the data, we created a username and password for the course.

We store the downloaded data in the object `Df`.

Inspect the `Df` object with

```
names(Df)
## [1] "type"   "label"  "lambda" "year"   "age"    "po
names(Df$pop)
## [1] "female" "male"   "total"
names(Df$rate)
## [1] "female" "male"   "total"
```

Verify that e.g.

```
str(Df$rate$female)
##  num [1:111, 1:180] 0.1516 0.0749 0.0417 0.0255 0.01
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:111] "0" "1" "2" "3" ...
##   ..$ : chr [1:180] "1841" "1842" "1843" "1844" ...
```

stores the $m_{x,t}$ central death rates for Belgian females, with x going from 0 to 110 in the rows and t going from 1841 to 2019 in the columns.

What's stored in the data set?

```
Df$type
## [1] "mortality"
```

```
Df$label
## [1] "Belgium"
```

```
min(Df$year)
## [1] 1841
max(Df$year)
## [1] 2020
```

```
min(Df$age)
## [1] 0
max(Df$age)
## [1] 110
```

Import a life table stored as .txt file

Alternatively, we can download a (period) life table, store it as .txt and import the data in R.

```
Belgium_female_2018 = read.table(file = "./data/Belgium_female_2018.txt", header = TRUE)
```

```
head(Belgium_female_2018)
##   Year Age      mx      qx   ax     lx  dx    Lx      Tx    ex
## 1 2018   0 0.00329 0.00329 0.14 100000 329 99718 8369226 83.69
## 2 2018   1 0.00022 0.00022 0.50  99671  22 99661 8269507 82.97
## 3 2018   2 0.00013 0.00013 0.50  99650  13 99643 8169847 81.99
## 4 2018   3 0.00015 0.00015 0.50  99636  15 99629 8070204 81.00
## 5 2018   4 0.00011 0.00011 0.50  99622  11 99616 7970575 80.01
## 6 2018   5 0.00008 0.00008 0.50  99611   8 99607 7870958 79.02
```

```
Belgium_male_2018 = read.table(file = "./data/Belgium_male_2018.txt", header = TRUE)
```

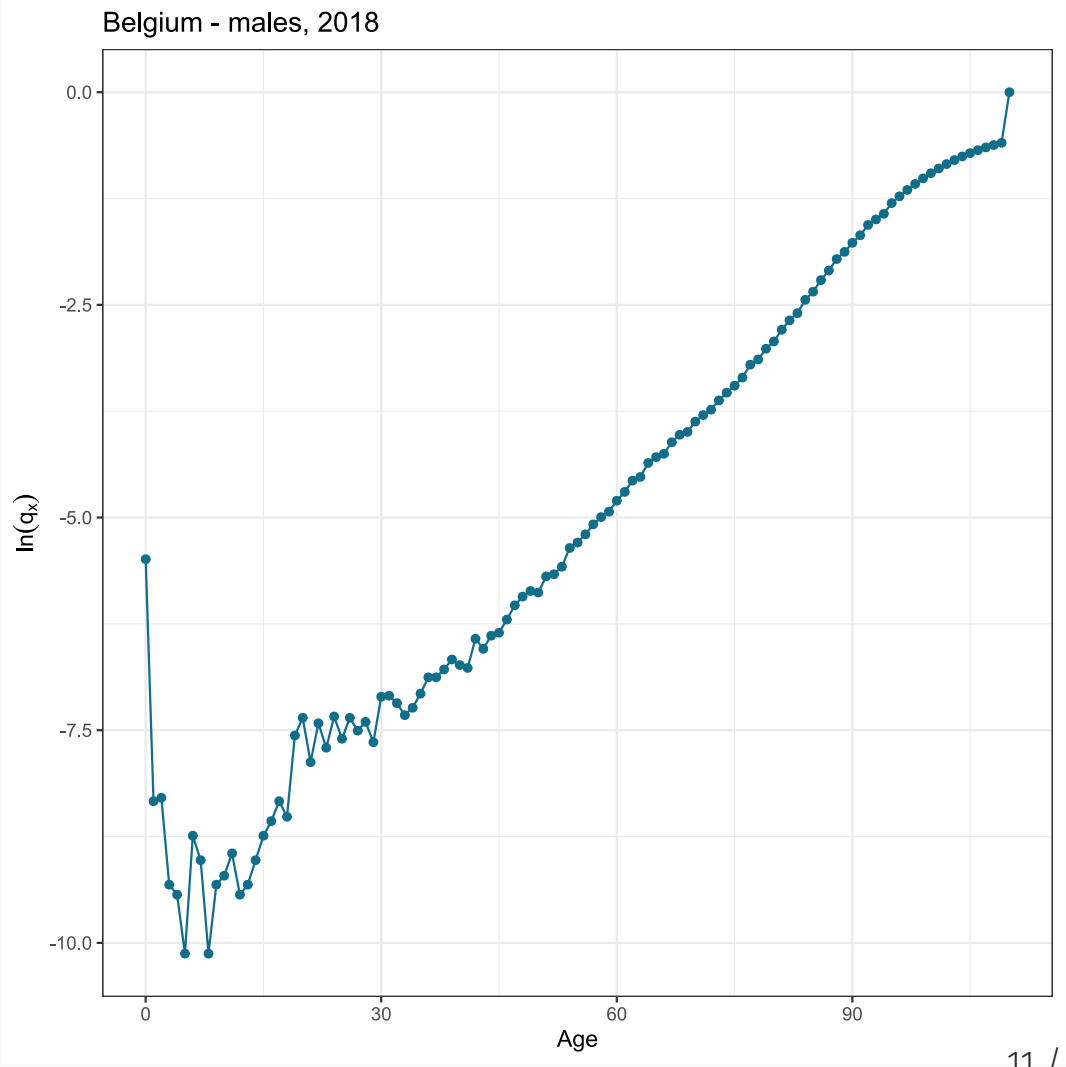
(Both tables were downloaded from the HMD on March 23, 2020.)

Visualize a period life table

We start with visualizing the q_x for M/F data in Belgium, period 2018.

Using `ggplot` instructions:

```
KULbg ← "#116E8A"
library(ggplot2)
g_male ←
  ggplot(Belgium_male_2018, aes(Age, log(qx))) +
  geom_point(col = KULbg) +
  geom_line(stat = "identity", col = KULbg) +
  theme_bw() +
  ggtitle("Belgium - males, 2018") +
  labs(y = bquote(ln(q[x])))

g_fem ←
  ggplot(Belgium_female_2018, aes(Age, log(qx))) +
  geom_point(col = KULbg) +
  geom_line(stat = "identity", col = KULbg) +
  theme_bw() +
  ggtitle("Belgium - females, 2018") +
  labs(y = bquote(ln(q[x])))
```

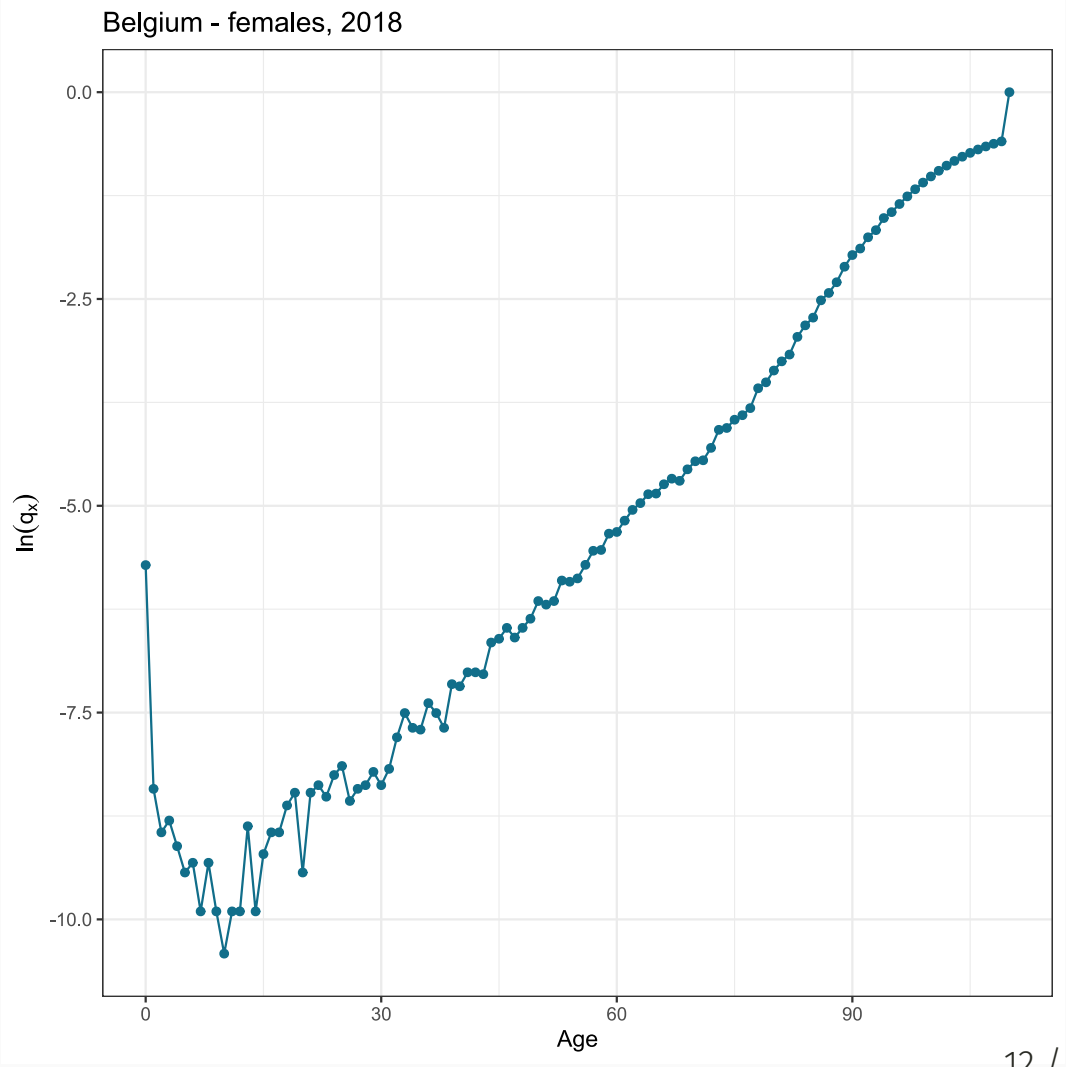


Visualize a period life table

We start with visualizing the q_x for M/F data in Belgium, period 2018.

Using `ggplot` instructions:

```
KULbg <- "#116E8A"
library(ggplot2)
g_male <-
  ggplot(Belgium_male_2018, aes(Age, log(qx))) +
  geom_point(col = KULbg) +
  geom_line(stat = "identity", col = KULbg) +
  theme_bw() +
  ggtitle("Belgium - males, 2018") +
  labs(y = bquote(ln(q[x])))

g_fem <-
  ggplot(Belgium_female_2018, aes(Age, log(qx))) +
  geom_point(col = KULbg) +
  geom_line(stat = "identity", col = KULbg) +
  theme_bw() +
  ggtitle("Belgium - females, 2018") +
  labs(y = bquote(ln(q[x])))
```



Parametric mortality laws

Makeham's law

We focus on fitting Makeham's parametric mortality law (see the lecture sheets):

$$\begin{aligned} \mu_x &= \theta_1 + \theta_2 \cdot \theta_3^x \\ {}_tp_x &= \theta_5^t \cdot \theta_6^{(\theta_3^t - 1)\theta_3^x} \\ -\log p_x &= \theta_1 + \theta_7 \cdot \theta_3^x \end{aligned}$$

In the following sheets we cover two different methods **to estimate** the parameters:

- Ballegeer (1973),
- De Vylder (1975).

These are coded in Sections 1.1.1 to 1.1.2 of the script `ALIM_PCsession1.R`.

You can jump to a Section using `Shift+Alt+J` (Windows/Linux) or `Cmd+Shift+Option+J` (Mac).

Makeham's law: Ballegeer's calibration method

This calibration strategy puts focus on minimizing

$$\sum_{x=x_{min}}^{x_{max}} \left(\log(\theta_5) + (\theta_3 - 1) \cdot \theta_3^x \cdot \log\left(\theta_6\right) - \log p_x\right)^2,$$

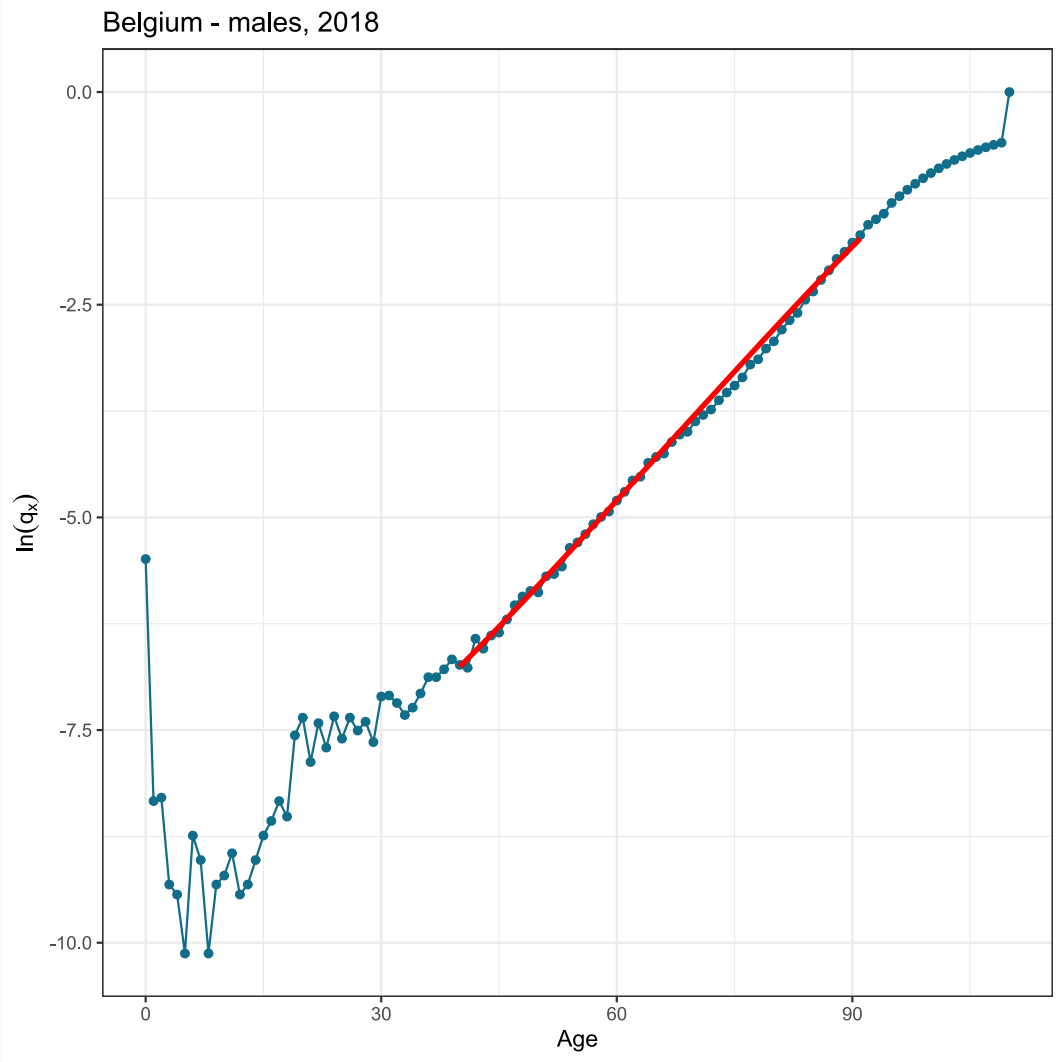
with $\theta_3 > 1, \theta_5 \leq 1$ and $\theta_6 < 1$. The parametrization used is connected to Makeham's initial parameters via

$$\begin{aligned}\theta_5 &= \exp\left(-\theta_1\right),\\ \theta_6 &= \exp\left(-\frac{\theta_2}{\log \theta_3}\right).\end{aligned}$$

Makeham's law: Ballegeer's calibration method

```
f ← function(par, x, px) {
  th5 = par[2]
  th3 = par[1]
  th6 = par[3]
  sum((log(th5) + (th3 - 1) * th3^x * log(th6) - log(px))^2)
}

# initial values
par.init = c(1.1049542, 0.9999124, 0.9998060)

# resulting parameters
(Ballegeer = optim(par.init, f, x = Belgium_male_2018$Age[41:92], lower = c(1, -Inf, -Inf), upper = c(Inf, 1, 1),
                   method = "L-BFGS-B", px = 1 - Belgium_male_2018$qx[41:92])$par)
## [1] 1.1082005 0.9998789 0.9998417
theta_3B = Ballegeer[1]
theta_5B = Ballegeer[2]
theta_6B = Ballegeer[3]

# tranform parameters Ballegeer to Makeham parameters original parametrization
th1B = -log(Ballegeer[2])
th2B = -log(Ballegeer[3]) * log(Ballegeer[1])
th3B = Ballegeer[1]

c(th1B, th2B, th3B)
## [1] 1.210855e-04 1.626377e-05 1.108200e+00
```

```r
g_male <-
  ggplot(Belgium_male_2018, aes(Age, log(qx))) +
  geom_point(col = KULbg) +
  geom_line(stat = "identity", col = KULbg) +
  theme_bw() +
  ggtitle("Belgium - males, 2018") +
  labs(y = bquote(ln(q[x])))

df_Ball <- data.frame(
            Age = 40:91,
            Makeham = log(1 - theta_5B * theta_6B ^
                           ((theta_3B - 1) *
                           theta_3B ^ (40:91)))
            )

g_male <- g_male + geom_line(data = df_Ball,
                     aes(Age, Makeham),
                     col = "red", lwd = 1.2)

g_male
```

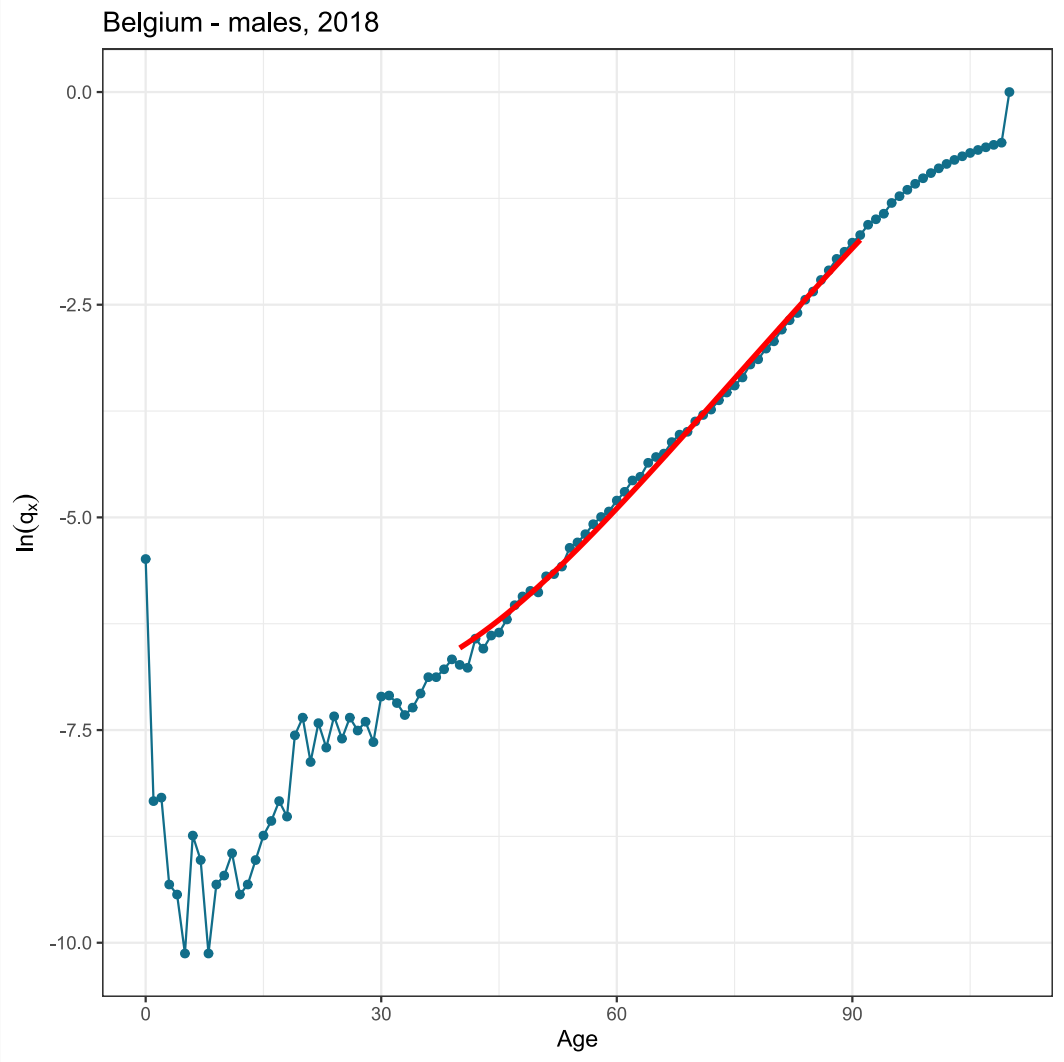


Makeham's law: De Vylder's calibration method

De Vylder optimizes a **binomial log-likelihood**, see

```
BinLL ← function(par, x, dx, lx) {
  px = exp(-(par[1] + par[3] * par[2]^x))
  -sum((lx - dx) * log(px) + dx * log(1 - px))
}

par.init = c(8.758055e-05, 1.104954, 2.036360e-05)
fit1     = optim(par.init, BinLL, x = Belgium_male_2018$Age[41:92], lx = Belgium_male_2018$lx[41:92],
                 dx = Belgium_male_2018$dx[41:92],
                 control = list(reltol = 1e-10))

theta_1DV = fit1$par[1] ; theta_3DV = fit1$par[2] ; theta_7DV = fit1$par[3]

theta_1M  = theta_1DV
theta_2M  = theta_7DV * log(theta_3DV) / (theta_3DV - 1)
theta_3M  = theta_3DV

c(theta_1M, theta_2M, theta_3M)
## [1] 6.565441e-04 1.016813e-05 1.113650e+00
```

```r
g_male <-
  ggplot(Belgium_male_2018, aes(Age, log(qx))) +
  geom_point(col = KULbg) +
  geom_line(stat = "identity", col = KULbg) +
  theme_bw() +
  ggtitle("Belgium - males, 2018") +
  labs(y = bquote(ln(q[x])))

df_DV <- data.frame(
            Age = 40:91,
            Makeham = log(1 - exp(-(theta_1DV +
                    theta_7DV * theta_3DV ^ (40:91))))
            )

g_male <- g_male + geom_line(data = df_DV,
                    aes(Age, Makeham),
                    col = "red", lwd = 1.2)

g_male
```



Simulating future lifetimes ~ Makeham

We simulate the remaining lifetime T_x of a person aged x, according to Makeham's law:

$$
\begin{aligned}
V_1, V_2 &\sim \mathrm{Exp}(1) \\
T_1 &= \frac{\log\left(\theta_2 \theta_3^x + V_1 \log \theta_2\right) - \log \theta_2}{\log \theta_3} - x \\
T_2 &= \frac{V_2}{\theta_1} \\
T &= \min(T_1, T_2)
\end{aligned}
$$

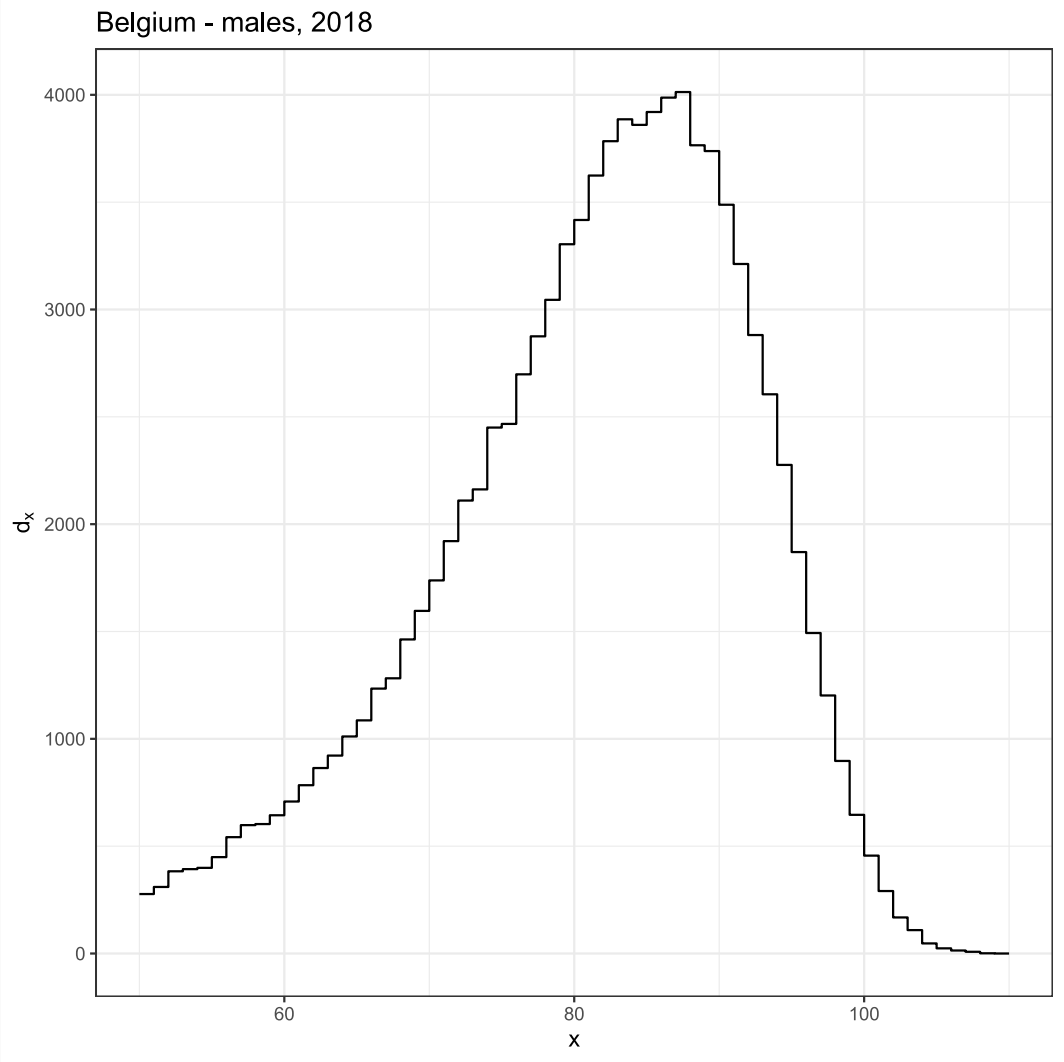
Our first experiment considers `nsim` individuals, aged `50`.

The vector `t` stores the simulated remaining lifetimes.

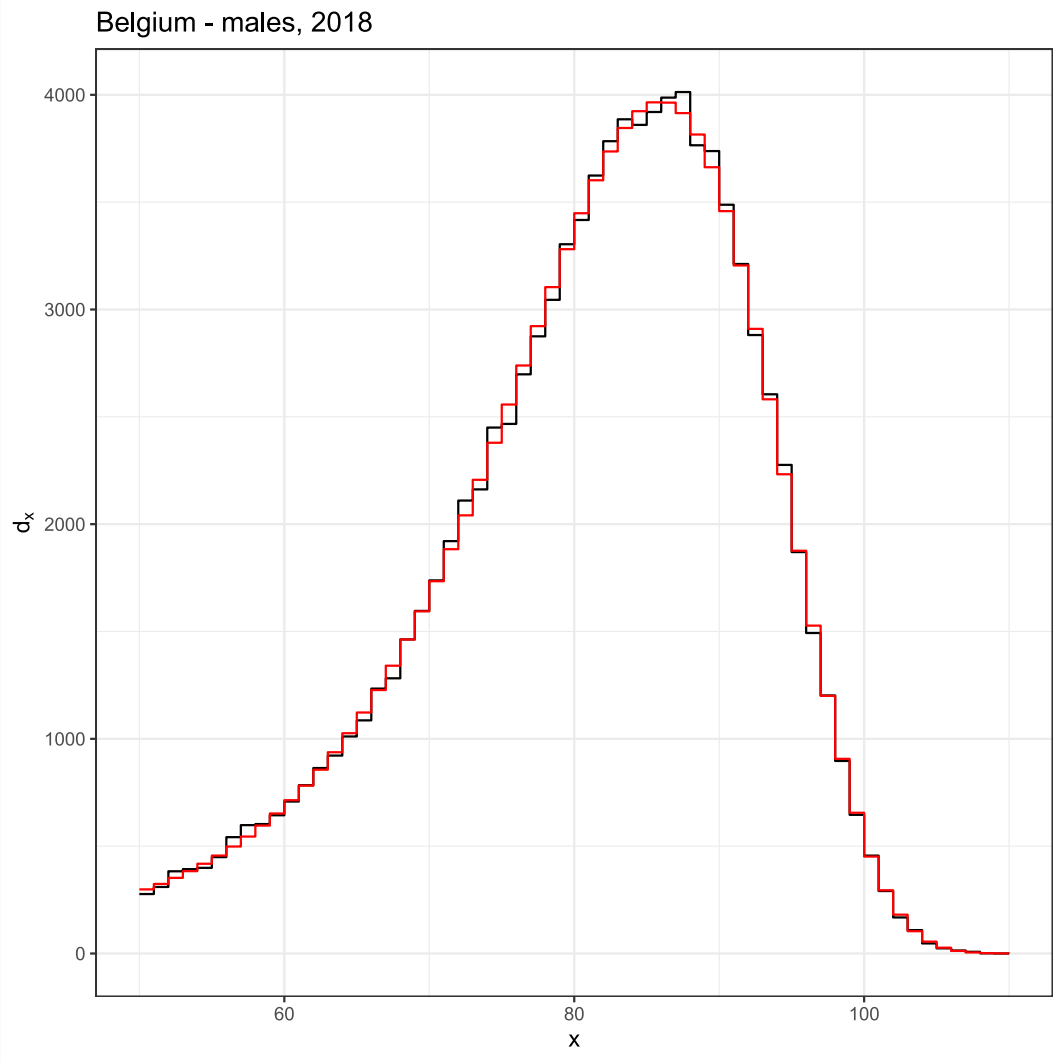
The vector `age_death` stores the simulated ages at death of the `nsim` individuals.

```
age  = 50

nsim = 100000
v1   = rexp(nsim, 1)
v2   = rexp(nsim, 1)
t1   = (log(theta_2M * theta_3M ^ age +
              v1 * log(theta_3M)) -
          log(theta_2M)) / log(theta_3M) - age
t2   = v2 / theta_1M
t    = pmin(t1, t2)
age_death    = t + age
```

```r
# group the simulated lifetimes in integer
#        classes
d_sim <- data.frame(
            x = age:110,
            sim = as.numeric(table(
              factor(floor(age_death),
              levels = age:110)))
         )

# and plot
g_sim_Makeham <- ggplot(d_sim) +
                 geom_step(aes(x, sim)) +
                 theme_bw() +
                 ggtitle("Belgium - males, 2018") +
                 labs(y = bquote(d[x]))

g_sim_Makeham
```



```r
# now we add the theoretical results,
# using Makeham's law
grid = age:110
tgrid = grid - age
p = exp(-theta_1M * tgrid -
          theta_2M * theta_3M ^ (age) *
          (theta_3M ^ tgrid - 1) / log(theta_3M))
l = nsim * p
d = c(-diff(l), 0) # add zero to match length of grid

d_theo <- data.frame(
             x = age:110,
             theo = d
)

g_sim_Makeham <- g_sim_Makeham +
                 geom_step(data = d_theo,
                           aes(x, theo),
                           color = "red")

g_sim_Makeham
```



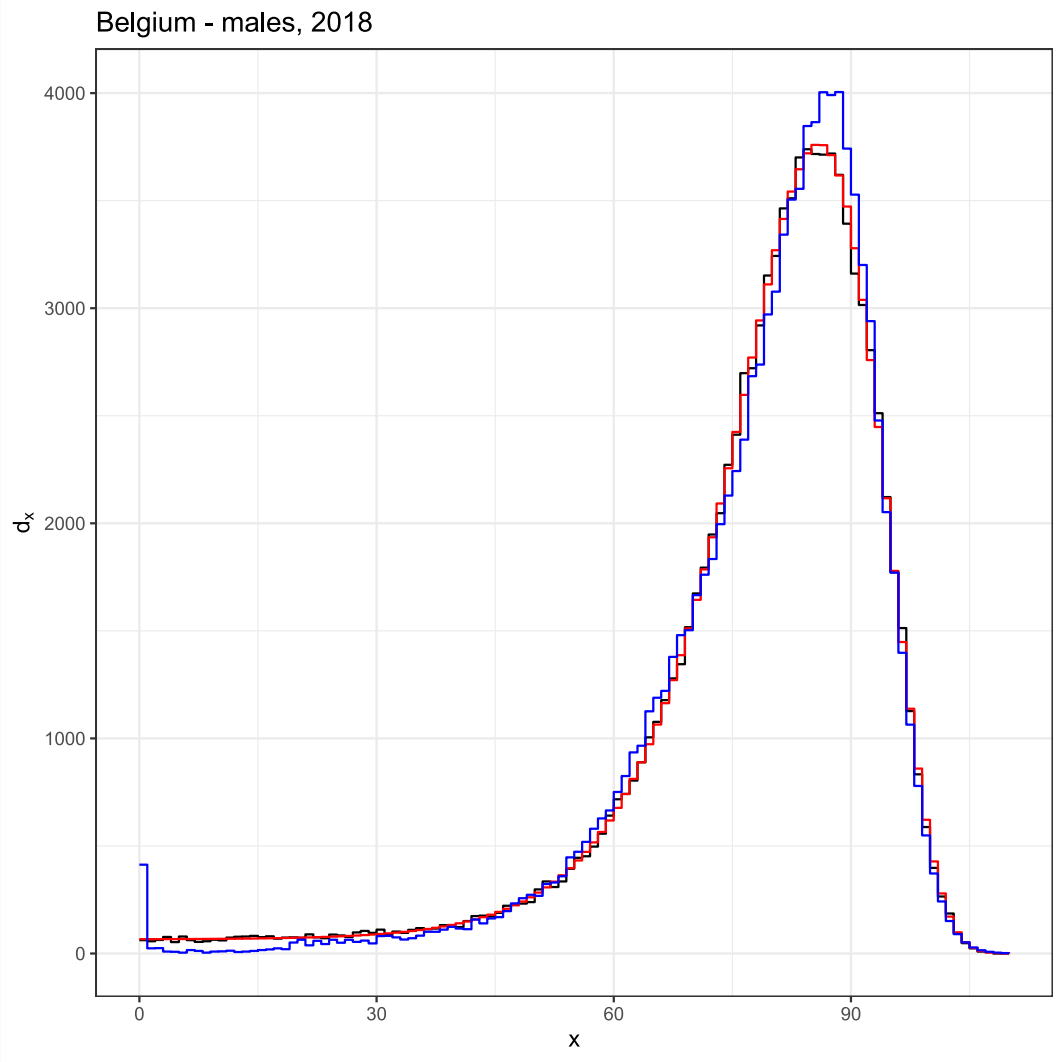
Now, we repeat the simulation for `age = 0`.

Recycling the previous instructions creates the plot on the right.

We can compare the generated age at death distribution with the observed ones, stored in

```
head(Belgium_male_2018)
##   Year Age      mx      qx   ax     lx  dx    Lx
## 1 2018   0 0.00414 0.00413 0.14 100000 413 99645 792
## 2 2018   1 0.00024 0.00024 0.50  99587  24 99575 782
## 3 2018   2 0.00025 0.00025 0.50  99563  25 99550 772
## 4 2018   3 0.00009 0.00009 0.50  99538   9 99533 762
## 5 2018   4 0.00008 0.00008 0.50  99529   8 99525 752
## 6 2018   5 0.00004 0.00004 0.50  99521   4 99519 742
```

```
g_sim_Makeham ← g_sim_Makeham +
                geom_step(data = Belgium_male_2018,
                          aes(Age, dx),
                          color = "blue")

g_sim_Makeham
```



Heligman & Pollard

We put focus on calibrating the parameters in the Heligman & Pollard parametric mortality law.

$$\frac{q_x}{p_x} = \theta_1^{(x+\theta_2)^{\theta_3}} + \theta_4 \cdot \exp\Big(-\theta_5(\log x - \log \theta_6)^2\Big) + \theta_7 \theta_8^x.$$

Our goals:

- estimate unknown parameters using MLE
- stress importance of starting values
- generate lifetimes using **Probability Integral Transform** and a non-integer age mortality assumption.

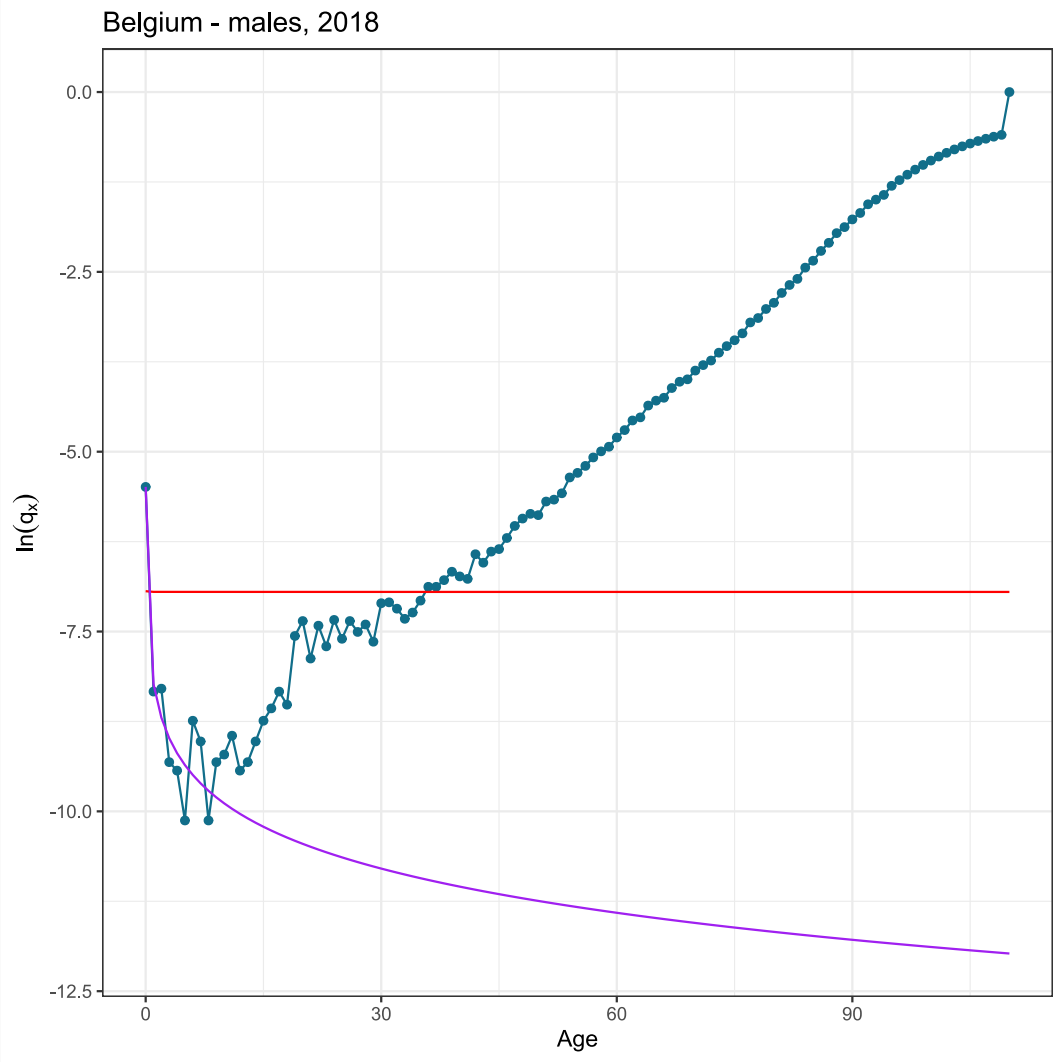
This is covered in Sections 1.1.5 and 1.1.6 of the code.

Heligman & Pollard - infant mortality

```r
HPLLInfMort <- function(par, dx, lx, x){
  theta_1 = par[1]
  theta_2 = par[2]
  theta_3 = par[3]

  qxHP1   = theta_1 ^ ((x + theta_2) ^ (theta_3))
  qxHP    = qxHP1 / (1 + qxHP1)
  pxHP    = 1 - qxHP

  -sum((lx - dx) * log(pxHP) + dx * log(qxHP))
}
```

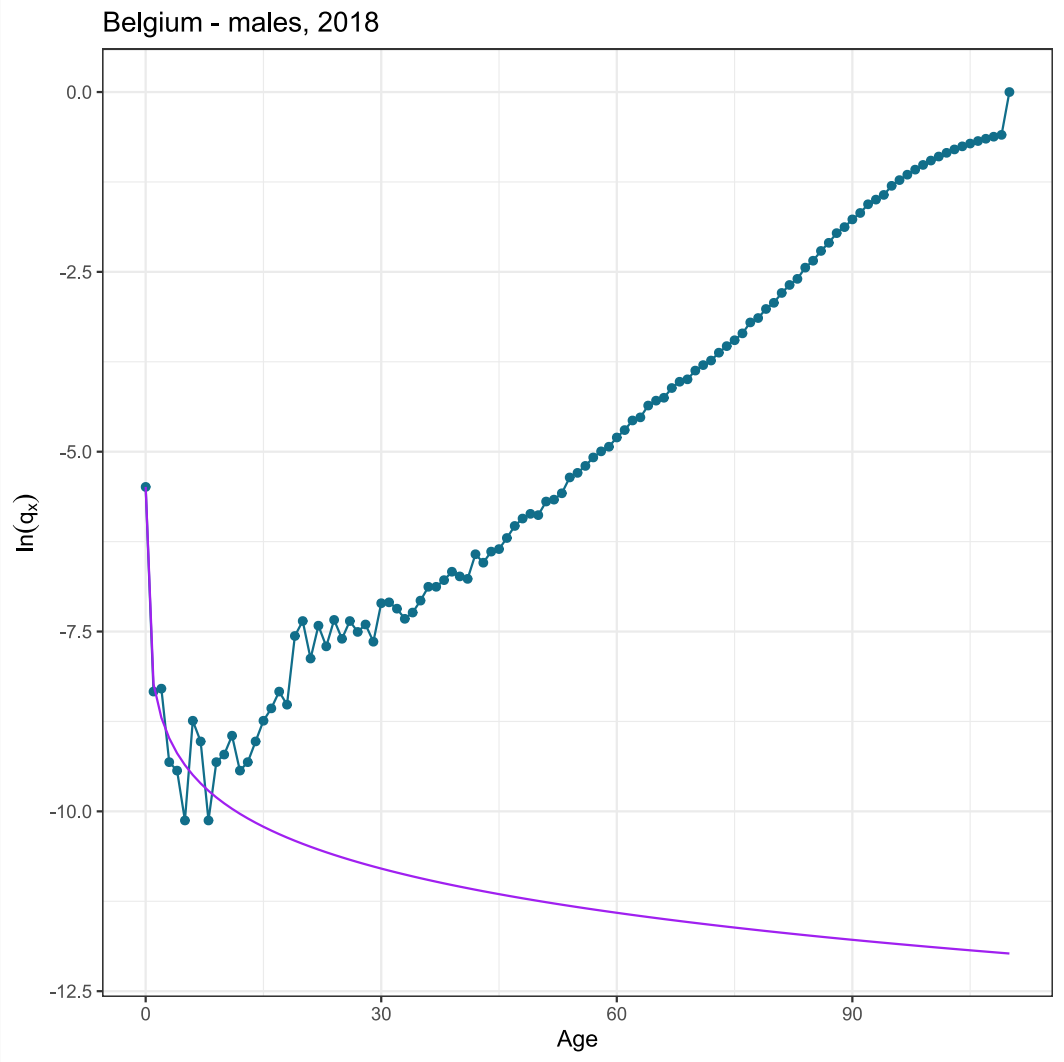
```r
fitHP1 = optim(c(1e-6, 1e-6, 1e-6), HPLLInfMort,
               dx = Belgium_male_2018$dx[1:5],
               lx = Belgium_male_2018$lx[1:5], x = Belgium_male_2018$Age[1:5])

fitHP2 = optim(c(1e-2, 1e-2, 1e-2), HPLLInfMort,
               dx = Belgium_male_2018$dx[1:5],
               lx = Belgium_male_2018$lx[1:5], x = Belgium_male_2018$Age[1:5])

fitHP1$value; fitHP2$value
## [1] 3803.466
## [1] 3314.435
# Note that this is -log-likelihood, so the lower, the better
```

```r
PH.lnqx <- function(par, grid) {
  th1 = par[1]
  th2 = par[2]
  th3 = par[3]
  qx1 = th1^((grid + th2)^th3)
  log(qx1 / (1 + qx1))
}

g_male <-
  ggplot(Belgium_male_2018, aes(Age, log(qx))) +
  geom_point(col = KULbg) +
  geom_line(stat = "identity", col = KULbg) +
  theme_bw() +
  ggtitle("Belgium - males, 2018") +
  labs(y = bquote(ln(q[x])))

df_HP <- data.frame(
          x = 0:110,
          HP1 = PH.lnqx(fitHP1$par, 0:110),
          HP2 = PH.lnqx(fitHP2$par, 0:110)
)

g_male <- g_male + geom_line(data = df_HP,
                   aes(x, HP1), col = "red") +
          geom_line(data = df_HP, aes(x, HP2),
                    col = "purple")
g_male
```



Finally, we store the optimal parameter values, to use these as starting values when calibrating the H&P law over ages [0,110].

```
theta_1HPInf = fitHP2$par[1]
theta_2HPInf = fitHP2$par[2]
theta_3HPInf = fitHP2$par[3]
```



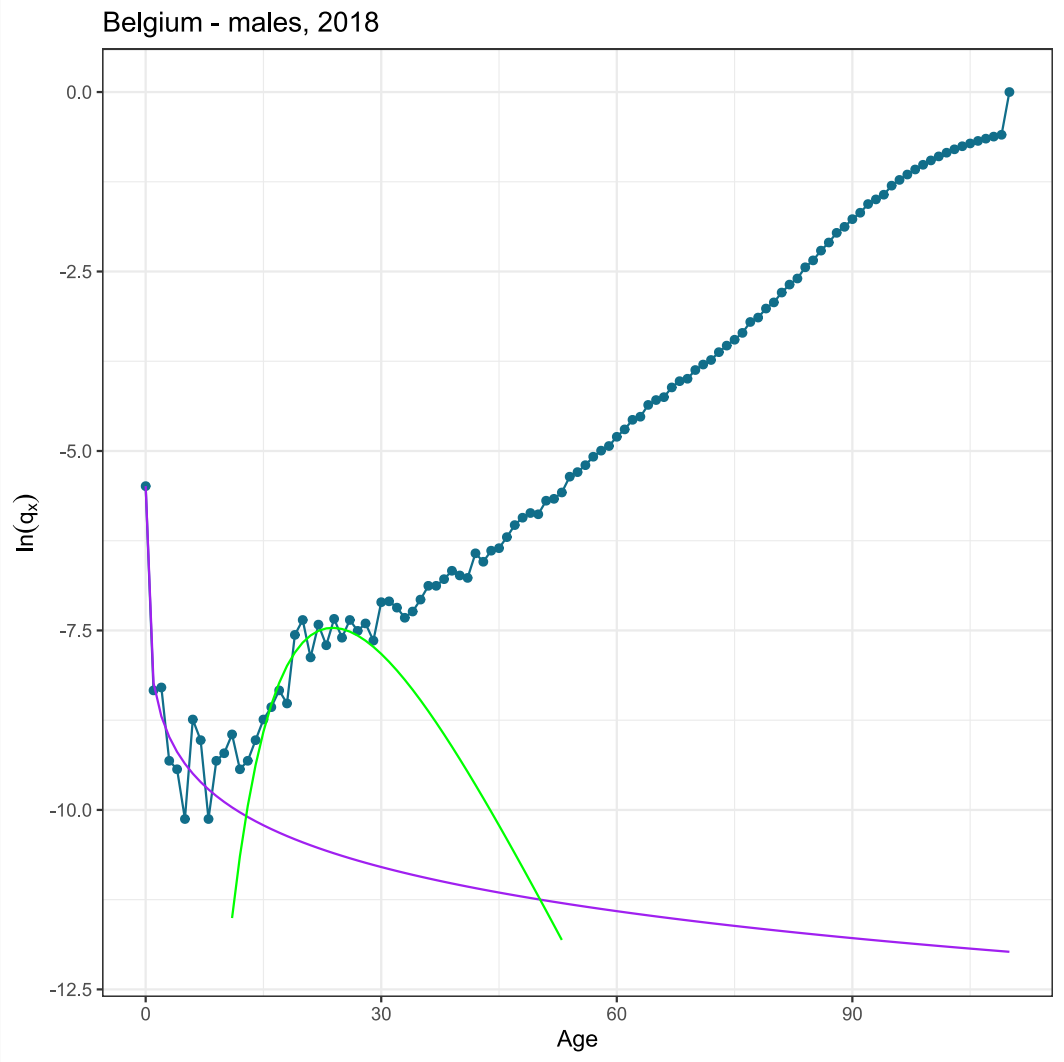
Heligman & Pollard - accident hump

```r
HPLLAccid <- function(par, dx, lx, x) {
  theta_4 = par[1]
  theta_5 = par[2]
  theta_6 = par[3]

  qxHP1 = theta_4 * exp(-theta_5 *
                          (log(x) - log(theta_6)) ^ 2)
  qxHP  = qxHP1 / (1 + qxHP1)
  pxHP  = 1 - qxHP

  -sum((lx - dx) * log(pxHP) + dx * log(qxHP))
}

par.init = c(0.0001, 10, 18)
par.init = c(0.01, 5, 18)
```

```r
fit1 = optim(par.init,
             HPLLAccid,
             dx = Belgium_male_2018$dx[16:25],
             lx = Belgium_male_2018$lx[16:25],
             x = Belgium_male_2018$Age[16:25])

(theta_4HPAcc = fit1$par[1])
## [1] 0.0005741263
(theta_5HPAcc = fit1$par[2])
## [1] 6.785977
(theta_6HPAcc = fit1$par[3])
## [1] 23.80243
```

We extend our previous graph with the fit for the accident hump.

```
df_HPAcc ← data.frame(
      x = 0:110,
      qxHP1 = theta_4HPAcc * exp(-theta_5HPAcc *
              (log(0:110) - log(theta_6HPAcc)) ^ 2)
)

df_HPAcc$qxHP ← df_HPAcc$qxHP1 / (1 + df_HPAcc$qxHP1)

g_male ← g_male + geom_line(data = df_HPAcc,
                            aes(x, log(qxHP)),
                            col = "green") +
          ylim(-12, 0)

g_male
```



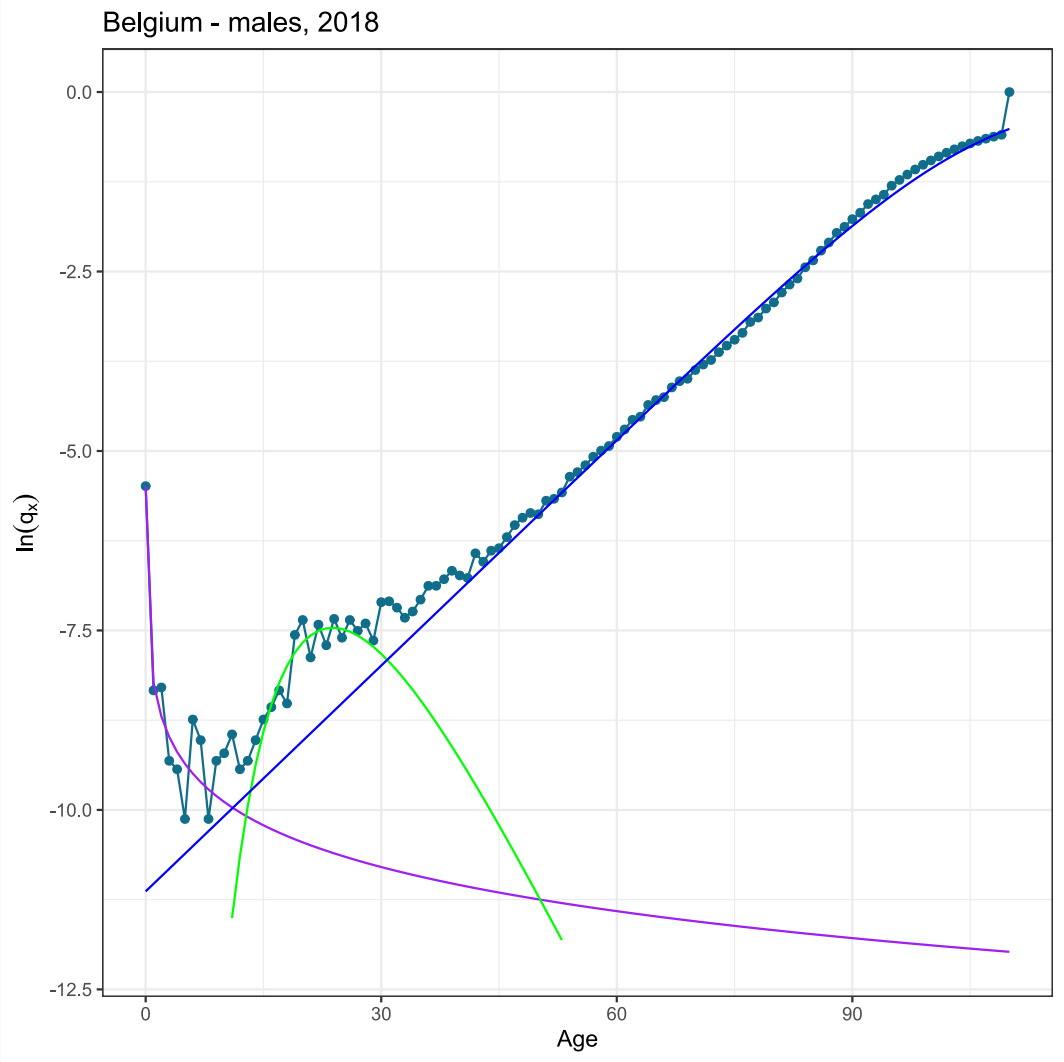
Heligman & Pollard - adult mortality

```r
HPLLAdult ← function(par,dx,lx,x){
  theta_7 = par[1]
  theta_8 = par[2]

  qxHP1 = theta_7 * theta_8 ^ x
  qxHP  = qxHP1 / (1 + qxHP1)
  pxHP  = 1 - qxHP

  -sum((lx - dx) * log(pxHP) + dx * log(qxHP))
}

par.init = c(0.01,1)
```

```r
fit1 = optim(par.init,
             HPLLAdult,
             dx = Belgium_male_2018$dx[25:99],
             lx = Belgium_male_2018$lx[25:99],
             x = Belgium_male_2018$Age[25:99])

(theta_7HPAd = fit1$par[1])
## [1] 1.459687e-05
(theta_8HPAd = fit1$par[2])
## [1] 1.110546
```

We extend our previous graph with the fit for the adult mortality.

```
df_HPAd ← data.frame(
      x = 0:110,
      qxHP1 = theta_7HPAd * theta_8HPAd ^ (0:110)
)

df_HPAd$qxHP ← df_HPAd$qxHP1 / (1 + df_HPAd$qxHP1)

g_male ← g_male + geom_line(data = df_HPAd,
                            aes(x, log(qxHP)),
                            col = "blue") +
          ylim(-12, 0)

g_male
```



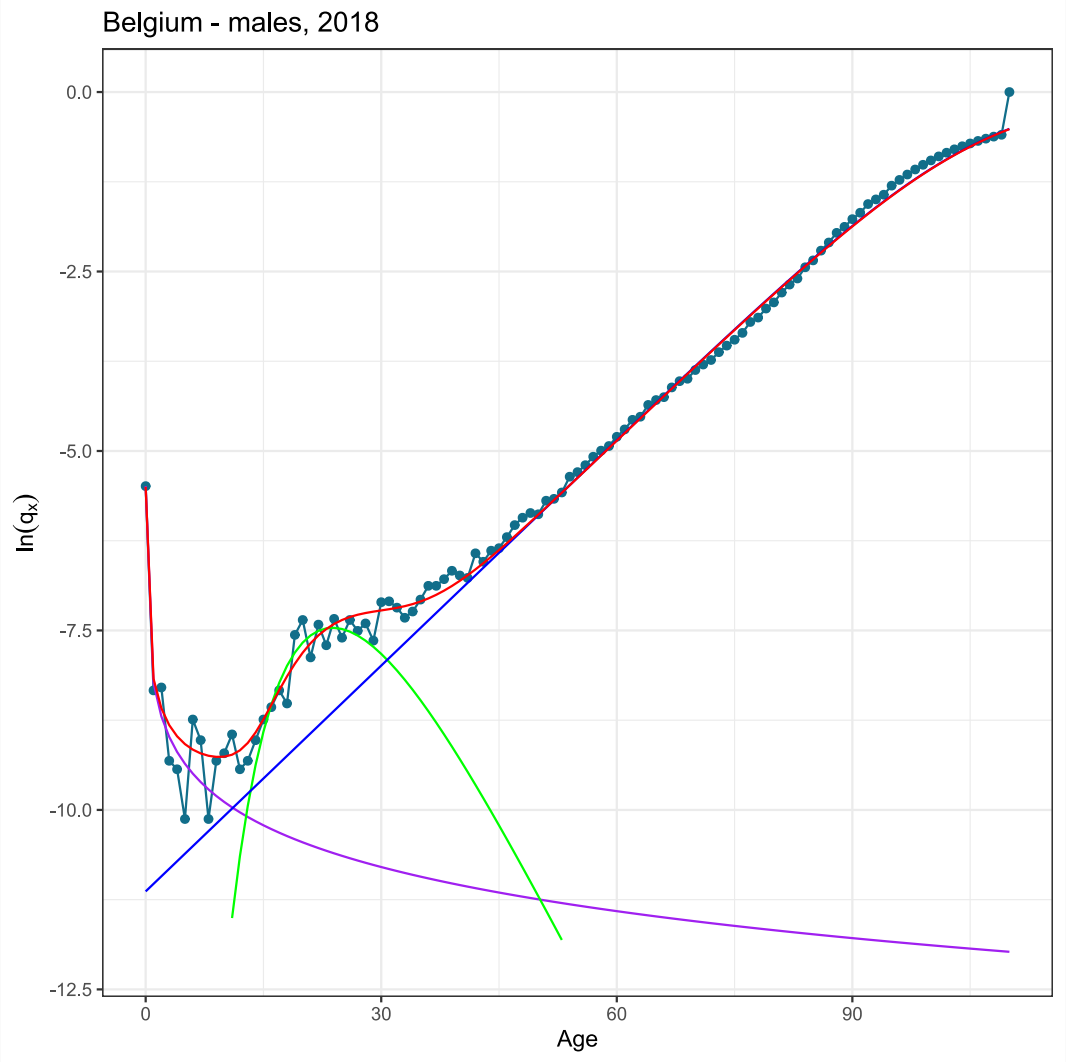
Heligman & Pollard - putting it all together

```
HPLL ← function(par,dx,lx,x){
  theta_1 = par[1]
  theta_2 = par[2]
  theta_3 = par[3]
  theta_4 = par[4]
  theta_5 = par[5]
  theta_6 = par[6]
  theta_7 = par[7]
  theta_8 = par[8]

qxHP1 = theta_1 ^ ((x + theta_2) ^ (theta_3)) +
theta_4 * exp(-theta_5 * (log(x) - log(theta_6)) ^ 2) +
theta_7 * theta_8 ^ x

qxHP  = qxHP1 / (1 + qxHP1)
pxHP  = 1 - qxHP

  -sum((lx - dx) * log(pxHP) + dx * log(qxHP))
}

par.init = c(theta_1HPInf,theta_2HPInf,theta_3HPInf,
             theta_4HPAcc,theta_5HPAcc,theta_6HPAcc,
             theta_7HPAd,theta_8HPAd)
```

```
fit1 = optim(par.init,
             HPLL,
             dx = Belgium_male_2018$dx[1:99],
             lx = Belgium_male_2018$lx[1:99],
             x = Belgium_male_2018$Age[1:99])

theta_1HP = fit1$par[1]
theta_2HP = fit1$par[2]
theta_3HP = fit1$par[3]
theta_4HP = fit1$par[4]
theta_5HP = fit1$par[5]
theta_6HP = fit1$par[6]
theta_7HP = fit1$par[7]
theta_8HP = fit1$par[8]
```

We plot the resulting fit.

```
df_HP ← data.frame(
x = 0:110,
qxHP1 = theta_1HP ^ ((0:110 + theta_2HP) ^ (theta_3HP)
  theta_4HP * exp(-theta_5HP *(log(0:110) - log(theta_6
  theta_7HP * theta_8HP ^ (0:110)
)

df_HP$qxHP ← df_HP$qxHP1 / (1 + df_HP$qxHP1)

g_male ← g_male + geom_line(data = df_HP,
                            aes(x, log(qxHP)),
                            col = "red") +
          ylim(-12, 0)

g_male
```



Simulating future lifetimes ~ Heligman & Pollard

We simulate the remaining lifetime T_x for person aged x, according to H&P law.

- if $U \sim \mathrm{Unif}(0,1)$, then $1-U$ is also uniformly distributed
- PIT:
  - find t such that $1-u = F_x(t)$ equivalent with
  - find t such that $u = S_x(t)$.

```
age = 0
x   = 0:110
p   = 1 / (1 + theta_1HP ^ ((x + theta_2HP) ^
                                (theta_3HP)) +
              theta_4HP *
              exp(-theta_5HP * (log(x) -
                              log(theta_6HP)) ^ 2) +
              theta_7HP * (theta_8HP ^ x))
n   = 110 - age

# pmat[1+i] is i_p_age
pmat = cumprod(p[(age + 1):length(p)])
```

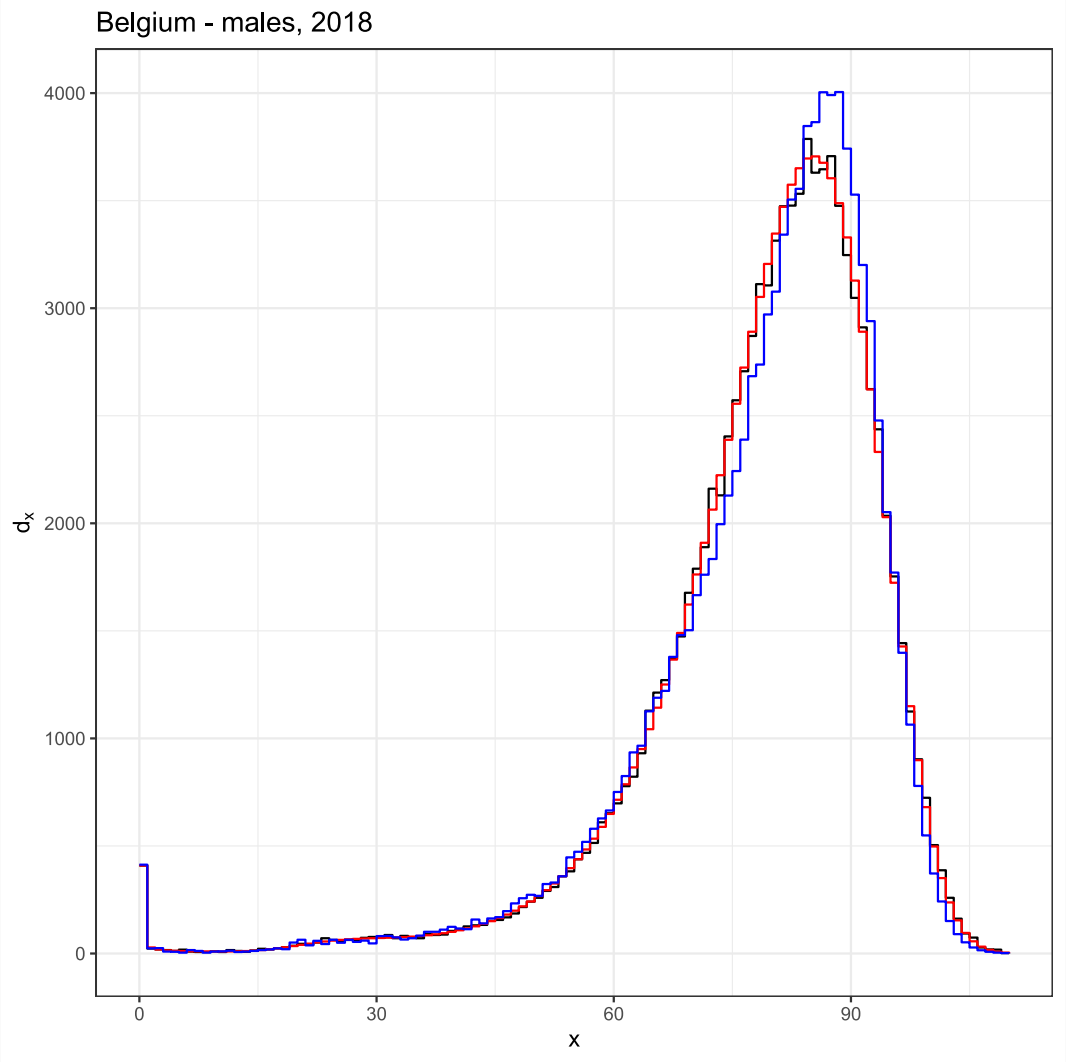
```r
nsim     = 100000
age_death = numeric(nsim)
u        = runif(nsim, 0, 1)
t1       = numeric(nsim)
t2       = numeric(nsim)
```

```r
for(j in 1:nsim) {
  if (u[j] > pmat[1]) {
    t1[j] = 0
  }
  else{
    i = 1
    flag = FALSE
    while (i ≤ (n - 1) & !flag) {
      if (pmat[i] ≥ u[j] && u[j] > pmat[i + 1]) {
        t1[j] = i
        flag = TRUE
      }
      i = i + 1
    }
  }
  t = t1[j]
  if (t == 0) {
    t2[j] = log(u[j]) / (log(p[t + 1]))
  }
  if (t > 0) {
    t2[j] = (log(u[j]) - log(pmat[t])) /
                        log(p[t + age + 1])
  }
  age_death[j] = age + t1[j] + t2[j]
}
```

Using plotting instructions similar to the ones introduced for Makeham, we visualize the generated distribution of ages at death.

```
# group the simulated lifetimes in integer classes
# and plot
d_sim ← data.frame(
            x = age:110,
            sim = as.numeric(table(
              factor(floor(age_death),
              levels = age:110)))
         )

# now we add the theoretical results, using H&P law
l = nsim * c(1, pmat)
d = -diff(l)

d_theo ← data.frame(
             x = age:110,
             theo = d
)
```

and plot ...



Thanks!

Slides created with the R package xaringan.

Course material available via

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