Modelling and quantifying mortality and longevity risk

Tutorial 1 (in brief)

Katrien Antonio, Michel Vellekoop, with input by Jens Robben
School of Actuarial Science, Warsaw | September 18-19, 2025

Goals of tutorial session 1

Goals of tutorial session 1

In this tutorial session you will learn to:

- download and visualize mortality data sets and life tables obtained from the Human Mortality Database, see HMD
- perform some basic calculations with period life tables
- calibrate (selected) single population mortality models, using the Poisson distributional assumption for death counts
- perform some checks to inspect the in-sample fit of the calibrated model.

Outline of the tutorial:

- Prologue
- Life table calculations and visualizations
- Download and import mortality data
- Fit single population mortality models by maximising a Poisson likelihood
- Model selection and goodness-of-fit

Life table calculations and visualizations

Import a life table stored as .txt file

We downloaded (on 17 September 2025) life tables from the HMD and stored it as .txt file on the GitHub repo. Now you can import it in R:

```
POL_female ← read.table(
  file = "../data/hmd/POL_female_life_table_1×1.txt",
  skip = 2, header = TRUE)
POL_male   ← read.table(
  file = "../data/hmd/POL_male_life_table_1×1.txt",
  skip = 2, header = TRUE)
```

We convert the age variable into an integer variable:

```
POL_male$Age   ← parse_number(POL_male$Age)   %>%
  as.integer()
POL_female$Age ← parse_number(POL_female$Age) %>%
  as.integer()
```

Mind the use of `parse_number()` to handle the entry '110+' in the `Age` column of the original object.

We extract the 2022 data (by means of example) using the function `filter` from the {dplyr} package:

```
POL_male_2022   ← POL_male   %>%
  dplyr::filter(Year == 2022)
POL_female_2022 ← POL_female %>%
  dplyr::filter(Year == 2022)
```

We display the structure of the R object `POL_female_2022` using `str(.)`

```
str(POL_female_2022)
## 'data.frame':    111 obs. of  10 variables:
##  $ Year: int  2022 2022 2022 2022 2022 2022 2022 202
##  $ Age : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ mx  : num  0.0033 0.00031 0.00015 0.00012 0.0001
##  $ qx  : num  0.00329 0.00031 0.00015 0.00012 0.000
##  $ ax  : num  0.14 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0
##  $ lx  : int  100000 99671 99640 99625 99613 99602 9
##  $ dx  : int  329 31 15 12 11 7 8 10 10 8 ...
##  $ Lx  : int  99718 99656 99633 99619 99608 995994 9
##  $ Tx  : int  8109270 8009552 7909896 7810264 771064
```

Let's take a closer look at the period life table for males in Poland in the year 2022:

```
head(POL_male_2022)
```

Year	Age	mx	qx	ax	lx	dx	Lx	Tx	ex
2022	0	0.00412	0.00410	0.14	100000	410	99647	7342567	73.43
2022	1	0.00033	0.00033	0.50	99590	33	99573	7242919	72.73
2022	2	0.00023	0.00023	0.50	99557	23	99546	7143346	71.75
2022	3	0.00018	0.00018	0.50	99534	18	99525	7043800	70.77
2022	4	0.00014	0.00014	0.50	99516	13	99510	6944275	69.78
2022	5	0.00010	0.00010	0.50	99503	10	99498	6844765	68.79

`mx`: death rate at age x, calculated as observed deaths divided by observed exposures at age x.

`qx`: mortality rate, i.e., the probability that an x-year old dies within a year.

`lx`: number of survivors at age x in the life table population.

`dx`: number of deaths at age x in the life table population.

`Tx`: remaining person-years for all individuals of age x.

`ex`: period life expectancy at age x.

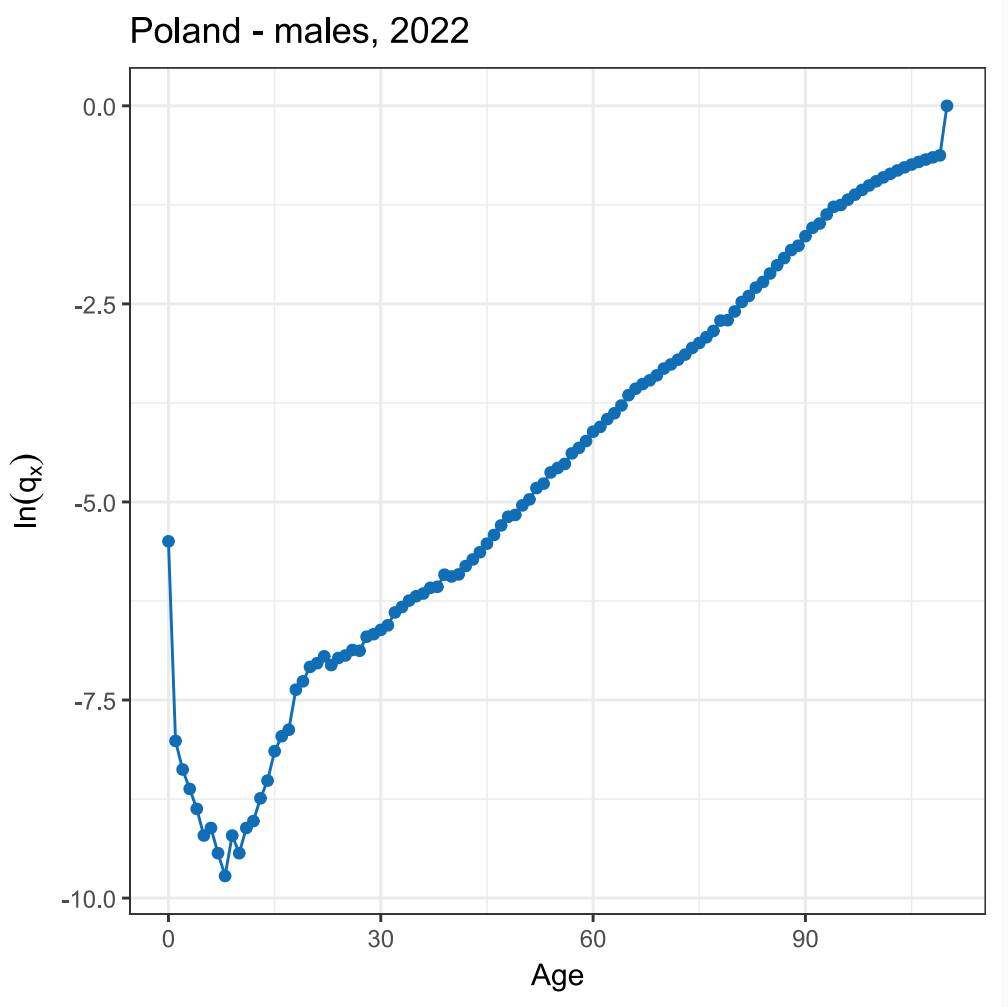
Visualize a period life table

We start with visualizing the $q_{x,t}$ for M/F data in Poland, year 2022.

Using `ggplot` instructions:

```
g_male ←
  ggplot(POL_male_2022, aes(Age, log(qx))) +
  geom_point(col = RCLRbg) +
  geom_line(col = RCLRbg) +
  theme_bw(base_size = 15) +
  ggtitle("Poland - males, 2022") +
  labs(y = bquote(ln(q[x])))

g_fem ←
  ggplot(POL_female_2022, aes(Age, log(qx))) +
  geom_point(col = RCLRbg) +
  geom_line(col = RCLRbg) +
  theme_bw(base_size = 15) +
  ggtitle("Poland - females, 2022") +
  labs(y = bquote(ln(q[x])))
```



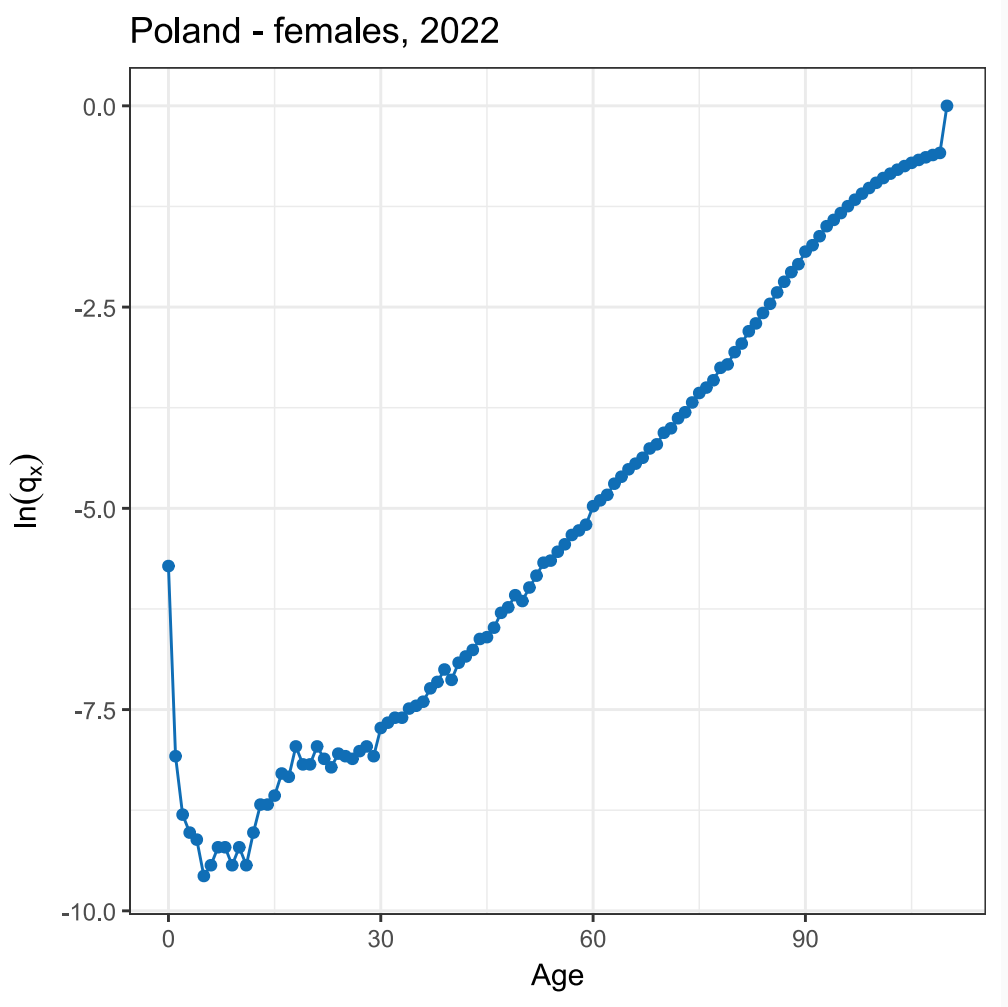
Visualize a period life table

We start with visualizing the $q_{x,t}$ for M/F data in Poland, year 2022.

Using `ggplot` instructions:

```
g_male ←
  ggplot(POL_male_2022, aes(Age, log(qx))) +
  geom_point(col = RCLRbg) +
  geom_line(col = RCLRbg) +
  theme_bw(base_size = 15) +
  ggtitle("Poland - males, 2022") +
  labs(y = bquote(ln(q[x])))

g_fem ←
  ggplot(POL_female_2022, aes(Age, log(qx))) +
  geom_point(col = RCLRbg) +
  geom_line(col = RCLRbg) +
  theme_bw(base_size = 15) +
  ggtitle("Poland - females, 2022") +
  labs(y = bquote(ln(q[x])))
```



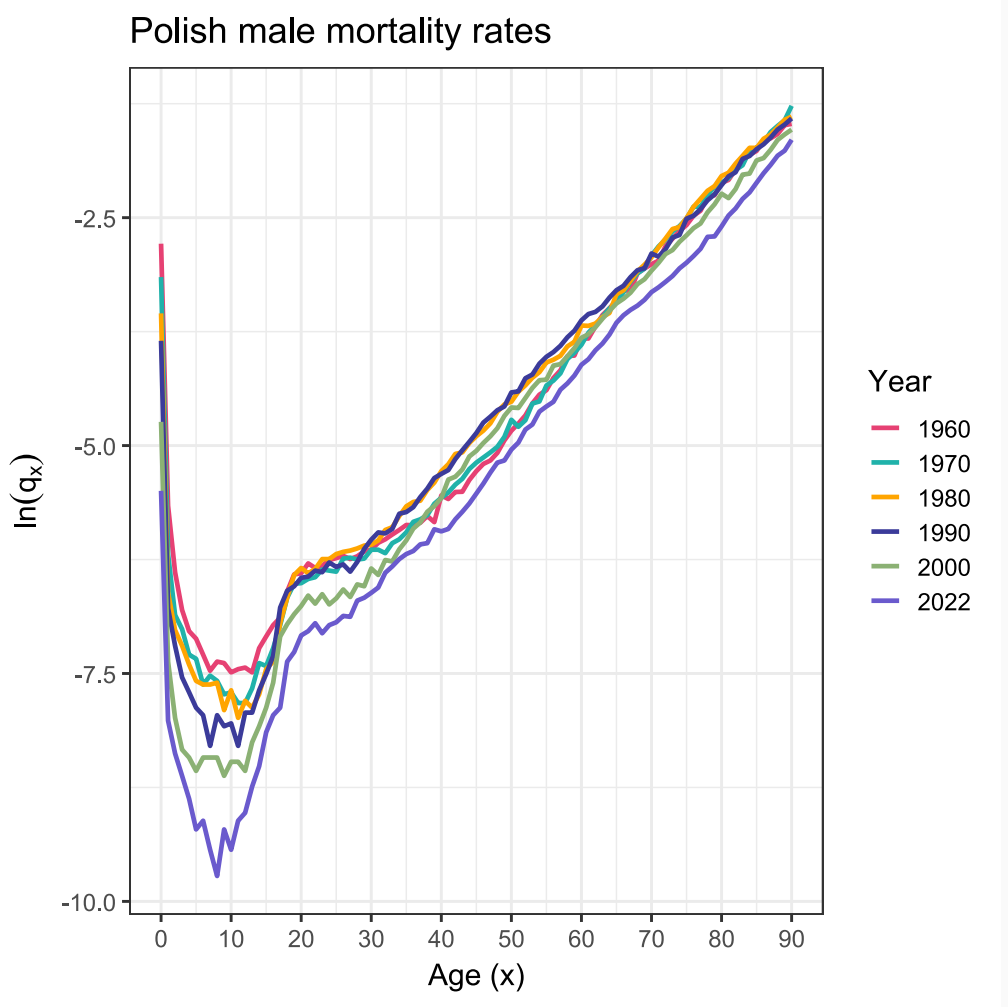
Visualize the mortality rates over time

We visualize the $q_{x,t}$ for male and female data in Poland, ages 0-90 and (selected) years 1960, 1970, 1980, 1990, 2000, 2022.

Using `ggplot` instructions:

```
years ← c(1960, 1970, 1980, 1990, 2000, 2022)
dfM   ← POL_male %>%
  dplyr::filter(Year %in% years,
                Age ⩽ 90)

ggplot(dfM, aes(x = Age, y = log(qx),
                group = Year, colour = factor(Year))) +
  geom_line(linewidth = 1.1)  +
  ggtitle('Polish male mortality rates') +
  xlab('Age (x)') + ylab(bquote(ln(q[x]))) +
  scale_color_manual(values = c(red_pink, turquoise,
                                orange, blue, green,
                                purple), name = 'Year') +
  scale_x_continuous(breaks = seq(0, 90, 10)) +
  theme_bw(base_size = 15)
```



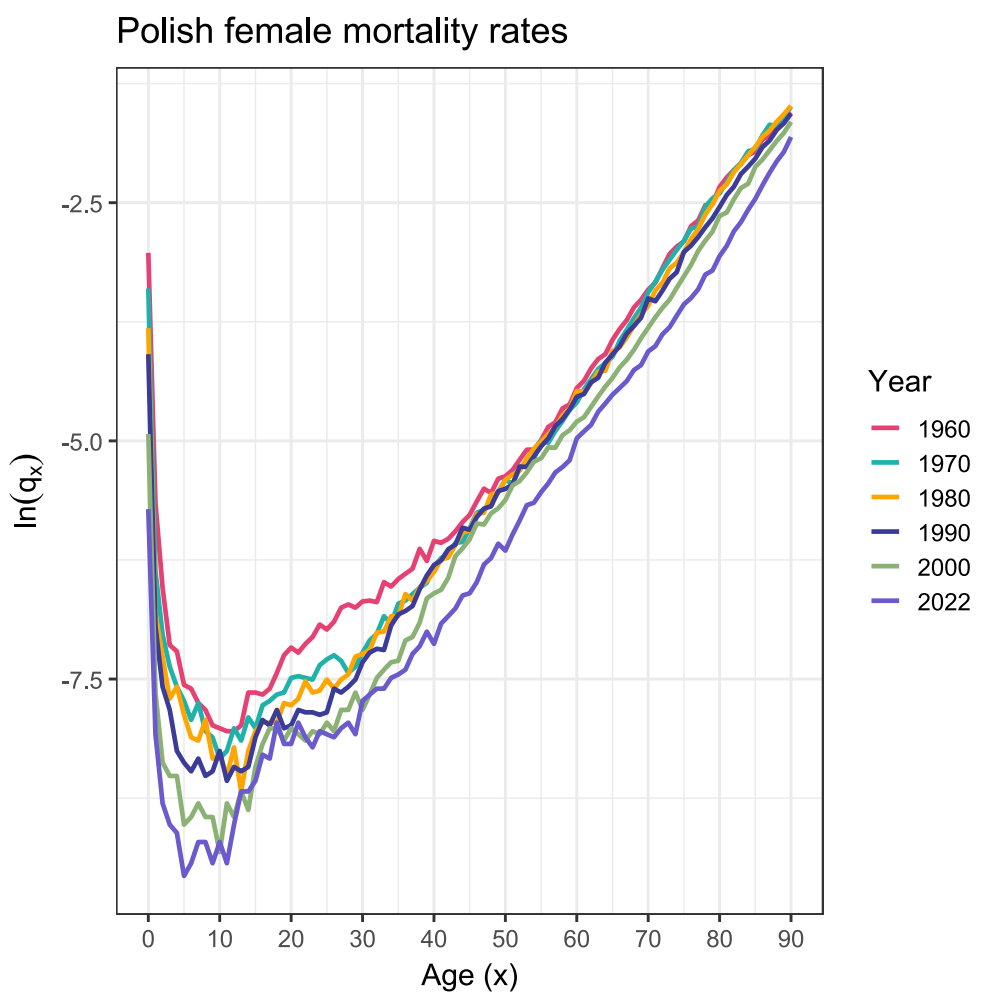
Visualize the mortality rates over time

We visualize the $q_{x,t}$ for male and female data in Poland, ages 0-90 and (selected) years 1960, 1970, 1980, 1990, 2000, 2022.

Using `ggplot` instructions:

```
years ← c(1960, 1970, 1980, 1990, 2000, 2022)
dfF   ← POL_female %>%
  dplyr::filter(Year %in% years,
                Age ⩽ 90)

ggplot(dfF, aes(x = Age, y = log(qx),
                group = Year, colour = factor(Year))) +
  geom_line(linewidth = 1.1)  +
  ggtitle('Polish female mortality rates') +
  xlab('Age (x)') + ylab(bquote(ln(q[x]))) +
  scale_color_manual(values = c(red_pink, turquoise,
                                orange, blue, green,
                                purple), name = 'Year') +
  scale_x_continuous(breaks = seq(0, 90, 10)) +
  theme_bw(base_size = 15)
```



Visualize the survival curve

We put focus on $S_{0,t}(.)$, the survival curve for a newborn, born in period (or year) t.

We calculate the survival probabilities $S_{0,t}(x)$ for a Polish male newborn, for selected time period t. Hereby we let x run from 0 to 100.

We consider a minimum age x of 0, a maximum age of 100, and the years of interest.

```
xmax  ← 100
xmin  ← 0
years ← c(1960, 1970, 1980, 1990, 2000, 2022)
surv_rate ← matrix(0, xmax-xmin+1, length(years))
surv_rate[1,] ← 1

for(t in 1:length(years)){
  for(x in 1:(xmax-xmin)){
    df.t ← POL_male %>%
      dplyr::filter(Year == years[t])
    px ← 1 - df.t[x,'qx']
    surv_rate[x+1,t] ← surv_rate[x,t]*px
  }
}
```

Visualize the survival curve

We put focus on $S_{0,t}(.)$, the survival curve for a newborn, born in period (or year) t.

We calculate the survival probabilities $S_{0,t}(x)$ for a Polish male newborn, for selected time period t. Hereby we let x run from 0 to 100.

```
xmax  ← 100
xmin  ← 0
years ← c(1960, 1970, 1980, 1990, 2000, 2022)
surv_rate ← matrix(0, xmax-xmin+1, length(years))
surv_rate[1,] ← 1

for(t in 1:length(years)){
  for(x in 1:(xmax-xmin)){
    df.t ← POL_male %>%
      dplyr::filter(Year == years[t])
    px ← 1 - df.t[x,'qx']
    surv_rate[x+1,t] ← surv_rate[x,t]*px
  }
}
```

We consider a minimum age x of 0, a maximum age of 100, and the years of interest.

We create an empty matrix `surv_rate` of dimension 101×6 to store the calculated survival probabilities in the years 1960, 1970, 1980, 1990, 2000, 2022.

Visualize the survival curve

We put focus on $S_{0,t}(.)$, the survival curve for a newborn, born in period (or year) t.

We calculate the survival probabilities $S_{0,t}(x)$ for a Polish male newborn, for selected time period t. Hereby we let x run from 0 to 100.

```
xmax  ← 100
xmin  ← 0
years ← c(1960, 1970, 1980, 1990, 2000, 2022)
surv_rate ← matrix(0, xmax-xmin+1, length(years))
surv_rate[1,] ← 1

for(t in 1:length(years)){
  for(x in 1:(xmax-xmin)){
    df.t ← POL_male %>%
      dplyr::filter(Year == years[t])
    px ← 1 - df.t[x,'qx']
    surv_rate[x+1,t] ← surv_rate[x,t]*px
  }
}
```

We consider a minimum age x of 0, a maximum age of 100, and the years of interest.

We create an empty matrix `surv_rate` of dimension 101×6 to store the calculated survival probabilities in the years 1960, 1970, 1980, 1990, 2000, 2022.

The probability of surviving zero years equals one, i.e. $S_{0,t}(0) = 1$.

Visualize the survival curve

We put focus on $S_{0,t}(.)$, the survival curve for a newborn, born in period (or year) t.

We calculate the survival probabilities $S_{0,t}(x)$ for a Polish male newborn, for selected time period t. Hereby we let x run from 0 to 100.

```
xmax  ← 100
xmin  ← 0
years ← c(1960, 1970, 1980, 1990, 2000, 2022)
surv_rate ← matrix(0, xmax-xmin+1, length(years))
surv_rate[1,] ← 1
```

```
for(t in 1:length(years)){
  for(x in 1:(xmax-xmin)){
    df.t ← POL_male %>%
      dplyr::filter(Year == years[t])
    px ← 1 - df.t[x,'qx']
    surv_rate[x+1,t] ← surv_rate[x,t]*px
  }
}
```

We consider a minimum age x of 0, a maximum age of 100, and the years of interest.

We create an empty matrix `surv_rate` of dimension 101×6 to store the calculated survival probabilities in the years 1960, 1970, 1980, 1990, 2000, 2022.

The probability of surviving zero years equals one, i.e. $S_{0,t}(0) = 1$.

For every year t under consideration, the one-year survival probability of (x) in year t is $S_{x,t}(1) = p_{x,t} = 1 - q_{x,t}$. We evaluate the survival probabilities recursively, as follows:

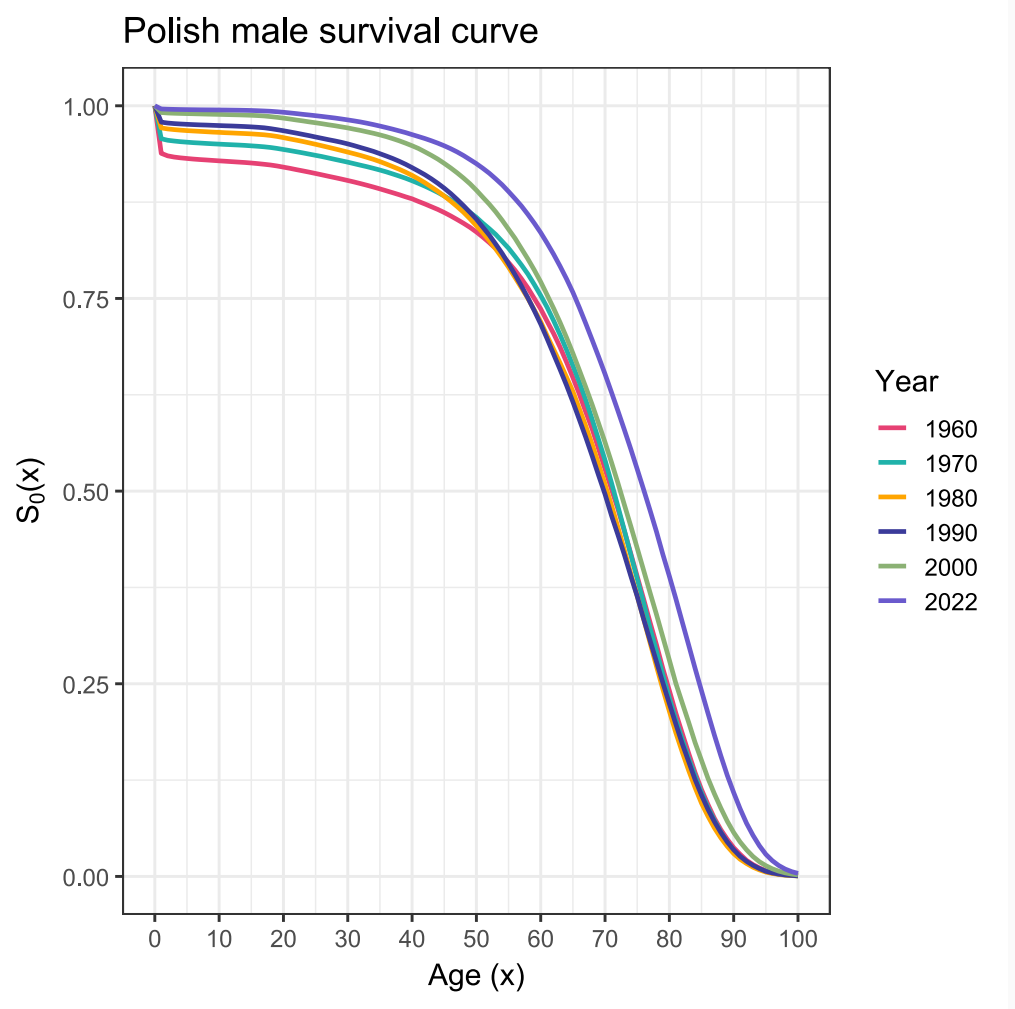
$$S_{0,t}(x+1) = S_{0,t}(x) \cdot p_{x,t}.$$

Hereby, we follow the period approach (and keep t fixed)!

We plot these survival probabilities calculated for a male Polish newborn in the selected years.

```
df ← data.frame('Age' = rep(0:100, times = length(yea
                'Year' = rep(years, each = length(0:1
                'Surv' = as.numeric(surv_rate))

ggplot(df, aes(x = Age, y = Surv,
               group = Year, colour = as.factor(Year))
  geom_line(linewidth = 1.1)  +
  ggtitle('Polish male survival curve') +
  xlab('Age (x)') + ylab(bquote(S[0]*'(x)')) +
  scale_color_manual(values = c(red_pink, turquoise,
                                orange, blue, green,
                                purple), name = 'Year'
  scale_x_continuous(breaks = seq(0, 100, 10)) +
  theme_bw(base_size = 15)
```



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.

To get started with the package, first install it:

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

Then, load the package:

```
library(demography)
```

Next, we use the `hmd.mx` function from {demography} to read the "Mx" (1 x 1) data from the HMD:

```
? hmd.mx
User = "summerschool.rclr2024@outlook.com"
pw   = "Test1234."
Df   = hmd.mx("POL", User , pw , "Poland")
```

We specify the country code (`POL`), the user name and password, along with the character string containing the country name from which the data is retrieved.

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

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

Constructing the mortality data set

```
User = "summerschool.rclr2024@outlook.com"
pw   = "Test1234."
Df   = hmd.mx("POL", User , pw , "Poland")
```

Use the `hmd.mx` function from {demography} to read the "Mx" (1 x 1) data from the HMD.

```
years ← 1970:2023
ages  ← 0:90
```

We define a calibration period `years` and an age range `ages` on which we will calibrate the Lee-Carter model.

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

We use `extract.years` and `extract.ages` from the {demography} package to subset `Df` according to the specified `years` and `ages`.

```
dim(Df$rate$male)
## [1] 91 54
dim(Df$pop$female)
## [1] 91 54
View(Df$rate$female)
```

The death rates are stored in `Df$rate` for males (`$male`) and females (`$female`) and the exposures are contained in `Df$pop`. These statistics are stored in a 91x54 matrix, with in the rows the ages, and in the columns the years in the calibration period.

The mortality data set `Df` contains the observed central death rates $m_{x,t}$ (`$rates`) for both males (`$male`) and females (`$female`). These rates correspond to the `mx` column in the life tables:

```
head(POL_female_2022$mx)
## [1] 0.00330 0.00031 0.00015 0.00012 0.00011 0.00007
head(Df$rate$female[,'2022'])
##        0        1        2        3        4
## 0.003299 0.000308 0.000152 0.000118 0.000112 0.00007
```

The `$pop` attribute in `Df` stores the exposure-to-risk $E_{x,t}$, representing the total amount of person-years lived by individuals aged x in year t.

```
head(Df$pop$female[,'2022'])
##        0        1        2        3        4
## 153374.7 165559.1 177734.0 185906.8 195566.6 198346.
```

Death counts can be calculated by multiplying the death rates with the exposure-to-risk, i.e., $d_{x,t} = E_{x,t} \cdot m_{x,t}$. In R we do:

```
round(head(Df$pop$female[,'2022'] * Df$rate$female[,'20
##   0   1   2   3   4   5
## 506  51  27  22  22  14
```

These computed death counts match the death counts from the Human Mortality Database (HMD) (see here).

What if {demography} does not work?

```
Df ← readRDS(file = "../data/hmd/Df_POL_hmd_mx.rds")
```

Read in the pre-downloaded mortality data set `Df` of Poland using the `readRDS` function.

```
row ← Df$age
col ← Df$year

Df$pop$female ← Df$pop$female[row ⩽ 90, col ⩾ 1970]
Df$pop$male   ← Df$pop$male[row ⩽ 90, col ⩾ 1970]
Df$pop$total  ← Df$pop$total[row ⩽ 90, col ⩾ 1970]

Df$rate$female ← Df$rate$female[row ⩽ 90, col ⩾ 1970
Df$rate$male   ← Df$rate$male[row ⩽ 90, col ⩾ 1970]
Df$rate$total  ← Df$rate$total[row ⩽ 90, col ⩾ 1970]
```

We filter the exposures and death rates, contained in `Df`, according to the specified age range and calibration period. We do this by filtering the rows on ages below 90 (maximum age in `ages`) and years beyond the year 1970 (minimum year in `years`). This is what happens inside the functions `extract.years` and `extract.ages` of the {demography} package.

```
dim(Df$rate$male)
## [1] 91 54
dim(Df$pop$female)
## [1] 91 54
View(Df$rate$female)
```

The mortality data object `Df` is now in the same structure as in the previous slide.

Fit single population mortality models by maximizing a Poisson likelihood

Fitting the Lee-Carter model

```
etx ← t(Df$pop$male)
dtx ← round(etx * t(Df$rate$male))
```

We extract the male exposures and store these in `etx`.
Note the use of the transpose function `t( ... )` to make sure the years are now in the rows and the ages in columns (! required by `fit701` function !).
We extract the male death rates and multiply these with the exposures to obtain the death counts `dtx`.

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

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

As inputs we have the age range (`ages`), the calibration period (`years`), the exposures (`etx`), the death counts (`dtx`), and a matrix of unit weights.

The `fit701(.)` function is from the `fitModels.R` script.

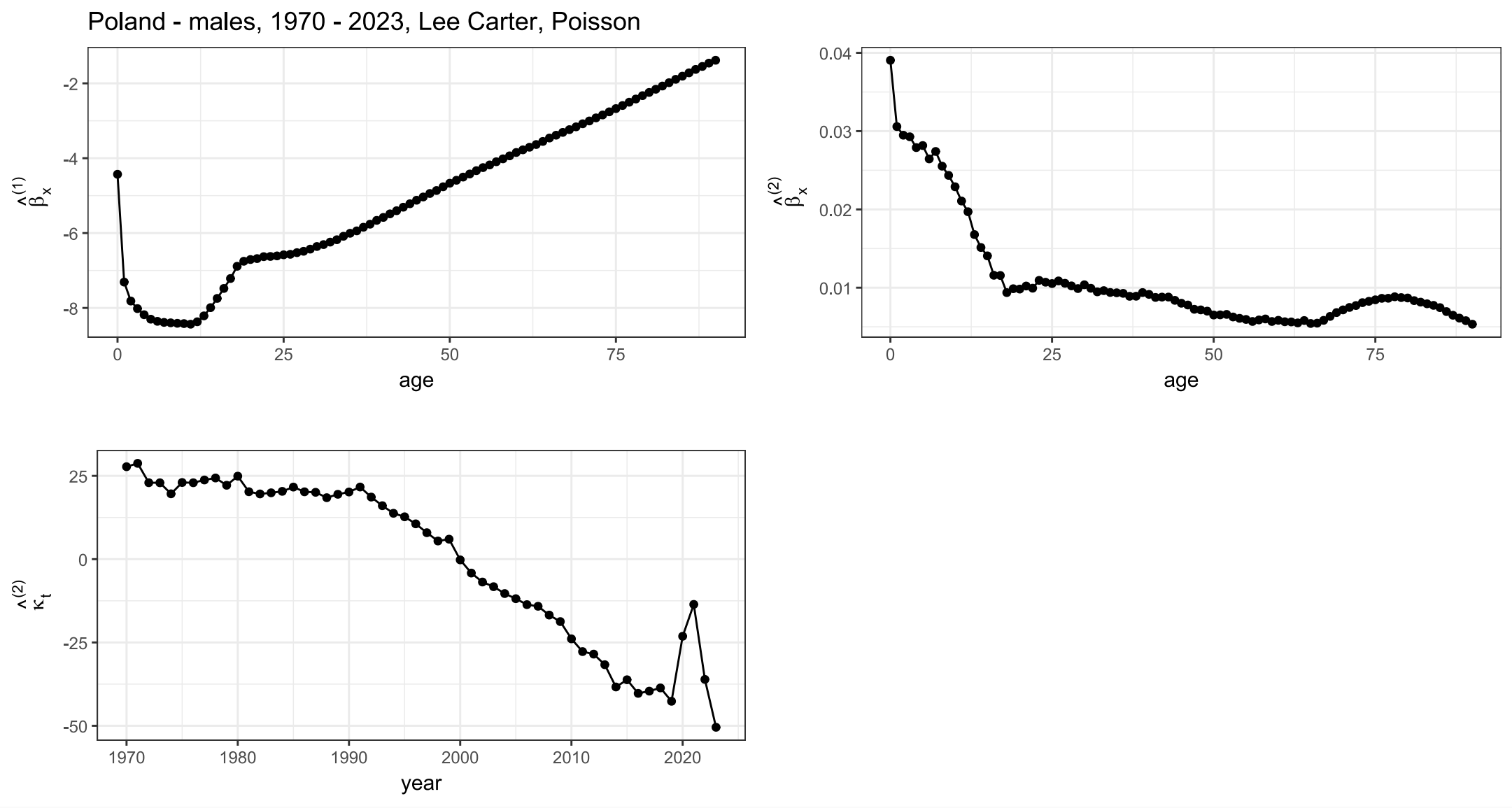
This method is written by Andrew Cairns, see LifeMetrics software, and uses univariate Newton-Raphson (NR) steps to optimize the log-likelihood.

`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.

Outputs from the fitted Lee-Carter model

```
LCfit701$beta1[1:4]
## [1] -4.429355 -7.309327 -7.816010 -8.017979
LCfit701$beta2[1:4]
## [1] 0.03905182 0.03058030 0.02947736 0.02927654
LCfit701$kappa2[1:4]
## [1] 27.73950 28.74882 22.94013 22.91997
```

The parameter estimates from the fitted Lee-Carter mortality model are retrieved by calling `$beta1` for the $\beta_x^{(1)}$, `$beta2` for the $\beta_x^{(2)}$, and `$kappa2` for the $\kappa_t^{(2)}$. We print the first four estimates.


Poland - males, 1970 - 2023, Lee Carter, Poisson
$\hat{\beta}_x^{(1)}$
-2
-4
-6
-8
0
25
50
75
age
$\hat{\beta}_x^{(2)}$
0.04
0.03
0.02
0.01
0
25
50
75
age
$\hat{\kappa}_t^{(2)}$
25
0
-25
-50
1970
1980
1990
2000
2010
2020
year

```
str(LCfit701$mhat)
##  num [1:54, 1:91] 0.0352 0.0366 0.0292 0.0292 0.0256
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:54] "1970" "1971" "1972" "1973" ...
##   ..$ : chr [1:91] "0" "1" "2" "3" ...
```

The `mhat` object contains the fitted force of mortality $\hat{\mu}_{t,x}$ and has dimension names (`dimnames`): the row names are the years in the calibration period and the column names are the ages in the considered age range.

```
LCfit701$mhat['2019','65']
## [1] 0.02490507
LCfit701$mhat[50,66]
## [1] 0.02490507
```

Using these row and column names, you can easily extract information from `mhat`. E.g., you can retrieve the force of mortality in the year 2019 for age 65 using character notations. Alternatively, you extract the 50th row and the 66th column to obtain $\hat{\mu}_{2019,65}$.

Outputs from the fitted Lee-Carter model

```
exp(LCfit701$beta1[66] +
  LCfit701$beta2[66]*LCfit701$kappa2[50])
## [1] 0.02490507
```

We can verify the value for $\hat{\mu}_{2019,65}$, by calculating it manually using the formula:

$$\hat{\mu}_{2019,65} = \exp\left(\hat{\beta}^{(1)}_{65} + \hat{\beta}^{(2)}_{65} \cdot \hat{\kappa}^{(2)}_{2019}\right).$$

```
qhat ← 1 - exp(-LCfit701$mhat)
qhat['2019','65']
## [1] 0.0245975
```

From the fitted forces of mortality, we can then calculate the mortality rates using the relation:

$$\hat{q}_{t,x} = 1 - \exp(-\hat{\mu}_{t,x}).$$

As an example, the estimated mortality rate in the year 2019 for age 65 can be extracted in a similar way.

```
ggplot( ... ) +
   ...
```

We can plot parameter estimates, estimated forces of mortality, mortality rates, survival probabilities,... using {ggplot} instructions.

We compute the observed and estimated mortality rates from the `LCfit701` object:

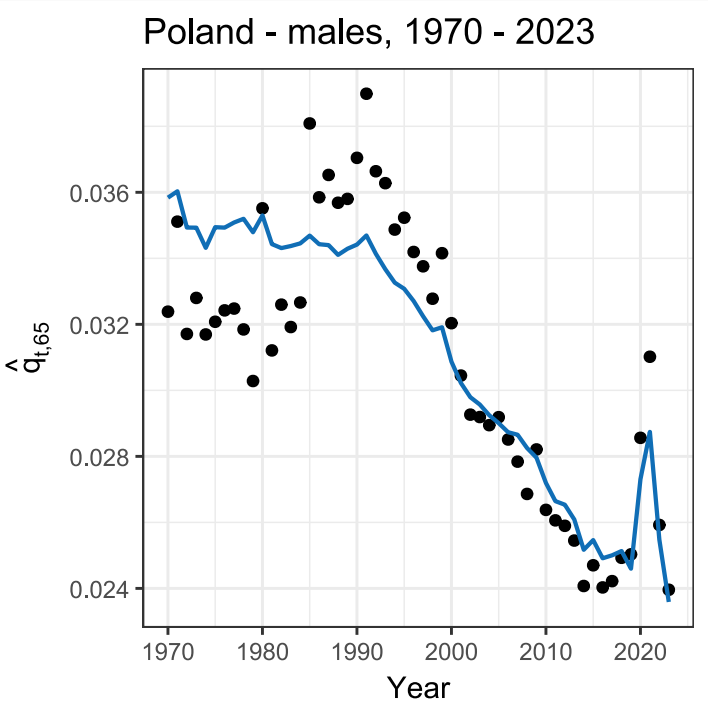
```
qobs ← 1 - exp(-dtx/etx)
qhat ← 1 - exp(-LCfit701$mhat)
```

We construct a data frame that stores the observed and estimated mortality rates at age 65:

```
age    ← '65'
df.age ← data.frame(Year = years, obs = qobs[,age],
                    fit = qhat[,age])
```

We visualize these using {ggplot}.

```
ggplot(df.age) +
  geom_point(aes(x = Year, y = obs)) +
  geom_line(aes(Year, fit), col = RCLRbg,
            linewidth = 1) +
  ggtitle("Poland - males, 1970 - 2023") +
  ylab(bquote(hat(q)['t,65'])) +
  theme_bw(base_size = 15)
```



We compute the observed and estimated mortality rates from the `LCfit701` object:

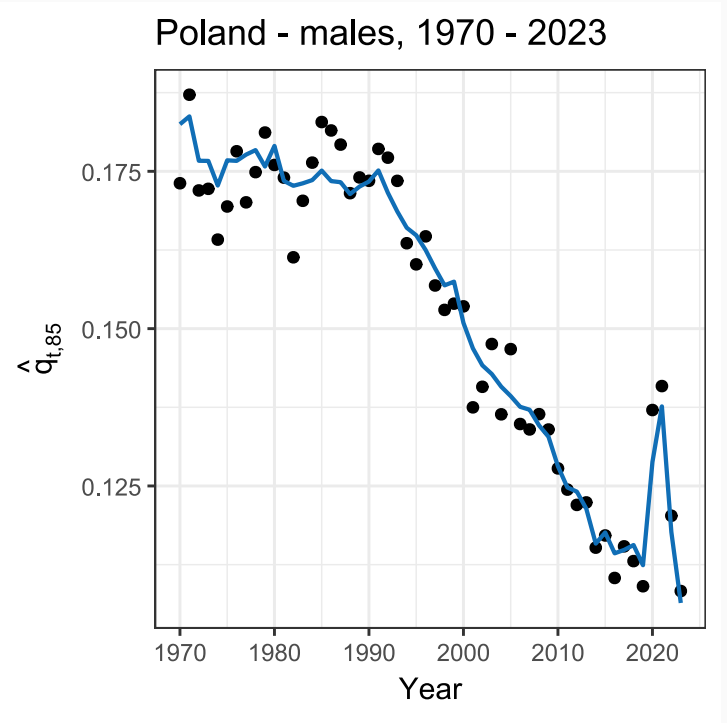
```
qobs ← 1 - exp(-dtx/etx)
qhat ← 1 - exp(-LCfit701$mhat)
```

We construct a data frame that stores the observed and estimated mortality rates at age 85:

```
age    ← '85'
df.age ← data.frame(Year = years, obs = qobs[,age],
                    fit = qhat[,age])
```

We visualize these using {ggplot}.

```
ggplot(df.age) +
  geom_point(aes(x = Year, y = obs)) +
  geom_line(aes(Year, fit), col = RCLRbg,
            linewidth = 1) +
  ggtitle("Poland - males, 1970 - 2023") +
  ylab(bquote(hat(q)['t,85'])) +
  theme_bw(base_size = 15)
```



Model selection and goodness-of-fit

Model selection

Various mortality model specifications have been proposed in the literature.

How to select an appropriate mortality model? Here are a few tools.

- **Goodness-of-Fit**: The model should accurately fit the historical, observed data. This can be evaluated using Pearson residuals:

$$\epsilon_{x,t} = \frac{d_{x,t} - \hat{\mu}_{x,t} \cdot E_{x,t}}{\sqrt{\hat{\mu}_{x,t} \cdot E_{x,t}}}.$$

- **Predictive Performance**: The model should perform well on both in-sample statistical measures as well as out-of-time back-tests.
- **Parsimony**: The model should have a good balance between complexity and goodness-of-fit. Compare the AIC/BIC values of different calibrated models to assess this balance.
- **Reasonable**: The chosen model should lead to mortality rates (and their evolutions) that are biologically and demographically reasonable.
- ...

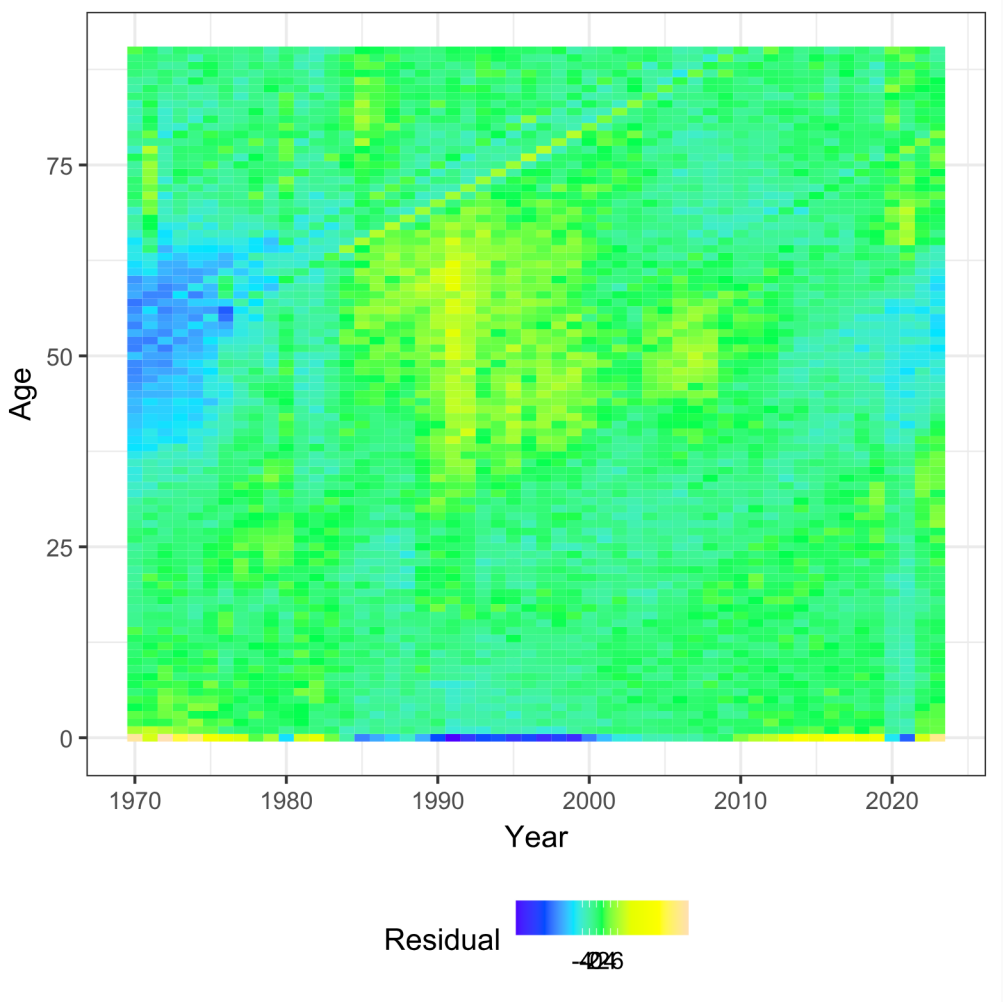
Pearson residuals

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")

ggplot(grid, aes(x = Year, y = Age)) +
  geom_tile(aes(fill = Residual)) +
  scale_fill_gradientn(colours = topo.colors(7),
                       breaks = c(-4,-2,0,2,4,6)) +
  theme_bw(base_size = 15) +
  theme(legend.position = "bottom")
```

```
head(grid)[1:3,]
##   Year Age  Residual
## 1 1970   0 24.760867
## 2 1971   0  8.865804
## 3 1972   0 26.241043
```



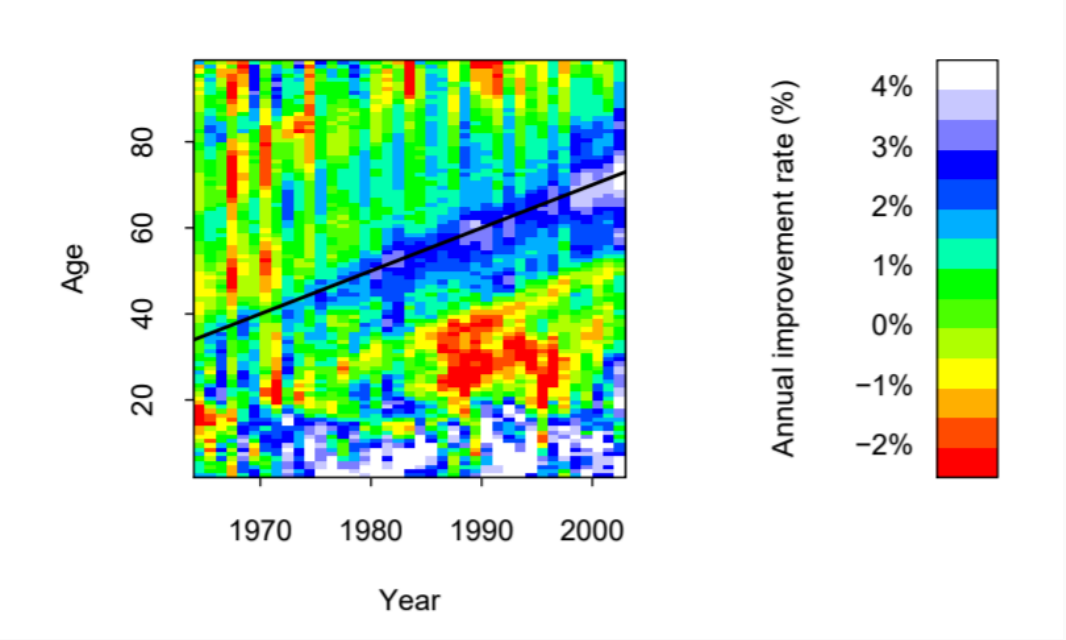
Diagonal patterns in heatmaps of residuals might hint to the inclusion of a cohort effect in a stochastic mortality model.

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 typically comes with a lot of difficulties!



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**.

Thanks!

Slides created with the R package xaringan.

Course material available via

https://github.com/katrienantonio/warsaw-2025