KOSTAT-UNFPA Summer Seminar on Population

Workshop 1. Introduction to Demography

Day 5: Projection

Instructor:
Tim Riffe `tim.riffe@ehu.eus`

Assistant:
Inchan Hwang `inchan@yonsei.ac.kr`

27 June 2025

Contents

1 Summary

This session is meant to show one important way to think about forecasting: to abstract the data until the time series can be linearized (easier to project), then transform back to the original scale. This script is in part inherited from Marie-Pier Bergeron Boucher, by now modified and edited numerous times. Her original version of this material forecasts both mortality and fertility using a Lee-Carter model, and then performs a population projection, as we did with stable populations, but using time-varying forecasted Leslie matrices. That's of course an excellent exercise, but there's no way we can fit it into this session. Instead we will just do a mortality forecast using the basic Lee-Carter method. Parts of this will be in base `R`, while others are reformulated using the tidy approach. You may see some neat tricks for toggling back and forth between tidy objects and matrices.

2 Demographic forecasting

In the previous class, we saw how to project population size, assuming that the population is stable, i.e. the age-specific fertility and mortality rates are constant over time. These assumptions are, however, rarely met. As shown in class 1 and 2 of this module, life expectancy has been increasing over time and TFR has been decreasing, although there are exceptions to both trends.

To obtain a plausible forecast, one must take into account these changes in fertility and mortality over time.

There are four main approaches to forecasting, as defined by Booth (2006):

1) Extrapolative methods: *This is the most common approach.* It assumes that the future trends will be a continuity of the past. It requires little to no subjective judgment.

2) Expectation methods: These methods use expectations of individuals and experts about their own and population level behaviors, respectively. The methods integrate expectations and informed judgment of experts in the forecast.

3) Structural modelling: This approach consists in explaining demographic rates by determinants, such as socio-economic determinants.

4) Decomposition and disaggregation: This approach consists in breaking down rates beyond age and sex. For example, decomposing mortality by causes of deaths and fertility by parity.

For matter of simplicity, migration forecasts are set aside in this class. We will assume zero net migration in our projections, even if this assumption is considered at best naive. However, please remember that migration should also be considered in population projections. In industrialized countries, simple assumptions about future migration are often made, e.g. zero net migration or continuing of the current level of migration (with fixed age pattern). More recently, forecasters also use informed judgment and extrapolation of past trend to predict future migration flow (Booth 2006).

3 Data

We will use the Spanish females' data from the HMD (Human Mortality Database 2018). You can read the output file `ESmx.csv` directly from the github site, see code in next chunk.

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to b
```

```r
ES <- read_csv("Data/ESmx.csv")
```

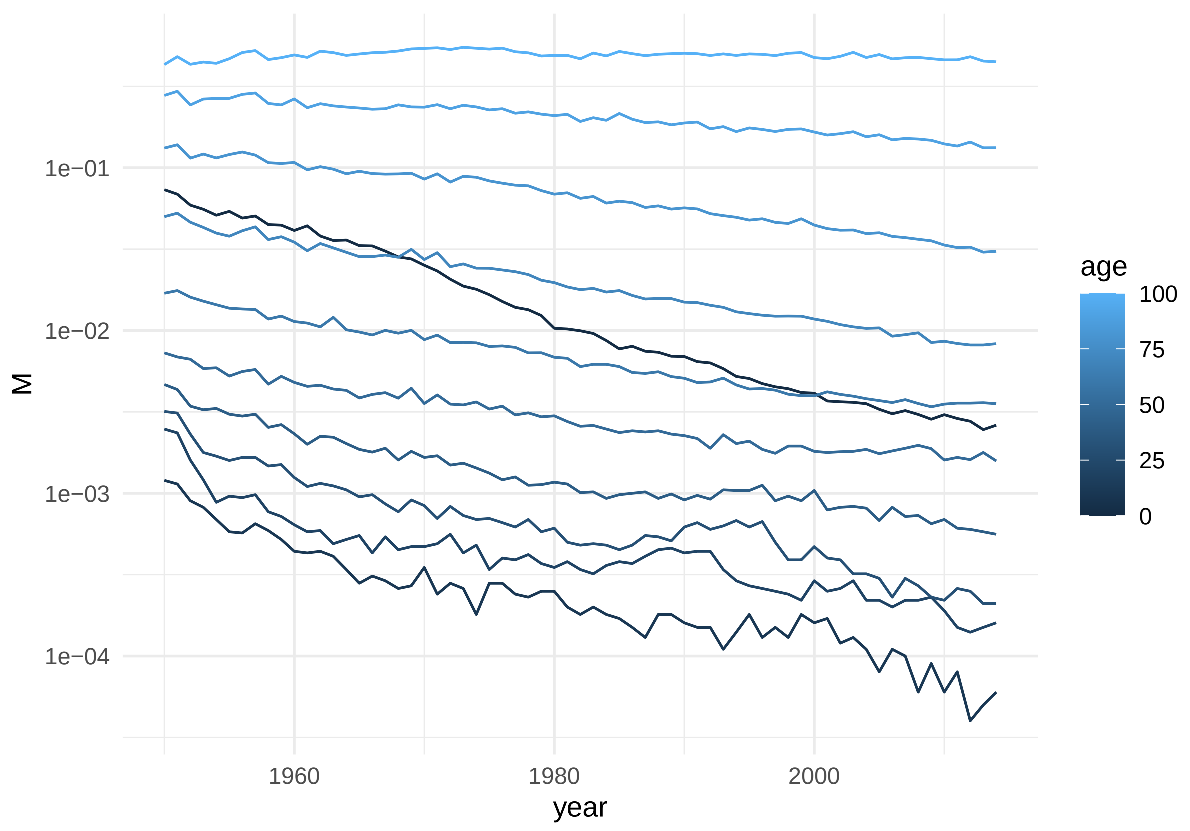
```
## Rows: 6565 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (5): year, age, exposure, deaths, M
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

4 Mortality forecasting

Since the 1980s, many models have been suggested to forecast mortality. From simple models, such as extrapolating life expectancy, to more complex models including cohort effects and parameterization functions.

A great deal of attention to mortality forecasting comes from the observation that death rates have been declining in a log-linear way over time, with (almost) no sign of deceleration.

```r
ES |>
  filter(age %% 10 == 0) |>
  ggplot(aes(x = year, y = M, color = age, group = age)) +
  geom_line() +
  scale_y_log10() +
  theme_minimal()
```



The most popular approach to mortality forecasting is the Lee-Carter model (Lee and Carter 1992). Since 1992, most development in mortality forecasting methods has focused on extending and improving the Lee-Carter model Booth (2006). We will focus on the original Lee-Carter model. This will help understand the mechanics of its extensions in case you ever go looking at those.

4.1 The Lee-Carter model

The Lee-Carter model (Lee and Carter 1992) forecasts the age-specific death rates in a log-linear way, using *one* age-pattern and *one* time-index of mortality changes. The general equation is written as:

$$ln(m_x(t)) = \alpha_x + \beta_x \kappa(t) + \epsilon_x(t)$$

where α_x is the age-specific average of the log death rates $(m_x(t))$. The β_x and κ_t are found with a Singular Value Decomposition (SVD) and consist in the normalized first right and left singular vectors. They are interpreted as the age-pattern (age-specific rate of change relative to $\kappa(t)$) of mortality change and the time-pattern of mortality change. Mortality is forecast by extrapolating $\kappa(t)$. The term $\epsilon_x(t)$ is the error of the model.

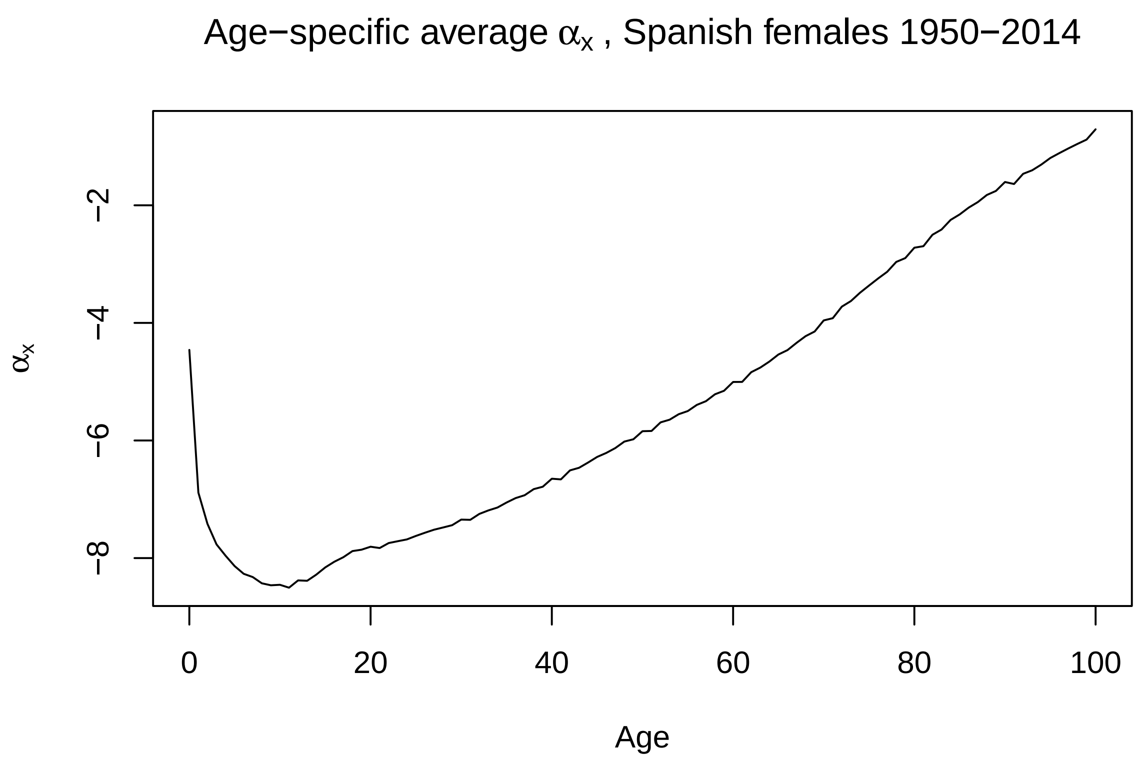
Let's go step by step!

Step 1: Calculate the age-specific average of the log death rates.

```r
#Log transformed data
lM <-
  ES |>
  select(year, age, M) |>
  pivot_wider(names_from = year, values_from = M) |>
  column_to_rownames("age") |>
  as.matrix() |>
  log()

ax  <- rowMeans(lM)
Age <- rownames(lM) |> as.integer()
```

Examine α_x in a base-tastic plot

```
plot(Age, ax, type="l",
     ylab=expression(alpha[x]), xlab="Age",
     main=expression("Age-specific average"~alpha[x]~", Spanish females 1950-2014"))
```



Step 2: Center the matrix on α_x

```
#Center the matrix
cent_lM <- sweep(lM, 1, ax, FUN = "-")
```

Step 3: Do SVD on the centered matrix

In simple terms, the Singular Value Decomposition (SVD) is a factorization of a matrix. It uses eigenvectors and eigenvalue and is seen as a generalization of the eigen decomposition.

It decomposes an $m\ x\ n$ matrix in an $m\ x\ m$ matrix, named U, (the columns consisting of left-singular vectors), an $m\ x\ n$ diagonal matrix with non-negative real numbers (singular values) and an $n\ x\ n$ matrix, named V, (the columns consisting of right-singular vectors).

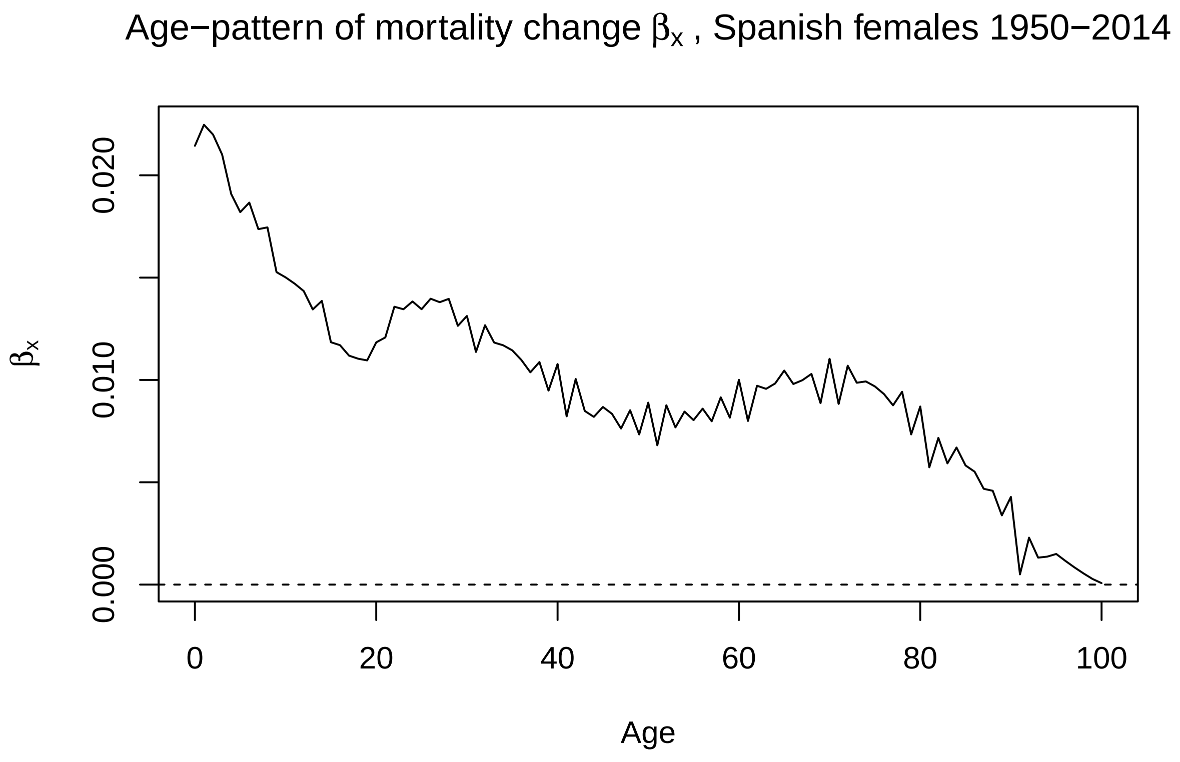
Here, the left-singular vectors capture the age-pattern of mortality change and the right-singular vectors, the time-pattern of mortality change. For further insight on SVD in demography, take a look at Monica Alexander's 2017 blog post https://www.monicaalexander.com/posts/2017-12-16-svd/

```
#SVD: using first vectors and value only

svd_lM <- svd(cent_lM)

u       <- svd_lM$u[, 1]
v       <- svd_lM$v[, 1]
d       <- svd_lM$d[1]

#Explained variance

exp.var<- cumsum((svd_lM$d)^2 / sum((svd_lM$d)^2))[1]
exp.var
```

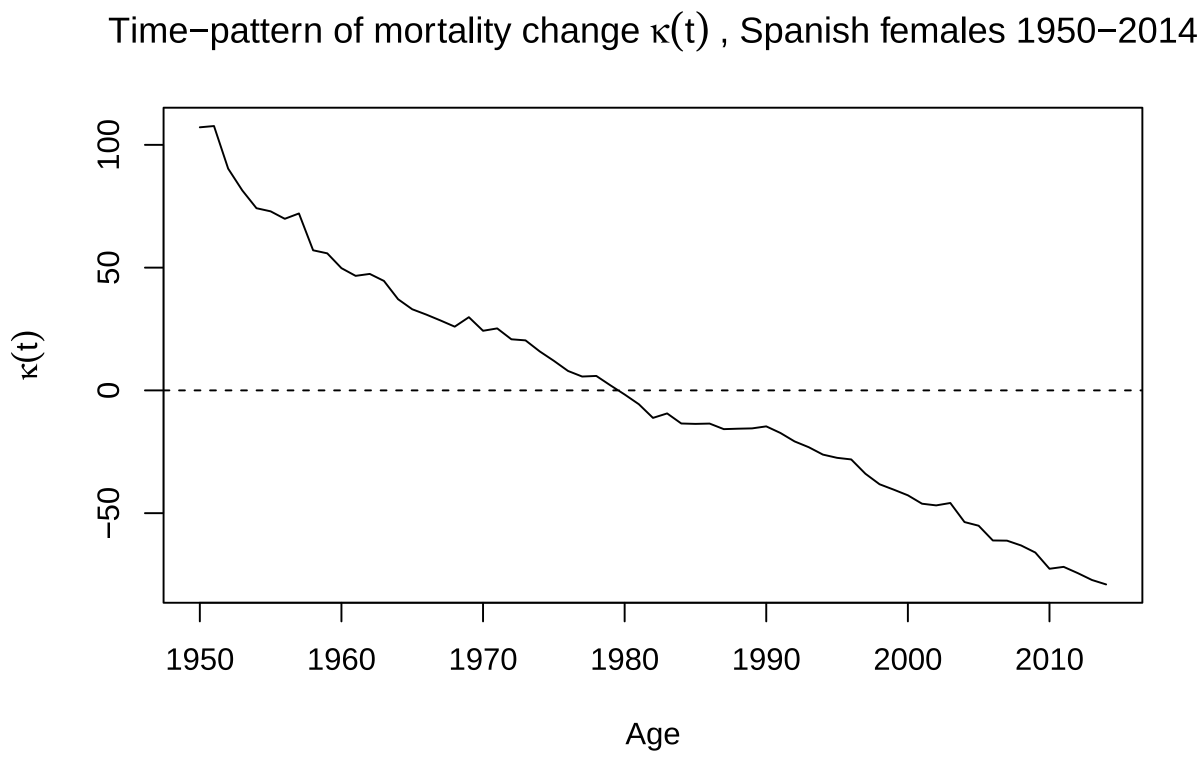
```
## [1] 0.9493005
```

In order to reach a unique solution, the parameters are normalized such that $\sum_x \beta_x = 1$ and $\sum_t \kappa(t) = 0$.

```
#Normalization

Bx <- u / sum(u)
plot(Age, Bx, type="l",
     ylab=expression(beta[x]), xlab="Age",
     main=expression("Age-pattern of mortality change"~beta[x]~", Spanish females 1950-2014"
abline(h=0, lty=2)
```



```
Kt<-v*sum(u)*d
plot(1950:2014, Kt, type="l",
     ylab=expression(kappa(t)), xlab="Age",
     main=expression("Time-pattern of mortality change"~kappa(t)~", Spanish females 1950-201
abline(h=0, lty=2)
```



Step 4: Forecast κ_t

Lee and Carter (1992) suggest forecasting κ_t with a random walk with drift. The method is written as:

$$\kappa(t+1) = \kappa(t) + D + \epsilon_\kappa(t)$$

Where D is the drift and $\epsilon_\kappa(t)$ is the errors from fitting a random walk with drift to $\kappa(t)$.

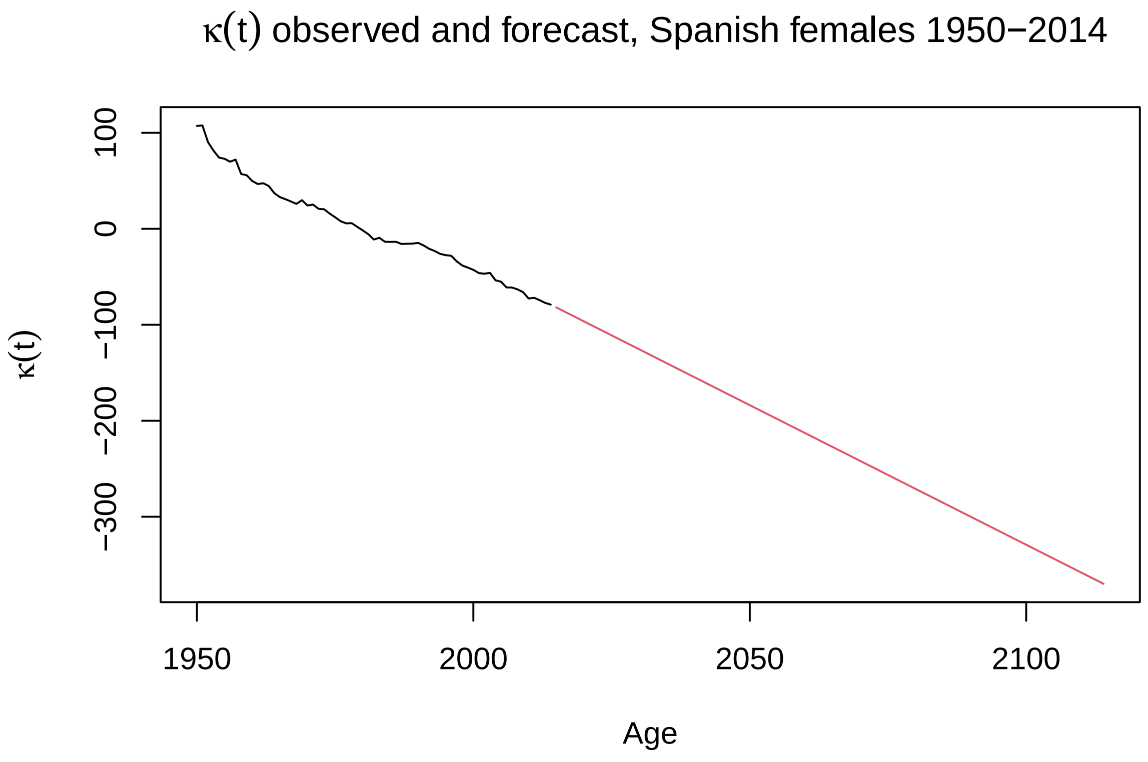
$$D = \frac{\kappa(T) - \kappa(0)}{T-1}$$

where T is the last year observed.

```
#Forecast horizon
h       <- 1:100
Year    <- 1950:2014
#Drift
lK      <- length(Kt)
drift   <- (Kt[lK] - Kt[1]) / (lK - 1)
drift
```

```
## [1] -2.908799
```

```r
#Forecast
Kt.fcst <- Kt[lK] + drift * h
plot(Year, Kt, type = "l", ylim=c(min(Kt.fcst), max(Kt)),
     xlim = c(Year[1], Year[length(Year)]+h[length(h)]),
     ylab = expression(kappa(t)), xlab="Age",
     main = expression(~kappa(t)~"observed and forecast, Spanish females 1950-2014"))
lines(Year[length(Year)]+h, Kt.fcst, col=2)
```



Step 5: Find the prediction intervals (PI)

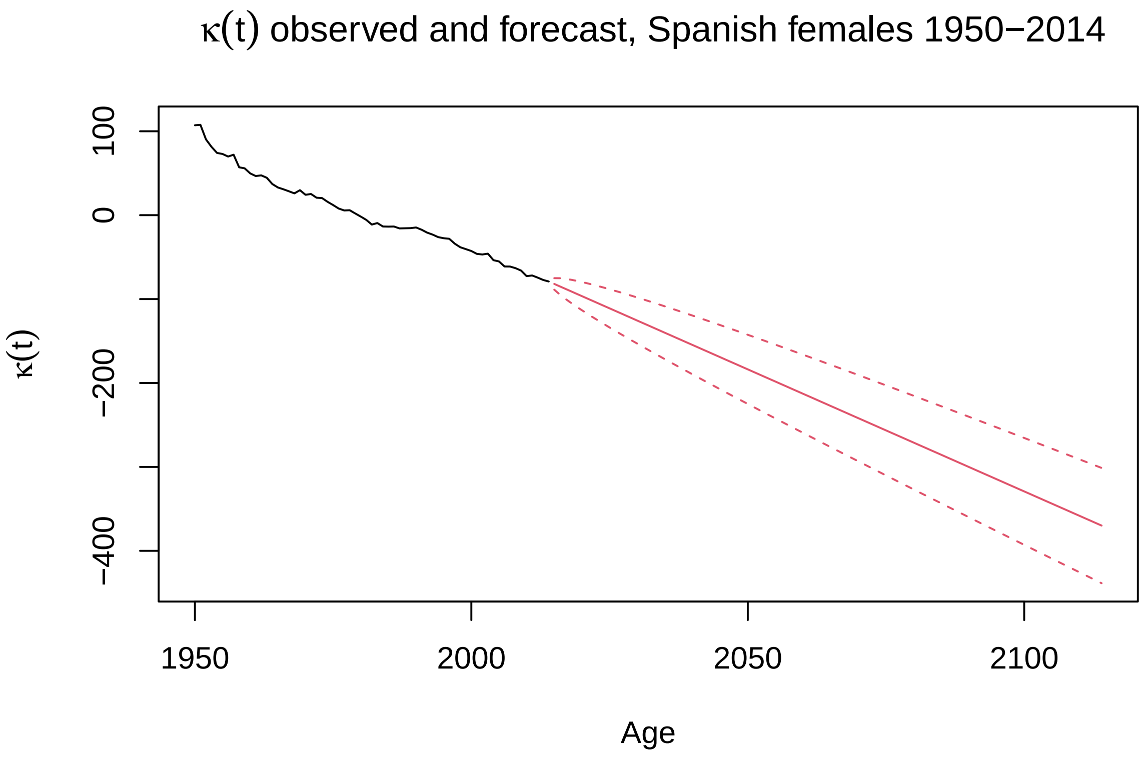
The prediction intervals (PI) are found by finding the standard errors of the estimates (SE) from the random walk with drift model. The SE indicates the uncertainty with one year forecast ahead. It is here assumed that the SE increase with the forecast horizon (h) square root.

$$PI(t+h) = \kappa(t+h) + / - z_{1-a} SE\sqrt{h}$$

where z is the z-score at level a

```r
# Fit random walk with drift
fit.Kt <- Kt[1:(lK - 1)] + drift

# Find SE
se     <- sqrt(sum((Kt[2:lK] - fit.Kt)^2) / (length(Kt) - 1))

# Calculate PI
Kt.95u <- Kt.fcst + 1.96 * se * sqrt(h)
Kt.95l <- Kt.fcst - 1.96 * se * sqrt(h)
```

```r
plot(Year, Kt, type="l", ylim=c(min(Kt.95l), max(Kt)),
     xlim=c(Year[1], Year[length(Year)]+h[length(h)]),
     ylab=expression(kappa(t)), xlab="Age",
     main=expression(~kappa(t)~"observed and forecast, Spanish females 1950-2014"))
lines(Year[length(Year)]+h, Kt.fcst, col=2)
lines(Year[length(Year)]+h, Kt.95u, col=2, lty=2)
lines(Year[length(Year)]+h, Kt.95l, col=2, lty=2)
```



Step 6: Reconstruct and back-transform the matrix

This is just taking the reverse steps of what we first did to M_x.

```r
Bx         <- matrix(Bx, ncol = 1)
mx.fit     <- exp(Bx %*% Kt + ax)
mx.fcst    <- exp(Bx %*% Kt.fcst + ax)
mx.fcst95u <- exp(Bx %*% Kt.95u + ax)
mx.fcst95l <- exp(Bx %*% Kt.95l + ax)
```

Step 8 Let's grab all those pieces and make it tidy again for further calculations and processing.

```r
# add dimension names
dimnames(mx.fit)     <- dimnames(lM)
dimnames(mx.fcst)    <- list(Age = 0:100, Year = 2014 + h)
dimnames(mx.fcst95u) <- list(Age = 0:100, Year = 2014 + h)
dimnames(mx.fcst95l) <- list(Age = 0:100, Year = 2014 + h)

# longer
```

```r
mx_fit <-
  mx.fit |>
  as_tibble(rownames = "Age") |>
  pivot_longer(-Age,
               names_to = "Year",
               values_to = "M") |>
  mutate(variant = "fitted")

mx_fcst <-
  mx.fcst |>
  as_tibble(rownames = "Age") |>
  pivot_longer(-Age,
               names_to = "Year",
               values_to = "forecast")
mx_fcst95u <-
  mx.fcst95u |>
  as_tibble(rownames = "Age") |>
  pivot_longer(-Age,
               names_to = "Year",
               values_to = "lower")
mx_fcst95l <-
  mx.fcst95l |>
  as_tibble(rownames = "Age") |>
  pivot_longer(-Age,
               names_to = "Year",
               values_to = "upper")

# bind the pieces together into single object
fcst <- mx_fcst |>
  left_join(mx_fcst95u, by = c("Age", "Year")) |>
  left_join(mx_fcst95l, by = c("Age", "Year")) |>
  pivot_longer(forecast:upper,
               names_to = "variant",
               values_to = "M") |>
  bind_rows(mx_fit) |>
  mutate(Age = as.integer(Age),
         Year = as.integer(Year))
```

Step 7: Calculate the life table

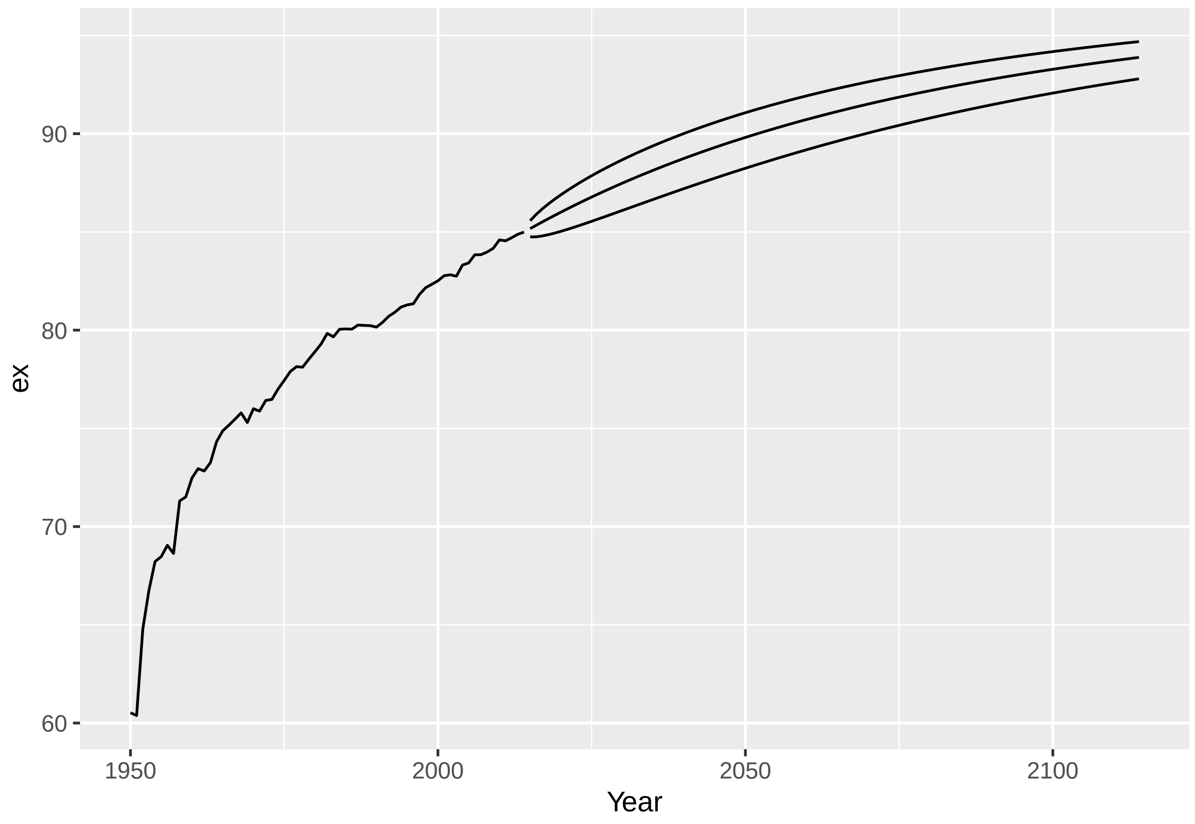
Here the lifetable code is verbatim from Tuesday solutions.

```r
LTfcst <-
  fcst |>
  group_by(Year, variant) |>
  mutate(M = ifelse(is.na(M), .5, M),         # hack
         n = 1,
         ax = case_when(
                Age == 0 & M < .02012 ~ .14916 - 2.02536 * M,
                Age == 0 & M < .07599 ~ 0.037495 + 3.57055 * M,
                Age == 0 & M >= .07599 ~ 0.30663,
                Age == 110 ~ 1 / M,
                TRUE ~ n / 2),
          ax = ifelse(is.infinite(ax),.5,ax),
          qx = (M * n) / (1 + (n - ax) * M),
          qx = ifelse(qx > 1, 1, qx),
          px = 1 - qx,
          lx = c(1, cumprod(px[-n()])),
          dx = qx * lx,
          Lx = lx - (n - ax) * dx,
          Tx = Lx |> rev() |> cumsum() |> rev(),
          ex = Tx / lx)
```

4.2 Visualize results

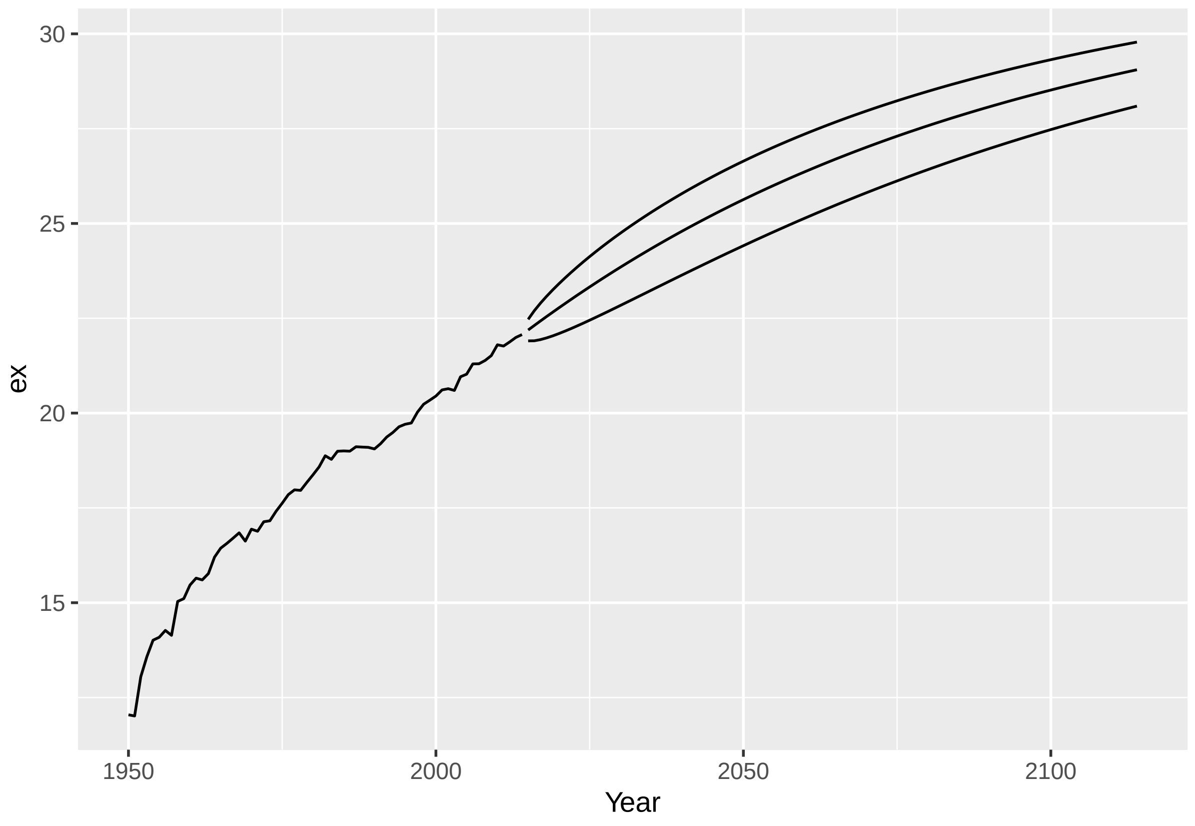
Life expectancy at birth

```
LTfcst |>
  filter(Age == 0) |>
  ggplot(aes(x = Year, y = ex, group = variant)) +
  geom_line()
```



Life expectancy at age 65

```
LTfcst |>
  filter(Age == 65) |>
  ggplot(aes(x = Year, y = ex, group = variant)) +
  geom_line()
```



References

Booth, Heather. 2006. "Demographic Forecasting: 1980 to 2005 in Review." *International Journal of Forecasting* 22 (3): 547–81.

Booth, Heather, and Leonie Tickle. 2008. "Mortality Modelling and Forecasting: A Review of Methods." *Annals of Actuarial Science* 3 (1-2): 3–43.

Human Mortality Database. 2018. "University of California, Berkeley (USA) and Max Planck Institute for Demographic Research (Germany)."

Lee, Ronald D, and Lawrence R Carter. 1992. "Modeling and Forecasting US Mortality." *Journal of the American Statistical Association* 87 (419): 659–71.