

POPULATION PROJECTIONS

Lecture 3

Matrix Projections & Dynamic Visualizations

Ugofilippo Basellini

basellini@demogr.mpg.de



MAX-PLANCK-INSTITUT
FÜR DEMOGRAFISCHE
FORSCHUNG

MAX PLANCK INSTITUTE
FOR DEMOGRAPHIC
RESEARCH

Barcelona
8th June 2022

Brief course summary

Lecture 1: introduction & first simple model of population projections

Lecture 2: cohort component method

Lecture 3:

- ▶ matrix projections
- ▶ dynamic visualizations and shiny apps

Lecture 4: extensions of matrix projections

Small recap

In Lecture 2, we have seen the most employed model for population projections, the cohort component method

- ▶ takes into account age composition of populations
- ▶ projection of age groups rather straightforward, except for the youngest and the oldest
- ▶ projected population from one interval becomes the baseline for next projection
- ▶ several projections can become cumbersome if done one at a time

⇒ we can speed things up with matrix algebra

Introduction

The cohort component method:

- ▶ can be compactly written in matrix notation (Bernardelli 1941, Lewis 1942, Leslie 1945, as reported by Smith and Keyfitz 1977)
- ▶ rewriting facilitates computer programming and shows the use of matrix algebra for population projections
- ▶ simplifies projections over multiple time intervals

Cohort component formulas I

We can write down the female-specific equations for each age group:

$$N_0(t+5) = \sum_x N_x(t) b_x$$

$$N_5(t+5) = N_0(t) s_0$$

$$N_{10}(t+5) = N_5(t) s_5$$

$$\vdots$$

$$N_{45}(t+5) = N_{40}(t) s_{40}$$

$$N_{50}(t+5) = N_{45}(t) s_{45}$$

$$\vdots$$

$$N_{80}(t+5) = N_{75}(t) s_{75}$$

$$N_{85}(t+5) = (N_{80}(t) + N_{85}(t)) s_{80}$$

with $s_x = \frac{L_{x+5}}{L_x}$, $s_{80} = \frac{T_{85}}{T_{80}}$ and $b_x = \frac{1}{1+SRB} \frac{L_0}{2\ell_0} (F_x + s_x F_{x+5})$

Cohort component formulas II

Note the structure of the equations:

$$\begin{array}{rcl}
 N_0(t+5) & = & N_{10}(t)b_{10} + \cdots + N_{45}(t)b_{45} \\
 N_5(t+5) & = & N_0(t)s_0 \\
 N_{10}(t+5) & = & N_5(t)s_5 \\
 N_{15}(t+5) & = & N_{10}(t)s_{10} \\
 & \vdots & \\
 & & \ddots \\
 N_{45}(t+5) & = & N_{40}(t)s_{40} \\
 N_{50}(t+5) & = & N_{45}(t)s_{45} \\
 & \vdots & \\
 & & \ddots \\
 N_{80}(t+5) & = & N_{75}(t)s_{75} \\
 N_{85}(t+5) & = & (N_{80}(t) + N_{85}(t))s_{80}
 \end{array}$$

⇒ this suggests the use of matrix notation

Matrix notation I

Let's rewrite this in matrix form:

$$\begin{bmatrix} N_0(t+5) \\ N_5(t+5) \\ N_{10}(t+5) \\ N_{15}(t+5) \\ \vdots \\ N_{45}(t+5) \\ N_{50}(t+5) \\ \vdots \\ N_{80}(t+5) \\ N_{85}(t+5) \end{bmatrix} = \begin{bmatrix} 0 & 0 & b_{10} & \dots & b_{45} & 0 & \dots & 0 & 0 & 0 \\ s_0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & s_5 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & s_{10} & \dots & 0 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & s_{45} & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & s_{50} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 & \dots & s_{75} & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & s_{80} & s_{80} \end{bmatrix} \begin{bmatrix} N_0(t) \\ N_5(t) \\ N_{10}(t) \\ N_{15}(t) \\ \vdots \\ N_{45}(t) \\ N_{50}(t) \\ \vdots \\ N_{80}(t) \\ N_{85}(t) \end{bmatrix}$$

Matrix notation II

More compactly:

$$\mathbf{N}(t + 5) = \mathbf{L}[t, t + 5]\mathbf{N}(t) \quad (1)$$

If we can assume that the projection (Leslie) matrix \mathbf{L} does not change over future projections, then:

$$\mathbf{N}(t + 5 \times n) = \mathbf{L}^n \mathbf{N}(t) \quad (2)$$

Using matrices - exercise

Exercise

Load the data that we saved yesterday, `EDSD.lect2.Rdata`. Project the female population by one period using the matrix notation in Equation (1). Check your results with the previously derived projection.

Hint: insert the vectors sFx and bFx in the matrix L and compute

$$N(t+5) = L[t, t+5]N(t)$$

Using matrices - one possible solution

Example

```
rm(list = ls())
library(tidyverse); library(viridis)
load("EDSD.lect2.Rdata")
## extract sx, bx and NFx
m <- nrow(dta.swe)
sFx <- dta.swe$sFx; bFx <- dta.swe$bFx; NFx <- dta.swe$NFx
## change NA to 0 in bFx and remove NA from sFx
bFx[is.na(bFx)] <- 0
sFx <- sFx[!is.na(sFx)]
## create empty Leslie matrix
L <- matrix(0,m,m)
## assign bFx and sFx
L[1,] <- bFx
diag(L[-1,]) <- sFx
L[m,m] <- sFx[m-1]
## matrix projection and comparison
NFx5.matrix <- c(L%*%NFx)
NFx5.manual <- dta.swe$NFx5
all.equal(NFx5.matrix,NFx5.manual)
```

[1] TRUE

Longer projections - exercise

Exercise

Now that we are sure that the matrix formulation works, let us project $n = 20$ periods ahead. Plot your results in a pyramid showing the starting population, the first and the last projected periods

Reminder:

$$\mathbf{N}(t + 5 \times n) = \mathbf{L}^n \mathbf{N}(t)$$

Hint: write a function containing a population matrix \mathbf{N} with $n + 1$ columns and loop through its columns with matrix multiplication. You can do this recursively (without the need of matrix exponentiation) by updating your baseline population and using the simpler formula

$$\mathbf{N}(t + 5) = \mathbf{L}[t, t + 5] \mathbf{N}(t)$$

Longer projections - one possible solution I

Example

```
## function to project several periods
pop.proj <- function(x, AgeGroup, Nx, sx, bx, n){
  ## number of age groups
  m <- length(x)
  ## create Leslie matrix
  L <- matrix(0, m, m)
  L[1,] <- bx
  diag(L[-1,]) <- sx
  L[m, m] <- sx[m-1]
  ## create population matrix
  N <- matrix(0, m, n+1)
  N[,1] <- Nx
  for (i in 1:n){
    N[,i+1] <- L%*%N[,i]
  }
  out <- cbind(data.frame(x=x, AgeGroup=AgeGroup), N)
  return(out)
}

## actual projection
n <- 20
my.proj <- pop.proj(x=dta.swe$Age, AgeGroup=dta.swe$AgeGroup,
Nx=NFx, sx=sFx, bx=bFx, n=n)
all.equal(my.proj$'1', NFx)
all.equal(my.proj$'2', NFx5.manual)
```

```
[1] TRUE
```

```
[1] TRUE
```

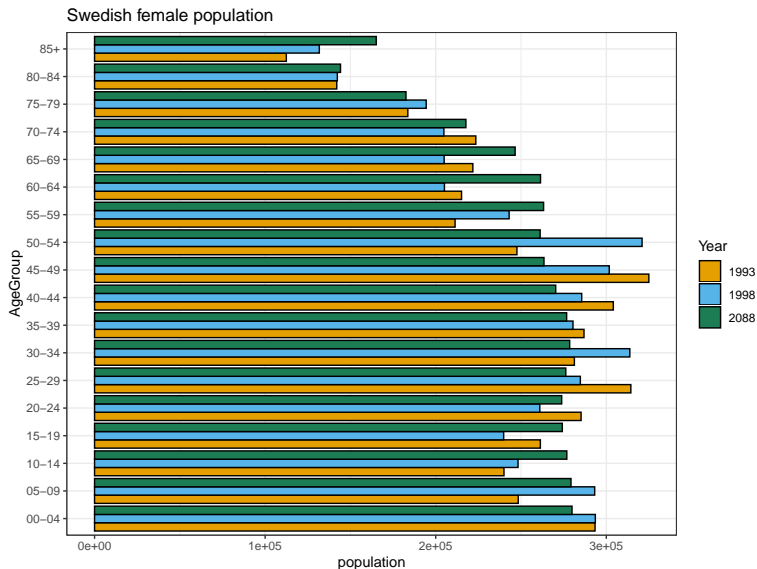
Longer projections - one possible solution II

Example

```
## long data
dta.swe.l <- my.proj %>%
  pivot_longer(-c(x, AgeGroup), names_to = "period", values_to = "population") %>%
  mutate(period=as.numeric(period),
         Year=1993 + (period-1)*5,
         YearF=as.factor(Year))

## plotting
ggplot(dta.swe.l, aes(x=AgeGroup, y=population, fill=YearF)) +
  geom_bar(data = subset(dta.swe.l, period %in% c(1,2,20)),
          stat = "identity", position = "dodge", color = "black") +
  coord_flip() +
  theme_bw() +
  ggtitle("Swedish female population") +
  scale_fill_manual(name = 'Year', values=c("#E69F00", "#56B4E9", "#1C7C54"))
```

Longer projections - one possible solution III



Intermezzo: animated visualizations & shiny-app

In some instances, it can be useful to introduce dynamic plots, animations and flexible outputs in your presentations or documents:

- ▶ can grab the audience's attention
- ▶ show time-series evolutions
- ▶ to describe your model [see my own example](#)
- ▶ sensitivity analysis of your results

⇒ We will now look at how to make these

Images: animation with `\animategraphics`

BEAMER supports animation images: `\animategraphics`

Images: animation with `\animategraphics`

```
\animategraphics[<options>]{frame rate}{  
file}{first}{last}
```

- ▶ Preamble: `\usepackage{animate}`
- ▶ need to create a multipage PDF file (next frame)
- ▶ Options:
 - ▶ `autoplay`: start animation after the page has opened
 - ▶ `loop`: animation restarts immediately after the end
 - ▶ `palindrome`: animation plays forwards and backwards
 - ▶ `step`: step through animation by mouse-click
 - ▶ `controls`: shows control buttons below the animation widget
- ▶ make sure to open file with Adobe Acrobat Reader (or the animation will fail)!!

Animated pyramids: an example

Example

```
plots <- list()
my.cols <- cividis(n+1)
my.years <- unique(dta.swe.l$Year)
for (i in 1:(n+1)){
  gg <- ggplot(dta.swe.l,aes(x=AgeGroup,y=population,fill=YearF)) +
    geom_bar(data = subset(dta.swe.l, period == i),
             stat = "identity",color = "black") +
    coord_flip() +
    theme_bw() +
    theme(legend.position = "none") +
    ggtitle(paste("Swedish female population, year",my.years[i])) +
    scale_fill_manual(values=my.cols[i])
  plots[[i]] <- gg
}
## saving plots in a single file
pdf("myAnimFig.pdf")
invisible(lapply(plots, print))
dev.off()
```

In \LaTeX :

```
\animategraphics[autoplay,scale=0.42]{3}{myAnimFig.pdf}{}{}
```

Animated pyramids: output

Images: animation with .GIFs

- ▶ if you use Powerpoint* for your presentations, you can include animations by creating .GIFs in R using:
 - ▶ the `gganimate` package (for `ggplots`)
 - ▶ a series of .png or .pdf files and converting them to a .gif
- ▶ .GIFs are also useful for other media outlets, for example Twitter, ...

*please consider switching to \LaTeX for your presentations. You can find here some materials on how to get started with BEAMER:

<https://github.com/ubasellini/LaTeXpresentations>

Animations with .GIFs: gganimate

Example

```
## gganimate
library(gganimate)
library(gifski)
gg <- ggplot(dta.swe.l,aes(x=AgeGroup,y=population,fill=YearF)) +
  geom_bar(stat = "identity",position = "dodge",color = "black") +
  coord_flip() +
  theme_bw() +
  theme(legend.position = "none") +
  scale_fill_viridis_d(option="cividis")

gg + transition_states(YearF) +
  ggtitle('Swedish female population, year {closest_state}')
```



```
anim_save("F2.gif")
```

Animations with .GIFs: converting .pdf or .png

Example

```
## general example using magick package
library(magick)

## create a temporary directory to which the images will be written
dir_out <- file.path(tempdir(), "temp_dir")
dir.create(dir_out, recursive = TRUE)

## loop through years and write plot to file
for (i in 1:(n+1)){
  gg <- ggplot(dta.swe.l, aes(x=AgeGroup, y=population, fill=YearF)) +
    geom_bar(data = subset(dta.swe.l, period == i), stat = "identity", color = "black") +
    coord_flip() + theme_bw() + theme(legend.position = "none") +
    ggtitle(paste("Swedish female population, year", my.years[i])) +
    scale_fill_manual(values=my.cols[i])
  fp <- file.path(dir_out, paste0(i, ".png"))
  ggsave(plot = gg,
    filename = fp,
    device = "png")
}

## list file names and read in
imgs <- list.files(dir_out, full.names = TRUE)
img_list <- lapply(imgs, image_read)
## join the images together
img_joined <- image_join(img_list)
## animate at 2 frames per second
img_animated <- image_animate(img_joined, fps = 2)
## view animated image
img_animated
## save to your pc
image_write(image = img_animated, path = "example.gif")
```

Animations with .GIFs: converting .pdf or .png

Example

```
## MAC-specific example
## convert the previously created .pdf into a .gif using ImageMagick.
## this is done internally by the terminal
## the "-delay" sets the time between the frames, i.e. the speed of the animation.
system("convert -delay 40 myAnimFig.pdf example.gif")
```

Shiny app

- ▶ shiny is an R package that makes it easy to build interactive web apps straight from R
- ▶ keep them on webpages or embed them in R Markdown documents
- ▶ a user-friendly interface to interact with your R analysis and show your results ([here](#) my own example)
- ▶ it is composed by
 - ▶ an UI (user interface), where you can create the inputs for your shiny and decide the outputs to display
 - ▶ a server, where you assemble the outputs from your given inputs
 - ▶ the shinyApp, putting the two together
- ▶ to learn more, visit <https://shiny.rstudio.com> to get started, plenty of videos and written tutorials

Shiny app - a simple example

Example

```
library("shiny")
library("ggplot2")
## general parameters
n1 <- max(dta.swe.l$period)
my.cols <- cividis(n1)
my.years <- unique(dta.swe.l$Year)
## build your user interface
ui <- fluidPage(
  ## title of your shiny
  titlePanel('My first shiny app'),
  ## display a slider that returns input$year to pass to the server function
  sliderInput(inputId = "year", label = "Year", step = 5,
    value = min(my.years), min = min(my.years), max = max(my.years)),
  ## display a plot returned from the server
  plotOutput("plot_pyr1")
)
```

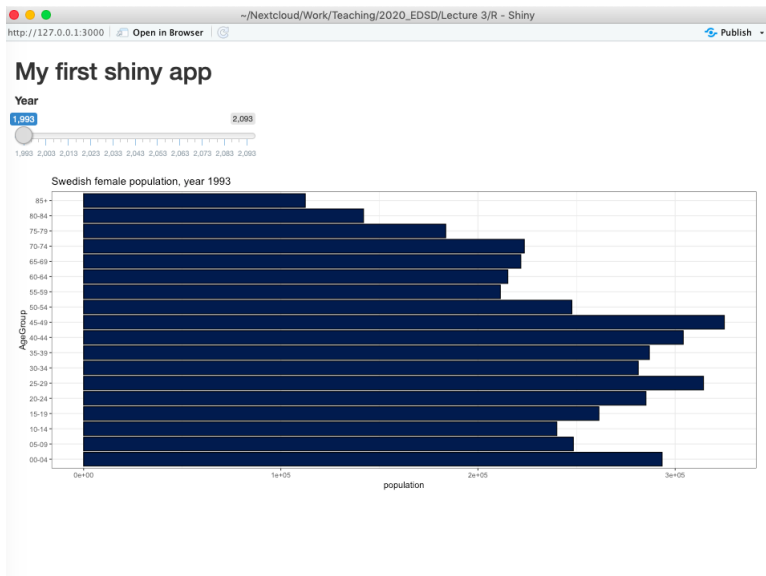
Shiny app - a simple example

Example

```
## build your server
server <- function(input, output){
  ## create an output that renders a plot
  output$plot_pyr1 <- renderPlot({
    ## any ggplot or plot,
    ## here subsetting the year of the given input$year
    ggplot(dta.swe.l,aes(x=AgeGroup,y=population,fill=YearF)) +
      geom_bar(data = subset(dta.swe.l, Year == input$year),
        stat = "identity",color = "black") +
      coord_flip() +
      theme_bw() +
      theme(legend.position = "none") +
      ggtitle(paste("Swedish female population, year",input$year)) +
      scale_fill_manual(values=my.cols[which(input$year==my.years)])
  })
}

## run the shiny app, which puts together the ui and server
shinyApp(ui = ui, server = server)
```

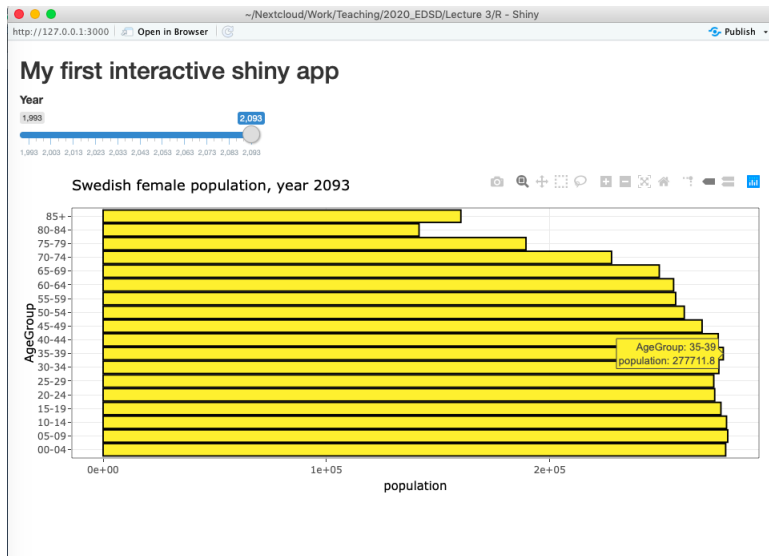
Shiny app - a simple example



Interactive Shiny apps

- ▶ interactivity can be easily implemented into shiny using plotly
- ▶ only need small changes to previous code:
 - ▶ load plotly package
 - ▶ from plotOutput to plotlyOutput
 - ▶ from renderPlot to renderPlotly
 - ▶ can select what to interactively show by using `ggplotly(fig, tooltip = c('AgeGroup', 'population'))`

Interactive Shiny apps - an example



Including the male population

- ▶ For males, we have seen that we can use (almost) the same formulas as for females
- ▶ we can express the equations for both sexes using matrix notation and a block diagonal Leslie matrix:

$$\begin{bmatrix} N^F(t+5) \\ N^M(t+5) \end{bmatrix} = \begin{bmatrix} L^F & 0 \\ B^M & L^M \end{bmatrix} \begin{bmatrix} N^F(t) \\ N^M(t) \end{bmatrix}$$

where

$$B^M = \begin{bmatrix} 0 & 0 & b_{10} & \dots & b_{45} & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix}, \quad L^M = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ s_0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & s_5 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & s_{10} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & s_{75} & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & s_{80} & s_{80} \end{bmatrix}$$

Two-sex projections - exercise

Exercise

Starting again from the data of the previous lecture, write a function to project both sexes contemporaneously. Check if your projection for the first period are the same as those of the previous class. Plot a pyramid of the projections after 20 periods.

Reminder:

$$\begin{bmatrix} N^F(t+5) \\ N^M(t+5) \end{bmatrix} = \begin{bmatrix} L^F & 0 \\ B^M & L^M \end{bmatrix} \begin{bmatrix} N^F(t) \\ N^M(t) \end{bmatrix}$$

Two-sex projections - one possible solution I

Example

```
sMx <- dta.swe$sMx; bMx <- dta.swe$bMx; NMx <- dta.swe$NMx
## similar adjustments to bMx and sMx
bMx[is.na(bMx)] <- 0
sMx <- sMx[!is.na(sMx)]
## female Leslie matrix (derived before)
LF <- L
## male Leslie matrix
BM <- LM <- matrix(0,m,m)
BM[1,] <- bMx
diag(LM[-1,]) <- sMx
LM[m,m] <- sMx[m-1]
## putting male and females together
ZEROS <- diag(0,m)
Lup <- cbind(LF,ZEROS)
Ldown <- cbind(BM,LM)
L <- rbind(Lup,Ldown)
## matrix projection and comparison
Nx <- c(NFx,NMx)
Nx5.matrix <- c(L%*%Nx)
NFx5.manual <- dta.swe$NFx5
NMx5.manual <- dta.swe$NMx5
all.equal(NFx5.manual,Nx5.matrix[1:m]) ## [1] TRUE
all.equal(NMx5.manual,Nx5.matrix[1:m+m]) ## [1] TRUE
```


Two-sex projections - one possible solution II

Example

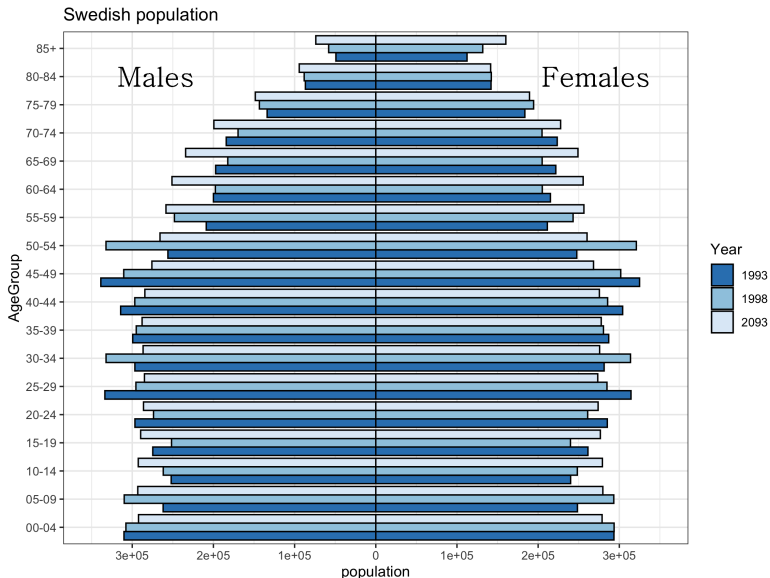
```
## function to project several periods
pop.proj.v2 <- function(x, AgeGroup, NFx, sFx, bFx, NMx, sMx, bMx, n){
  ## number of age groups
  m <- length(x); m2 <- m*2
  ## female Leslie matrix
  LF <- matrix(0, m, m)
  LF[1,] <- bFx
  diag(LF[-1,]) <- sFx
  LF[m, m] <- sFx[m-1]
  ## male Leslie matrix
  BM <- LM <- matrix(0, m, m)
  BM[1,] <- bMx
  diag(LM[-1,]) <- sMx
  LM[m, m] <- sMx[m-1]
  ## putting them together
  ZEROS <- diag(0, m)
  Lup <- cbind(LF, ZEROS)
  Ldown <- cbind(BM, LM)
  L <- rbind(Lup, Ldown)
  ## create population matrix
  N <- matrix(0, m2, n+1)
  N[, 1] <- c(NFx, NMx)
  for (i in 1:n){
    N[, i+1] <- L %*% N[, i]
  }
  out <- cbind(data.frame(x=rep(x, 2), AgeGroup=rep(AgeGroup, 2),
    sex=rep(c("Females", "Males"), each=m)), N)
  return(out)
}
```

Two-sex projections - one possible solution III

Example

```
my.proj <- pop.proj.v2(x=dta.swe$Age, AgeGroup=dta.swe$AgeGroup, NFx=NFx, sFx=sFx, bFx=bFx,
  NMx=NMx, sMx=sMx, bMx=bMx, n=20)
## long data
dta.swe.l <- my.proj %>%
  pivot_longer(-c(x, AgeGroup, sex), names_to = "period", values_to = "population") %>%
  mutate(period=as.numeric(period),
    Year=1993 + (period-1)*5,
    YearF=as.factor(Year))
## plotting
ggplot(dta.swe.l, aes(x=AgeGroup, y=population, fill=YearF)) +
  geom_bar(data = subset(dta.swe.l, period %in% c(1,2,21) & sex == "Males"),
    stat = "identity", position = "dodge", color = "black", mapping = aes(y = -population)) +
  geom_bar(data = subset(dta.swe.l, period %in% c(1,2,21) & sex == "Females"),
    stat = "identity", position = "dodge", color = "black") +
  coord_flip() +
  theme_bw() +
  ggtitle("Swedish population") +
  scale_y_continuous(limits=c(-3.5e5, 3.5e5),
    breaks = seq(-4e5, 4e5, 1e5),
    labels = abs(seq(-4e5, 4e5, 1e5))) +
  scale_fill_brewer(name="Year", palette = 'Blues', direction = -1) +
  geom_text(data = subset(dta.swe.l, period %in% c(1)),
    aes(y = max(population)/1.25, x = 17, label='Females'), size=7) +
  geom_text(data = subset(dta.swe.l, period %in% c(1)),
    aes(y = -max(population)/1.25, x = 17, label='Males'), size=7)
```

Two-sex projections - one possible solution IV



Two-sex projections - one possible solution V

Some final remarks

- ▶ matrix algebra can significantly speed up your population projections

Some final remarks

- ▶ matrix algebra can significantly speed up your population projections
- ▶ can incorporate female and male populations within the same setting

Some final remarks

- ▶ matrix algebra can significantly speed up your population projections
- ▶ can incorporate female and male populations within the same setting
- ▶ great care is needed to construct the Leslie matrix

Some final remarks

- ▶ matrix algebra can significantly speed up your population projections
- ▶ can incorporate female and male populations within the same setting
- ▶ great care is needed to construct the Leslie matrix
- ▶ we still have not considered migration, nor time-specific assumptions on future demographic components

Some final remarks

- ▶ matrix algebra can significantly speed up your population projections
- ▶ can incorporate female and male populations within the same setting
- ▶ great care is needed to construct the Leslie matrix
- ▶ we still have not considered migration, nor time-specific assumptions on future demographic components

⇒ we'll look at these in tomorrow's lecture

Assignment

Exercise #5

Take again the population that you used for Exercise #2. Project the population for 5 years ahead, but this time use matrix formulas. Compare your results with the projections that you obtained in Exercise #2. Are they the same?

Exercise #6

Project your chosen population by sex for $n = 20$ periods ahead and show your results.

Bonus: present your results with the aid of a shiny app or an animation.

References

- ▶ Basellini, U. and Camarda, C.G. (2019). Modelling and forecasting adult age-at-death distributions. *Population Studies*, **73**(1), 119—138.
- ▶ Bernardelli, H. (1941). Population Waves. *Journal of the Bourma Research Society*, **31**(1), 1–18
- ▶ Lewis, E.G. (1942). On the generation and growth of a population. *Sankhya*, **6**, 93–96
- ▶ Leslie, P.H. (1945). On the use of matrices in certain population dynamics. *Biometrika*, **33**, 183–212
- ▶ Preston, S. H., Heuveline, P., and Guillot, M. (2001). *Demography. Measuring and Modeling Population Processes*. Blackwell.
- ▶ Smith, D.P. and Keyfitz, N. (eds.) (1977). *Mathematical Demography: Selected Papers*. Berlin: Springer Verlag

Animation example: the STAD model (Basellini & Camarda 2019)

[go back](#)