Summative Assessment PDP

```
echo: false
```

Install packages

```
# Install necessary packages
# install.packages("tidyverse")
# install.packages("ggplot2")
# install.packages("HMDHFDplus")
# install.packages('MortalityLaws')
# install.packages('DemoDecomp')
```

Load packages

```
# load necessary packages
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ ggplot2   3.5.1     ✔ tibble    3.2.1
✔ lubridate 1.9.3     ✔ tidyr     1.3.1
✔ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to
become errors
```

```
library(HMDHFDplus)
library(ggplot2)
library(MortalityLaws)
library(DemoDecomp)
library(readxl)
library(stringr)

# don't display in scientific notation
options(scipen = 999)
```

Question 1 - Mortality

Obtain single year LTs for Latvia and Italy from HMD

```
# Obtain 1 year LTs for Latvia and Italy
```

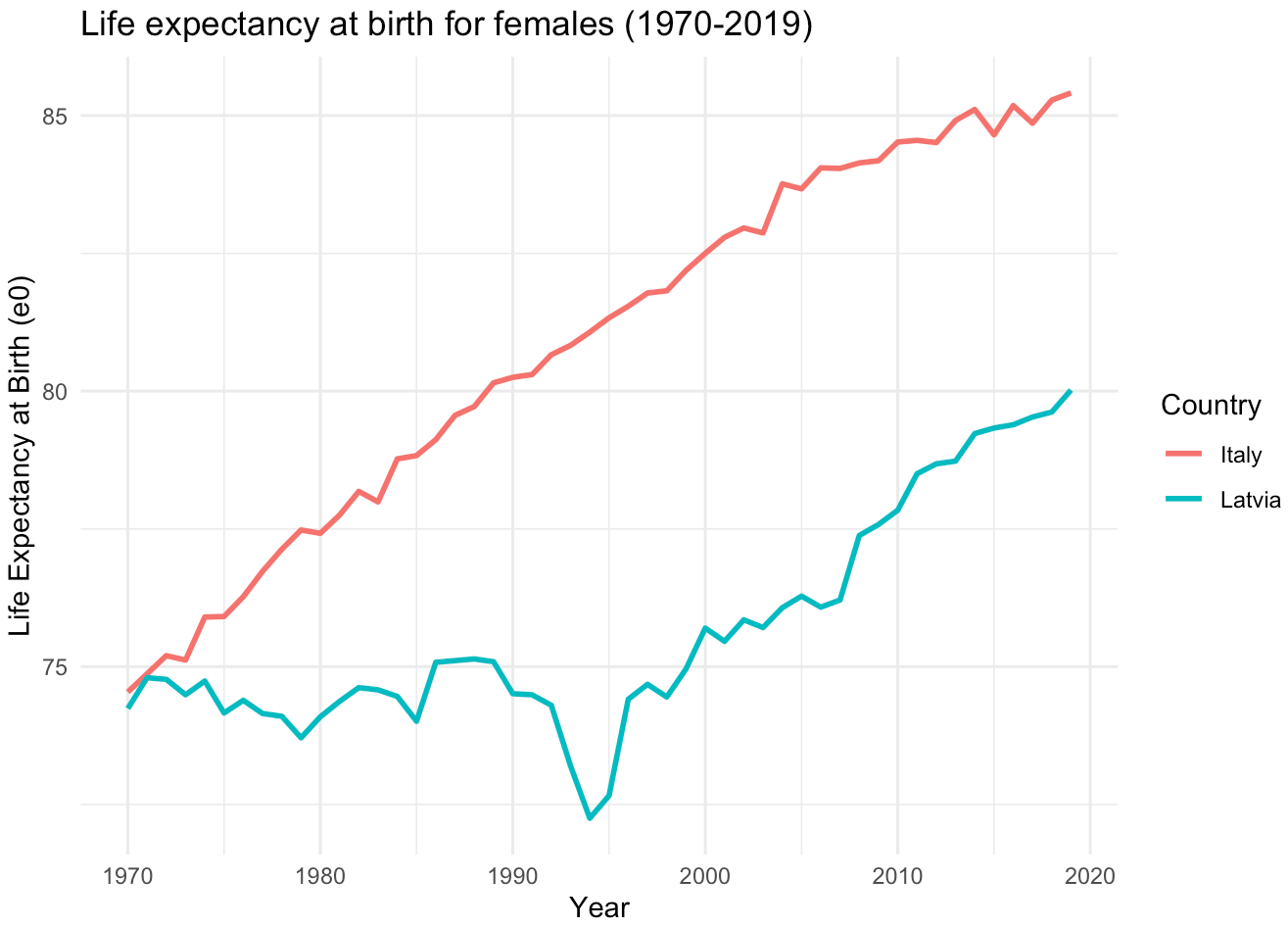
```
# code in comments for anonymity purposes

# LT_females_Latvia <- readHMDweb(CNTRY = "LVA", item = "fltper_1x1", username = "***", p

# LT_females_Italy <- readHMDweb(CNTRY = "ITA", item = "fltper_1x1", username = "***", pa
```

Create data frames that give e0 for both Latvia and Italy from 1970 to 2019. We will also merge these data frames so that we can plot e0 for both countries over time.

```
# Filter data for e0 from 1970 to 2019 - Latvia
e0_females_Latvia_DF <- LT_females_Latvia %>%
  filter(Age == 0, Year >= 1970, Year <= 2019) %>%
  select(Year, ex) %>%
  mutate(Country = "Latvia")

# change name of ex column to e0
colnames(e0_females_Latvia_DF) <- c("Year","e0","Country")

# Filter data for e0 from 1970 to 2019 - Italy
e0_females_Italy_DF <- LT_females_Italy %>%
  filter(Age == 0, Year >= 1970, Year <= 2019) %>%
  select(Year, ex) %>%
  mutate(Country = "Italy")

# change name of ex column to e0
colnames(e0_females_Italy_DF) <- c("Year","e0","Country")

# Combine both datasets
e0_Latvia_and_Italy_DF <- bind_rows(e0_females_Latvia_DF, e0_females_Italy_DF)
```

1. Plot e0 for both countries from 1970 - 2019 and describe the results

```
# Plot life expectancy at birth over time
ggplot(e0_Latvia_and_Italy_DF, aes(x = Year, y = e0, color = Country)) +
  geom_line(linewidth = 1) +
  labs(title = "Life expectancy at birth for females (1970-2019)",
       x = "Year",
       y = "Life Expectancy at Birth (e0)",
       color = "Country") +
  theme_minimal()
```



Describe results:

Italy e0 appears to be improving at a constant rate over time. On the other hand, Latvia e0 stagnates somewhat from 1970 - 1990 and then e0 sharply drops in 1990. Around 1995, e0 in Latvia improves back to pre-1990 levels and then improves at a constant rate. It is also worthwhile to note that both Italy and Latvia have similar e0 in 1970, when the first year is specified. However, after 50 years, Latvia e0 in 2019, which is ~80, falls behind Italy by about 5 years, which is ~85.

2. What was the difference in life expectancy at birth between these two countries in 1970, and in 2019?

```
# Latvia
e0_Latvia_1970 <- e0_females_Latvia_DF %>%
  filter(Year == 1970) %>%
  pull(e0)

e0_Latvia_2019 <- e0_females_Latvia_DF %>%
  filter(Year == 2019) %>%
  pull(e0)

e0_Latvia_2019 - e0_Latvia_1970 # 5.78 year difference in e0 from 1970 - 2019
```

```
[1] 5.78
```

```
# Italy
e0_Italy_1970 <- e0_females_Italy_DF %>%
  filter(Year == 1970) %>%
  pull(e0)

e0_Italy_2019 <- e0_females_Italy_DF %>%
  filter(Year == 2019) %>%
  pull(e0)

e0_Italy_2019 - e0_Italy_1970 # 10.87 year difference in e0 from 1970 - 2019
```

```
[1] 10.87
```

3.

1). subset the data for both countries in 2019 and for ages from 10 onward

```
# Subset data for year 2019 and ages 10 and on - Latvia
LT_females_Latvia_2019 <- LT_females_Latvia %>%
  filter(Age >= 10, Year == 2019)

# Subset data for year 2019 and ages 10 and on - Italy
LT_females_Italy_2019 <- LT_females_Italy %>%
  filter(Age >= 10, Year == 2019)
```

2). randomly assign NA values to 30 ages

```
# Latvia - Randomly assign NA values to 30 ages
set.seed(123)  # For reproducibility
random_rows <- sample(nrow(LT_females_Latvia_2019), 30)  # Select 30 random row indices

# Assign NA to all columns in the selected rows
LT_females_Latvia_2019[random_rows, ] <- NA

# Italy - Randomly assign NA values to 30 ages
set.seed(123)  # For reproducibility
random_rows <- sample(nrow(LT_females_Italy_2019), 30)  # Select 30 random row indices

# Assign NA to all columns in the selected rows
LT_females_Italy_2019[random_rows, ] <- NA
```

3). fit a Makeham model for both countries.

Latvia:

```
# first, remove rows with NA values as we did in the first week's HW.
# MortalityLaw() function will not operate properly otherwise.
LT_females_Latvia_2019_clean <- na.omit(LT_females_Latvia_2019)

# obtain mxs from LT_females_Latvia_2019_clean
mx_values_Latvia <- LT_females_Latvia_2019_clean$mx

# obtain ages from LT_females_Latvia_2019_clean
ages_Latvia <- LT_females_Latvia_2019_clean$Age

# create an object that transforms the mx values from LT_females_Latvia_2019_clean into l
log_mx_values_Latvia <- log(mx_values_Latvia)

# append this object to the LT_females_Latvia_2019_clean dataset
LT_females_Latvia_2019_clean$log_mx <- log_mx_values_Latvia

# See available Mortality laws for reference
# availableLaws(law = NULL)
# We can see that the makeham mortality model is typically used to
# estimate adult mortality based on the legend (3 makeham)

# Fit the makeham model, using poisson regression - Latvia
makeham_fitted_Latvia <- MortalityLaw(x = LT_females_Latvia_2019_clean$Age, mx = mx_value
```

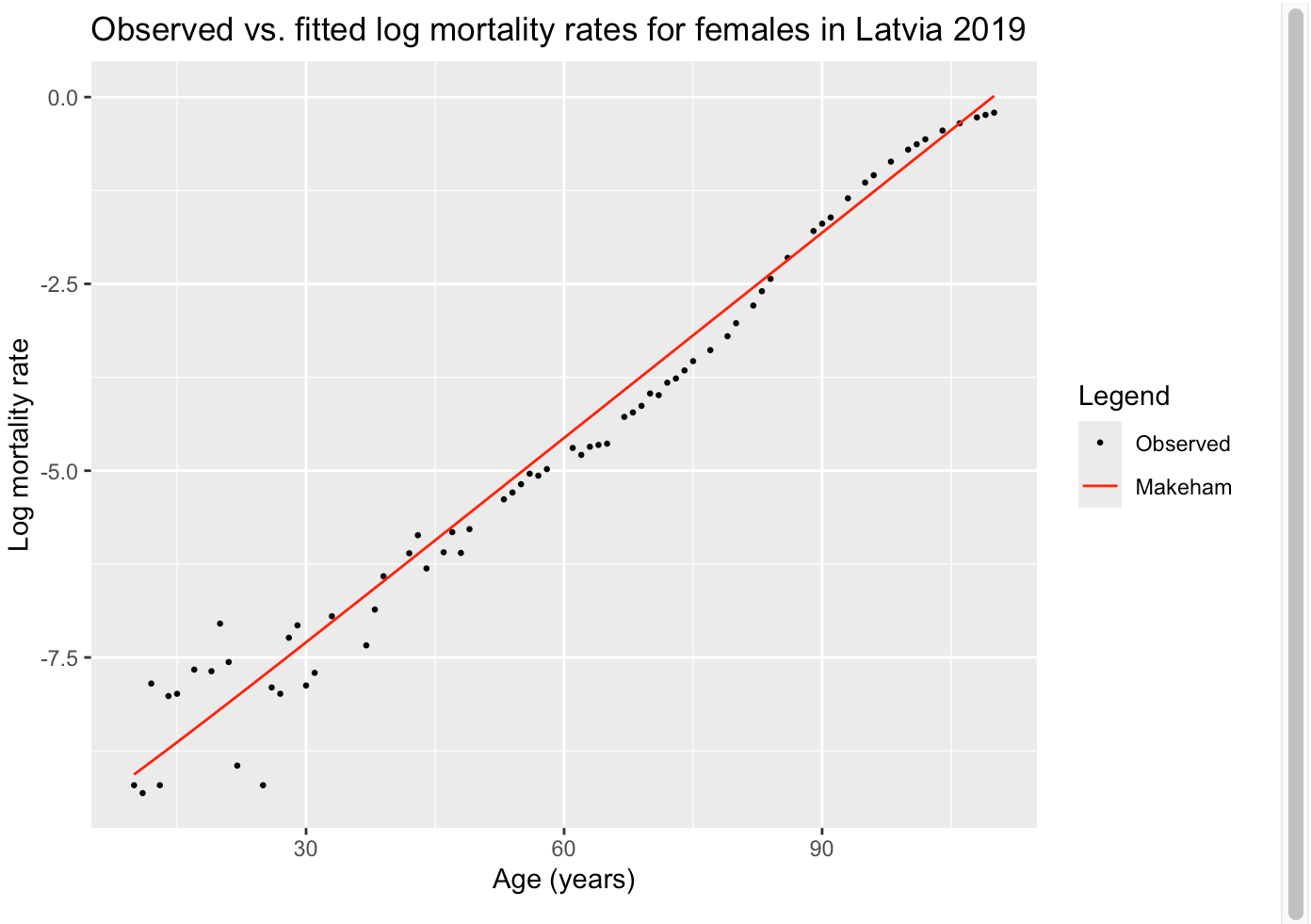
Italy:

```
# first, remove rows with NA values
LT_females_Italy_2019_clean <- na.omit(LT_females_Italy_2019)

# obtain mxs from LT_females_Italy_2019_clean
mx_values_Italy <- LT_females_Italy_2019_clean$mx

# obtain ages from LT_females_Italy_2019_clean
ages_Italy <- LT_females_Italy_2019_clean$Age

# create an object that transforms the mx values from LT_females_Italy_2019_clean into lo
log_mx_values_Italy <- log(mx_values_Italy)

# append this object to the LT_females_Italy_2019_clean dataset
LT_females_Italy_2019_clean$log_mx <- log_mx_values_Italy

# Fit the makeham model, using poisson regression - Italy
makeham_fitted_Italy <- MortalityLaw(x = LT_females_Italy_2019_clean$Age, mx = mx_values_
```

4. Plot observed and fitted values, and describe your findings. How good is the fit?

Latvia:

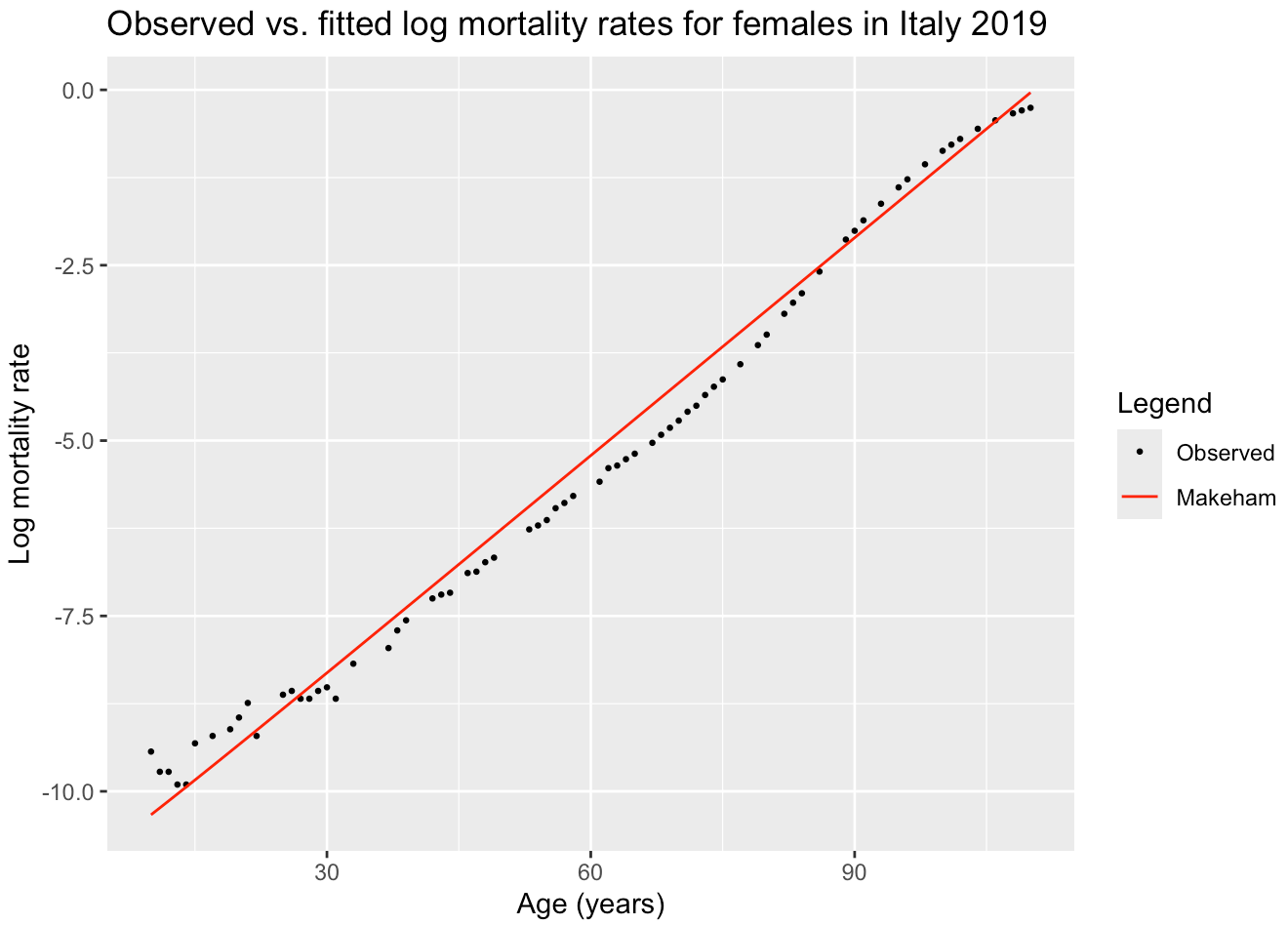
```
# create DF that includes ages, makeham mxs, and log makeham mxs
# will use this DF to plot results
makeham_DF_Latvia <- data.frame(Age = ages_Latvia,
                                makeham_fitted_Latvia = predict(makeham_fitted_Latvia, x
  mutate(log_makeham_fitted_mx_Latvia = log(makeham_fitted_Latvia))


# Create a plot of the observed log mx rates and the makeham fitted log mx rates
makeham_Latvia <- ggplot() +
  geom_point(data = LT_females_Latvia_2019_clean, aes(x=Age, y=log_mx, color = "Observed"
  geom_line(data = makeham_DF_Latvia, aes(x = Age, y = log_makeham_fitted_mx_Latvia, colo
  scale_color_manual(name = "Legend",
                     values = c("Observed" = "black",
                                "Makeham" = "red"),
                     limits = c("Observed", "Makeham")) +
  labs(title = "Observed vs. fitted log mortality rates for females in Latvia 2019",
       y = "Log mortality rate",
       x = 'Age (years)') +
  theme(legend.position.inside = c(0.85, 0.25))

makeham_Latvia
```



For Latvia, this model fits relatively well for ages 30 and on. However, from ages 60-90, it seems to overestimate the mortality rates, which may be a cause for concern.

Italy:

```
# create DF that includes ages, makeham mxs, and log makeham mxs
makeham_DF_Italy <- data.frame(Age = ages_Italy,
                               makeham_fitted_Italy = predict(makeham_fitted_Italy, x = a
  mutate(log_makeham_fitted_mx_Italy = log(makeham_fitted_Italy))

# Create a plot of the observed log mx rates and the makeham fitted log mx rates - Italy
makeham_Italy <- ggplot() +
  geom_point(data = LT_females_Italy_2019_clean, aes(x=Age, y=log_mx, color = "Observed")
  geom_line(data = makeham_DF_Italy, aes(x = Age, y = log_makeham_fitted_mx_Italy, color
  scale_color_manual(name = "Legend",
                     values = c("Observed" = "black",
                                "Makeham" = "red"),
                     limits = c("Observed", "Makeham")) +
  labs(title = "Observed vs. fitted log mortality rates for females in Italy 2019",
       y = "Log mortality rate",
       x = 'Age (years)') +
  theme(legend.position.inside = c(0.85, 0.25))

makeham_Italy
```



For Italy, this model fits relatively well, and actually performs better for ages < 30. However, similar to what we saw with Latvia, the model overestimates mortality from ages 45 - 90, which may be a cause for concern.

5. Write down the Makeham model, what does each parameter mean?

$\mu(x) = A * \exp(Bx) + C$ where,

- $\mu(x)$ is the death rate or force of mortality at age x
- A is a parameter that determines the initial mortality rate
- B is a parameter that determines the rate at which mortality increases with age
- C represents mortality due to factors independent of age (makeham term)
- Note: the Makeham model extends on the Gompertz model, $\mu(x) = A * \exp(Bx)$, by incorporating C, the age-independent mortality component.

6. Say you have data for a country from 1970 to 2019, and you would like to forecast life expectancy to 2040. How would you do it?

To answer this question, I would use the Lee-Carter model (Lecture 9: Forecasting, Slide 4) to forecast life expectancy as we did in the forecasting practical in class on March 14th. The model is specified by

the function: ln(mx,t) = αx+βxκt+ϵx,t where

- αx is the general shape of the log-mortality at age x
- βx is the rate of mortality improvement at age x
- κt is the general level of mortality at time t
- ϵx,t is the error with mean 0 and variance σ^2,ϵ, which reflects the age-specific influences that aren't captured in the model

We first can use the data we have for the country of interest from 1970 to 2019 and model mortality by inputting the data for the relevant parameters. The main parameter of interest in forecasting mortality is κt, as this is the only parameter that contains time in its dimension (error does as well, but we measure this independently). We can identify how κt changes from 1970 to 2019 to see if mortality conditions are improving or worsening over time (Slide 11).

We can then forecast mortality rates for each age group from 2020 to 2040 (20 years) using the κt values alongside the αx and βx values.

Lastly, through constructing a life table, we can then convert these forecasted mortality rates into forecasted life expectancy values.

Question 2 - Lifespan inequality and decomposition

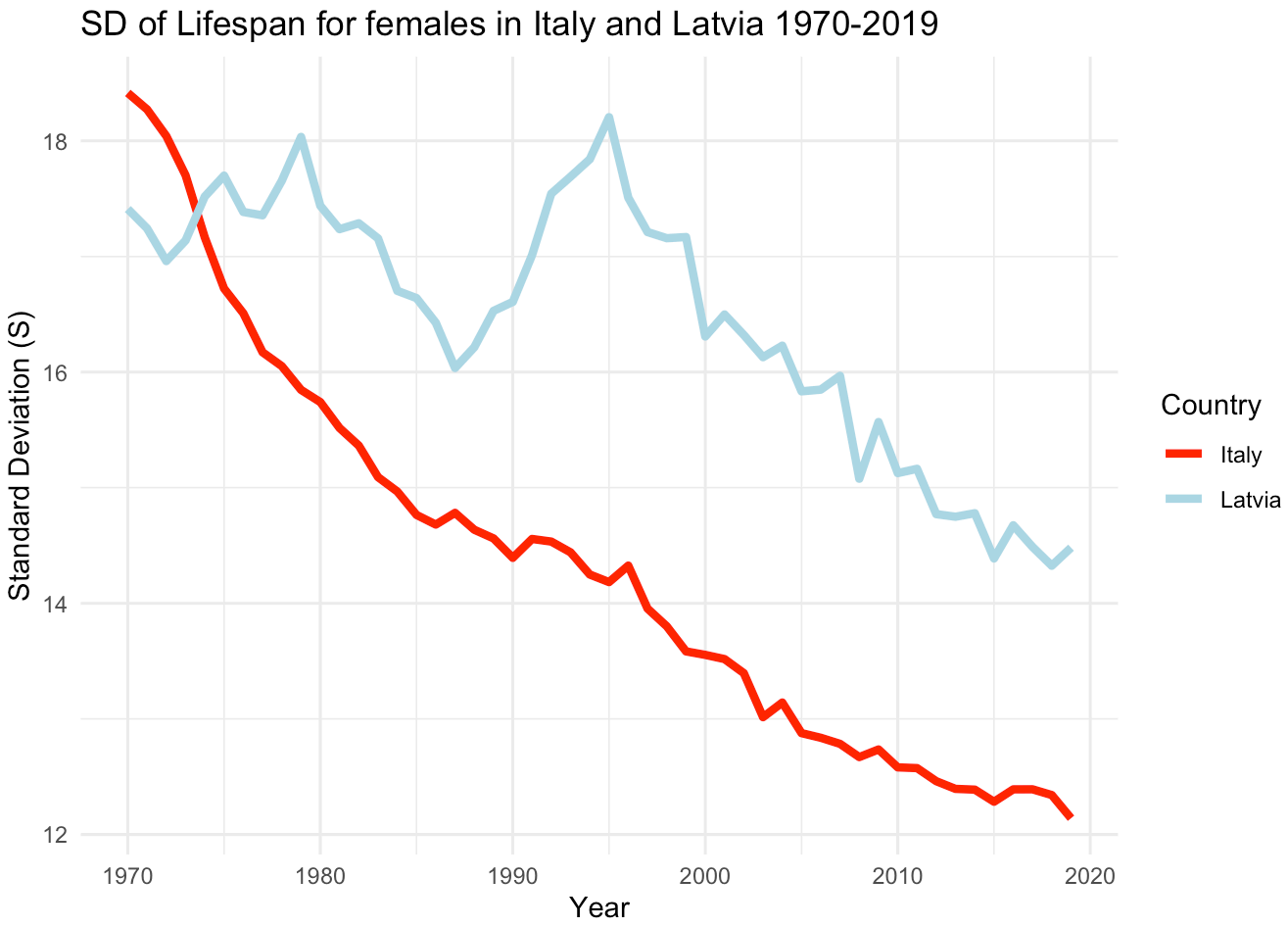
1. Build a function to calculate the standard deviation from a set of mortality rates.

```
sd_function <- function(age, dx, ex, ax, l0) {
  alphax <- age + ax
  eta <- ex[1] +age[1]
  inner <- dx * (alphax - eta)^2
  V <- sum(inner) / l0
  S <- sqrt(V)
  return(S)
}
```

2. Calculate the standard deviation for both countries from 1970 to 2019. [describe the results, maybe a plot is useful]

```
# Latvia - create a DF that stores e0 and S
results_Latvia <- LT_females_Latvia %>%
  filter(Year >= 1970 & Year <= 2019) %>%
  group_by(Year) %>%
  summarize(
    ex = first(ex),
    ax = list(ax),
```

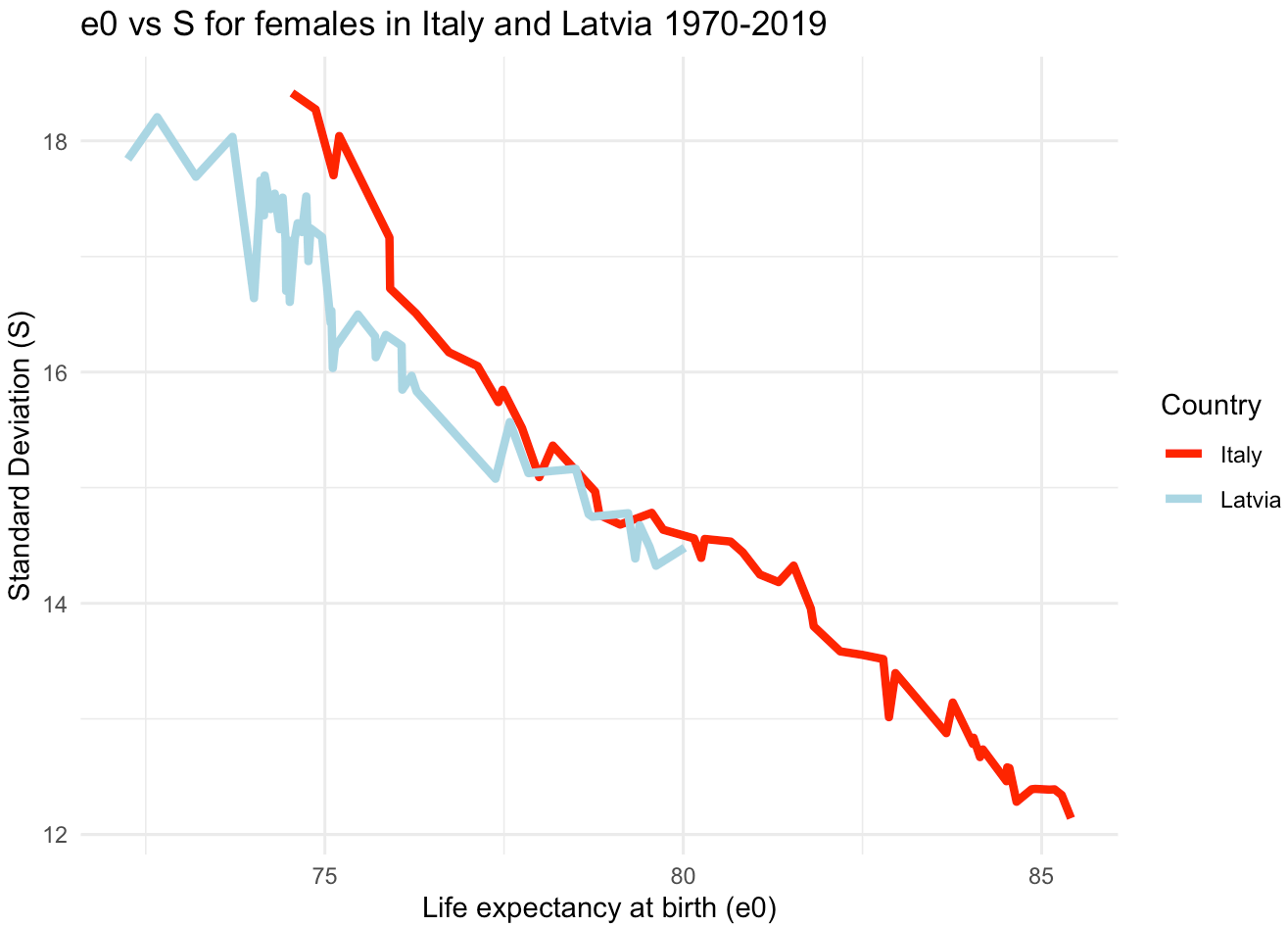
```
    dx = list(dx),
    lx = list(lx),
    S = sd_function(
      age = Age,
      dx = dx[[1]],
      ex = ex,
      ax = ax[[1]],
      l0 = 100000
    )
  ) %>%
  ungroup() %>%
  select(Year, e0 = ex, S)

# Italy - create a DF that stores e0 and S
results_Italy <- LT_females_Italy %>%
  filter(Year >= 1970 & Year <= 2019) %>%
  group_by(Year) %>%
  summarize(
    ex = first(ex),
    ax = list(ax),
    dx = list(dx),
    lx = list(lx),
    S = sd_function(
      age = Age,
      dx = dx[[1]],
      ex = ex,
      ax = ax[[1]],
      l0 = 100000
    )
  ) %>%
  ungroup() %>%
  select(Year, e0 = ex, S)

# Plot both countries together

# add a country label
results_Italy$Country <- "Italy"
results_Latvia$Country <- "Latvia"

# Merge the two data frames
results_combined <- rbind(results_Italy, results_Latvia)

# Plot
ggplot(results_combined, aes(x = Year, y = S, color = Country)) +
  geom_line(linewidth = 1.5) +
  labs(title = "SD of Lifespan for females in Italy and Latvia 1970-2019",
       x = "Year",
       y = "Standard Deviation (S)") +
  scale_color_manual(values = c("Italy" = "red", "Latvia" = "lightblue")) +
  theme_minimal()
```



3. Plot life expectancy at birth vs standard deviation, what do you observe?

Simple plot of e0 vs. S

```
# first: a simple plot with e0 on the x-axis and S on the y-axis
# this follows what we have seen in the alternative measures
# of longevity lecture (Slide 27)

ggplot(results_combined, aes(x = e0, y = S, color = Country)) +
  geom_line(linewidth = 1.5) +
  labs(title = "e0 vs S for females in Italy and Latvia 1970-2019",
       x = "Life expectancy at birth (e0)",
       y = "Standard Deviation (S)") +
  scale_color_manual(values = c("Italy" = "red", "Latvia" = "lightblue")) +
  theme_minimal()
```



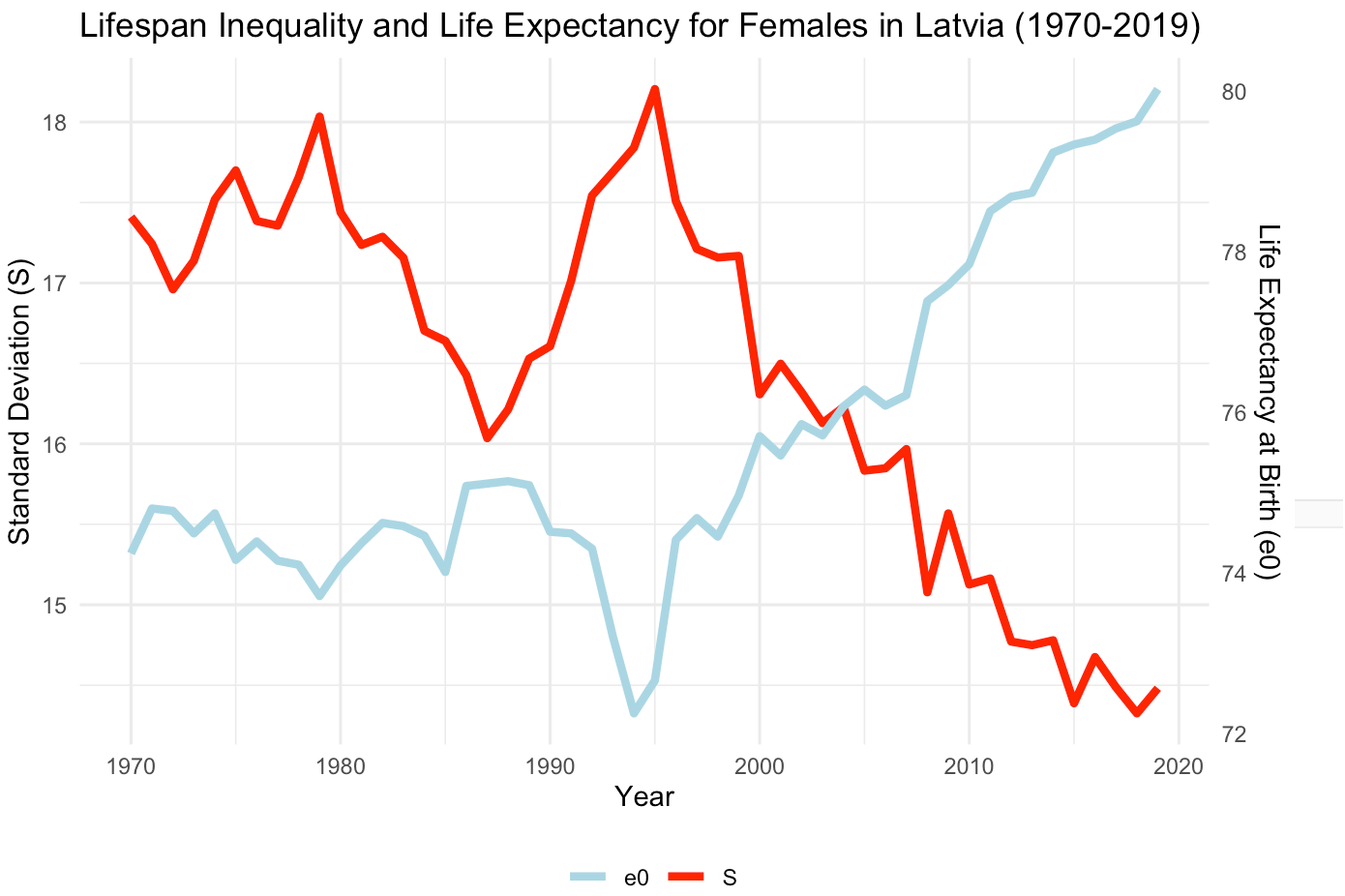
Observations:

- This plot matches what we saw in the Alternative measures of longevity lecture (Slide 27: Life expectancy and life span inequality (σ)).
- Generally, we see that as life expectancy at birth improves, life span inequality reduces, which is the case in plot above.

Can also be useful to see the relationship between e0 and S by plotting Latvia and Italy individually.

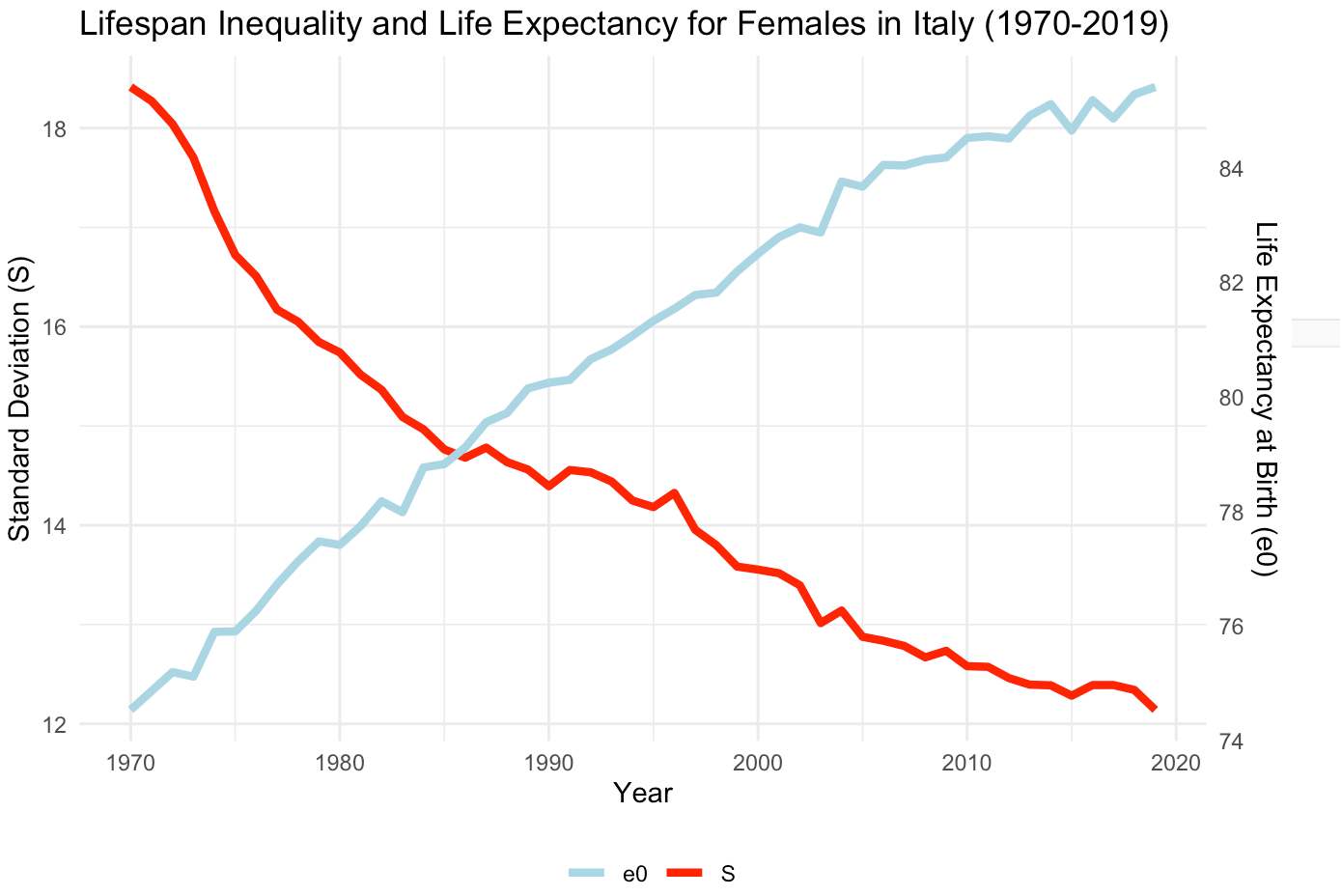
```
# Latvia
# to plot e0 and SD on the y-axis, we need to scale both approrpriately
# scaling factors for e0 to match S's range
min_S_Latvia <- min(results_Latvia$S, na.rm = TRUE)
max_S_Latvia <- max(results_Latvia$S, na.rm = TRUE)
min_e0_Latvia <- min(results_Latvia$e0, na.rm = TRUE)
max_e0_Latvia <- max(results_Latvia$e0, na.rm = TRUE)

# Scale e0 to match S's scale
results_Latvia$e0_scaled <- (results_Latvia$e0 - min_e0_Latvia) / (max_e0_Latvia - min_e0

# plot e0 vs S
ggplot(results_Latvia, aes(x = Year)) +
  geom_line(aes(y = S, color = "S"), linewidth = 1.5) +
```

```
  geom_line(aes(y = e0_scaled, color = "e0"), linewidth = 1.5) +
  scale_y_continuous(
    name = "Standard Deviation (S)",
    sec.axis = sec_axis(~ (. - min_S_Latvia) / (max_S_Latvia - min_S_Latvia) * (max_e0_La
                        name = "Life Expectancy at Birth (e0)")
  ) +
  scale_color_manual(values = c("S" = "red", "e0" = "lightblue")) +
  labs(title = "Lifespan Inequality and Life Expectancy for Females in Latvia (1970-2019)
       x = "Year",
       color = "") +
  theme_minimal() +
  theme(legend.position = "bottom")
```




```
# Italy
# scaling factors for e0 to match S's range
min_S_Italy <- min(results_Italy$S, na.rm = TRUE)
max_S_Italy <- max(results_Italy$S, na.rm = TRUE)
min_e0_Italy <- min(results_Italy$e0, na.rm = TRUE)
max_e0_Italy <- max(results_Italy$e0, na.rm = TRUE)

# Scale e0 to match S's scale
results_Italy$e0_scaled <- (results_Italy$e0 - min_e0_Italy) / (max_e0_Italy - min_e0_Ita
```

```
# plot e0 vs S
ggplot(results_Italy, aes(x = Year)) +
  geom_line(aes(y = S, color = "S"), linewidth = 1.5) +
  geom_line(aes(y = e0_scaled, color = "e0"), linewidth = 1.5) +
  scale_y_continuous(
    name = "Standard Deviation (S)",
    sec.axis = sec_axis(~ (. - min_S_Italy) / (max_S_Italy - min_S_Italy) * (max_e0_Italy
                        name = "Life Expectancy at Birth (e0)")
  ) +
  scale_color_manual(values = c("S" = "red", "e0" = "lightblue")) +
  labs(title = "Lifespan Inequality and Life Expectancy for Females in Italy (1970-2019)"
       x = "Year",
       color = "") +
  theme_minimal() +
  theme(legend.position = "bottom")
```



4. Using the Horiuchi model for decomposition, decompose the change in lifespan inequality between 2000 and 2019 in both countries by age.

To answer this question, we will need to first define an sd.frommx function in R

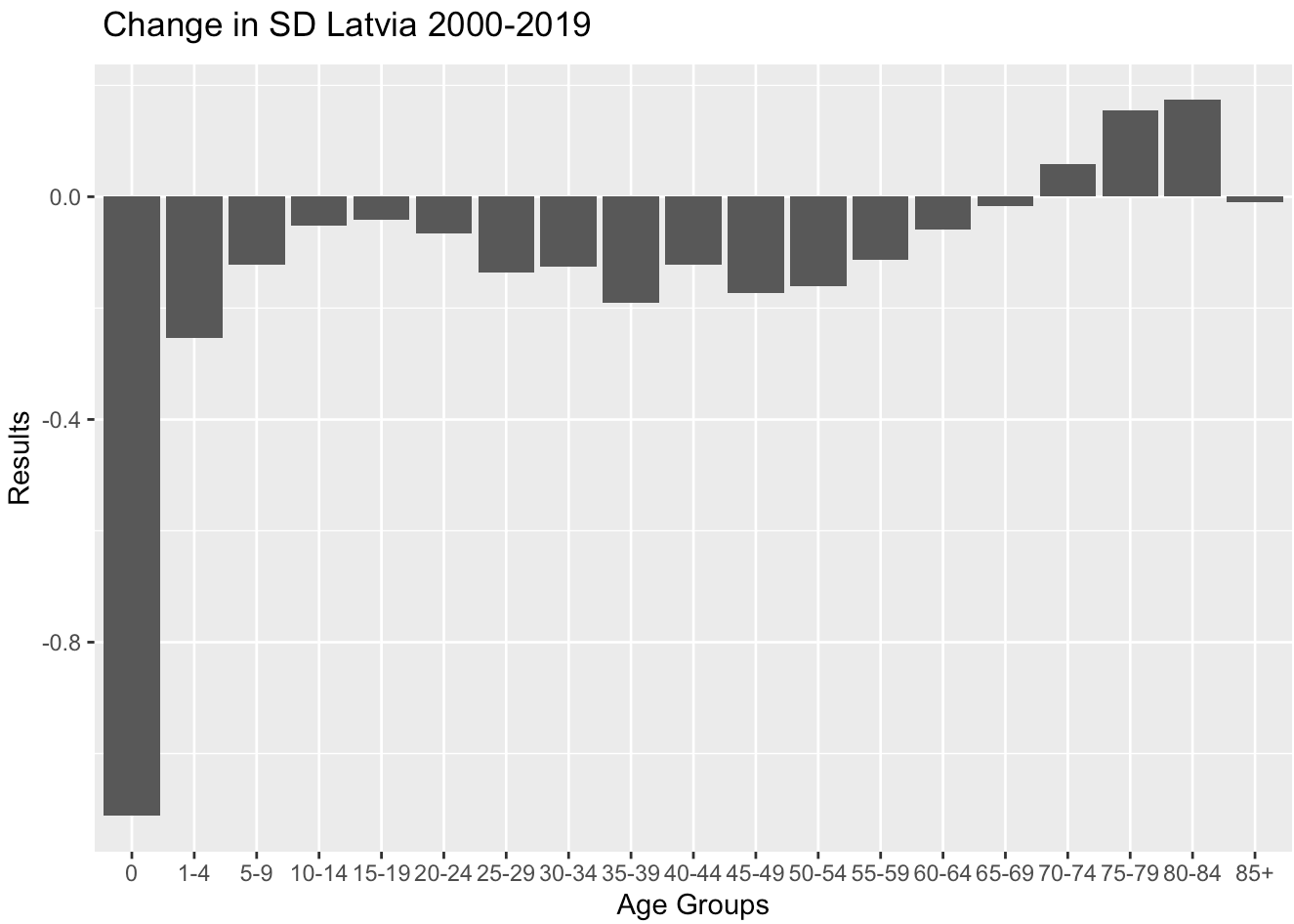
```r
# sd.frommx is defined as we saw in the practical on Change in LSV 1996-2023
sd.frommx <- function(nmx =  mx, sex=1, age = c(0, 1, seq(5, 85, 5)), nax = NULL){
  n   <- c(diff(age), 999)

  if (is.null(nax)) {
    nax <- 0.5 * n
    if (n[2] == 4) {
      if (sex == 1) {
        if (nmx[1] >= 0.107) {
          nax[1] <- 0.33
          nax[2] <- 1.352
        }
        else {
          nax[1] <- 0.045 + 2.684 * nmx[1]
          nax[2] <- 1.651 - 2.816 * nmx[1]
        }
      }
      if (sex == 2) {
        if (nmx[1] >= 0.107) {
          nax[1] <- 0.35
          nax[2] <- 1.361
        }
        else {
          nax[1] <- 0.053 + 2.8 * nmx[1]
          nax[2] <- 1.522 - 1.518 * nmx[1]
        }
      }
    }
  }
  nqx          <- (n * nmx)/(1 + (n - nax) * nmx)
  nqx          <- c(nqx[-(length(nqx))], 1)
  nqx[nqx > 1] <- 1

  npx <- 1 - nqx
  lx <- cumprod(c(1, npx))
  ndx <- -diff(lx)
  lxpn <- lx[-1]
  nLxpn <- n * lxpn + ndx * nax
  nLx <- c(nLxpn[-length(nLxpn)], lxpn[length(lxpn)-1]/nmx[length(nmx)])
  Tx <- rev(cumsum(rev(nLx)))
  lx <- lx[1:length(age)]
  ex <- Tx/lx
  nax[length(nax)] <- ex[length(ex)]
  vx <- sum(ndx*(age+nax-ex[1L])^2)
  sd <- sqrt(vx)
  return(sd)
}
```

We also need to create an object that gives us labels for age groups

```
#labels for age-groups
age_names<-c("0"="0","1"="1-4","5"="5-9","10"="10-14","15"="15-19","20"="20-24",
             "25"="25-29","30"="30-34","35"="35-39","40"="40-44","45"="45-49",
             "50"="50-54","55"="55-59","60"="60-64","65"="65-69","70"="70-74",
             "75"="75-79","80"="80-84","85"="85+")
```

Latvia:

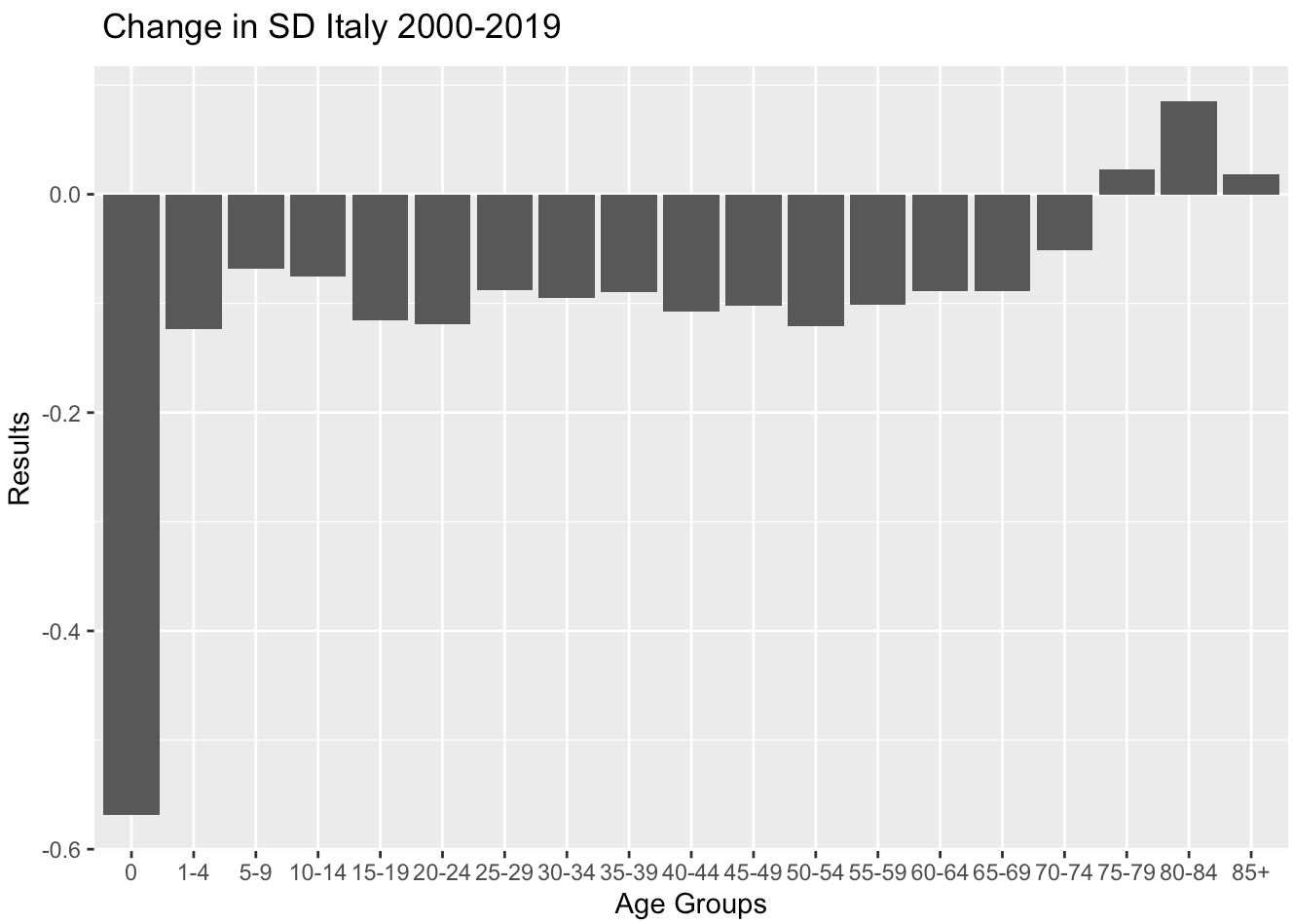
```
# obtain the mortality rates for each age
# Note that the length of these vectors is 111. This is because we
# have mortality rates for each individual age
mx1_Latvia <- LT_females_Latvia[LT_females_Latvia$Year == 2000,]$mx
mx2_Latvia <- LT_females_Latvia[LT_females_Latvia$Year == 2019,]$mx

# In order for the horiuchi model to work properly, the pars1 and pars2
# parameters need to be of length 19, not 111. We need to transform the mx1_Latvia and mx

# here, we create a vector that will be used to transform our data into
# the typical unabridged LT format
age_groups <- c(0, rep(1, 4), rep(seq(2, 18), each = 5), rep(18, 21))

# Compute mean mortality rates for each group
mx1_Latvia_19 <- tapply(mx1_Latvia, age_groups, mean)
mx2_Latvia_19 <- tapply(mx2_Latvia, age_groups, mean)

# Convert to numeric vector
mx1_Latvia_19 <- as.numeric(mx1_Latvia_19)
mx2_Latvia_19 <- as.numeric(mx2_Latvia_19)

# age specific contributions to the change in Lifespan variation
Results_horiuchi_Latvia <- horiuchi(func = sd.frommx,
                                    pars1 = mx1_Latvia_19,
                                    pars2 = mx2_Latvia_19, N = 100)

# Extract just the labels
age_labels <- as.character(age_names)

# Convert to an ordered factor, resolves issues had beforehand with plotting
age_labels <- factor(age_labels, levels = age_labels)

# Plot Results
    ggplot() +
      ggtitle(bquote(~"Change in SD Latvia 2000-2019")) +
      geom_bar(aes(x = age_labels, y = Results_horiuchi_Latvia),
               stat = "identity", position = "stack") +
      xlab("Age Groups") +
      ylab("Results")
```



Italy:

```
# Italy
mx1_Italy <- LT_females_Italy[LT_females_Italy$Year == 2000,]$mx
mx2_Italy <- LT_females_Italy[LT_females_Italy$Year == 2019,]$mx

# Compute mean mortality rates for each group
mx1_Italy_19 <- tapply(mx1_Italy, age_groups, mean)
mx2_Italy_19 <- tapply(mx2_Italy, age_groups, mean)

# Convert to numeric vector
mx1_Italy_19 <- as.numeric(mx1_Italy_19)
mx2_Italy_19 <- as.numeric(mx2_Italy_19)

# age specific contributions to the change in Lifespan variation
Results_horiuchi_Italy <- horiuchi(func = sd.frommx,
                                   pars1 = mx1_Italy_19,
                                   pars2 = mx2_Italy_19, N = 100)

# Plot results
ggplot() +
  ggtitle(bquote(~"Change in SD Italy 2000-2019")) +
  geom_bar(aes(x = age_labels, y = Results_horiuchi_Italy),
           stat = "identity", position = "stack") +
```

```
  xlab("Age Groups") +
  ylab("Results")
```



5. Which age groups contribute to the changes observed? interpret your results and speculate what causes of death may be driving the observed trends.

It appears that the first age group from ages 0 to 1 is contributing the most to changes in SD for both countries from 2000 to 2019. It may be that infant mortality was heavily reduced during this time, thus making the contribution to changes in SD more noticeable for this particular age group.

As far as possible causes of death that may be driving this trend, it could be that less infants are dying from conditions of the perinatal period. Additionally, it may be that this age group is experiencing less deaths due to infectious disease, possibly due to improvements in treatment.

6. How do we interpret lifespan inequality and why do authors think it is important?

Lifespan inequality can be seen as an alternative measure of longevity, particularly alongside life expectancy. While life expectancy can be a useful measure for monitoring improvements in health and development across the globe, it doesn't tell us the whole story. Understanding lifespan inequality,

which is interested in capturing the variation in when people die, is equally important for researchers and authors to consider.

For example, two countries can have an improved life expectancy at birth of 15 years from 1950 to 2020, but they can experience different lifespan variations over this same time period. This is an important factor in measuring inequality because it involves going a step further to unpack whether deaths are more concentrated or dispersed around a specific age. In an ideal state, deaths would be more clustered around a specific age so as to indicate that individuals are dying "equally", for lack of a better term.

Question Three - Projections

Load the data for France

```
# set working directory to access France, WW1.xlsx
setwd("~/Desktop/Term 2/Population Dynamics & Projections/Summative Assessment - work/")

# load France, WW1 excel file
France_WWI <- read_excel("France, WW1.xlsx")
```

Some data handling:

```
# change column names, will make it simpler to follow code from practical
colnames(France_WWI) <- c("age","nNx_m","nNx_m1920.5","nNx_f", "nNx_f1920.5", "nLx_m","nL

# Read the data as a matrix
France_WWI <- as.matrix(France_WWI)

# create separate matrices for females and males
female <- France_WWI[,c("age","nLx_f","nFx","nNx_f")]
male <- France_WWI[,c("age","nLx_m","nNx_m")]
```

Important definitions:

```
year <- 1910.5 # mid-year population
ages <- nrow(female) # The number of age groups
radix <- 1 # The radix of a life table
n <- 1 # The length of the age intervals—which is the same for all
SRB <- 1.05 # The Sex Ratio at Birth
```

Female population, 1910.5:

```
nLx_f <- female[,"nLx_f"] # creates a vector of nLx values for females in 1910.5
nSx_f <- c(nLx_f[1]/(n*radix),nLx_f[2:(ages - 1)]/nLx_f[1:(ages - 2)],nLx_f[ages]/sum(nLx
nFx <- female[,"nFx"] # creates a vector of nFx values for females in 1910.5
```

```
# Incorporates specific assumptions about fertility.
nGx <- (nFx + c(nFx[2:ages]*nSx_f[2:ages],0))*n/2
```

Create LESliE matrix which will be used for population projections:

```
# Define a space for the Leslie matrix
lESliE <- matrix(0,ages,ages)

# Inserts multipliers to project the first age group—depending on the age distribution of
lESliE[1,] <- nGx*1/(1 + SRB)*nSx_f[1]

# Inserts multipliers to project (part of) the open-ended age group
lESliE[ages,ages] <- nSx_f[ages]

# Inserts multipliers to project the remaining age groups
for (x in 1:(ages - 1)) {
  lESliE[x + 1,x] <- nSx_f[x + 1]
}
```

Create sub-matrices (lESliE_ff, lESliE_mm, lESliE_fm, and lESliE_mf):

```
# Note we use the subscript ff to define a Leslie matrix to project female population usi
lESliE_ff <- lESliE

# Defines a space for the Leslie matrix to project male population using only male inform
lESliE_mm <- matrix(0,ages,ages)

# Defines a space for the Leslie matrix to project male population using only female info
lESliE_fm <- matrix(0,ages,ages)

# Defines a null space for the Leslie matrix to project female population using only male
lESliE_mf <- matrix(0,ages,ages)
```

Male population, 1910.5:

```
# Male population
nLx_m <- male[,"nLx_m"]

# Calculation of survivorship ratios is the same as for female population, just using nLx
nSx_m <- c(nLx_m[1]/(n*radix),nLx_m[2:(ages - 1)]/nLx_m[1:(ages - 2)],nLx_m[ages]/sum(nLx

# Inserts multipliers to project the first age group—depending on the age distribution of
lESliE_fm[1,] <- nGx*SRB/(1 + SRB)*nSx_m[1]

# Inserts multipliers to project (part of) the open-ended age group
lESliE_mm[ages,ages] <- nSx_m[ages]
```

```
# Inserts multipliers to project the remaining age groups
for (x in 1:(ages - 1)) {
  lESliE_mm[x + 1,x] <- nSx_m[x + 1]
}
```

Now, extend LESLiE matrix to 202x202 to account for both sexes:

```
# Concatenates the Leslie matrices for the two sexes
lESliE <- rbind(cbind(lESliE_ff,lESliE_mf),cbind(lESliE_fm,lESliE_mm))
```

Current vs. projected:

```
current <- rbind(matrix(female[,"nNx_f"],ages),matrix(male[,"nNx_m"],ages))
sum(current) # 39,185,119
```

```
[1] 39185119
```

```
projected <- lESliE%*%current
sum(projected) # 39,261,845
```

```
[1] 39261845
```

Now create a data frame called pop that takes the columns of interest:

```
pop <- France_WWI[,c("age","nNx_m","nNx_f")] #Retains only relevant information for the t
colnames(pop)[colnames(pop) == "nNx_f"] <- str_c("nNx_f",sprintf("%.1f",year)) # Renames
colnames(pop)[colnames(pop) == "nNx_m"] <- str_c("nNx_m",sprintf("%.1f",year))

# don't execute command more than once, otherwise, pop object will change
# for loop iterates every 5 years (from 1910.5-1915.5 and 1915.5-1920.5)


for (t in seq(year, year + 5*n, 5*n)) {
  current <- rbind(matrix(pop[,str_c("nNx_f",sprintf("%.1f",t))],ages),matrix(pop[,str_c(
  projected <- lESliE%*%current #Projects the population n years using the most recent da
  female <- matrix(projected[1:ages,],ages) # Extracts female population
  colnames(female)[1] <- str_c("nNx_f",sprintf("%.1f",t + 5*n)) # Gives a name to the pro
  male <- matrix(projected[(1 + ages):(2*ages),],ages) # Extracts male population
  colnames(male)[1] <- str_c("nNx_m",sprintf("%.1f",t + 5*n)) # Gives a name to the proje
  pop <- cbind(pop,male,female) # Concatenates the table to include the new set of projec
}

pop # lists the current population for males and females in 1910.5,
```

```
      age nNx_m1910.5 nNx_f1910.5  nNx_m1915.5  nNx_f1915.5  nNx_m1920.5
 [1,]   0   367716.62   358234.02 368144.49208 356954.12272 368372.32812
 [2,]   1   339158.18   334671.47 346740.93085 340321.51302 347144.39578
 [3,]   2   327918.23   324483.68 332405.17053 328068.69769 339836.94053
```

[4,]	3	329206.46	326062.55	324714.45433	321309.73080	329157.55724
[5,]	4	331455.83	328307.88	327138.94564	323917.93444	322675.15110
[6,]	5	332074.81	329878.16	329911.08719	326715.38751	325614.32158
[7,]	6	331059.81	328866.72	330827.36280	328599.77583	328671.76807
[8,]	7	337225.14	334094.89	330040.41541	327743.21761	329808.68395
[9,]	8	337009.72	333224.65	336321.55936	333088.16935	329156.08594
[10,]	9	336030.96	331817.71	336187.57815	332356.52446	335501.09629
[11,]	10	335039.61	331396.77	335293.22094	331033.14904	335449.49524
[12,]	11	324803.11	322115.12	334339.64242	330632.22404	334592.72351
[13,]	12	330785.48	326452.84	324115.76181	321331.53948	333632.11303
[14,]	13	333849.14	328227.50	330057.74814	325544.36189	323402.70341
[15,]	14	326656.03	322815.44	333086.18215	327199.42575	329303.45490
[16,]	15	326395.44	323320.21	325774.37628	321656.60256	332187.17327
[17,]	16	327013.43	325556.47	325266.65408	322012.98846	324647.73821
[18,]	17	321388.54	321406.39	325641.04640	324194.50369	323901.60121
[19,]	18	308707.11	312820.16	319797.64282	319918.89063	324029.09900
[20,]	19	303616.91	314456.78	306997.78042	311253.70453	318026.90431
[21,]	20	311757.44	318102.02	301846.32748	312826.36739	305207.48191
[22,]	21	315925.65	313602.77	309745.72823	316355.69337	299898.57024
[23,]	22	311329.86	316305.93	313699.64728	311762.56148	307563.26905
[24,]	23	301101.06	311400.09	309171.16209	314396.53892	311524.51775
[25,]	24	308463.53	318786.05	299139.08984	309527.74520	307156.60727
[26,]	25	310244.61	319359.46	306511.94045	316843.75719	297246.49424
[27,]	26	303369.22	309734.18	308170.03859	317369.58299	304462.32900
[28,]	27	303717.59	310290.47	301288.51268	307778.62113	306056.40408
[29,]	28	302512.45	307428.73	301602.21725	308323.56679	299190.05829
[30,]	29	299051.31	304847.03	300340.36057	305483.78480	299436.66344
[31,]	30	298914.33	304950.67	296906.05619	302927.44204	298185.85972
[32,]	31	292003.32	295310.19	296733.72570	302985.89551	294740.10241
[33,]	32	295319.75	296588.36	289824.06338	293362.41755	294519.16547
[34,]	33	296552.22	297976.70	293005.02419	294577.07731	287552.41294
[35,]	34	305302.02	308539.09	294118.62758	295934.66861	290600.54107
[36,]	35	299747.60	301699.41	302770.90618	306394.21235	291680.22995
[37,]	36	285173.08	284471.86	297161.98597	299485.28299	300159.21320
[38,]	37	288074.44	288746.86	282632.87251	282389.61626	294514.98611
[39,]	38	256053.28	258388.06	285481.91591	286645.14915	280089.31978
[40,]	39	257118.22	260164.71	253569.64245	256445.64680	282712.82970
[41,]	40	264841.32	272857.94	254452.78137	258209.69196	250940.99046
[42,]	41	249376.34	256001.15	262008.69895	270751.27550	251731.27135
[43,]	42	246522.64	251948.79	246584.76359	253950.03666	259075.71299
[44,]	43	253769.90	258053.24	243644.53026	249860.20521	243705.92857
[45,]	44	253555.46	259742.07	250673.75502	255824.38604	240671.92087
[46,]	45	247243.77	253033.52	250300.89310	257379.70107	247456.17688
[47,]	46	241970.54	248094.06	243946.68816	250633.84627	246963.04346
[48,]	47	240378.55	249729.40	238646.02125	245754.78433	240595.01840
[49,]	48	233978.27	240641.08	236866.15223	247248.40720	235158.93908
[50,]	49	235950.71	243661.37	230369.00495	238051.87814	233212.33973
[51,]	50	229428.33	238370.50	232161.93031	240907.76772	226669.85352
[52,]	51	218551.83	226963.33	225499.98718	235574.67407	228186.78195
[53,]	52	201916.00	210380.17	214551.19634	224153.75338	221372.16616
[54,]	53	198738.00	208312.33	198027.67262	207622.15668	210419.55105

```
      [55,]  54   197078.83   210295.50 194783.65164 205451.03750 194087.45785
      [56,]  55   188158.00   202582.50 192910.89277 207282.12238 190664.25417
      [57,]  56   182971.50   192670.67 183946.16558 199484.89605 188592.66693
      [58,]  57   184477.00   196200.00 178617.54794 189472.22420 179569.02058
      [59,]  58   185019.17   202564.50 179751.81806 192777.99956 174042.44962
      [60,]  59   182784.83   197319.83 179996.60697 198787.04220 174872.24349
      [61,]  60   172232.67   187457.67 177485.05418 193180.45487 174777.67460
      [62,]  61   160620.50   190740.50 166830.53291 183223.55746 171918.17425
      [63,]  62   148549.83   176629.00 155246.42252 186152.69236 161248.67873
      [64,]  63   138892.67   158605.00 143173.62034 171897.93193 149627.85456
      [65,]  64   144584.00   167075.17 133414.82298 153932.00399 137526.93510
      [66,]  65   142125.50   163646.83 138451.70775 161605.98926 127756.25298
      [67,]  66   132664.00   150351.83 135659.47373 157706.31659 132152.82134
      [68,]  67   127861.50   145854.00 126133.14836 144405.16957 128981.15936
      [69,]  68   122982.83   141801.17 120999.24793 139648.92285 119363.65591
      [70,]  69   117002.17   137517.83 115801.90575 135379.26829 113934.14434
      [71,]  70   107989.33   127668.33 109540.01106 130688.13295 108416.29720
      [72,]  71    96270.00   113733.67 100464.91015 120584.97815 101907.54372
      [73,]  72    86433.00   105734.83  88957.92601 106785.07394  92834.21672
      [74,]  73    80159.17    98107.50  79257.50993  98568.46553  81572.82178
      [75,]  74    77232.67    95528.50  72881.36869  90738.16705  72061.57203
      [76,]  75    67464.17    84599.50  69650.15849  87560.20288  65726.05712
      [77,]  76    56859.33    72289.83  60191.86504  76768.45901  62142.21475
      [78,]  77    49526.83    64150.83  50109.75700  64977.75348  53046.69841
      [79,]  78    43301.67    56347.00  43117.73510  57003.78388  43625.22754
      [80,]  79    40756.17    55811.50  37389.54282  49756.15751  37230.72119
      [81,]  80    33737.05    47465.01  34684.31909  48815.92741  31819.25175
      [82,]  81    27411.41    39178.28  28208.46032  40954.54740  29000.49763
      [83,]  82    22321.73    32931.17  22606.47965  33387.26985  23263.81547
      [84,]  83    17987.65    26937.56  18095.44944  27622.17846  18326.28607
      [85,]  84    14106.12    21618.41  14307.65692  22188.68981  14393.40228
      [86,]  85    10745.87    17230.10  10989.82083  17475.86243  11146.83457
      [87,]  86     8004.64    13622.71   8172.49968  13695.67020   8358.03032
      [88,]  87     5854.93    10429.38   5959.88252  10646.89811   6084.86303
      [89,]  88     4229.02     7695.74   4265.80429   8077.41313   4342.27094
      [90,]  89     3024.22     5739.63   3012.41291   5824.75885   3038.61507
      [91,]  90     2026.90     4163.03   2133.77049   4201.87132   2125.43987
      [92,]  91     1301.18     2763.13   1368.17020   2974.67889   1440.30845
      [93,]  92      846.78     1833.49    852.96563   1932.98389    896.87988
      [94,]  93      559.29     1295.41    557.07241   1273.45251    561.14176
      [95,]  94      341.67      881.15    355.61688    887.77225    354.20686
      [96,]  95      195.67      560.73    206.18191    573.58255    214.59820
      [97,]  96      125.27      339.91    118.44758    359.71996    124.81089
      [98,]  97       70.05      179.69     82.08982    219.64980     77.61906
      [99,]  98       36.10      112.18     43.53255    111.74949     51.01468
     [100,]  99       21.92       73.05     16.48085     65.69640     19.87405
     [101,] 100       22.35       64.67     19.22504     69.46724     15.50591
             nNx_f1920.5
       [1,] 357175.03331
       [2,] 339105.61349
       [3,] 333607.27036
```

```
 [4,] 324859.68151
 [5,] 319196.37602
 [6,] 322346.73585
 [7,] 325449.26010
 [8,] 327477.18539
 [9,] 326755.63632
[10,] 332220.39938
[11,] 331570.68951
[12,] 330269.44197
[13,] 329827.92473
[14,] 320437.31333
[15,] 324524.69176
[16,] 326024.85075
[17,] 320356.10718
[18,] 320665.84631
[19,] 322694.10066
[20,] 318316.88808
[21,] 309639.89940
[22,] 311109.00320
[23,] 314499.33080
[24,] 309880.59658
[25,] 312506.17747
[26,] 307641.86119
[27,] 314869.55512
[28,] 315365.81671
[29,] 305827.64030
[30,] 306372.96042
[31,] 303560.18727
[32,] 300975.70306
[33,] 300987.49654
[34,] 291373.01125
[35,] 292558.34350
[36,] 293877.41338
[37,] 304145.63088
[38,] 297293.14576
[39,] 280334.17808
[40,] 284490.31537
[41,] 254518.57581
[42,] 256216.12274
[43,] 268581.98230
[44,] 251844.86210
[45,] 247702.11602
[46,] 253497.64867
[47,] 254938.80981
[48,] 248270.62299
[49,] 243313.27825
[50,] 244588.11314
[51,] 235361.66838
[52,] 238082.18240
[53,] 232658.49771
[54,] 221215.17301
```

```
 [55,] 204770.34412
 [56,] 202507.07742
 [57,] 204112.65848
 [58,] 196173.33011
 [59,] 186167.56551
 [60,] 189183.04211
 [61,] 194616.88789
 [62,] 188817.08160
 [63,] 178816.55194
 [64,] 181166.52894
 [65,] 166833.28486
 [66,] 148893.06283
 [67,] 155739.56003
 [68,] 151468.77420
 [69,] 138261.73012
 [70,] 133324.49226
 [71,] 128655.78095
 [72,] 123437.23503
 [73,] 113217.79916
 [74,]  99547.52733
 [75,]  91164.50721
 [76,]  83169.44489
 [77,]  79455.10134
 [78,]  69003.37162
 [79,]  57738.57979
 [80,]  50336.11816
 [81,]  43519.57882
 [82,]  42120.16838
 [83,]  34900.98406
 [84,]  28004.74826
 [85,]  22752.61566
 [86,]  17936.86448
 [87,]  13891.01911
 [88,]  10703.92053
 [89,]   8245.87795
 [90,]   6113.63996
 [91,]   4264.19250
 [92,]   3002.43282
 [93,]   2080.97569
 [94,]   1342.55610
 [95,]    872.72432
 [96,]    577.89329
 [97,]    367.96514
 [98,]    232.45099
 [99,]    136.60056
[100,]     65.44428
[101,]     68.17779
```

```
# and projects for years 1915.5 and 1920.5
```

1. Calculate the Total Fertility Rate of this population

```
TFR <- sum(nFx)*n
TFR # TFR = 2.61 births per woman
```

```
[1] 2.61
```

2. Calculate the Net Reproduction Rate of this population

```
NRR <- sum(nFx*nLx_f/radix)*(1/(1+SRB)) # Formula on Slide 27 from Cohort Compoent Model
NRR # NRR = 0.9700091 , NRR < 1 posing that the population will decline
```

```
[1] 0.9700091
```

3. Calculate the projected number of births from 1910 to 1920 by age group of the mother

The following command only gives us the projected number of births by age group of the mother one year in advance:

```
# projected_births <- ((current + lESliE%*%current)/2) * nFx * n
```

So, we must create a for loop that gives us the number of births by age group per year from 1910.5 - 1920.5

```
# Set up parameters
years = seq(1910.5, 1919.5, by = 1)

# Initialize birth_by_age_group object
births_by_age_group <- matrix(0, nrow <-  length(current), ncol <-  length(years))  # Sto

# Loop from 1910.5 to 1920.5 (each step projects to the next year)
# only complete loop once, otherwise results are conflated
for (t in 1:length(years)) {
  # Project population one year forward
  projected_population <- lESliE %*% current

  # Compute births by age group for this particular year
  births_by_age_group[, t] <-  ((current + projected) / 2) * nFx * n # this is the formul

  # Update population for next iteration
  current <-  projected
}


# Here, I've specified that I'm only interested in looking at the age groups of the femal
births_by_age_group <- births_by_age_group[0:101, ]
```

```
# Let's also change the column names
colnames(births_by_age_group) <- c("1911.5", "1912.5", "1913.5", "1914.5", "1915.5", "191

births_by_age_group
```

	1911.5	1912.5	1913.5	1914.5
[1,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[2,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[3,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[4,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[5,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[6,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[7,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[8,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[9,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[10,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[11,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[12,]	0.00002529399	0.0000252801	0.0000252801	0.0000252801
[13,]	0.29279154363	0.2966119131	0.2966119131	0.2966119131
[14,]	18.17185727700	18.0281928957	18.0281928957	18.0281928957
[15,]	282.63130536955	281.4713612519	281.4713612519	281.4713612519
[16,]	2042.02804098327	2055.8003749630	2055.8003749630	2055.8003749630
[17,]	7469.75410422321	7450.4871503488	7450.4871503488	7450.4871503488
[18,]	15774.45518026885	15688.1378130325	15688.1378130325	15688.1378130325
[19,]	23344.11002601087	23444.9246830915	23444.9246830915	23444.9246830915
[20,]	29595.70858342334	29927.7442984236	29927.7442984236	29927.7442984236
[21,]	33752.65274713150	33579.8694924202	33579.8694924202	33579.8694924202
[22,]	37694.82033628836	37379.6264391938	37379.6264391938	37379.6264391938
[23,]	39280.53283723181	39452.1890700824	39452.1890700824	39452.1890700824
[24,]	39952.47736626870	39663.4661667081	39663.4661667081	39663.4661667081
[25,]	40032.63034420982	40224.3151923856	40224.3151923856	40224.3151923856
[26,]	39949.84180646899	39361.1744636780	39361.1744636780	39361.1744636780
[27,]	40143.44413226272	39984.7071558146	39984.7071558146	39984.7071558146
[28,]	38824.42467881190	39297.1377220982	39297.1377220982	39297.1377220982
[29,]	37207.79921747027	37056.5857530008	37056.5857530008	37056.5857530008
[30,]	35959.56322184078	36011.8211513523	36011.8211513523	36011.8211513523
[31,]	34379.25931310617	34415.1270064979	34415.1270064979	34415.1270064979
[32,]	32751.59887783608	32642.5902636539	32642.5902636539	32642.5902636539
[33,]	30654.84852869089	31048.1280360710	31048.1280360710	31048.1280360710
[34,]	28707.46296737608	28550.4861026931	28550.4861026931	28550.4861026931
[35,]	27280.97307101899	27124.4556060329	27124.4556060329	27124.4556060329
[36,]	26190.03652691485	25643.9247200807	25643.9247200807	25643.9247200807
[37,]	24681.58102932907	24872.1358062061	24872.1358062061	24872.1358062061
[38,]	22028.80587605257	22595.1621310196	22595.1621310196	22595.1621310196
[39,]	19909.18426615099	19687.5777985240	19687.5777985240	19687.5777985240
[40,]	17502.53313776178	18409.9469080526	18409.9469080526	18409.9469080526
[41,]	14624.23387578521	14518.9544339669	14518.9544339669	14518.9544339669
[42,]	13109.97908949860	12748.3712378157	12748.3712378157	12748.3712378157
[43,]	10598.02851469406	10894.7945912356	10894.7945912356	10894.7945912356

```
[44,]  7978.20972447021  8009.7701169539  8009.7701169539  8009.7701169539
[45,]  5864.91344800369  5770.3078801611  5770.3078801611  5770.3078801611
[46,]  3984.03753444747  3953.7636487822  3953.7636487822  3953.7636487822
[47,]  2420.93341812078  2441.5477254259  2441.5477254259  2441.5477254259
[48,]  1463.90445795357  1471.3594336924  1471.3594336924  1471.3594336924
[49,]   849.83317133883   843.0160814221   843.0160814221   843.0160814221
[50,]   424.69909665958   430.4506571413   430.4506571413   430.4506571413
[51,]   109.01269603706   107.7432565240   107.7432565240   107.7432565240
[52,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[53,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[54,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[55,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[56,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[57,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[58,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[59,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[60,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[61,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[62,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[63,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[64,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[65,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[66,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[67,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[68,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[69,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[70,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[71,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[72,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[73,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[74,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[75,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[76,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[77,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[78,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[79,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[80,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[81,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[82,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[83,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[84,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[85,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[86,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[87,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[88,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[89,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[90,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[91,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[92,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[93,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
[94,]     0.00000000000     0.0000000000     0.0000000000     0.0000000000
```

[95,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[96,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[97,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[98,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[99,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[100,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000
[101,]	0.00000000000	0.0000000000	0.0000000000	0.0000000000

	1915.5	1916.5	1917.5	1918.5
[1,]	0.0000000000	0.0000000000	0.0000000000	0.0000000000
[2,]	0.0000000000	0.0000000000	0.0000000000	0.0000000000
[3,]	0.0000000000	0.0000000000	0.0000000000	0.0000000000
[4,]	0.0000000000	0.0000000000	0.0000000000	0.0000000000
[5,]	0.0000000000	0.0000000000	0.0000000000	0.0000000000
[6,]	0.0000000000	0.0000000000	0.0000000000	0.0000000000
[7,]	0.0000000000	0.0000000000	0.0000000000	0.0000000000
[8,]	0.0000000000	0.0000000000	0.0000000000	0.0000000000
[9,]	0.0000000000	0.0000000000	0.0000000000	0.0000000000
[10,]	0.0000000000	0.0000000000	0.0000000000	0.0000000000
[11,]	0.0000000000	0.0000000000	0.0000000000	0.0000000000
[12,]	0.0000252801	0.0000252801	0.0000252801	0.0000252801
[13,]	0.2966119131	0.2966119131	0.2966119131	0.2966119131
[14,]	18.0281928957	18.0281928957	18.0281928957	18.0281928957
[15,]	281.4713612519	281.4713612519	281.4713612519	281.4713612519
[16,]	2055.8003749630	2055.8003749630	2055.8003749630	2055.8003749630
[17,]	7450.4871503488	7450.4871503488	7450.4871503488	7450.4871503488
[18,]	15688.1378130325	15688.1378130325	15688.1378130325	15688.1378130325
[19,]	23444.9246830915	23444.9246830915	23444.9246830915	23444.9246830915
[20,]	29927.7442984236	29927.7442984236	29927.7442984236	29927.7442984236
[21,]	33579.8694924202	33579.8694924202	33579.8694924202	33579.8694924202
[22,]	37379.6264391938	37379.6264391938	37379.6264391938	37379.6264391938
[23,]	39452.1890700824	39452.1890700824	39452.1890700824	39452.1890700824
[24,]	39663.4661667081	39663.4661667081	39663.4661667081	39663.4661667081
[25,]	40224.3151923856	40224.3151923856	40224.3151923856	40224.3151923856
[26,]	39361.1744636780	39361.1744636780	39361.1744636780	39361.1744636780
[27,]	39984.7071558146	39984.7071558146	39984.7071558146	39984.7071558146
[28,]	39297.1377220982	39297.1377220982	39297.1377220982	39297.1377220982
[29,]	37056.5857530008	37056.5857530008	37056.5857530008	37056.5857530008
[30,]	36011.8211513523	36011.8211513523	36011.8211513523	36011.8211513523
[31,]	34415.1270064979	34415.1270064979	34415.1270064979	34415.1270064979
[32,]	32642.5902636539	32642.5902636539	32642.5902636539	32642.5902636539
[33,]	31048.1280360710	31048.1280360710	31048.1280360710	31048.1280360710
[34,]	28550.4861026931	28550.4861026931	28550.4861026931	28550.4861026931
[35,]	27124.4556060329	27124.4556060329	27124.4556060329	27124.4556060329
[36,]	25643.9247200807	25643.9247200807	25643.9247200807	25643.9247200807
[37,]	24872.1358062061	24872.1358062061	24872.1358062061	24872.1358062061
[38,]	22595.1621310196	22595.1621310196	22595.1621310196	22595.1621310196
[39,]	19687.5777985240	19687.5777985240	19687.5777985240	19687.5777985240
[40,]	18409.9469080526	18409.9469080526	18409.9469080526	18409.9469080526
[41,]	14518.9544339669	14518.9544339669	14518.9544339669	14518.9544339669
[42,]	12748.3712378157	12748.3712378157	12748.3712378157	12748.3712378157
[43,]	10894.7945912356	10894.7945912356	10894.7945912356	10894.7945912356

```
[44,]  8009.7701169539  8009.7701169539  8009.7701169539  8009.7701169539
[45,]  5770.3078801611  5770.3078801611  5770.3078801611  5770.3078801611
[46,]  3953.7636487822  3953.7636487822  3953.7636487822  3953.7636487822
[47,]  2441.5477254259  2441.5477254259  2441.5477254259  2441.5477254259
[48,]  1471.3594336924  1471.3594336924  1471.3594336924  1471.3594336924
[49,]   843.0160814221   843.0160814221   843.0160814221   843.0160814221
[50,]   430.4506571413   430.4506571413   430.4506571413   430.4506571413
[51,]   107.7432565240   107.7432565240   107.7432565240   107.7432565240
[52,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[53,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[54,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[55,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[56,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[57,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[58,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[59,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[60,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[61,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[62,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[63,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[64,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[65,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[66,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[67,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[68,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[69,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[70,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[71,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[72,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[73,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[74,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[75,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[76,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[77,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[78,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[79,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[80,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[81,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[82,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[83,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[84,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[85,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[86,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[87,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[88,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[89,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[90,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[91,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[92,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[93,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
[94,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
```

```
      [95,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
      [96,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
      [97,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
      [98,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
      [99,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
     [100,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
     [101,]     0.0000000000     0.0000000000     0.0000000000     0.0000000000
                     1919.5           1920.5
       [1,]     0.0000000000     0.0000000000
       [2,]     0.0000000000     0.0000000000
       [3,]     0.0000000000     0.0000000000
       [4,]     0.0000000000     0.0000000000
       [5,]     0.0000000000     0.0000000000
       [6,]     0.0000000000     0.0000000000
       [7,]     0.0000000000     0.0000000000
       [8,]     0.0000000000     0.0000000000
       [9,]     0.0000000000     0.0000000000
      [10,]     0.0000000000     0.0000000000
      [11,]     0.0000000000     0.0000000000
      [12,]     0.0000252801     0.0000252801
      [13,]     0.2966119131     0.2966119131
      [14,]    18.0281928957    18.0281928957
      [15,]   281.4713612519   281.4713612519
      [16,]  2055.8003749630  2055.8003749630
      [17,]  7450.4871503488  7450.4871503488
      [18,] 15688.1378130325 15688.1378130325
      [19,] 23444.9246830915 23444.9246830915
      [20,] 29927.7442984236 29927.7442984236
      [21,] 33579.8694924202 33579.8694924202
      [22,] 37379.6264391938 37379.6264391938
      [23,] 39452.1890700824 39452.1890700824
      [24,] 39663.4661667081 39663.4661667081
      [25,] 40224.3151923856 40224.3151923856
      [26,] 39361.1744636780 39361.1744636780
      [27,] 39984.7071558146 39984.7071558146
      [28,] 39297.1377220982 39297.1377220982
      [29,] 37056.5857530008 37056.5857530008
      [30,] 36011.8211513523 36011.8211513523
      [31,] 34415.1270064979 34415.1270064979
      [32,] 32642.5902636539 32642.5902636539
      [33,] 31048.1280360710 31048.1280360710
      [34,] 28550.4861026931 28550.4861026931
      [35,] 27124.4556060329 27124.4556060329
      [36,] 25643.9247200807 25643.9247200807
      [37,] 24872.1358062061 24872.1358062061
      [38,] 22595.1621310196 22595.1621310196
      [39,] 19687.5777985240 19687.5777985240
      [40,] 18409.9469080526 18409.9469080526
      [41,] 14518.9544339669 14518.9544339669
      [42,] 12748.3712378157 12748.3712378157
      [43,] 10894.7945912356 10894.7945912356
```

```
      [44,]  8009.7701169539  8009.7701169539
      [45,]  5770.3078801611  5770.3078801611
      [46,]  3953.7636487822  3953.7636487822
      [47,]  2441.5477254259  2441.5477254259
      [48,]  1471.3594336924  1471.3594336924
      [49,]   843.0160814221   843.0160814221
      [50,]   430.4506571413   430.4506571413
      [51,]   107.7432565240   107.7432565240
      [52,]     0.0000000000     0.0000000000
      [53,]     0.0000000000     0.0000000000
      [54,]     0.0000000000     0.0000000000
      [55,]     0.0000000000     0.0000000000
      [56,]     0.0000000000     0.0000000000
      [57,]     0.0000000000     0.0000000000
      [58,]     0.0000000000     0.0000000000
      [59,]     0.0000000000     0.0000000000
      [60,]     0.0000000000     0.0000000000
      [61,]     0.0000000000     0.0000000000
      [62,]     0.0000000000     0.0000000000
      [63,]     0.0000000000     0.0000000000
      [64,]     0.0000000000     0.0000000000
      [65,]     0.0000000000     0.0000000000
      [66,]     0.0000000000     0.0000000000
      [67,]     0.0000000000     0.0000000000
      [68,]     0.0000000000     0.0000000000
      [69,]     0.0000000000     0.0000000000
      [70,]     0.0000000000     0.0000000000
      [71,]     0.0000000000     0.0000000000
      [72,]     0.0000000000     0.0000000000
      [73,]     0.0000000000     0.0000000000
      [74,]     0.0000000000     0.0000000000
      [75,]     0.0000000000     0.0000000000
      [76,]     0.0000000000     0.0000000000
      [77,]     0.0000000000     0.0000000000
      [78,]     0.0000000000     0.0000000000
      [79,]     0.0000000000     0.0000000000
      [80,]     0.0000000000     0.0000000000
      [81,]     0.0000000000     0.0000000000
      [82,]     0.0000000000     0.0000000000
      [83,]     0.0000000000     0.0000000000
      [84,]     0.0000000000     0.0000000000
      [85,]     0.0000000000     0.0000000000
      [86,]     0.0000000000     0.0000000000
      [87,]     0.0000000000     0.0000000000
      [88,]     0.0000000000     0.0000000000
      [89,]     0.0000000000     0.0000000000
      [90,]     0.0000000000     0.0000000000
      [91,]     0.0000000000     0.0000000000
      [92,]     0.0000000000     0.0000000000
      [93,]     0.0000000000     0.0000000000
      [94,]     0.0000000000     0.0000000000
```

```
 [95,]     0.0000000000     0.0000000000
 [96,]     0.0000000000     0.0000000000
 [97,]     0.0000000000     0.0000000000
 [98,]     0.0000000000     0.0000000000
 [99,]     0.0000000000     0.0000000000
[100,]     0.0000000000     0.0000000000
[101,]     0.0000000000     0.0000000000
```

4. Calculate the projected total number of births for each projected year

```
# Calculate the projected total number of births for each year
total_births <- colSums(births_by_age_group)
total_births
```

```
  1911.5   1912.5   1913.5   1914.5   1915.5   1916.5   1917.5   1918.5
786839.4 787057.4 787057.4 787057.4 787057.4 787057.4 787057.4 787057.4
  1919.5   1920.5
787057.4 787057.4
```

5. Calculate how many are projected to be male births

```
male_births <- total_births * SRB / (1 + SRB)
male_births
```

```
  1911.5   1912.5   1913.5   1914.5   1915.5   1916.5   1917.5   1918.5
403015.3 403127.0 403127.0 403127.0 403127.0 403127.0 403127.0 403127.0
  1919.5   1920.5
403127.0 403127.0
```

6. Using a plot of overlapping population pyramids, compare the projected population of 1920.5 with the population counts of the same year (post-exposure).

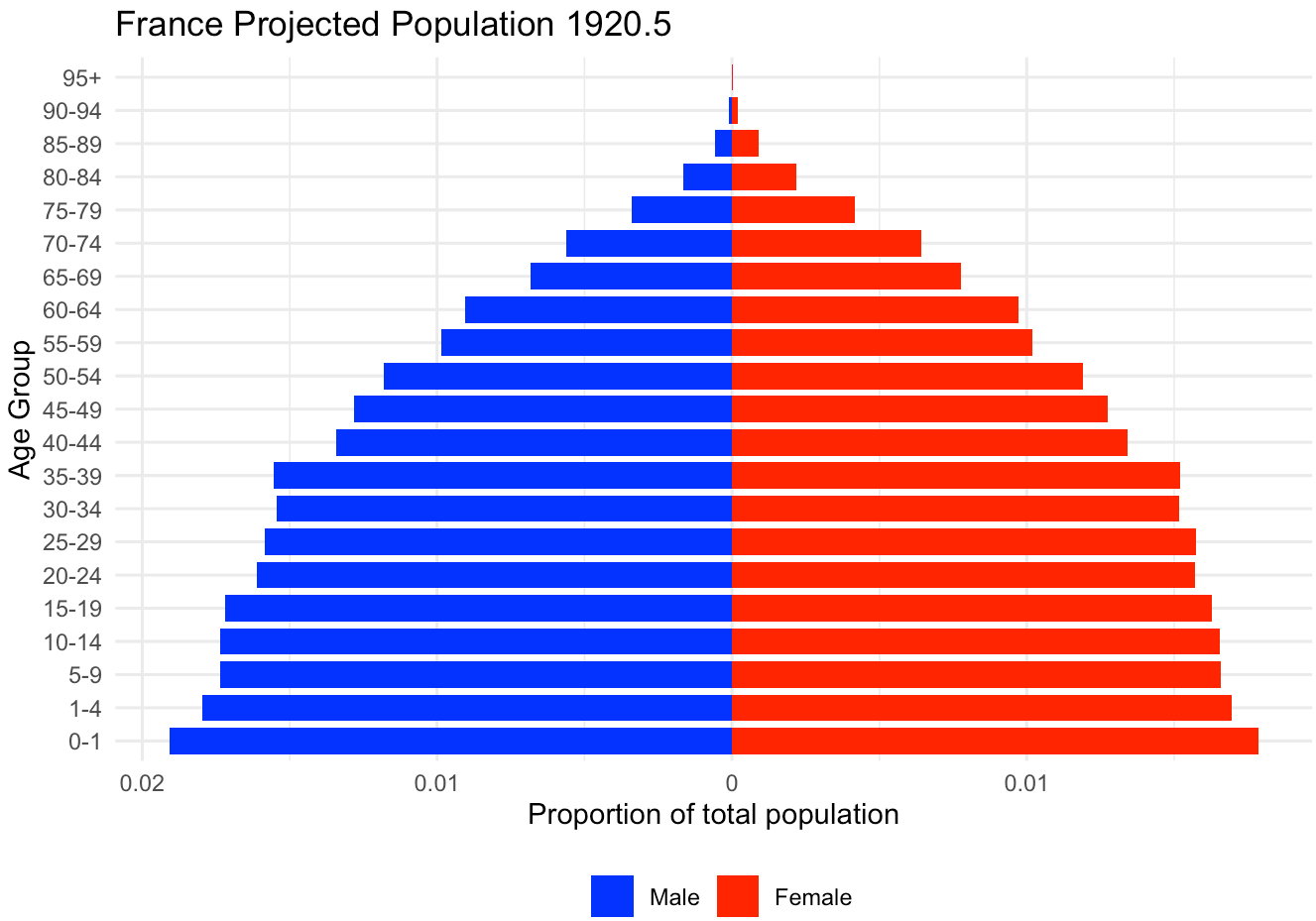
Projected Population - 1920.5:

```
# projected population for 1920.5

pop <- data.frame(pop)
projected_pop_1920.5 <- pop

# remove non-relevant column years
projected_pop_1920.5 <- projected_pop_1920.5 %>%
  select(-c(nNx_m1910.5, nNx_f1910.5, nNx_m1915.5, nNx_f1915.5))

# Convert male population to negative values
projected_pop_1920.5 <- projected_pop_1920.5 %>%
  mutate(nNx_m1920.5 = -nNx_m1920.5)
```

```
# create an object projected_pop_long that converts projected_pop_1920.5 to long format
projected_pop_long <- projected_pop_1920.5 %>%
  pivot_longer(cols = c(nNx_m1920.5, nNx_f1920.5),
               names_to = "Sex",
               values_to = "Population")

# Recode Sex labels
projected_pop_long$Sex <- recode(projected_pop_long$Sex,
                                 nNx_m1920.5 = "Male",
                                 nNx_f1920.5 = "Female")

# add a column to specify the percentage of pop instead of absolute value
projected_pop_long <- projected_pop_long %>%
  group_by(Sex) %>%  # Ensure we calculate proportions separately for each sex
  mutate(Proportion = Population / sum(abs(Population))) %>%
  ungroup()  # Remove grouping after calculation
```

Plot a population pyramid for just the projected population in 1920.5:

```
# Transform data frame into age groups and not single year age-intervals
# to make more interpretable on a plot
projected_pop_long <- projected_pop_long %>%
  mutate(Age_Group = cut(age,
         breaks = c(0, 1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80,
         labels = c("0-1", "1-4", "5-9", "10-14", "15-19", "20-24","25-29", "30-34", "35-
         right = FALSE))  # Ensures left-inclusive intervals

# Plot
ggplot(projected_pop_long, aes(x = Proportion, y = Age_Group, fill = Sex)) +
          geom_col(width = 0.8, position = "identity") +  # Use geom_col() instead of geo
          scale_fill_manual(values = c("Male" = "blue", "Female" = "red"), breaks = c("Ma
          scale_x_continuous(labels = abs) +  # Keep x-axis values positive
          labs(title = "France Projected Population 1920.5",
               x = "Proportion of total population",
               y = "Age Group",
               fill = "") +
          theme_minimal() +
          theme(legend.position = "bottom")
```



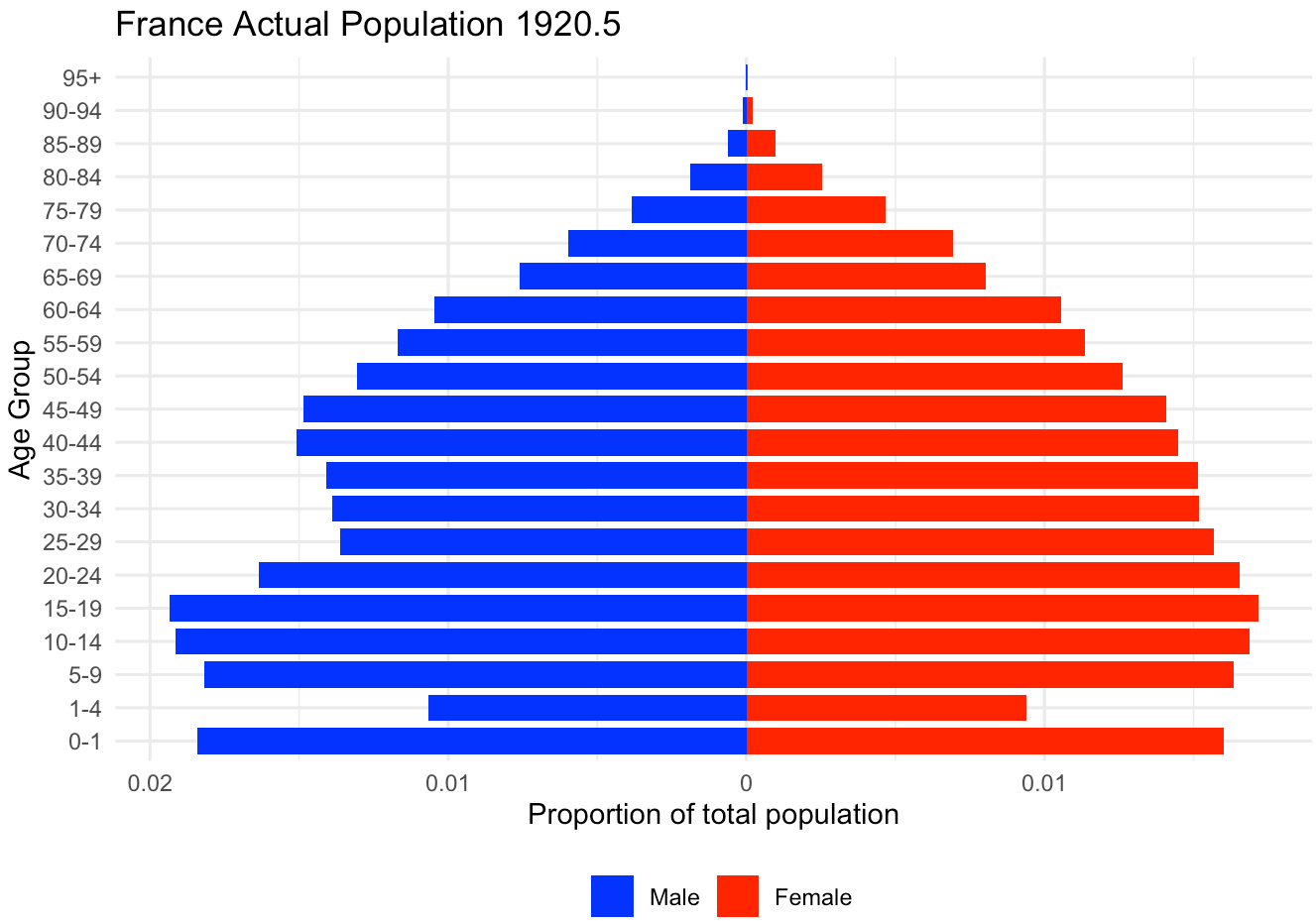
Actual Population - 1920.5:

```
# transform France_WWI into a data frame to work with
France_WWI <- data.frame(France_WWI)
actual_pop_1920.5 <- France_WWI

# remove non-relevant column years
actual_pop_1920.5 <- actual_pop_1920.5 %>%
  select(-c(nNx_m, nNx_f, nLx_m, nLx_f, nFx))

# Convert male population to negative values
actual_pop_1920.5 <- actual_pop_1920.5 %>%
  mutate(nNx_m1920.5 = -nNx_m1920.5)

# create an object call actual_pop_long that converts to long format
actual_pop_long <- actual_pop_1920.5 %>%
  pivot_longer(cols = c(nNx_m1920.5, nNx_f1920.5),
               names_to = "Sex",
               values_to = "Population")

# Recode Sex labels
actual_pop_long$Sex <- recode(actual_pop_long$Sex,
                              nNx_m1920.5 = "Male",
                              nNx_f1920.5 = "Female")
```

```r
# add a column to specify the percentage of pop instead of absolute value
actual_pop_long <- actual_pop_long %>%
  group_by(Sex) %>%  # Ensure we calculate proportions separately for each sex
  mutate(Proportion = Population / sum(abs(Population))) %>%
  ungroup()  # Remove grouping after calculation
```

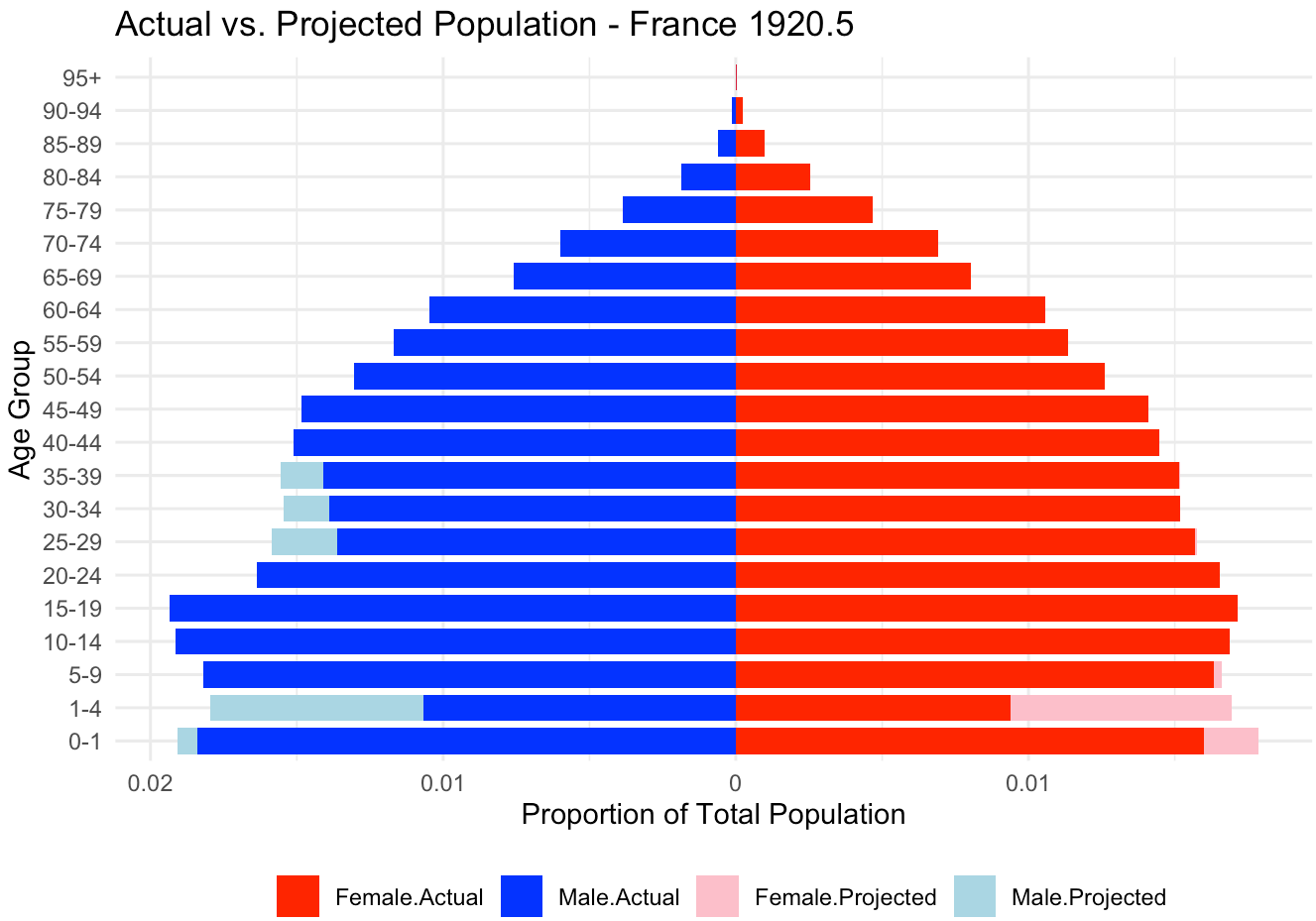
Plot a population pyramid for just the actual population in 1920.5:

```r
# Transform data frame into age groups and not single year age-intervals
actual_pop_long <- actual_pop_long %>%
  mutate(Age_Group = cut(age,
         breaks = c(0, 1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80,
         labels = c("0-1", "1-4", "5-9", "10-14", "15-19", "20-24","25-29", "30-34", "35-
         right = FALSE))  # Ensures left-inclusive intervals


# Plot
ggplot(actual_pop_long, aes(x = Proportion, y = Age_Group, fill = Sex)) +
          geom_col(width = 0.8, position = "identity") +  # Use geom_col() instead of geo
          scale_fill_manual(values = c("Male" = "blue", "Female" = "red"), breaks = c("Ma
          scale_x_continuous(labels = abs) +  # Keep x-axis values positive
          labs(title = "France Actual Population 1920.5",
               x = "Proportion of total population",
               y = "Age Group",
               fill = "") +
          theme_minimal() +
          theme(legend.position = "bottom")
```



Now, overlap the two population pyramids to identify where differences exist:

```
# combined population pyramids - actual vs. projected
# Merge actual and projected data
    combined_pop_long <- bind_rows(
      projected_pop_long %>% mutate(Type = "Projected"),
      actual_pop_long %>% mutate(Type = "Actual")
    )


# Create the overlapping population pyramids
ggplot(combined_pop_long, aes(x = Proportion, y = Age_Group, fill = interaction(Sex, Type
  geom_col(width = 0.8, position = "identity") +
  scale_fill_manual(values = c("Male.Actual" = "blue", "Male.Projected" = "lightblue",
                               "Female.Actual" = "red", "Female.Projected" = "pink")) +
  scale_x_continuous(labels = abs) +
  labs(title = "Actual vs. Projected Population - France 1920.5",
       x = "Proportion of Total Population",
       y = "Age Group",
       fill = "") +
  theme_minimal() +
  theme(legend.position = "bottom")
```



7. Identify the age groups that were mostly affected during the Great War

Comparing the two population pyramids, it appears that males ages 25 - 40 were severely affected during the Great War. This intuitively makes sense as more men of these age groups were likely fighting during the Great War, which started in 1914 and went until 1919. We can see that this pattern is not noticed with females.

Looking at the plot, it is clear to see that the age group 1-4 was also heavily impacted from the Great War. This makes sense as it is understandable that at this time of crisis, many families are not having as many children.

The difference between the projected population and the actual population for males in this age interval is 630,017. For females, the difference between the projected population and the actual population is 618,130. This is for a total population difference of 1,248,147, which is quite the difference (see R code below for how results were obtained)

```
# Summing the male population in the 1-4 age group - projected
male_pop_age_group_1_4_projected <- projected_pop_long %>%
  filter(Sex == "Male", Age_Group == "1-4") %>%
  summarise(total_population = abs(sum(Population, na.rm = TRUE)))
```

```r
# Summing the male population in the 1-4 age group - actual
male_pop_age_group_1_4_actual <- actual_pop_long %>%
  filter(Sex == "Male", Age_Group == "1-4") %>%
  summarise(total_population = abs(sum(Population, na.rm = TRUE)))

# Difference between the two - males
male_pop_age_group_1_4_projected - male_pop_age_group_1_4_actual
```

```
  total_population
1         630017.3
```

```r
# Summing the female population in the 1-4 age group - projected
female_pop_age_group_1_4_projected <- projected_pop_long %>%
  filter(Sex == "Female", Age_Group == "1-4") %>%
  summarise(total_population = abs(sum(Population, na.rm = TRUE)))

# Summing the male population in the 1-4 age group - actual
female_pop_age_group_1_4_actual <- actual_pop_long %>%
  filter(Sex == "Female", Age_Group == "1-4") %>%
  summarise(total_population = abs(sum(Population, na.rm = TRUE)))

# Difference between the two - females
female_pop_age_group_1_4_projected - female_pop_age_group_1_4_actual
```

```
  total_population
1           618130
```

```r
# Total difference
(female_pop_age_group_1_4_projected + male_pop_age_group_1_4_projected) - (female_pop_age
```

```
  total_population
1          1248147
```

References:

Aburto, J.M., Villavicencio, F., Basellini, U., Kjærgaard, S., & Vaupel, J.W. (2020). Dynamics of life expectancy and life span equality. *Proceedings of the National Academy of Sciences of the United States of America*, *117*(10), 5250–5259.

Aburto, J.M. (2025) “Alternative measures of longevity” [Lecture] 2429 Population Dynamics & Projections. London School of Hygiene & Tropical Medicine. 07 March.

Aburto, J.M. (2025) “Mortality forecasting: Lee-Carter (based on Ugo Basellini’s notes)” [Lecture] 2429 Population Dynamics & Projections. London School of Hygiene & Tropical Medicine. 14 March.

Statement to acknowledge AI use:

I used ChatGPT (OpenAI, 2025) to assist with the debugging of R code as specified on the cover sheet form.