Demographic Methods - Practical 11 (Migration)

2025-11-24

Exercise 1: Survival Method

The heading of the R script

The R script begins by clearing the workspace. It requires the data.table package for high-performance data manipulation—please ensure this packages are installed.

```
rm(list = ls())
#install.packages("data.table")
#install.packages("tidyverse")
#install.packages("ggplot2")
library(data.table)
library(tidyr)
library(ggplot2)
```

The data

In this exercise, we are given census enumerations by 5-year age groups (${}_nN_x$) for the female population of England and Wales in 1891 and 1901 (in thousands), as well as the number of person-years lived in each age interval (${}_nL_x$) from a life table representing the mortality conditions between the two censuses.

```
x                   = seq(5, 35, by = 5)
n                   = rep(5,7)
nNx_1891            = c(1702, 1613, 1486, 1399, 1239, 1049, 916)
nNx_1901            = c(1748, 1671, 1639, 1648, 1496, 1274, 1111)
nLx                 = c(3885, 3812, 3747, 3665, 3564, 3445, 3255)
data                = data.table(x, n, nLx, nNx_1891, nNx_1901)
as.data.frame(data)
```

```
##    x n  nLx nNx_1891 nNx_1901
## 1  5 5 3885     1702     1748
## 2 10 5 3812     1613     1671
## 3 15 5 3747     1486     1639
## 4 20 5 3665     1399     1648
## 5 25 5 3564     1239     1496
## 6 30 5 3445     1049     1274
## 7 35 5 3255      916     1111
```

One method to quantify migration is to use the balancing equation to calculate the number of net migrants indirectly, based on population counts at two points in time and the number of deaths between them. A life table can serve as a substitute for the number of deaths by providing survivorship ratios (i.e., ${}_nL_{x+n}/{}_nL_x$). Hence, the number of net migrants in a cohort can be calculated as the difference between the observed population in a census and the projected population of the same cohort from a previous census when it was younger, as shown:

$$ {}_nNM_{x+n}(t+n) = {}_nN_{x+n}(t+n) - {}_nN_x(t) \cdot \frac{{}_nL_{x+n}}{{}_nL_x} \tag{1} $$

Using a backward projection, the number of net migrants of the same cohort at the exact time t can be also calculated as:

$$ {}_nNM_x(t) = {}_nN_{x+n}(t+n) \cdot \frac{{}_nL_x}{{}_nL_{x+n}} - {}_nN_x(t) \tag{2} $$

Example Questions:

a. Using the ${}_5L_x$ values, calculate survivorship ratios (${}_5S_x$) for a five-year period.

Hint: this is a straightforward calculation, considering the five-year length of the age intervals.
${}_5S_x = \frac{{}_5L_{x+5}}{{}_5L_x}$

```
nSx                 = tail(data[["nLx"]],-1)/head(data[["nLx"]],-1)
x                   = seq(5, 30, by = 5)
n                   = rep(5,6)
S5                  = data.table(x, n, nSx)
print(data.frame(S5), row.names = FALSE)
```

```
##   x n       nSx
##   5 5 0.9812098
##  10 5 0.9829486
##  15 5 0.9781158
##  20 5 0.9724420
##  25 5 0.9666105
##  30 5 0.9448476
```

b. Calculate survivorship ratios (${}_{10}S_x$) for a 10-year period.

Hint: this can be approached as the ratio of survival across two consecutive age intervals of five years each.
${}_{10}S_x = \frac{{}_5L_{x+10}}{{}_5L_{x+5}} \cdot \frac{{}_5L_{x+5}}{{}_5L_x} = \frac{{}_5L_{x+10}}{{}_5L_x}$

```
nSx                 = tail(data[["nLx"]],-2)/head(data[["nLx"]],-2)
x                   = seq(5, 25, by = 5)
n                   = rep(10,5)
S10                 = data.table(x, n, nSx)
print(data.frame(S10), row.names = FALSE)
```

```
##   x  n       nSx
##   5 10 0.9644788
##  10 10 0.9614376
##  15 10 0.9511609
##  20 10 0.9399727
##  25 10 0.9132997
```

c. Using the survivorship ratios and the population in 1891, calculate the number of survivors after five years of exposure to the risk of dying (i.e., the expected population in 1896).

Hint: Mind the ages. The population from 5 to 9 in 1896 cannot be calculated, as the population from 0 to 4 five years earlier was not available for this exercise.
${}_5N_{x+5} = \frac{{}_5L_{x+5}}{{}_5L_x} \cdot {}_5N_x$

```
data[["nNx_1896p"]]  =  c(NA,S5[["nSx"]]*head(data[["nNx_1891"]],-1))
print(data.frame(data[, c("x", "n", "nNx_1896p")]), row.names = FALSE)
```

```
##   x n nNx_1896p
```

```
##   5 5        NA
##  10 5 1670.0190
##  15 5 1585.4961
##  20 5 1453.4801
##  25 5 1360.4464
##  30 5 1197.6305
##  35 5  991.1451
```

Congratulations! If you have reached this point, you have learned how to project a population five years forward using a life table with five-year age intervals. This topic will be revisited in *Population Dynamics and Projections*.

Compulsory Questions:

a. Using the survivorship ratios and the population in 1891, calculate the number of survivors after 10 years of exposure to the risk of dying (i.e., the expected population in 1909).

Hint: Mind the ages. The population from 5 to 14 in 1901 cannot be calculated, as the number of births and population from 0 to 4, 10 years earlier was not available for this exercise. Report your results using the function "print()".

${}_5N_{x+10} = \frac{{}_5L_{x+10}}{{}_5L_x} \cdot {}_5N_x$

b. Using the result of the previous question and the population in 1901, calculate the estimated number of (net) migrants in 1901.

Hint: Calculate the difference between observed and projected population in 1901.

c. Why are these values treated as estimates? Mention one key assumption.

d. Calculate the total sum of (net) migrants aged 15 to 39 in 1901.

Hint: Sum all estimates of net migration, either by collapsing the table or by using the "sum()" function.

e. Use the survivorship ratios to project the 1901 population 10 years backward.

Hint: Divide the population by the survivorship ratios to project the population backward.

Exercise 2: Migration Expectations (Application of Sullivan's Method to Migration Data)

The data

In this exercise, we are given age-specific, one-year mobility data for the United States—excluding immigration from abroad—for the period 2020–2021. The dataset includes mid-year population totals and three types of relocation (in thousands): within the same county, within the same state but a different county (intra-state), and between different states (inter-state); Source: U.S. Census Bureau, *Current Population Survey: 2012 Annual Social and Economic Supplement*, https://www.census.gov/data/tables/2021/demo/geographic-mobility/cps-2021.html. To quantify lifetime exposure, we use the l_x and ${}_na_x$ columns from the 2020 U.S. life table for both sexes combined as input, which also reports a life expectancy of 6.106 years at age 85 and above; Source: *National Vital Statistics Reports: United States Life Tables, 2020*, https://stacks.cdc.gov/view/cdc/118055/cdc_118055_DS1.pdf. For ease of access, the file has been temporarily stored in a GitHub repository and can be retrieved using the following lines of code. No additional data preparation is required.

```
GitHub                = "https://raw.githubusercontent.com/Romero-Prieto/teaching/main/Demographic%20Metl
data                  = read.csv(GitHub)
data                  = as.data.table(data)
e85                   = 6.106
print(data.frame(data), row.names = FALSE)
```

```
##   x population within_counties within_states between_states       lx     nax
##   0         NA              NA            NA             NA 100000.0 0.12903
##   1      15504            1168           408            267  99460.6 0.41357
##   5      20194            1015           375            269  99377.3 0.46679
##  10      20651             884           249            261  99323.1 0.61840
##  15      12839             525           192            119  99241.6 0.54687
##  18       7816             470           195            109  99098.3 0.51705
##  20      21022            2158          1025            592  98951.7 0.53045
##  25      22592            2514           879            563  98414.8 0.51776
##  30      22691            1760           678            522  97725.1 0.51702
##  35      21607            1198           523            343  96855.7 0.51535
##  40      20397             897           325            267  95793.7 0.52079
##  45      19406             752           241            214  94470.9 0.52687
##  50      20273             694           262            204  92680.5 0.53153
##  55      20682             457           327            147  90114.9 0.52840
##  60       8677             189           144             67  86375.5 0.50829
##  62      12417             327           136            110  84469.9 0.51076
##  65      18162             273           133            141  81181.3 0.52171
##  70      14878             198            77             87  74465.9 0.52451
##  75      10172             162            38             23  65564.9 0.52443
##  80       6632              68            26             32  53346.0 0.51624
##  85       5992              95            36             37  37700.2      NA
```

Example Questions:

a. Using the columns l_x and ${}_na_x$, calculate the number of person-years lived in the age interval (${}_nL_x$).

Hint: Reuse code from previous lectures.

```
x                     = data[["x"]]
lx                    = data[["lx"]]
nax                   = data[["nax"]]
n                     = c(diff(x,1),NA)
sEL                   = !is.na(n)
ndx                   = c(-diff(lx,1),lx[!sEL])
nLx                   = n*(lx - ndx) + n*nax*ndx
nLx[!sEL]             = e85*lx[!sEL]
print(data.frame(x, n, nax, lx, ndx, nLx), row.names = FALSE)
```

```
##   x  n     nax       lx     ndx      nLx
##   0  1 0.12903 100000.0   539.4  99530.2
##   1  4 0.41357  99460.6    83.3 397647.0
##   5  5 0.46679  99377.3    54.2 496742.0
##  10  5 0.61840  99323.1    81.5 496460.0
##  15  3 0.54687  99241.6   143.3 297530.0
##  18  2 0.51705  99098.3   146.6 198055.0
##  20  5 0.53045  98951.7   536.9 493498.0
```

```
##  25  5 0.51776  98414.8   689.7 490411.0
##  30  5 0.51702  97725.1   869.4 486526.0
##  35  5 0.51535  96855.7  1062.0 481705.0
##  40  5 0.52079  95793.7  1322.8 475799.0
##  45  5 0.52687  94470.9  1790.4 468119.0
##  50  5 0.53153  92680.5  2565.6 457393.0
##  55  5 0.52840  90114.9  3739.4 441757.0
##  60  2 0.50829  86375.5  1905.6 170877.0
##  62  3 0.51076  84469.9  3288.6 248583.0
##  65  5 0.52171  81181.3  6715.4 389847.0
##  70  5 0.52451  74465.9  8901.0 351167.8
##  75  5 0.52443  65564.9 12218.9 298769.8
##  80  5 0.51624  53346.0 15645.8 228885.9
##  85 NA      NA  37700.2 37700.2 230197.4
```

b. Calculate the number of person-years lived above age x (T_x) and the life expectancy (e_x), to complete a life table.

```
Tx                  = rev(cumsum(rev(nLx)))
ex                  = Tx/lx
print(data.frame(x, n, nax, lx, ndx, nLx,Tx, ex), row.names = FALSE)
```

```
##   x  n     nax       lx     ndx      nLx        Tx        ex
##   0  1 0.12903 100000.0   539.4  99530.2 7699500.1 76.995001
##   1  4 0.41357  99460.6    83.3 397647.0 7599969.9 76.411864
##   5  5 0.46679  99377.3    54.2 496742.0 7202322.9 72.474527
##  10  5 0.61840  99323.1    81.5 496460.0 6705580.9 67.512803
##  15  3 0.54687  99241.6   143.3 297530.0 6209120.9 62.565707
##  18  2 0.51705  99098.3   146.6 198055.0 5911590.9 59.653807
##  20  5 0.53045  98951.7   536.9 493498.0 5713535.9 57.740654
##  25  5 0.51776  98414.8   689.7 490411.0 5220037.9 53.041188
##  30  5 0.51702  97725.1   869.4 486526.0 4729626.9 48.397258
##  35  5 0.51535  96855.7  1062.0 481705.0 4243100.9 43.808479
##  40  5 0.52079  95793.7  1322.8 475799.0 3761395.9 39.265587
##  45  5 0.52687  94470.9  1790.4 468119.0 3285596.9 34.778931
##  50  5 0.53153  92680.5  2565.6 457393.0 2817477.8 30.399899
##  55  5 0.52840  90114.9  3739.4 441757.0 2360084.9 26.189730
##  60  2 0.50829  86375.5  1905.6 170877.0 1918327.9 22.209167
##  62  3 0.51076  84469.9  3288.6 248583.0 1747450.9 20.687261
##  65  5 0.52171  81181.3  6715.4 389847.0 1498867.9 18.463217
##  70  5 0.52451  74465.9  8901.0 351167.8 1109021.0 14.893004
##  75  5 0.52443  65564.9 12218.9 298769.8  757853.1 11.558824
##  80  5 0.51624  53346.0 15645.8 228885.9  459083.4  8.605769
##  85 NA      NA  37700.2 37700.2 230197.4  230197.4  6.106000
```

Compulsory Questions:

a. If you aim to collect similar data for a different country, what question would you add to a population census or survey?

b. Why are mobility data missing for individuals less than one year old? How are mobility data collected for children who can be included in a survey or census but cannot be actually interviewed?

c. In your opinion, what would be the reason for reporting mobility data for ages 15–19 and 60–64 using irregular breakdowns?

d. Which types of migratory movements are missing from these data?

e. Calculate the migration rates.

Hint: As usual, age-specific migration rates can be calculated by dividing the number of moves during the past year by the mid-year population. The reported moves are mutually exclusive; therefore, the total number of internal moves is equal to the sum of all categories. Create a subsidiary table of migratory movements and divide it by the mid-year population to estimate rates. Define the total migration rate as the sum of all categories. Report your results using the function "print()".

f. Calculate the lifetime migration expectancy at age 1 for migratory movements within counties, within states, between states, and for all internal migration movements. Interpret your results.

Hint: Following Sullivan's method, multiply the table of migration rates by the number of person-years lived within the age interval (${}_nL_x$) to estimate the person-years lived while changing place of residence. Then, calculate the corresponding number of person-years lived above age x and the lifetime expectancy associated with each category of internal migration. To do so, reuse code from previous lectures to calculate T_x as the reverse cumulative sum of multiple columns, each representing a category of internal migration.

Optional Questions:

a. An expert suggests that 2020–21 is a problematic period for analyzing migration due to the COVID-19 pandemic and the resulting lockdowns and mobility restrictions. Propose a research design to test this hypothesis, quantifying the effect of the pandemic on lifetime migration expectancy (use 250 words or fewer).

b. Plot the (internal) migration rates.

Exercise 3: Researching Migration (Optional)

The **Living Standards and Measurement Surveys** (LSMS) have been conducted by the World Bank since 1980 in order to improve data collection systems in low- and middle-income countries. The first Ghana LSMS was undertaken in 1988-89 and a sample of about 3,200 households was randomly selected. The survey included the following 10 questions on migration that are posed to all household members 7 years of age and older:

1. Were you born in [PRESENT PLACE OF RESIDENCE]?
2. Have you ever lived anywhere else?
3. At the time of your birth, was your birthplace a city, large town, medium town, small town, large village, small village, or other?

4. How old were you when you left your place of birth for the first time to live somewhere else? (recorded in years)
5. What was the main reason you left? a. To follow or join family b. Work related c. Marriage d. School e. Adventure f. Escape family problems g. Other
6. How long have you lived in [PRESENT PLACE OF RESIDENCE] since your last move? (recorded in years or months if less than 1 year)
7. What was the main reason you came to [PRESENT PLACE OF RESIDENCE]? a. To follow or join family b. Work related c. Marriage d. School e. Adventure f. Escape family problems g. Other
8. From which region of the country were you coming from? (including somewhere else in Africa or somewhere outside of Africa)
9. Was the place you were living before coming here a city, large town, medium town, small town, large village, small village, or other?
10. How many different places have you lived in for periods of more than 3 months in your life?

Comment on this migration module in the LSMS. What is good about it, what is not so good? What would you do differently?

Appendix

The R Demographer's Corner: Syntax Examples for Fast Calculations

Let's define x as a sequence of numbers from 0 to 9, in increments of 1, as follows:

```
x                    = seq(0, 9, 1)
x
```

```
##  [1] 0 1 2 3 4 5 6 7 8 9
```

and r as a vector of the same dimension, indicating a constant value of 3.

```
r                    = rep(3, length(x))
r
```

```
##  [1] 3 3 3 3 3 3 3 3 3 3
```

The first difference of x is computed using the function "diff(*vector*, *order*)", as given below:

```
diff(x, 1)
```

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

The sequence can also be reversed using:

```
rev(x)
```

```
##  [1] 9 8 7 6 5 4 3 2 1 0
```

The function "head(x, k)" returns the first k elements of x, for example the first two:

```
head(x, 2)
```

```
## [1] 0 1
```

When using a negative number, head(x, -k) returns all elements except the last two, as shown below:

```
head(x, -2)
```

```
## [1] 0 1 2 3 4 5 6 7
```

The function "tail(x, k)" is the opposite of "head(x, k)", and returns the last k elements of x, for example the last three:

```
tail(x, 3)
```

```
## [1] 7 8 9
```

Similarly, when using a negative number, tail(x, -k) returns all elements except the first three, as follows:

```
tail(x, -3)
```

```
## [1] 3 4 5 6 7 8 9
```

The function "c($object_1$,.., $object_n$)", consolidates n individual elements (either numbers or strings) into a single vector:"

```
c(2, 5, 4, 7)
```

```
## [1] 2 5 4 7
```

```
c("a", "b", "c")
```

```
## [1] "a" "b" "c"
```

The function "cumsum()" returns the cumulative sum of a vector:

```
cumsum(r)
```

```
##  [1]  3  6  9 12 15 18 21 24 27 30
```

In the case of a table, consisting of multiple vectors, the function "rowSums()" returns the sum by rows:

```
rowSums(data.frame(x, r))
```

```
##  [1]  3  4  5  6  7  8  9 10 11 12
```

Similarly, the function "colSums()" returns the sum by columns:

```
colSums(data.frame(x, r))
```

```
##  x  r
## 45 30
```

The function "apply(*object*, *dimension*, *function*)" executes a specified function across a set of rows or columns (i.e., dimension 1 or 2) of a given table-like object, for example, the cumulative sum by columns of a table consisting of x and r.

```
apply(data.frame(x, r), 2, cumsum)
```

```
##        x  r
##  [1,]  0  3
##  [2,]  1  6
##  [3,]  3  9
##  [4,]  6 12
##  [5,] 10 15
##  [6,] 15 18
##  [7,] 21 21
##  [8,] 28 24
##  [9,] 36 27
## [10,] 45 30
```

The syntax "df %>% pivot_longer(cols = c(var_1, ..., var_n), names_to ="name of variables", values_to = "name of values")" can be used to transform **wide form** to **long form** data. For example, using as an input a data frame consisting of x and two other variables y, z (defined as 10 random numbers from 1 to 25):

```
y                    = sample(1:25, length(x))
z                    = sample(1:25, length(x))
df                   = data.frame(x, y, z)
df
```

```
##    x  y  z
## 1  0  2  6
## 2  1 15 22
## 3  2  4 19
## 4  3 11 15
## 5  4 22 16
## 6  5  5  3
## 7  6 14  5
## 8  7  8  2
## 9  8 13 21
## 10 9 20 13
```

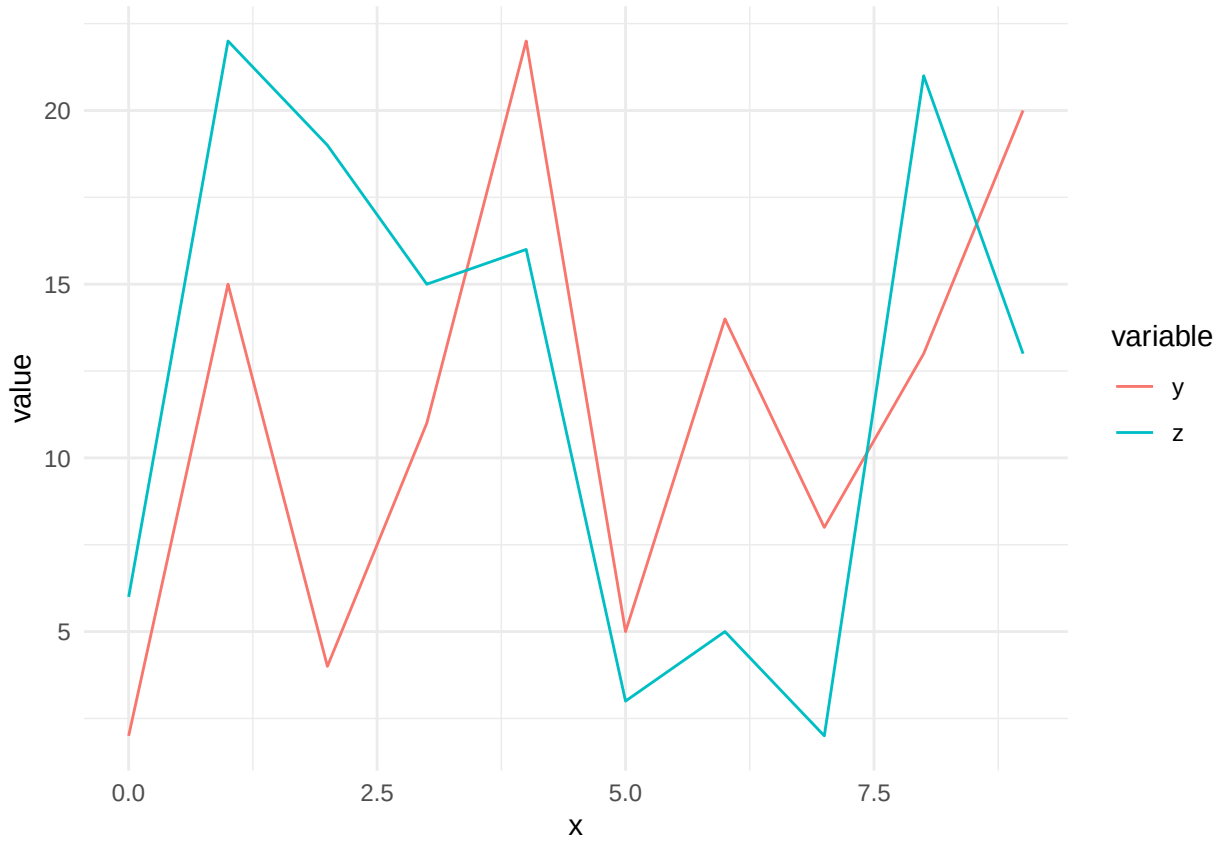
The function returns a column vector of y and z, all associated with the values of the variable x.

```
df                   = df %>% pivot_longer(cols = c(y, z),
                                           names_to = "variable",
                                           values_to = "value")
print(as.data.frame(df), row.names = FALSE)
```

```
##  x variable value
##  0        y     2
##  0        z     6
##  1        y    15
##  1        z    22
##  2        y     4
##  2        z    19
##  3        y    11
##  3        z    15
##  4        y    22
##  4        z    16
##  5        y     5
##  5        z     3
##  6        y    14
##  6        z     5
##  7        y     8
##  7        z     2
##  8        y    13
##  8        z    21
##  9        y    20
##  9        z    13
```

Lastly, the function "ggplot(*data frame*, aes(x = *Abscissa or range*, y = *Ordinate or response*, color = *pivot*)) + geom_line() + theme_minimal()" plots multiple lines using a data frame as input.

```
ggplot(df, aes(x = x, y = value, color = variable)) +
  geom_line() + theme_minimal()
```


value
20
15
10
5
0.0
2.5
5.0
7.5
x
variable
y
z