KOSTAT-UNFPA Summer Seminar on Population

Workshop 1. Introduction to Demography

Day 2: Mortality and Fertility

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

Assistant:
Inchan Hwang `inchanhwang@utexas.edu`

24 June 2025

Contents

1 Mortality

Mortality sets a fundamental constraint on population well-being by defining a longevity envelope within which all life happens. Mortality levels vary over age by orders of magnitude, and can also vary between populations. Demography delivers tools to understand mortality levels in terms of metrics in different units, and to adjust these metrics to be able to make valid comparisons between populations. The lifetable is the basic analytic tool to allow for valid and comparable summary metrics at the population level.

1.1 Lifetable transformations as functions

I describe the lifetable together with function-writing because lifetable transformations give simple practice in functionalizing mathematical formulas. When function-writing, it is desirable to work with test data handy. For this, go ahead and load the`KOR2014` file from yesterday like so:

```
library(tidyverse)
KOR2014 <- read_csv("Data/KOR2014.csv",
                    show_col_types = FALSE)
```

We will take what we need from this file as we go.

1.2 Death probabilities between age x and $x + n$ ${}_nq_x$

The first and key step is to transform a set of age-specific death rates into a set of age-specific probabilities of dying (${}_nq_x$). The relationship between ${}_nM_x$ and ${}_nq_x$ has been established based on analyses of actual cohorts (for mathematical proof, see Preston, Heuveline, and Guillot (2001), p. 42-43).

$$ {}_nq_x = \frac{n \cdot {}_nM_x}{1 + (n - {}_nA_x) \cdot {}_nM_x} $$

where ${}_nA_x$ is the average number of person-years lived in the interval by those dying in the interval and n is the width of the age-interval.

For single ages or when we're pragmatic, we define ${}_nA_x = n/2$ with the exceptions of the first and the last age group. Other approximations are also available, but these only matter when age groups are wider than a year. In our case, we're working with abridged lifetables, some of which represent high mortality settings, and the ${}_nA_x$ assumptions are consequential. In our case, I provide this value so that we don't need to work so hard at deriving it in class. You could find several popular ${}_nA_x$ approximations in the `DemoTools` package Riffe et al. (2021).

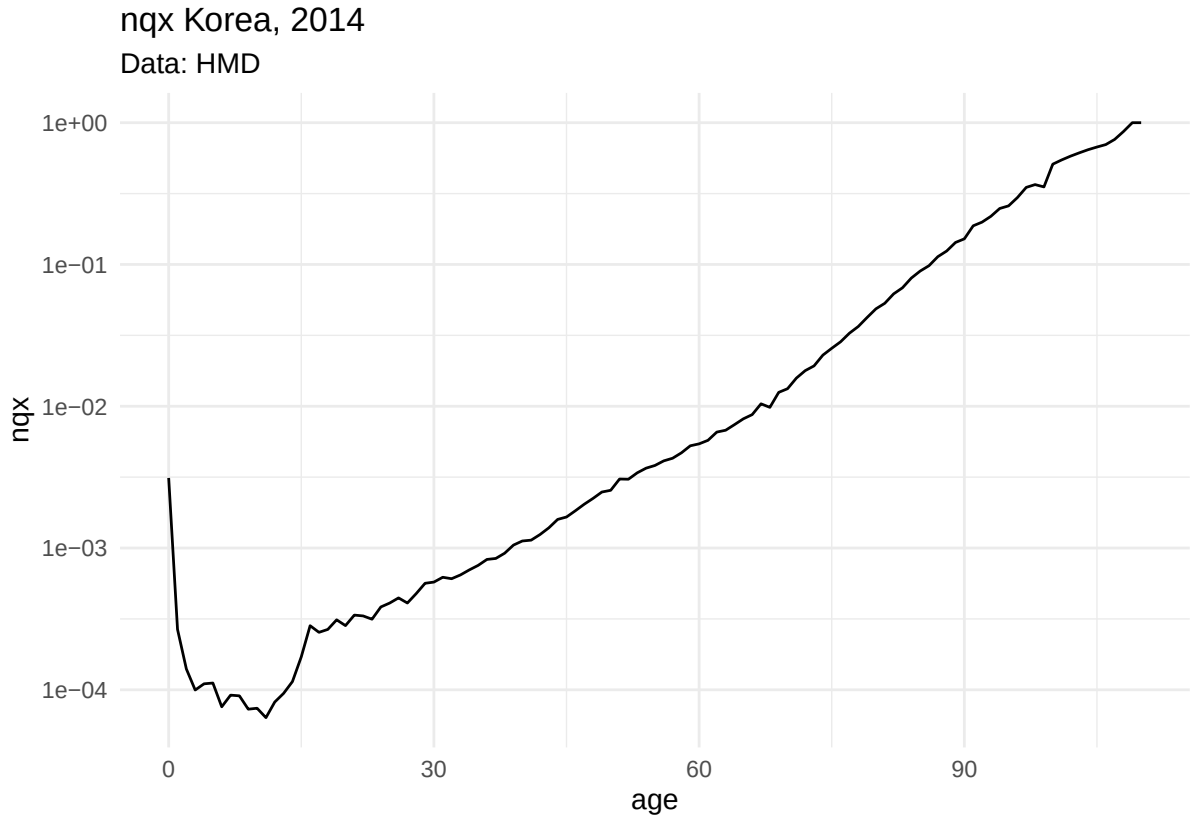
Getting down to business, we can rather directly convert the ${}_nq_x$ formula to an `R` function:

```
calc_nqx <- function(nMx, nAx, n){
  qx         <- (n * nMx) / (1 - (n - nAx) * nMx)

  # these are kludges, necessary to ensure nqx results as a probability
  qx[qx > 1] <- 1
  qx[qx < 0] <- 0
  qx
}
```

Here's how we can use this function in tidyverse syntax and plot the result:

```
# Example of what nqx looks like over age
KOR2014 |>
  filter(sex == "total") |>
  mutate(nMx = deaths / exposure,
         nAx = if_else(age == 0, .1, .5),
         n = 1,
         nqx = calc_nqx(nMx = nMx, nAx = nAx, n = n)) |>
  ggplot(aes(x = age, y = nqx)) +
  geom_line() +
  scale_y_log10() +
  labs(title = "nqx Korea, 2014",
       subtitle = "Data: HMD") +
  theme_minimal()
```



Often we're sure to *close out* the lifetable by making the *final* ${}_nq_x$ value equal to 1. You could optionally modify the function to impute 1 like so `nqx[length(nqx)] <- 1`.

Note, this $q(x)$ formula does not necessarily need to be committed to memory. This is something we either derive or look up as needed. For our needs we are satisfied with our values of $q(x)$, $m(x)$ and $a(x)$, and the remainder of the lifetable is now determined.

1.3 Survival probabilities between age x and $x + n$, ${}_np_x$

The survival probabilities between age x and $x+n$ (${}_np_x$) is simply one minus ${}_nq_x$. It is interpreted as the chance of surviving from age x to age $x + n$.

$$ {}_np_x = 1 - {}_nq_x $$

Really there's no need to program a function for this column, as we can just use ${}_nq_x$ as the function argument and take its complement as needed.

1.4 Survival probabilities to age x, l_x

This lifetable column indicates the chance of surviving from birth to age x (l_x) OR the number of survivors at age x relative to the radix r of the life table. The $l_0 = r$ is interpreted as the initial size of the synthetic lifetable population, generally set to 1 or 100,000. Think of this as the number of *sims* in your lifetable. Here's one of several ways to calculate it given what we have so far:

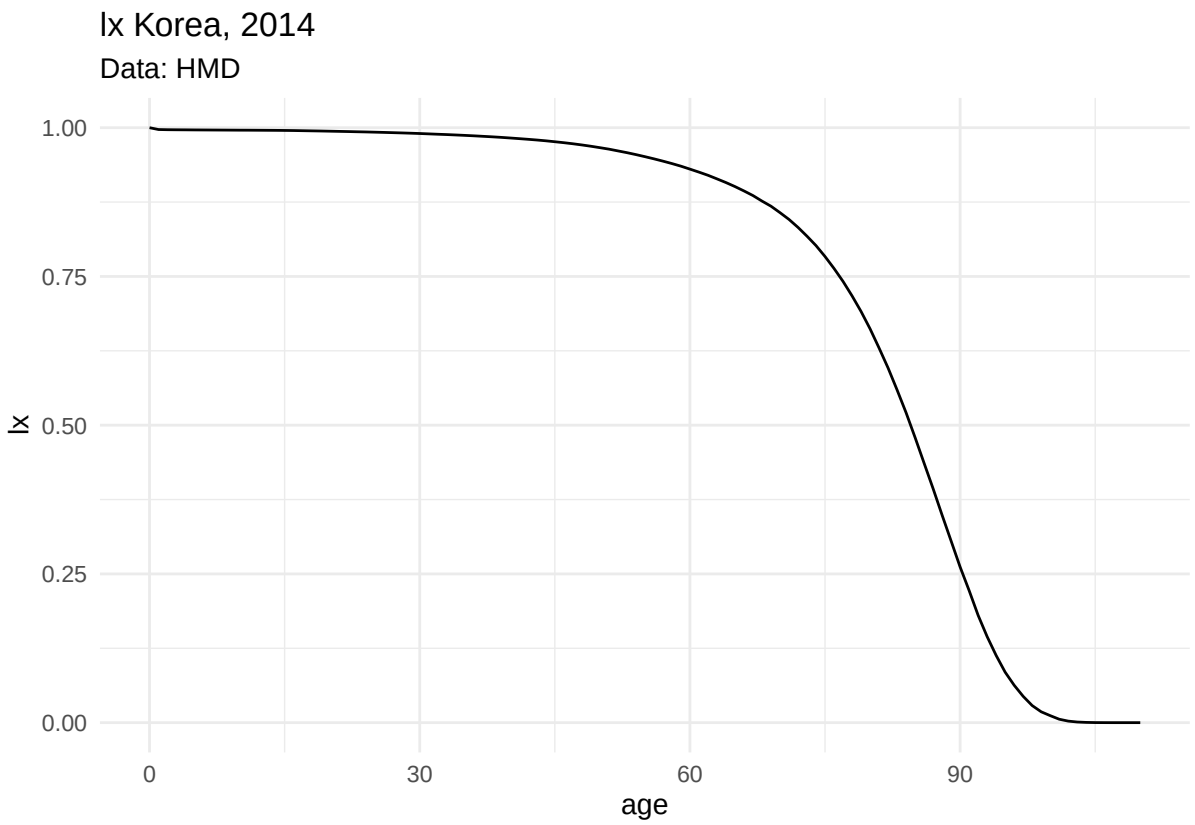
$$ l_{x+n} = r \prod_{y=0}^{x} {}_np_y $$

where $r = {}_nl_0$ is the radix. To program this, our arguments should be ${}_nq_x$ (`nqx`) or ${}_np_x$ (`npx`) and `radix`. In this case, we can assign a default value for the radix when defining the function using `radix = 1`. Whenever the argument isn't specified by the user, 1 will be assumed.

```
# nqx is a full vector over age.
calc_lx <- function(nqx, radix = 1){
  npx <- 1 - nqx
  n   <- length(nqx)
  lx  <- cumprod(npx)
  # shift it back 1, as we start with 100%!
  # also ensure outgoing vector is the same length.
  lx  <- radix * c(1, lx[-n])
  lx
}
```

And here is an application of the new function to data, as per before:

```
KOR2014 |>
  filter(sex == "total") |>
  mutate(nMx = deaths / exposure,
         nAx = if_else(age == 0, .1, .5),
         n = 1,
         nqx = calc_nqx(nMx = nMx, nAx = nAx, n = n),
         lx = calc_lx(nqx = nqx)) |>
  ggplot(aes(x = age, y = lx)) +
  geom_line() +
  ylim(0,1) +
  labs(title = "lx Korea, 2014",
       subtitle = "Data: HMD") +
  theme_minimal()
```



1.5 Death distribution, ${}_nd_x$

The life table deaths (${}_nd_x$) is the number of (synthetic) persons dying between age x and $x+n$, relative to the radix, and represents the distribution of deaths over age. There are two ways of calculating ${}_nd_x$. When programming, this is the most pragmatic way of calculating it:

$$ {}_nd_x = {}_nq_x * l_x $$

One could ask, do we really need this function? This is something we can remember, right?

```
calc_ndx <- function(nqx, lx){
   nqx * lx
}
```

1.6 Person-years lived between age x and $x+n$, ${}_nL_x$

The number of person-years between age x and $x+n$ (${}_nL_x$) is calculated as:

$$ \begin{aligned} {}_nL_x &= n(l_x - {}_nd_x) + {}_na_x \cdot {}_nd_x \\ &= n \cdot l_x - (n - {}_na_x){}_nd_x \end{aligned} $$

Note

${}_nm_x = {}_nd_x / {}_nL_x$

and

${}_nq_x = {}_nd_x / l_x$

```r
calc_nLx <- function(lx, ndx, nAx, n){
  N        <- length(lx)
  nLx      <- n[-N] * lx[-1] + nAx[-N] * ndx[-N]
  # special treatment for open age
  nLx[N]     <- lx[N] * nAx[N]
  nLx
}

# and application with no plot:
KOR2014 |>
  filter(sex == "total") |>
  mutate(nMx = deaths / exposure,
         nAx = if_else(age == 0, .1, .5),
         n = 1,
         nqx = calc_nqx(nMx = nMx, nAx = nAx, n = n),
         lx = calc_lx(nqx = nqx),
         ndx = lx * nqx,
         nLx = calc_nLx(lx = lx, ndx = ndx, nAx = nAx, n = n)) |>
  head()
```

```
## # A tibble: 6 x 13
##    year   age sex   exposure deaths births       nMx   nAx     n       nqx    lx
##   <dbl> <dbl> <chr>    <dbl>  <dbl>  <dbl>     <dbl> <dbl> <dbl>     <dbl> <dbl>
## 1  2014     0 total  419382. 1305.       0 0.00311     0.1     1 0.00312   1
## 2  2014     1 total  464824.  124.       0 0.000267    0.5     1 0.000267  0.997
## 3  2014     2 total  477676.   67.0      0 0.000140    0.5     1 0.000140  0.997
## 4  2014     3 total  481716.   48        0 0.0000996   0.5     1 0.0000996 0.996
## 5  2014     4 total  453779.   50.0      0 0.000110    0.5     1 0.000110  0.996
## 6  2014     5 total  457061    51.0      0 0.000112    0.5     1 0.000112  0.996
## # i 2 more variables: ndx <dbl>, nLx <dbl>
```

1.7 Person-years lived above age x T_x

Calculating the number person-years lived above age x (T_x) is a key step to calculate life expectancy. It consists in finding the sum of ${}_nL_x$ from age x:

$$T_x = \sum_{y=x}^{\infty} {}_nL_y$$

```r
calc_Tx <- function(nLx){
  # to understand this, look at the nLx curve,
  # then imagine integrating from the right
  # to the left. Then compare with the formula!
  nLx |>
    rev() |>
    cumsum() |>
    rev()
}
```

1.8 Life expectancy e_x

The last indicator in the life table is probably one of the most used in demographic analysis. The life expectancy is the average number of years lived by a (synthetic) cohort reaching age x. It consists in dividing the number of person-years lived above age x by the number of people alive at age x:

$$e_x = \frac{T_x}{l_x}$$

Since `mutate()` let's you make columns in a sequentially dependent way, we can actually do this whole lifetable inside a single `mutate()` statement. However, each combination of `Year` and `Sex` is an independent lifetable, so we need to declare groups beforehand using `group_by()`:

```
calc_ex <- function(Tx, lx){
  Tx / lx
}

KOR2014 |>
  filter(sex == "total") |>
  mutate(nMx = deaths / exposure,
         nAx = if_else(age == 0, .1, .5),
         n = 1,
         nqx = calc_nqx(nMx = nMx, nAx = nAx, n = n),
         lx = calc_lx(nqx = nqx),
         ndx = lx * nqx,
         nLx = calc_nLx(lx = lx, ndx = ndx, nAx = nAx, n = n),
         Tx = calc_Tx(nLx = nLx),
         ex = calc_ex(Tx = Tx, lx = lx)) |>
  filter(age == 0) |>
  pull(ex)
```

```
## [1] 81.49283
```

1.9 Calculating a lifetable for grouped data

If your data has many subgroups, we can calculate a lifetable for each of them by declaring groups on the data object with `group_by()`:

```
KOR2014 |>
  group_by(sex) |>
  mutate(nMx = deaths / exposure,
         nAx = if_else(age == 0, .1, .5),
         n = 1,
         nqx = calc_nqx(nMx = nMx, nAx = nAx, n = n),
         lx = calc_lx(nqx = nqx),
         ndx = lx * nqx,
         nLx = calc_nLx(lx = lx, ndx = ndx, nAx = nAx, n = n),
         Tx = calc_Tx(nLx = nLx),
         ex = calc_ex(Tx = Tx, lx = lx)) |>
  ungroup() |>
```

```
  filter(age == 0) |>
  select(sex, ex)
```

```
## # A tibble: 3 x 2
##   sex       ex
##   <chr>  <dbl>
## 1 female  84.5
## 2 male    78.2
## 3 total   81.5
```

2 A Lifetable function

You probably noticed that a whole lifetable operation can fit in a single `mutate()` call! Note, you could calculate all lifetables for each subset by simply including a `group_by()`. Well, that may be so, but there's still value in creating a wrapper function that does the whole thing:

```
calc_LT <- function(nMx, nAx, n, radix){
  N <- length(nMx)
  nqx  <- calc_nqx(nMx, nAx, n)
  lx   <- calc_lx(nqx = nqx, radix = 1)
  ndx  <- calc_ndx(nqx = nqx, lx = lx)
  nLx  <- calc_nLx(lx = lx, ndx = ndx, nAx = nAx, n = n)
  Tx   <- calc_Tx(nLx = nLx)
  ex   <- calc_ex(Tx = Tx, lx = lx)
  Age  <- cumsum(c(0,n))[1:N]


  tibble(Age = Age,
         nMx = nMx,
         nAx = nAx,
         nqx = nqx,
         lx = lx,
         ndx = ndx,
         nLx = nLx,
         Tx = Tx,
         ex = ex)
}
```

This function can be used as-is in a tidy pipeline using the `reframe()` verb. `reframe()` is like a more flexible version of `mutate()` or `summarize()`. The issue is here we have a function that takes vectors as its arguments, but returns a whole `data.frame` as its output.

```
KOR2014 |>
  mutate(nMx = deaths / exposure,
         nAx = case_when(age == 0 ~ .1,
                         age == 110 ~ 1/nMx,
                         TRUE ~ .5),
         n = 1) |>
  reframe(calc_LT(nMx,nAx,n, radix = 100000)) |>
  head()
```

```
## # A tibble: 6 x 9
##     Age       nMx   nAx       nqx    lx       ndx   nLx    Tx    ex
##   <dbl>     <dbl> <dbl>     <dbl> <dbl>     <dbl> <dbl> <dbl> <dbl>
## 1     0 0.00289     0.1 0.00290   1     0.00290   0.997   NaN   NaN
## 2     1 0.000296    0.5 0.000296  0.997 0.000295  0.997   NaN   NaN
## 3     2 0.000108    0.5 0.000108  0.997 0.000107  0.997   NaN   NaN
## 4     3 0.0000941   0.5 0.0000941 0.997 0.0000938 0.997   NaN   NaN
## 5     4 0.000100    0.5 0.000100  0.997 0.0000997 0.997   NaN   NaN
## 6     5 0.000108    0.5 0.000108  0.997 0.000108  0.996   NaN   NaN
```

2.1 reformulate as `data.frame`-in `data.frame`-out

You could also set the function up to work in a tidy pipeline by making a function that takes a whole data.frame as its input, and that also returns a data.frame. we'll just want to be sure that the input consists in a whole group (or chunk) of data:

```
# data.frame in, data.frame out!
calc_LT_tidy <- function(data, radix){
  # this is hacky, but works.
  # just pick out the needed vectors from the group of data
  calc_LT(nMx = data$nMx,
          nAx = data$nAx,
          n = data$n,
          radix = radix)
}
```

Now, this is something you can easily apply in bulk using `group_modify()` (`reframe()` also still works here!). You could design this in many different ways actually.

```
KOR2014 |>
  mutate(nMx = deaths / exposure,
         # You might change the way nAx is made...
         nAx = if_else(age == 0, .1, .5),
         n = 1) |>
  group_by(sex) |>
  # this also works, but notice we don't use the ~ for reframe,
  # and that the anonymous name of the incoming data is .data ...
  # reframe(calc_LT_tidy(data=.data, radix = 1e5))
  group_modify(~calc_LT_tidy(data = .x, radix = 1e5)) |>
  ungroup()
```

```
## # A tibble: 333 x 10
##    sex      Age       nMx   nAx       nqx    lx       ndx   nLx    Tx    ex
##    <chr>  <dbl>     <dbl> <dbl>     <dbl> <dbl>     <dbl> <dbl> <dbl> <dbl>
##  1 female     0 0.00289     0.1 0.00290   1     0.00290   0.997  84.5  84.5
##  2 female     1 0.000296    0.5 0.000296  0.997 0.000295  0.997  83.5  83.8
##  3 female     2 0.000108    0.5 0.000108  0.997 0.000107  0.997  82.5  82.8
##  4 female     3 0.0000941   0.5 0.0000941 0.997 0.0000938 0.997  81.5  81.8
##  5 female     4 0.000100    0.5 0.000100  0.997 0.0000997 0.997  80.5  80.8
##  6 female     5 0.000108    0.5 0.000108  0.997 0.000108  0.996  79.6  79.8
##  7 female     6 0.0000634   0.5 0.0000634 0.996 0.0000631 0.996  78.6  78.8
```

```
##  8 female     7 0.0000881   0.5 0.0000881 0.996 0.0000878 0.996  77.6  77.8
##  9 female     8 0.0000846   0.5 0.0000846 0.996 0.0000842 0.996  76.6  76.9
## 10 female     9 0.0000643   0.5 0.0000643 0.996 0.0000641 0.996  75.6  75.9
## # i 323 more rows
```

3 Fertility

In English, fertility refers to observed births, whereas fecundity refers to the capacity to give birth. The basic elements of fertility data include

- Events: births
- Exposure: every women alive in their reproductive age ($\approx$ 15 to 50 years old)

Births information most often comes from vital registration systems. This is the case for Korea. Countries without vital registration systems, or with incomplete vital registration rely on survey data to estimate fertility indicators.

3.1 Crude birth rate

The crude birth rate (CBR) is a rough measure of the occurrence/exposure of fertility.

$$CBR[0,T] = \frac{Number\ of\ births\ in\ the\ population\ between\ times\ T\ and\ T+t}{Number\ of\ person-years\ lived\ in\ the\ population\ between\ times\ T\ and\ T+t}$$

Does this look familiar? It's just like the crude death rate:

```r
# Person-years
KOR2014 |>
  filter(sex == "total") |>
  summarize(exposure = sum(exposure),
            births = sum(births, na.rm = TRUE)) |>
  mutate(CBR = births / exposure)
```

```
## # A tibble: 1 x 3
##    exposure  births     CBR
##       <dbl>   <dbl>   <dbl>
## 1 50765887. 435435. 0.00858
```

Often CBR is multiplied by 1000, so we'd have 8.58/1000

3.2 General fertility rate

The general fertility rate (GFR) is generally considered a better measure of fertility, as only women in their reproductive ages can give birth, and are thus at risk of experiencing the event. Sometimes the upper age is truncated at 45 rather than 50.

$$GFR[0,T] = \frac{Number\ of\ births\ in\ the\ population\ between\ times\ T\ and\ T+t}{Number\ of\ person-years\ lived\ by\ women\ aged\ 15\ to\ 50\ between\ times\ T\ and\ T+t}$$

This solution looks tricky because births are associated with total sex in our data, but here we need to relate them to women. For this, we use `pivot_wider()` to get all combinations of sex and measure side-by-side, then we shift births to females in `mutate()`, then we pivot the data back to its original form, filter to the desired subset, and calculate the GFR:

```
# Female population
KOR2014_2 <-
  KOR2014 |>
  select(-deaths) |>
  pivot_wider(names_from = sex,
              values_from = c(exposure,births)) |>
  mutate(births_female = births_total) |>
  pivot_longer(-c(year,age),
               names_to = c("measure","sex"),
               values_to= "value",
               names_sep="_") |>
  pivot_wider(names_from = "measure",
              values_from="value") |>
  filter(sex == "female")

KOR2014_2 |>
  filter(between(age, 15,50)) |>
  summarize(exposure = sum(exposure),
            births = sum(births)) |>
  mutate(GFR = births / exposure)
```

```
## # A tibble: 1 x 3
##    exposure  births    GFR
##       <dbl>   <dbl>  <dbl>
## 1 13308014. 435396. 0.0327
```

That is to say, about 33 per thousand. The GFR does not give unbiased information, however, as the age structure between ages 15 and 50 can vary wildly between populations, as can the shape and location of fertility rates within this range. Ideally, we would like a measure free of the effects of observed population structure.

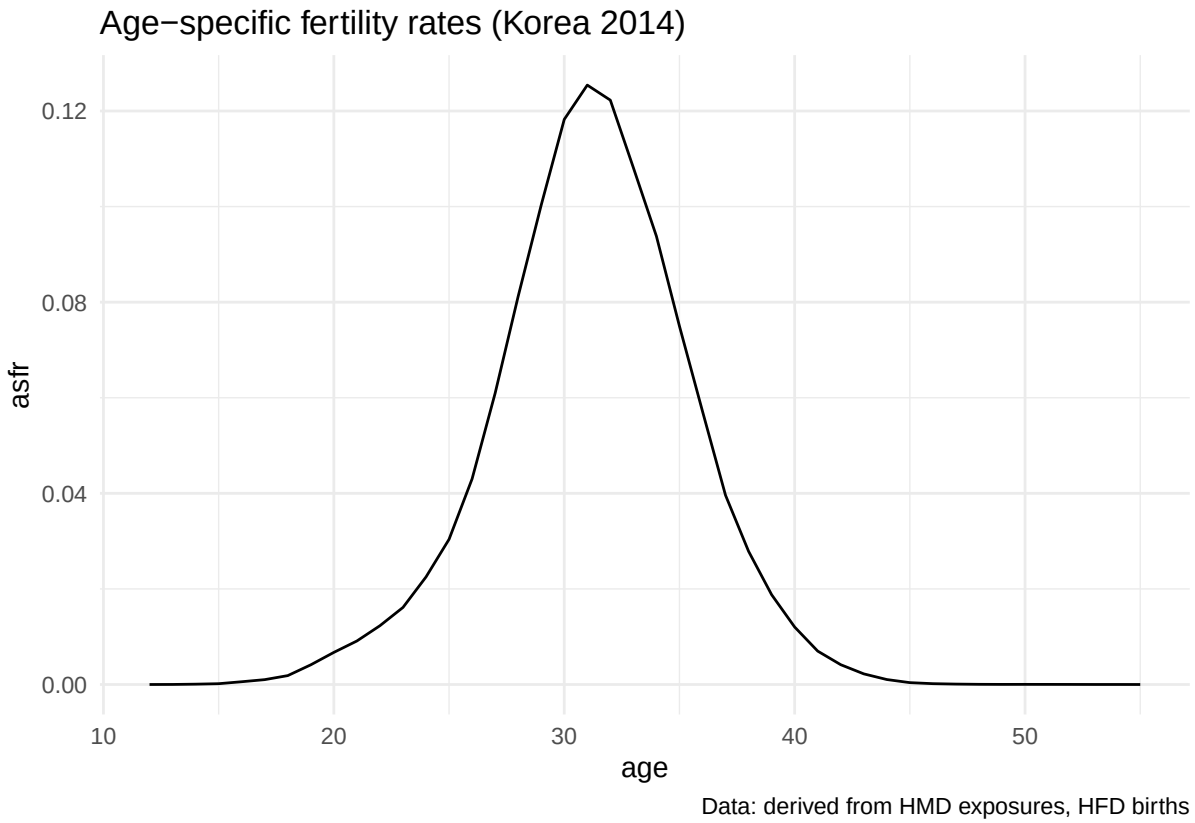
3.3 Age-specific fertility rates

As with age-specific death rates, age-specific fertility rates (F) are less sensitive to the age structure of the population. This measure provides the rate of giving birth for women age x to $x + n$:

$$ {}_nF_x[0,T] = \frac{Number\ of\ births\ between\ times\ T\ and\ T+t\ to\ women\ aged\ x\ to\ x+n}{Number\ of\ person-years\ lived\ by\ women\ aged\ x\ to\ x+n\ between\ times\ T\ and\ T+t} $$

```
KOR2014_2 |>
  mutate(asfr = births / exposure) |>
  ggplot(aes(x = age, y = asfr)) +
  geom_line() +
  labs(title = "Age-specific fertility rates (Korea 2014)",
```

```
       caption = "Data: derived from HMD exposures, HFD births") +
  xlim(12,55) +
  theme_minimal()
```



There can still be unobserved heterogeneity hiding in this population: age is not the only structural determinant of fertility. For instance, fertility may be considered separately by parity, or for other subpopulations.

3.4 Total fertility rate

The total fertility rate (TFR) is the average number of children a woman would have if she experienced the a particular set of age-specific fertility rates and survived until the end of her reproductive age. "*The TFR is the single most important indicator of fertility*" (Preston, Heuveline, and Guillot 2001). It is also the area under the ASFR curve.

$$TFR[T, T+t] = n \sum_{x=a}^{B-n} {}_nF_x[T, T+t]$$

where a and B are the minimum and maximum age at childbearing.

```
# TFR
KOR2014_2 |>
  mutate(asfr = births / exposure,
         n = 1) |> # make this 5 if you have 5-year age groups!!
  summarize(tfr = sum(asfr * n))
```

```
## # A tibble: 1 x 1
```

```
##     tfr
##   <dbl>
## 1  1.20
```

This the most commonly calculated and cited fertility metric, but it is not without criticism. For instance, (i) we should not discount the leverage mortality can have on population reproductivity (although it Korea this bias is very low), (ii) if fertility patterns are changing over time a period TFR may not give the best signal of fertility levels, and (iii) TFR is unfortunately sometimes presented as a target.

3.5 Mean age at childbearing

The mean age at childbearing is not a rate, but is based on the age-specific fertility rates. The mean age at childbearing (MA) is the average age of mothers at childbearing, standardized for the age-structure of the female population at reproductive age (Human Fertility Database 2018).

$$MA[T, T+t] = \frac{\sum_{x=a}^{B-n} \bar{x} * {}_nF_x[T, T+t]}{\sum_{x=a}^{B-n} {}_nF_x[T, T+t]}$$

where $\bar{x}$ is the mid-age of interval $x : x+n$, i.e. $\bar{x} = x + n/2$.

```
# MA
KOR2014_2 |>
  mutate(asfr = births / exposure,
         n = 1,
         xbar = age + n / 2) |>
  summarize(MA = sum(xbar * asfr) / sum(asfr))
```

```
## # A tibble: 1 x 1
##      MA
##   <dbl>
## 1  31.7
```

4 Exercises

In practical exercises, we will calculate trends for different populations based on different data.

References

Human Fertility Database. 2018. "Max Planck Institute for Demographic Research (Germany) and Vienna Institute of Demography (Austria)."

Preston, S, Patrick Heuveline, and Michael Guillot. 2001. "Demography: Measuring and Modeling Population Processes." *Malden, MA: Blackwell Publishers*.

Riffe, Tim, José Manuel Aburto, Ilya Kashnitsky, Monica Alexander, Marius D. Pascariu, Sara Hertog, and Sean Fennell. 2021. *DemoTools: Standardize, Evaluate, and Adjust Demographic Data*.