Tuesday class notes

Tim Riffe

7/6/2021

Review of little programming helper functions

```
# install.packages("demography")
```

```
library(demography)
```

```
## Loading required package: forecast
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## Registered S3 methods overwritten by 'demography':
##   method      from
##   print.lca   e1071
##   summary.lca e1071
```

```
## This is demography 1.22
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

cumprod() and cumsum()

```
X <- seq(2,10,2)
cumprod(X)
```

```
## [1]    2    8   48  384 3840
```

```
cumsum(X)
```

```
## [1]  2  6 12 20 30
```

ifelse()

1. give a logical expression or vector
2. what to return if it's `TRUE`
3. what to return if it's `FALSE`

`Ctrl + Alt + i`

```
ifelse(X > 5,"A","B")
```

```
## [1] "B" "B" "A" "A" "A"
```

case_when()

```
thing2 <- letters[1:5]
case_when(X < 3 ~ "first condition was met",
          X > 3 & thing2 == "d" ~ "second condition was met",
          TRUE ~ "these are the leftovers")
```

```
## [1] "first condition was met"  "these are the leftovers"
## [3] "these are the leftovers"  "second condition was met"
## [5] "these are the leftovers"
```

rev()

`rev()` flips a vector

```
rev(X)
```

```
## [1] 10  8  6  4  2
```

for(){}

Value iterator vs index iterator

```
# value iterator
A <- rep(0,5)
for (i in X){
  # this positional assignmet is hacky
  A[i / 2] <- i ^ 2
}
X / 2
```

```
## [1] 1 2 3 4 5
```

```
# index iterator
A <- rep(0,5)
for (i in 1:length(X)){
  # this is cleaner and less fragile
  A[i] <- X[i] ^ 2
}
```

These are various helper functions that will be used in the following lifetable code. The for-loop part is just for the data prep, which I'd actually like to fly over.

HMD intro

We'll use the demography package to load the data straight into R.

```
# us <- "this is my username"
# pw <- "this is my clever unhackable unbreakable maximum entropy password just forget about you'll nev
data <-
  demography::hmd.mx("ESP", us, pw)
```

```
## Warning in demography::hmd.mx("ESP", us, pw): NAs introduced by coercion
```

```
str(data)
```

```
## List of 7
##  $ type  : chr "mortality"
##  $ label : chr "ESP"
##  $ lambda: num 0
##  $ year  : int [1:111] 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917 ...
##  $ age   : num [1:111] 0 1 2 3 4 5 6 7 8 9 ...
##  $ pop   :List of 3
##   ..$ female: num [1:111, 1:111] 295481 263968 250296 242430 244758 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:111] "0" "1" "2" "3" ...
##   .. .. ..$ : chr [1:111] "1908" "1909" "1910" "1911" ...
##   ..$ male  : num [1:111, 1:111] 307918 268323 254684 246566 246617 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:111] "0" "1" "2" "3" ...
##   .. .. ..$ : chr [1:111] "1908" "1909" "1910" "1911" ...
##   ..$ total : num [1:111, 1:111] 603399 532291 504980 488996 491376 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:111] "0" "1" "2" "3" ...
##   .. .. ..$ : chr [1:111] "1908" "1909" "1910" "1911" ...
##  $ rate  :List of 3
##   ..$ female: num [1:111, 1:111] 0.1598 0.0827 0.0452 0.0247 0.0163 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:111] "0" "1" "2" "3" ...
##   .. .. ..$ : chr [1:111] "1908" "1909" "1910" "1911" ...
##   ..$ male  : num [1:111, 1:111] 0.1893 0.0863 0.0455 0.0258 0.0166 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:111] "0" "1" "2" "3" ...
##   .. .. ..$ : chr [1:111] "1908" "1909" "1910" "1911" ...
##   ..$ total : num [1:111, 1:111] 0.1748 0.0846 0.0453 0.0253 0.0164 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:111] "0" "1" "2" "3" ...
##   .. .. ..$ : chr [1:111] "1908" "1909" "1910" "1911" ...
##  - attr(*, "class")= chr "demogdata"
```

This will get pasted in the google doc

```
library(tidyverse)
sexes <- data$pop %>% names()

# two containers, columns given, but no rows
ESpop   <- tibble(Year = NULL, Age = NULL, Sex = NULL, Exposure = NULL)
ESrates <- tibble(Year = NULL, Age = NULL, Sex = NULL, M = NULL)
```

```
for (i in sexes){
  ESpop <- data$pop[[i]] %>%
    as_tibble() %>%
    rownames_to_column("Age") %>%
    pivot_longer(cols = -Age,
                 names_to = "Year",
                 values_to = "Exposure") %>%
    mutate(Sex = i,
           Age = as.integer(Age) - 1) %>%
    bind_rows(ESpop)

  ESrates  <- data$rate[[i]] %>%
    as_tibble() %>%
    rownames_to_column("Age") %>%
    pivot_longer(cols = -Age, names_to = "Year", values_to = "M") %>%
    mutate(Sex = i,
           Age = as.integer(Age) - 1) %>%
    bind_rows(ESrates)
}

ES <- left_join(ESpop,
                ESrates,
                by = c("Age", "Year", "Sex")) %>%
  select(Year, Sex, Age, Exposure, M) %>%
  arrange(Year, Sex, Age)
```

The above code shouldn't be a hangup, it's there jsut to get the data live and not have to include big files in the repository. A quick narration may have added value to someone, but if you REALLY want to learn about data wrangling of this kind, then consider going through by EDSD data wrangling module materials.

Introducing the lifetable

We start with m_x, conditional risk, which we often assume is somehow constant within age intervals, but this assumption is relaxable.

We would like q_x, the conditional death probabilities, which are interpreted as the probability of dying in the year (or interval) given that you survive into the age group. Ok, or the probability of not making it to the next birthday.

We need an instrumental piece of information in order to help us convert m_x into q_x, and somewhere in this conversion we'll end up applying assumptions, at least for discrete data of this kind.

The helper variable we need is called a_x, and it stands for the average time spent in the interval of those not making it to the end of the interval.

$$q_x = \frac{m_x}{1 + (1 - a_x) * m_x}$$

First helper variable is p_x, the probability of surviving from one birthday to the next.

$$p_x = 1 - q_x$$

The survival curve, or lifetable survivorship is the probability of surviving from birth until age x, l_x. Is is the cumuative product of the probabilities of surviving from each birthday to the next, i.e. the probability of surviving through successions of ages.

$$l_x = radix * \prod_{t=0}^{x} p_t$$

The death distribution, d_x is the size of the decrement in l_x between each successive age, $l_{x+1} - l_x$, or $l_x * q_x$, or something analogous.

One of the final things we need is actually the survival curve integrals, i.e. lifetable *exposure*, or fictitious person-years lived in the age interval. We can approximate it well using a_x as a helper again, or just use the trapezoid rule, for example.

$$L_x = l_{x+1} + (1 - a_x) * d_x$$
$$L_x = l_{x+1} + .5 * d_x$$

T_x is ALL the remaining years lived by those surviving to age x. It's the integral *going up* fro age x. In order to condition on survival and treat it as something we can get remaining life exectancy from, we divide out l_x as we go.

$$T_x = \sum_{x}^{\omega} L_t$$

$$e_x = T_x / l_x$$
$$e_x = 1/l_x * \sum(x * d_x)$$

Now it's time to calculate this

Note, the age 0 steps in `case_when()` are a simplified version of what HMD does, and you could actually faithfully implement their a_0 approach in the same `case_when()` setup. *However* this and other a_0 and a_x approaches are implemented in the `DemoTools` package (check my github), so you actually don't need to code it yourself!

```
radix <- 1e5
LT <-
  ES %>%
  group_by(Year, Sex) %>%
  mutate(M = ifelse(is.na(M), .5, M),          # hack nr 1
         n = 1,
         ax = case_when(
                Age == 0 & M < .02012 ~ .14916 - 2.02536 * M,
                Age == 0 & M < .07599 ~ 0.037495 + 3.57055 * M,
                Age == 0 & M >= .07599 ~ 0.30663,
                Age == 110 ~ 1 / M,
                TRUE ~ n / 2),
         ax = ifelse(is.infinite(ax),.5,ax),   # hack nr 2
         qx = (M * n) / (1 + (n - ax) * M),
         qx = ifelse(qx > 1, 1, qx),           # hack nr 3
         px = 1 - qx,
         lx = radix * c(1, cumprod(px[-n()])),
         dx = qx * lx,
         Lx = n * lx - (n - ax) * dx,
         Tx = Lx %>% rev() %>% cumsum() %>% rev(),
         ex = Tx / lx,
```

```
        ex = ifelse(is.nan(ex),ax,ex))  %>%  # hack nr 4
ungroup()
```

I've labelled the lines that constitute hacks that are acceptable for our purposes. These almost exclusively affect the closeout age 110, and other very high ages, and that's why I'm not bothering doing anything more sophisticated. For serious work, I'd probably rather fit a parametric model to mortality in the oldest ages and overwrite the oldest mortality results with it (see the `MortalityLaws` package if you're into that!). Doing that to M_x before starting the lifetable calcs would likely solve each of the steps that required a hacky solution. Finally, I'd consider smoothing mortality rates for early years where age heaping appears to be an issue. This is something you can learn to do by taking Giancarlo Camarda's PHDS/IDEM smoothing course offered periodically at the MPIDR, or by following examples in `R` packages such as `MortalitySmooth`, `ungroup`, or `DemoTools`.

The important columns worth developing intuition on first are 1. the M_x pattern (not level, unless comparing curves) and being able to visually judge whether one adheres to your expectations 2. l_x, especially with a starting population of 1, where it obtains a nice interpretation as the probability of surviving from birth until age x 3. d_x the death distribution get's you thinking about lifespan variability: is it wide(r) or narrow(er)? 4. e_x if for no reason other than the fact it represents a nice annualization of rates! It's a nice thing to be aware of generally.