KOSTAT-UNFPA Summer Seminar on Population

Workshop 1. Demography in R

Day 5: Advanced pipelines

Instructor: Tim Riffe
tim.riffe@gmail.com

Assistant: Rustam Tursun-Zade
rustam.tursunzade@gmail.com

30 July 2021

Contents

1 Summary

1.1 Design and motivation

One of the Monday exercises was to vote on data to use for today, Friday. I'll go with whatever signal I get. Eight votes were cast:

- Human Lifetable Database (3)
- World Values Survey (2)
- Global Burden of Disease (2)
- Human Mortality Database (1)

Based on this, I've decided to try and combine data from the Human Lifetable Database (HLD) and Global Burden of Disease (GBD) 2017 version. While many analyses are possible, I've decided to try to approximate healthy life expectancy using these two sources. It's likely that these two sources have never been combined in this way.

The purpose of today's exercise is to walk through a raw analysis that provokes some unforeseen issues that need to be resolved. Resolving unforeseen issues will force us to find solutions that will reveal further flexibility and power in the tidy data approach. And of course in the course we will see new `dplyr` *verbs* and `ggplot2` features, indeed even an old base `R` programming concept.

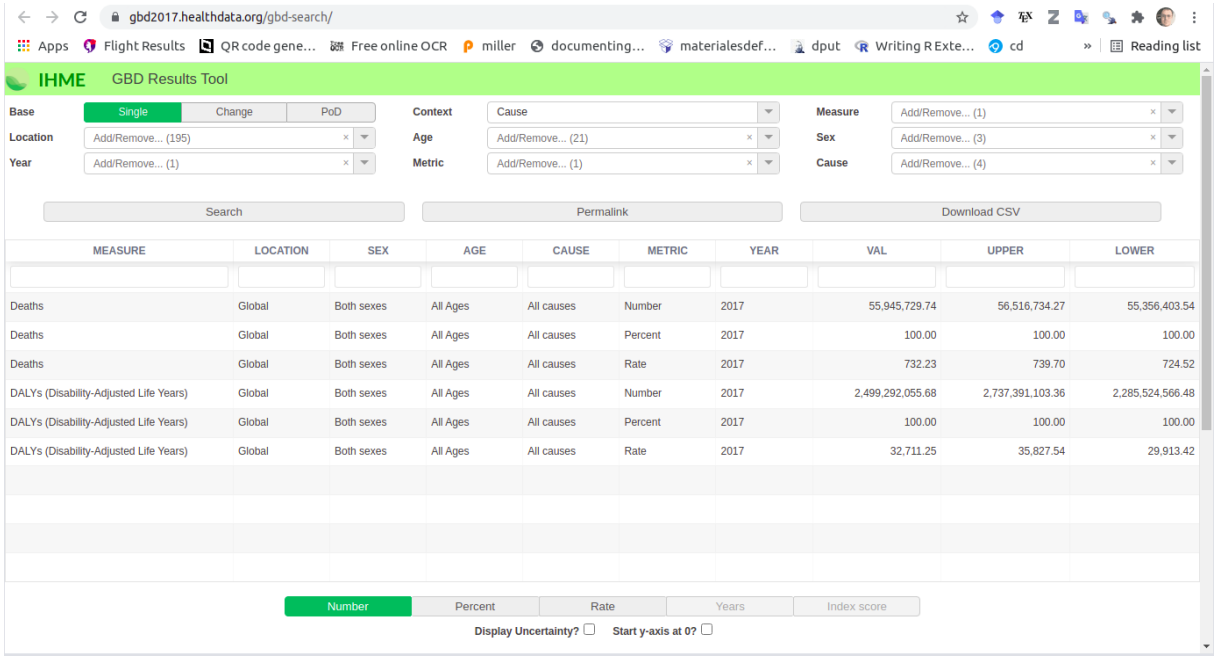
1.2 Healthy life expectancy

The purpose of healthy life expectancy is to tell us the average length of a healthy life under a given set of mortality and health conditions. There are different approaches to estimating it, of which we'll take the most widely applicable Sullivan (1971)

2 Downloading data

2.1 GBD

You can download the GBD data selection I made, which consists in all countries and territories in 2017 by abridged ages and sex, and gives the prevalence of four conditions. The data selection was made from this webpage: https://gbd2017.healthdata.org/gbd-search/, which looks like this:

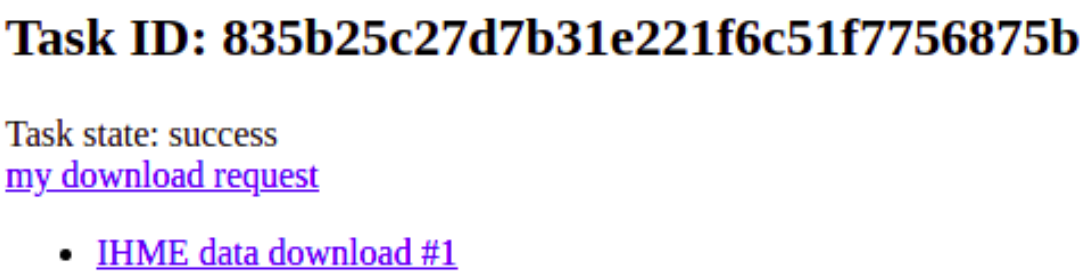


My selections were:

- Base: `single`
- Location: `all countries and territories`
- Year: `2017`
- Context: `Cause`
- Age: manual selection of 0, 1-4, 5-9 ...
- Metric: `percent`
- Measure: `prevalence`
- Sex: `Male`, `Female`, `Both`
- Cause: `Total All causes`, `B.2.6 Cardiomyopathy and myocarditis`, `B.4 Digestive diseases`, 'B.8 Diabetes and kidney diseases`

I then clicked `permalink`. They generate the file, and then send an email when it's ready. This is the location they sent me to: https://gbd2017.healthdata.org/gbd-search/result/835b25c27d7b31e221f6c51f7756875b, which, if true to the name *perma*link this ought to be available for a reasonably long period of time...

This is what it looks like:



I right-clicked where it says `IHME data download #1` and selected `copy link address`. Said link is what you see used in the following code chunk, which downloads the data as a zip file called `GBD_prevalence.zip` and sticks it in our `Data` folder. Before downloading, we check that we haven't already downloaded it using `if(file.exists())`. Make sense? We don't want to download over and over. What you see in the `{}` is the body of code subject to the `if` condition. That's how conditional code works ;-)

```
GBD_url <- "https://s3.healthdata.org/gbd-api-2017-public/835b25c27d7b31e221f6c51f7756875b_f
local_file <- here::here("Data","GBD_prevalence.zip")
# Only download the file once!
if (!file.exists(local_file)){
  download.file(GBD_url, destfile = local_file)
}
```

The zip file contains a suggested citation (Network (2018)) and a csv that can be read directly with `readr::read_csv()`:

```
library(tidyverse)
```

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

```
## v ggplot2 3.3.5     v purrr   0.3.4
```

```
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.0     v forcats 0.5.1
```

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

```
library(readr)
GBD  <- read_csv(local_file)
```

```
## Multiple files in zip: reading 'IHME-GBD_2017_DATA-835b25c2-1.csv'
```

```
## Rows: 49140 Columns: 10
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): measure, location, sex, age, cause, metric
## dbl (4): year, val, upper, lower
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
glimpse(GBD)
```

```
## Rows: 49,140
## Columns: 10
## $ measure  <chr> "Prevalence", "Prevalence", "Prevalence", "Prevalence", "Prev~
## $ location <chr> "Ghana", "Ghana", "Ghana", "Ghana", "Ghana", "Ghana", "Ghana"~
## $ sex      <chr> "Male", "Female", "Both", "Male", "Female", "Both", "Male", "~
## $ age      <chr> "1 to 4", "1 to 4", "1 to 4", "5 to 9", "5 to 9", "5 to 9", "~
## $ cause    <chr> "Diabetes and kidney diseases", "Diabetes and kidney diseases~
## $ metric   <chr> "Percent", "Percent", "Percent", "Percent", "Percent", "Perce~
## $ year     <dbl> 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2~
## $ val      <dbl> 0.0006772738, 0.0013323786, 0.0010007344, 0.0037088414, 0.004~
## $ upper    <dbl> 0.001216079, 0.002191553, 0.001592305, 0.005200965, 0.0065402~
## $ lower    <dbl> 0.0002696305, 0.0007487865, 0.0005419410, 0.0024902408, 0.003~
```

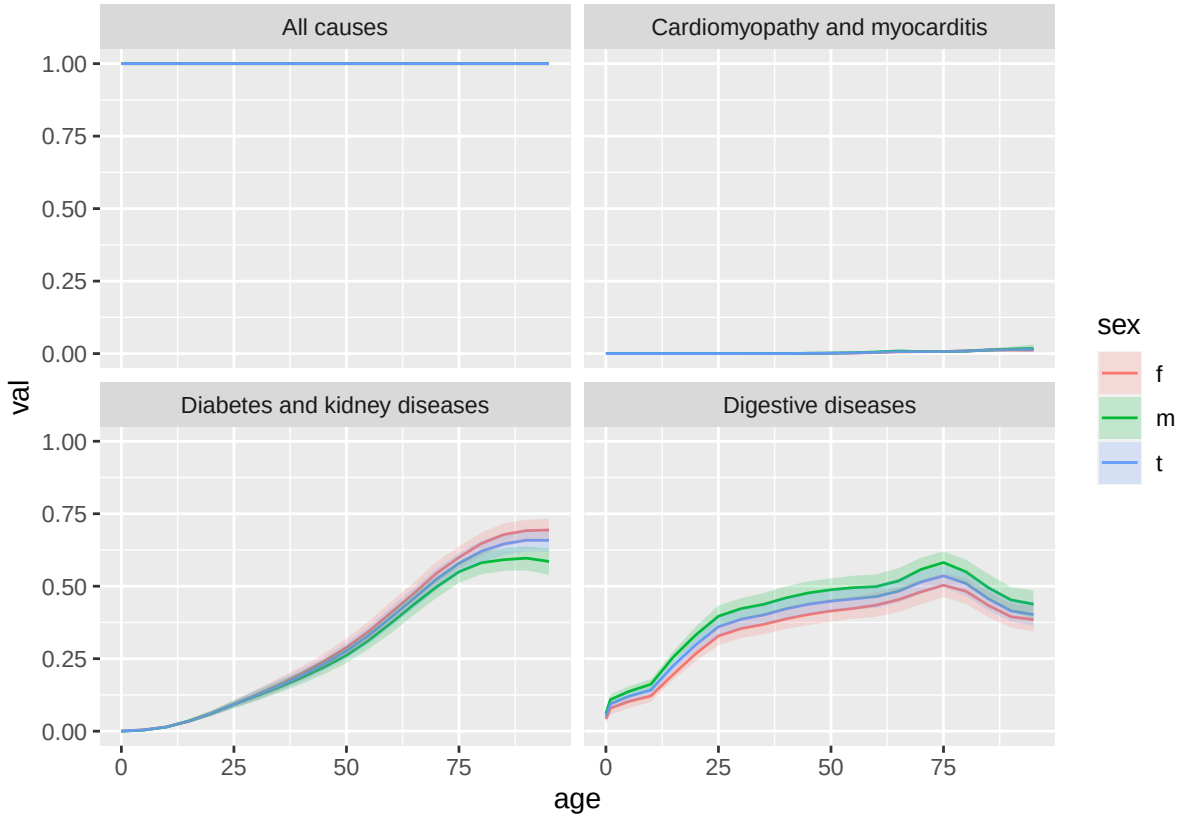
This is a rather tidy dataset already, so this saves us lots of work :-)

Let's first change the `age` coding to integer lower bounds. Again, there might be many strategies to do this. What I did was to look at the unique values `GBD %>% pull(age) %>% unique()`. I noticed the pattern that the lower age bound was always contained in the first two characters, which I extract using `substr()`. At this point `age` is almost ready for `parse_number()`, except age 0 is left at `"<1"`, so we handle it first using `ifelse()`. We can take care to recode `sex` at the same time, using `case_when()`.

```
GBD <-
  GBD %>%
  mutate(age = substr(age, start = 1, stop = 2),
         age = ifelse(age == "<1", "0", age),
         age = parse_number(age),
         sex = case_when(sex == "Male" ~ "m",
                         sex == "Female" ~ "f",
                         sex == "Both" ~ "t"))
```

Now let's have a preliminary look at some prevalence age patterns. Since the data have confidence intervals, I'll go ahead and add on the confidence bands to get a sense of it.

```
GBD %>%
  filter(location == "Ghana") %>%
  ggplot(aes(x = age, y = val, color = sex)) +
  geom_line() +
  geom_ribbon(aes(ymin = lower,
                  ymax = upper,
                  fill = sex),
              alpha = .2,
              color = NA) +
  facet_wrap(~cause)
```



From this picture, I think we can definitely delete `"All causes"`, which apparently contains no useful information. I have no instincts or background knowledge on `Cardiomyopathy and myocarditis`, but it seems harmless to keep it for now.

```
GBD <- GBD %>%
  filter(cause != "All causes")
```

We'll see how to handle country codes when we see what the codes and country names are for HLD data

2.2 HLD

The Human Lifetable Database https://www.lifetable.de/cgi-bin/index.php is the lesser-know cousin of the Human Mortality Database, and it is subject to irregular updates. This database does not estimate its own lifetables. Instead it collates lifetables from many sources and harmonizes them to a standard form, at times necessitating recalculation. Lifetables in this

database are exclusively based on vital registration, but may have been subject to various kinds of adjustment from the original producers. For years I never saw anyone use this database in research, and I suspect this is because it required lots of manual clicking to get all the files. Now they offer a pooled data file. Let's get it!

```
HLD_url <- "https://www.lifetable.de/data/hld.zip"

HLD_local_file <- here::here("Data","HLD.zip")
if (!file.exists(HLD_local_file)){
  download.file(HLD_url, destfile = HLD_local_file)
}
```

It's rather straightforward to get this into `R` using `read_csv()`. It doesn't even matter that the contents of the zip file don't have a file extension!

```
HLD <- read_csv(HLD_local_file)
```

3 Harmonize to join

These data include quite a large range of years. Since GBD data is just 2017 in what I downloaded, let's first filter HLD down to data just after, say, 2012. Now, for each country, we should try to select the year closest to 2017. We also filter down to just national data (no subpopulations). This filtering routine took some iteration to derive to make sure there was just one lifetable per `Country` and `Sex`.

```
HLD <-
  HLD %>%
  filter(Ethnicity == "0",    # had to look at code book
         Residence == "0",
         Region == "0",
         TypeLT > 1,
         Year1 >= 2012) %>%   # design choice

  # Calculate temporal distance to mid 2017
  mutate(Year = (Year1 + Year2 + 1) / 2,
         Diff = 2017.5 - Year,
         Dist = abs(Diff)) %>%

  # selection is independent per Country and Sex
  group_by(Country, Sex) %>%

  # create logical variable to help with filter
  mutate(keep = Dist == min(Dist)) %>%

  # if a country has 2016 and 2018 but no 2017, then take 2018?
  group_by(Country, Sex, keep) %>%

  # If we're in TRUE, then take max Year. If there's only one Year
  # then this is innocuous.
  mutate(keep2 = keep & Year == max(Year)) %>%
  ungroup() %>%
  filter(keep2) %>%
```

```r
  # renaming for purpose of joining
  select(ISO3 = Country, sex = Sex, age = Age, nLx = `L(x)`) %>%

  # ensure sex codes match
  mutate(sex = ifelse(sex == 1, "m", "f"))
```

On inspection, we learn that the HLD data only has males and females, so we can safely remove total sex from the GBD data. Finally, let's derive ISO3 country codes from the GBD names.

```r
# install.packages("countrycode")
library(countrycode)
GBD <- GBD %>%
  filter(sex != "t") %>%
  mutate(ISO3 = countryname(location, destination = "iso3c")) %>%
  arrange(location, sex, cause, age)
```

3.1 Join the datasets!

This was satisfyingly little upfront work to get ready to join. Double-checking: We have the same sex codes, the same age classes (open ages not checked, not essential here), and the same ISO3 codes. I think an inner join will get the job done. Note, since in the GBD data we have 3 causes per `age-sex-year-location` combination, `Lx` will be repeated for each of them. It appears we can calculate a few kinds of healthy life expectancy using the Sullivan method (Sullivan (1971)), how exciting!

```r
DAT <-
  inner_join(GBD, HLD, by = c("ISO3", "sex", "age"))

DAT %>% pull(location) %>% unique()
```

```
##  [1] "Algeria"                  "Australia"
##  [3] "Austria"                  "Bangladesh"
##  [5] "Belgium"                  "Bosnia and Herzegovina"
##  [7] "Botswana"                 "Brazil"
##  [9] "Bulgaria"                 "Canada"
## [11] "Costa Rica"               "Cyprus"
## [13] "Czech Republic"           "Denmark"
## [15] "Estonia"                  "Finland"
## [17] "France"                   "Georgia"
## [19] "Germany"                  "Hungary"
## [21] "Iceland"                  "India"
## [23] "Israel"                   "Italy"
## [25] "Japan"                    "Kazakhstan"
## [27] "Latvia"                   "Luxembourg"
## [29] "Macedonia"                "Malaysia"
## [31] "Malta"                    "Mauritius"
## [33] "Netherlands"              "New Zealand"
## [35] "Norway"                   "Poland"
## [37] "Portugal"                 "Russian Federation"
## [39] "Serbia"                   "Singapore"
## [41] "Slovakia"                 "Slovenia"
## [43] "South Korea"              "Spain"
## [45] "Sweden"                   "Switzerland"
```

```
## [47] "Taiwan (Province of China)" "Tajikistan"
## [49] "Turkey"                     "United Kingdom"
## [51] "United States"
```

4 Calculate HLE

4.1 The Sullivan approach

Now we can illustrate the Sullivan approach visually. As before, in order to plot abridged lifetable exposure, we should divide out the radix (`100000`) and age interval widths (`n`).

Given a stationary stock defined by ${}_nL_x$ and prevalence of a poor health state ${}_n\pi_x$, the unhealthy survivors ${}_nU_x$ in each age are defined as:

$$ {}_nU_x = {}_nL_x * {}_n\pi_x $$

And the healthy survivors ${}_nH_x$ are:

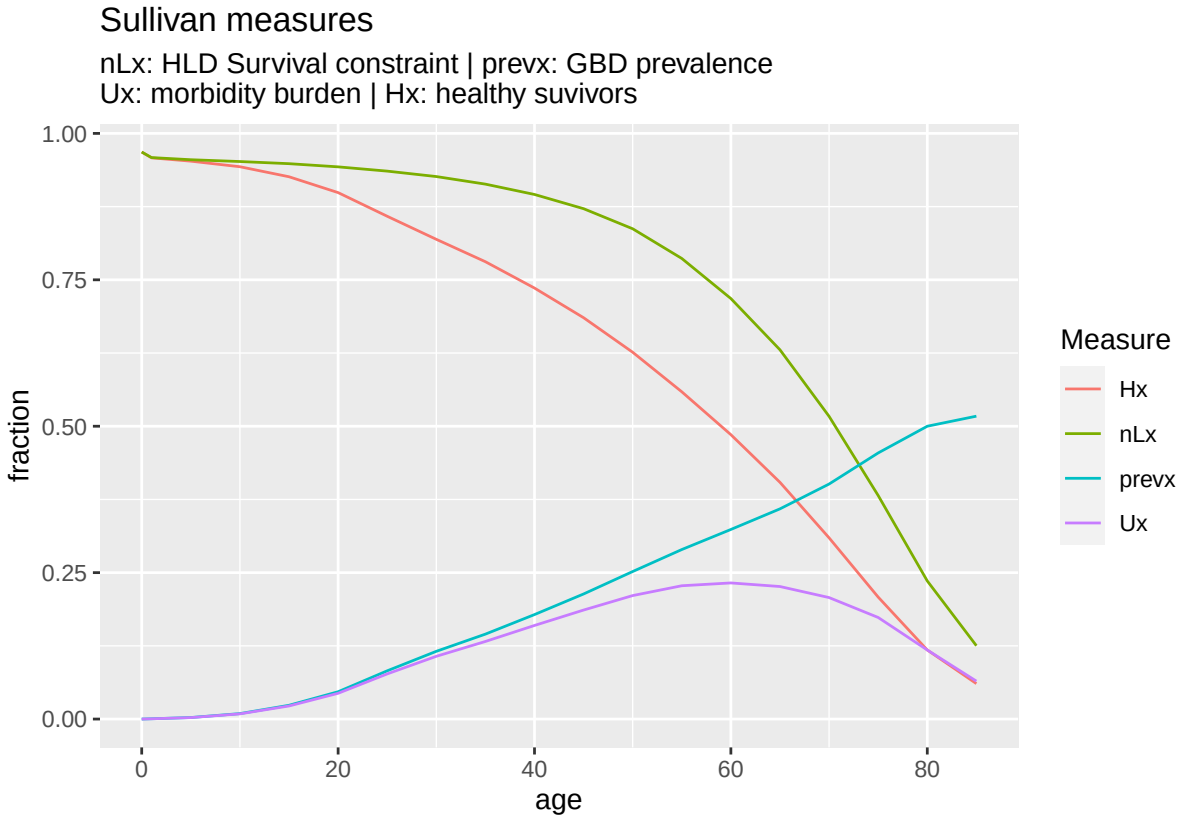
$$ {}_nH_x = {}_nL_x * (1 - {}_n\pi_x) $$

And we have:

$$ e_x = \sum {}_nH_x + \sum {}_nU_x $$

That is, since in each age prevalence and its compliment sum to 1, we use it to split lifetable survivors in a constrained way. Observe:

```
DAT %>%
  filter(location == "India",
         cause == "Diabetes and kidney diseases",
         sex == "m") %>%
  mutate(nLx = nLx / 1e5,
         n = case_when(age == 0 ~1,
                       age == 1 ~ 4,
                       TRUE ~ 5),
         nLx = nLx / n,
         Ux = val * nLx,
         Hx = nLx - Ux) %>%
  select(age, nLx, prevx = val, Ux, Hx) %>%
  pivot_longer(2:5, names_to = "Measure", values_to = "value") %>%
ggplot(mapping = aes(x = age, y = value, color = Measure)) +
  geom_line() +
  labs(y = "fraction",
       title = "Sullivan measures",
       subtitle = "nLx: HLD Survival constraint | prevx: GBD prevalence\nUx: morbidity burde
```



4.2 Diagnostic and re-join

Note: We see that for this subset, the lifetable closes out at age 85, so everything will be an underestimate. If we were being more rigorous, we would pre-extrapolate both prevalence and the lifetable. Out of curiosity, do we always have the same closeout ages? Because this might affect rankings:

```
DAT %>%
  select(ISO3, age) %>%
  group_by(ISO3) %>%
  filter(age == max(age)) %>%
  pull(age) %>% table()
```

```
## .
##  80  85  90  95
##  30  54  12 210
```

This begs the question, which source is constraining the upper age?

```
HLD %>%
  select(ISO3, age) %>%
  group_by(ISO3) %>%
  filter(age == max(age)) %>%
  pull(age) %>%
  table()
```

```
## .
##  80  85  90  95 100 105 110
##  10  18   4  12  42  14   4
```

```
GBD %>%
  select(ISO3, age) %>%
  group_by(ISO3) %>%
  filter(age == max(age)) %>%
  pull(age) %>%
  table()
```

```
## .
##   95
## 1170
```

In this situation, in practice, on realizing that the lifetables do not extend to at least age 95+ in all cases, I would opt to extrapolate them using the `MortalityLaws` package (Pascariu (2020)). However, we should remember the the final value of ${}_nL_x$ represents the integral of an open age group. Therefore the sum of ${}_nL_x$ for the populations that close out at age 80 is probably OK. For these 30 lifetables (15 populations I presume), the only compromise is that prevalence is being assumed constant beyond age 80, which may or may not be true, but which doesn't introduce much bias for HLE calculated over the full age range. Therefore I propose to drop the closeout age of all life tables to 80+, then re-merge the sources, in order to better compare apples with apples.

```
DAT <-
  HLD %>%
  # recode age
  mutate(age = ifelse(age >= 80, 80, age)) %>%
  # sum within age groups
  group_by(ISO3, sex, age) %>%
  summarize(nLx = sum(nLx),
            .groups = "drop") %>%
  # join back to GBD
  inner_join(GBD, by = c("ISO3", "sex", "age"))
```

4.3 Bulk calculation

Let's calculate life expectancy as the sum of survivorship, as well as point estimates and intervals for healthy life expectancy. Note, we do **not** divide out n when summing to get overall expectancy! That step was just for plotting!

```
HLE <-
  DAT %>%
  mutate(nLx = nLx / 1e5,
         Hx = nLx * (1 - val),
         # High prevalence of a bad condition maps
         # to lower health life, ergo the switcheroo
         Hx_lower = nLx * (1 - upper),
         Hx_upper = nLx * (1 - lower)) %>%
  # Aggregation step
  group_by(location, sex, cause) %>%
  summarize(LE = sum(nLx),
            HLE = sum(Hx),
            HLE_upper = sum(Hx_upper),
            HLE_lower = sum(Hx_lower),
            .groups = "drop")
```

5 Visualize HLE in a dotplot?

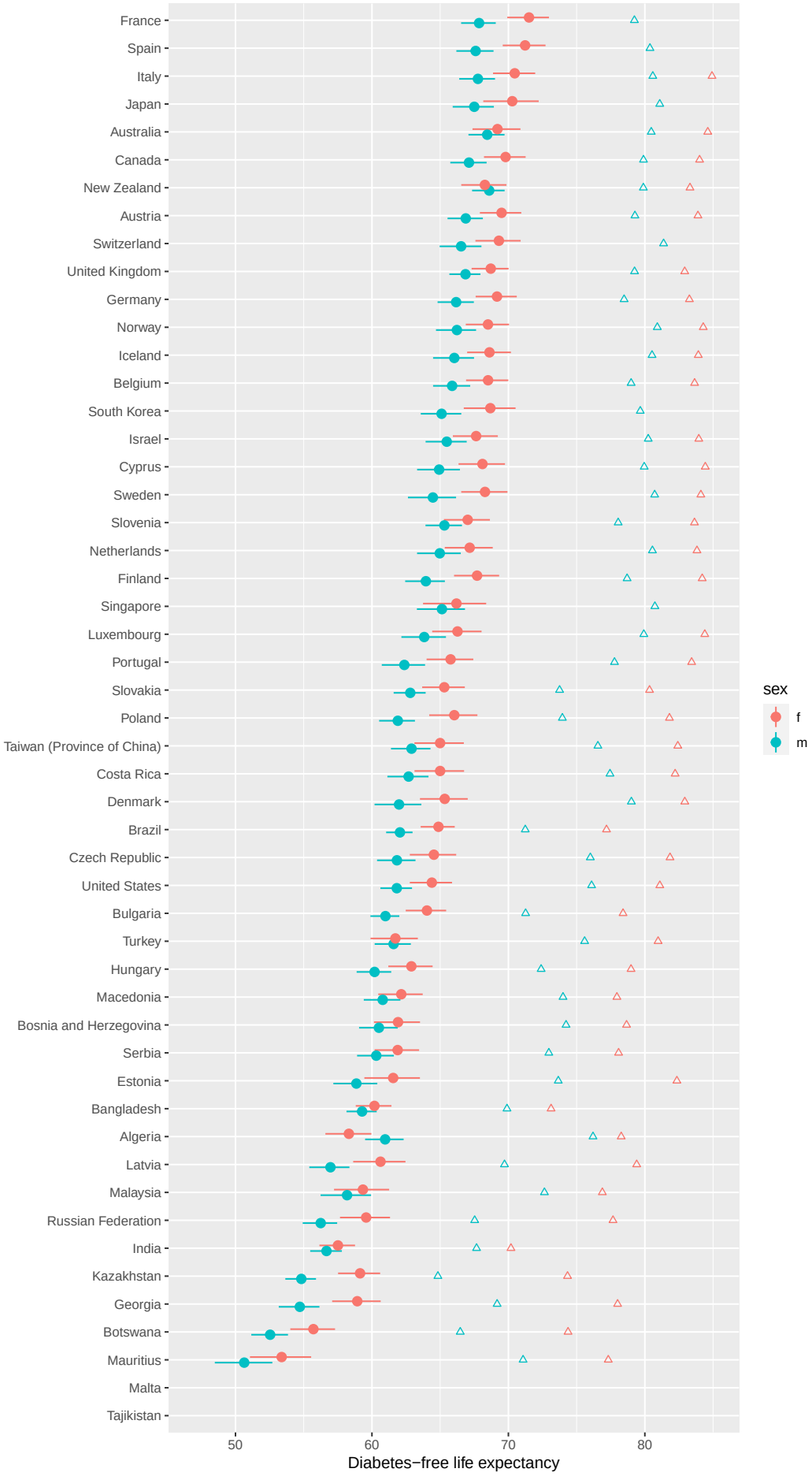
I'm going to plot HLE and life expectancy on the x axis, mostly in order to add full country names without having to rotate them. Each country will be a row. We will give HLE with point estimates and non-overlapping error bars. LE will be added to the plot in a second pass. Here I pick out some of the new idioms in this plot:

1. `y = reorder(location, HLE)` if we were to just say `y = HLE` then countries would plot alphabetically. This isn't a look-up table, it's a data visualization that may or may not reveal a pattern in the data. I always prefer to order based on a variable in the data. This statement just says that the country rows should follow the rank order of HLE.
2. `position = position_dodge2(width = .4, reverse = TRUE)` since the y coordinate comes from the country, all dots for a country would by default be on the same line. Even worse, error bars would overlap. This line applies a controlled jitter to avoid this overplotting.
3. `geom_pointrange()` adds the error bars. It needs `xmin` and `xmax` arguments. We could have also mapped these above in `ggplot()`, but I preferred to keep them in the `geom` where they get used. These need to follow the same jitter as the point estimates.
4. `geom_point(data = filter(HLE, cause == "Diabetes and kidney diseases")...` We can add new points on the same plot! Just use the `data` argument. This trick is also handy in other contexts to be able to highlight a single series. Be sure to order `y` in the same way!
5. `xlim()` I manually set `x` limits because of the Malta / Tajikistan issue spreading out the axis.
6. In order to plot the figure tall, I use parameters in the R markdown code chunk `fig.width=8`, `fig.height=14`. I think the units are in cm.

```
HLE$cause %>% unique()
```

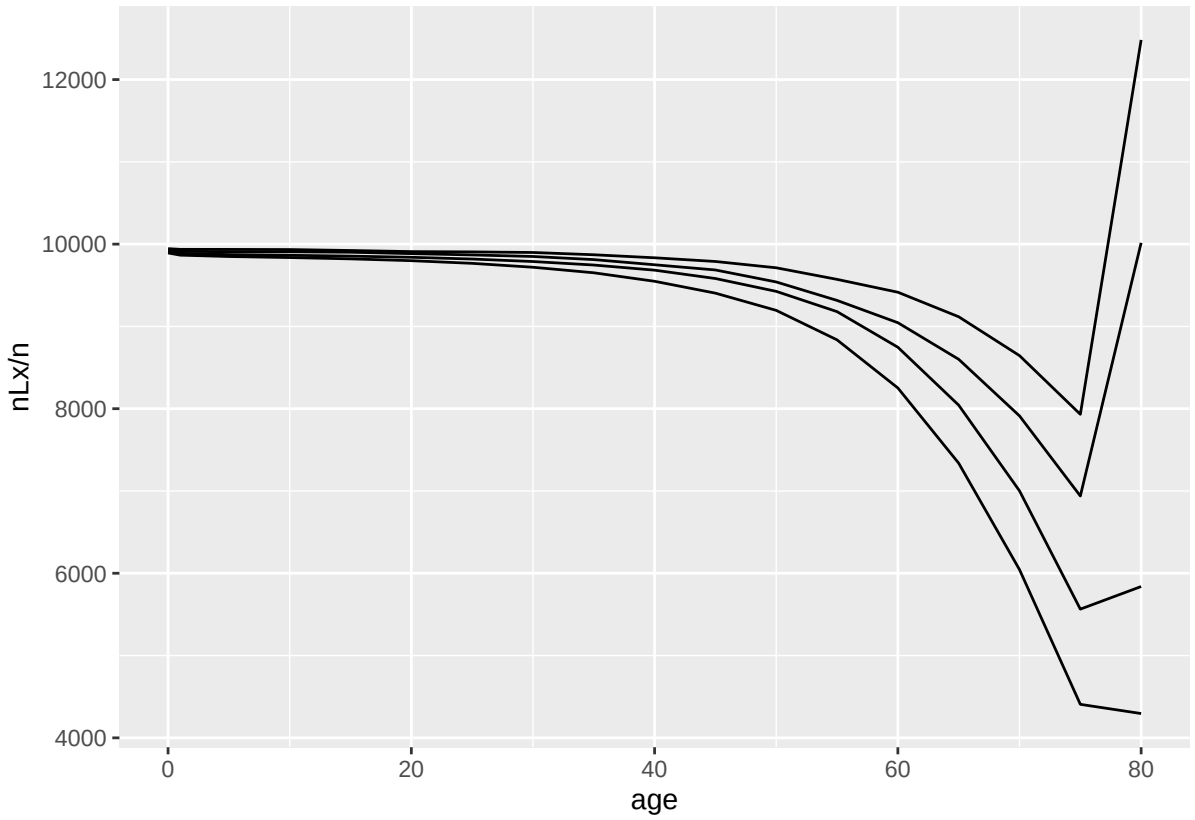
```
## [1] "Cardiomyopathy and myocarditis" "Diabetes and kidney diseases"
## [3] "Digestive diseases"
```

```
HLE %>%
  filter(cause == "Diabetes and kidney diseases") %>%
  ggplot(aes(x = HLE, y = reorder(location, HLE), color = sex)) +
           geom_point(position = position_dodge2(width = .4, reverse = TRUE)) +
           geom_pointrange(aes(xmin = HLE_lower, xmax = HLE_upper),
                           position = position_dodge2(width = .4, reverse = TRUE)) +
  # new data!
  geom_point(data = filter(HLE, cause == "Diabetes and kidney diseases"),
             mapping = aes(x = LE, y = reorder(location, HLE), color = sex),
             shape = 2) +
  xlim(47,85) +
  labs(y = "",
       x = "Diabetes-free life expectancy")
```


France
Spain
Italy
Japan
Australia
Canada
New Zealand
Austria
Switzerland
United Kingdom
Germany
Norway
Iceland
Belgium
South Korea
Israel
Cyprus
Sweden
Slovenia
Netherlands
Finland
Singapore
Luxembourg
Portugal
Slovakia
Poland
Taiwan (Province of China)
Costa Rica
Denmark
Brazil
Czech Republic
United States
Bulgaria
Turkey
Hungary
Macedonia
Bosnia and Herzegovina
Serbia
Estonia
Bangladesh
Algeria
Latvia
Malaysia
Russian Federation
India
Kazakhstan
Georgia
Botswana
Mauritius
Malta
Tajikistan
50
60
70
80
Diabetes−free life expectancy
sex
f
m

Something appears to be off with Malta and Tajikistan. Let's investigate:

```
DAT %>%
  filter(location %in% c("Malta","Tajikistan"),
         cause == "Diabetes and kidney diseases") %>%
  mutate(n = case_when(age == 0 ~ 1,
                       age == 1 ~ 4,
                       TRUE ~ 5)) %>%
  ggplot(aes(x = age, y = nLx / n, group = interaction(location, sex))) +
  geom_line()
```

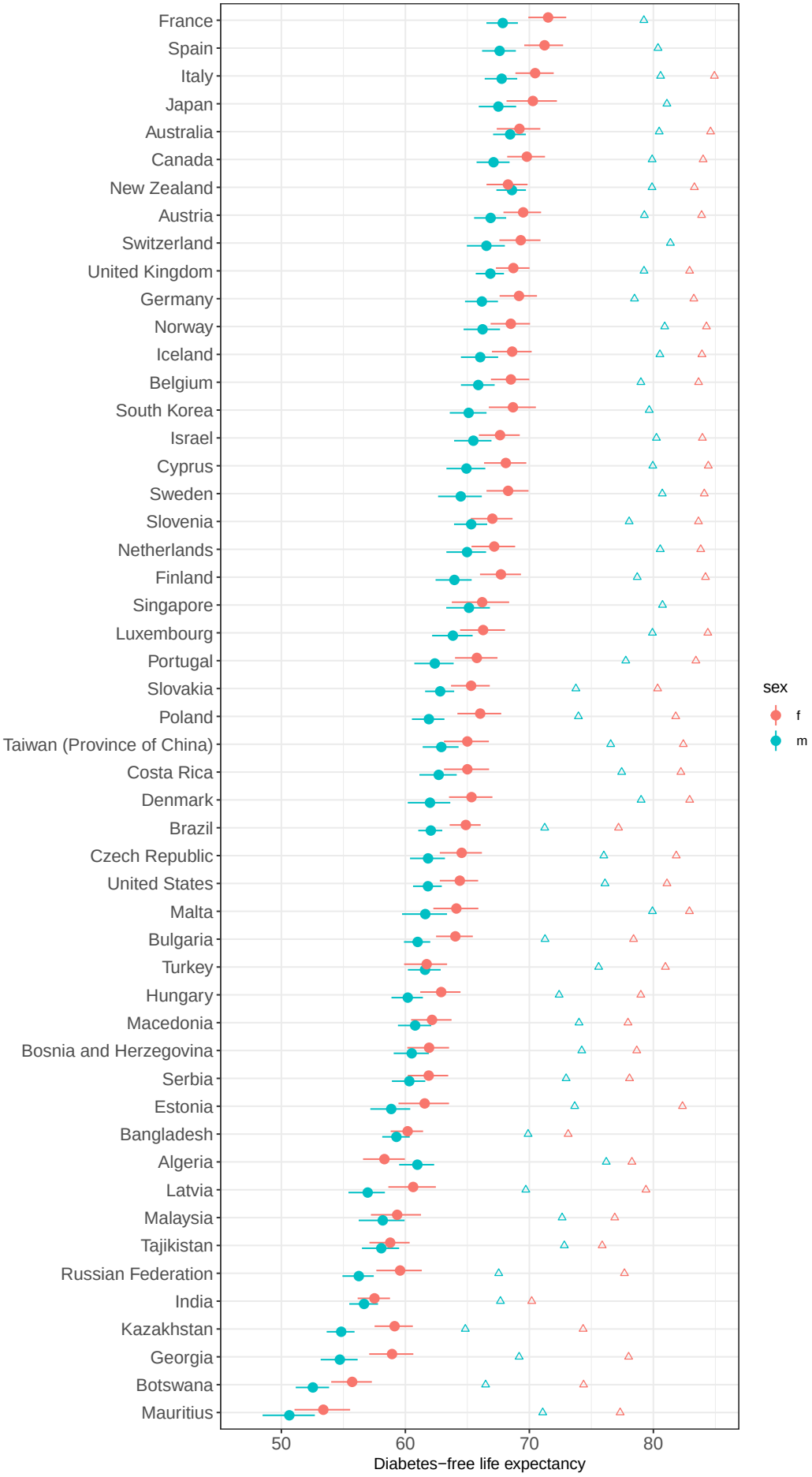


Ah! The issue here is that these two lifetables had a radix of 10000 and not 100000! That's clear to the eye. Rather than dividing by 100000 for all populations, we should have divided by l_0! separately for each population. That change would need to happen early in the HLD processing. The jump at age 80 is expected (it's an open age integral).

Here, we can be pragmatic and just multiply these two series by 10 and re-plot:

```
HLE <-
  DAT %>%
  mutate(radix = ifelse(location %in% c("Malta","Tajikistan"),10000, 100000),
         nLx = nLx  / radix,
         Hx = nLx * (1 - val),
         # High prevalence of a bad condition maps
         # to lower health life, ergo the switcheroo
         Hx_lower = nLx * (1 - upper),
         Hx_upper = nLx * (1 - lower)) %>%
  # Aggregation step
  group_by(location, sex, cause) %>%
  summarize(LE = sum(nLx),
```

```
            HLE = sum(Hx),
            HLE_upper = sum(Hx_upper),
            HLE_lower = sum(Hx_lower),
            .groups = "drop")
HLE %>%
  filter(cause == "Diabetes and kidney diseases") %>%
  ggplot(aes(x = HLE, y = reorder(location, HLE), color = sex)) +
          geom_point(position = position_dodge2(width = .4, reverse = TRUE)) +
          geom_pointrange(aes(xmin = HLE_lower, xmax = HLE_upper),
                          position = position_dodge2(width = .4, reverse = TRUE)) +
  # new data!
  geom_point(data = filter(HLE, cause == "Diabetes and kidney diseases"),
             mapping = aes(x = LE, y = reorder(location, HLE), color = sex),
             shape = 2) +
  xlim(47,85) +
  labs(y = "",
       x = "Diabetes-free life expectancy") +
  theme_bw() +
  theme(axis.text = element_text(size = 12))
```


France
Spain
Italy
Japan
Australia
Canada
New Zealand
Austria
Switzerland
United Kingdom
Germany
Norway
Iceland
Belgium
South Korea
Israel
Cyprus
Sweden
Slovenia
Netherlands
Finland
Singapore
Luxembourg
Portugal
Slovakia
Poland
Taiwan (Province of China)
Costa Rica
Denmark
Brazil
Czech Republic
United States
Malta
Bulgaria
Turkey
Hungary
Macedonia
Bosnia and Herzegovina
Serbia
Estonia
Bangladesh
Algeria
Latvia
Malaysia
Tajikistan
Russian Federation
India
Kazakhstan
Georgia
Botswana
Mauritius
50
60
70
80
Diabetes−free life expectancy
sex
f
m

References

Network, GBD Collaborative. 2018. "Global Burden of Disease Study 2017 (GBD 2017) Results." *Seattle, United States*. http://ghdx.healthdata.org/gbd-results-tool.

Pascariu, Marius D. 2020. *MortalityLaws: Parametric Mortality Models, Life Tables and HMD*. https://CRAN.R-project.org/package=MortalityLaws.

Sullivan, Daniel F. 1971. "A Single Index of Mortality and Morbidity." *HSMHA Health Reports* 86 (4): 347.