Here’s a common situation: you have a folder full of similarly-formatted CSV or otherwise structured text files that you want to get into R quickly and easily. Reading data into R is one of those tasks that can be a real source of frustration for beginners, so I like collecting real-life examples of the many ways it’s become much easier.

This week in class It was working with country-level historical mortality rate estimates. These are available from [mortality.org](http://mortality.org/), a fabulous resource. They have a variety of data available but I was interested in the 1×1 year estimates of mortality for all available countries. By “1×1” I mean that the tables show (for men, women, and in total) age-specific morality rate estimates for yearly ages from 0 to 110 and above, for every available historical year (e.g. from 1850 to 2016 or what have you). So you can have an estimate of the mortality rate for, say, 28 year olds in France in 1935.

Downloading this data gives me a folder of text files, one for each country. (Or rather, country-like unit: there are separate series for, e.g. East Germany, West Germany, and Germany as a whole, for example, along with some countries where sub-populations are broken out historically.) The names of the files are consistently formatted, as is the data inside them, and they have a .txt extension. What I wanted to do was get each one of these files into R, ideally putting them all into a single big table that could be the jumping-off point for subsetting and further analysis.

We know from the documentation provided by [mortality.org](http://mortality.org/) that the files all have the same basic format, which of course makes things much easier. The data is already clean. It’s just a matter of loading it all in efficiently, or “ingesting” it, to use the charming image that seems to be preferred at present.

Here we go. First, some libraries.

library(tidyverse)

library(janitor)

library(here)

## here() starts at /Users/kjhealy/Source/demog

We get a list of the filenames in our raw data folder, along with their full paths. Then we take a look at them.

filenames <- dir(path = here("rawdata"),

pattern = "\*.txt",

full.names = TRUE)

filenames

## [1] "/Users/kjhealy/Source/demog/rawdata/AUS.Mx\_1x1.txt"

## [2] "/Users/kjhealy/Source/demog/rawdata/AUT.Mx\_1x1.txt"

## [3] "/Users/kjhealy/Source/demog/rawdata/BEL.Mx\_1x1.txt"

## [4] "/Users/kjhealy/Source/demog/rawdata/BGR.Mx\_1x1.txt"

## [5] "/Users/kjhealy/Source/demog/rawdata/BLR.Mx\_1x1.txt"

## [6] "/Users/kjhealy/Source/demog/rawdata/CAN.Mx\_1x1.txt"

## [7] "/Users/kjhealy/Source/demog/rawdata/CHE.Mx\_1x1.txt"

## [8] "/Users/kjhealy/Source/demog/rawdata/CHL.Mx\_1x1.txt"

## [9] "/Users/kjhealy/Source/demog/rawdata/CZE.Mx\_1x1.txt"

## [10] "/Users/kjhealy/Source/demog/rawdata/DEUTE.Mx\_1x1.txt"

## [11] "/Users/kjhealy/Source/demog/rawdata/DEUTNP.Mx\_1x1.txt"

## [12] "/Users/kjhealy/Source/demog/rawdata/DEUTW.Mx\_1x1.txt"

## [13] "/Users/kjhealy/Source/demog/rawdata/DNK.Mx\_1x1.txt"

## [14] "/Users/kjhealy/Source/demog/rawdata/ESP.Mx\_1x1.txt"

## [15] "/Users/kjhealy/Source/demog/rawdata/EST.Mx\_1x1.txt"

## [16] "/Users/kjhealy/Source/demog/rawdata/FIN.Mx\_1x1.txt"

## [17] "/Users/kjhealy/Source/demog/rawdata/FRACNP.Mx\_1x1.txt"

## [18] "/Users/kjhealy/Source/demog/rawdata/FRATNP.Mx\_1x1.txt"

## [19] "/Users/kjhealy/Source/demog/rawdata/GBR\_NIR.Mx\_1x1.txt"

## [20] "/Users/kjhealy/Source/demog/rawdata/GBR\_NP.Mx\_1x1.txt"

## [21] "/Users/kjhealy/Source/demog/rawdata/GBR\_SCO.Mx\_1x1.txt"

## [22] "/Users/kjhealy/Source/demog/rawdata/GBRCENW.Mx\_1x1.txt"

## [23] "/Users/kjhealy/Source/demog/rawdata/GBRTENW.Mx\_1x1.txt"

## [24] "/Users/kjhealy/Source/demog/rawdata/GRC.Mx\_1x1.txt"

## [25] "/Users/kjhealy/Source/demog/rawdata/HRV.Mx\_1x1.txt"

## [26] "/Users/kjhealy/Source/demog/rawdata/HUN.Mx\_1x1.txt"

## [27] "/Users/kjhealy/Source/demog/rawdata/IRL.Mx\_1x1.txt"

## [28] "/Users/kjhealy/Source/demog/rawdata/ISL.Mx\_1x1.txt"

## [29] "/Users/kjhealy/Source/demog/rawdata/ISR.Mx\_1x1.txt"

## [30] "/Users/kjhealy/Source/demog/rawdata/ITA.Mx\_1x1.txt"

## [31] "/Users/kjhealy/Source/demog/rawdata/JPN.Mx\_1x1.txt"

## [32] "/Users/kjhealy/Source/demog/rawdata/KOR.Mx\_1x1.txt"

## [33] "/Users/kjhealy/Source/demog/rawdata/LTU.Mx\_1x1.txt"

## [34] "/Users/kjhealy/Source/demog/rawdata/LUX.Mx\_1x1.txt"

## [35] "/Users/kjhealy/Source/demog/rawdata/LVA.Mx\_1x1.txt"

## [36] "/Users/kjhealy/Source/demog/rawdata/NLD.Mx\_1x1.txt"

## [37] "/Users/kjhealy/Source/demog/rawdata/NOR.Mx\_1x1.txt"

## [38] "/Users/kjhealy/Source/demog/rawdata/NZL\_MA.Mx\_1x1.txt"

## [39] "/Users/kjhealy/Source/demog/rawdata/NZL\_NM.Mx\_1x1.txt"

## [40] "/Users/kjhealy/Source/demog/rawdata/NZL\_NP.Mx\_1x1.txt"

## [41] "/Users/kjhealy/Source/demog/rawdata/POL.Mx\_1x1.txt"

## [42] "/Users/kjhealy/Source/demog/rawdata/PRT.Mx\_1x1.txt"

## [43] "/Users/kjhealy/Source/demog/rawdata/RUS.Mx\_1x1.txt"

## [44] "/Users/kjhealy/Source/demog/rawdata/SVK.Mx\_1x1.txt"

## [45] "/Users/kjhealy/Source/demog/rawdata/SVN.Mx\_1x1.txt"

## [46] "/Users/kjhealy/Source/demog/rawdata/SWE.Mx\_1x1.txt"

## [47] "/Users/kjhealy/Source/demog/rawdata/TWN.Mx\_1x1.txt"

## [48] "/Users/kjhealy/Source/demog/rawdata/UKR.Mx\_1x1.txt"

## [49] "/Users/kjhealy/Source/demog/rawdata/USA.Mx\_1x1.txt"

What does each of these files look like? Let’s take a look at the first one, using read\_lines() to show us the top of the file.

read\_lines(filenames[1], n\_max = 5)

## [1] "Australia, Death rates (period 1x1), \tLast modified: 26 Sep 2017; Methods Protocol: v6 (2017)"

## [2] ""

## [3] " Year Age Female Male Total"

## [4] " 1921 0 0.059987 0.076533 0.068444"

## [5] " 1921 1 0.012064 0.014339 0.013225"

All the files have a header section like this. When we read the data in we’ll want to ignore that and go straight to the data. But seeing as it’s there, we can make use of it to grab the name of the country. It saves us typing it ourselves. Let’s say we’d also like to have a code-friendly version of those names (i.e., in lower-case with underscores instead of spaces). And finally—while we’re at it—let’s grab those all-caps country codes used in the file names, too. We write three functions:

* get\_country\_name() grabs the first word or words on the first line of each file, up to the first comma. That’s our country name.
* shorten\_name() makes the names lower-case and replaces spaces with underscores, and also shortens “The United States of America” to “USA”.
* make\_ccode() wraps a regular expression that finds and extracts the capitalized country codes in the file names.

get\_country\_name <- function(x){

read\_lines(x, n\_max = 1) %>%

str\_extract(".+?,") %>%

str\_remove(",")

}

shorten\_name <- function(x){

str\_replace\_all(x, " -- ", " ") %>%

str\_replace("The United States of America", "USA") %>%

snakecase::to\_any\_case()

}

make\_ccode <- function(x){

str\_extract(x, "[:upper:]+((?=\\.))")

}

Now we create a tibble of summary information by mapping the functions to the filenames.

countries <- tibble(country = map\_chr(filenames, get\_country\_name),

cname = map\_chr(country, shorten\_name),

ccode = map\_chr(filenames, make\_ccode),

path = filenames)

countries

## # A tibble: 49 x 4

## country cname ccode path

##

## 1 Australia australia AUS /Users/kjhealy/Source/demog/rawdata/AUS.M…

## 2 Austria austria AUT /Users/kjhealy/Source/demog/rawdata/AUT.M…

## 3 Belgium belgium BEL /Users/kjhealy/Source/demog/rawdata/BEL.M…

## 4 Bulgaria bulgaria BGR /Users/kjhealy/Source/demog/rawdata/BGR.M…

## 5 Belarus belarus BLR /Users/kjhealy/Source/demog/rawdata/BLR.M…

## 6 Canada canada CAN /Users/kjhealy/Source/demog/rawdata/CAN.M…

## 7 Switzerland switzerland CHE /Users/kjhealy/Source/demog/rawdata/CHE.M…

## 8 Chile chile CHL /Users/kjhealy/Source/demog/rawdata/CHL.M…

## 9 Czechia czechia CZE /Users/kjhealy/Source/demog/rawdata/CZE.M…

## 10 East Germa… east\_germa… DEUTE /Users/kjhealy/Source/demog/rawdata/DEUTE…

## # … with 39 more rows

Nice. We could have written each of those operations as anonymous functions directly inside of map\_chr(). This would have been more compact. But often it can be useful to break out the steps as shown here, for clarity—especially if map() operations have a tendency to break your brain, as they do mine.

We still haven’t touched the actual data files, of course. But now we can just use this countries table as the basis for reading in, I mean *ingesting*, everything in the files. We’re going to just add a list column named data to the end of the table and put the data for each country in it. We’ll temporarily unnest it to clean the column names and recode the age variable, then drop the file paths column and nest the data again.

The hard work is done by the map() call. This time we will use ~ formula notation inside map() to write what we want to do. We’re going to feed every filename in path to read\_table(), one at a time. We tell read\_table() to skip the first two lines of every file it reads, and also tell it that in these files missing data are represented by a . character. Everything read in ends up in a new list column named data.

mortality <- countries %>%

mutate(data = map(path,

~ read\_table(., skip = 2, na = "."))) %>%

unnest(cols = c(data)) %>%

clean\_names() %>%

mutate(age = as.integer(recode(age, "110+" = "110"))) %>%

select(-path) %>%

nest(data = c(year:total))

mortality

## # A tibble: 49 x 4

## country cname ccode data

## >

## 1 Australia australia AUS [10,434 × 5]

## 2 Austria austria AUT [7,881 × 5]

## 3 Belgium belgium BEL [19,425 × 5]

## 4 Bulgaria bulgaria BGR [7,104 × 5]

## 5 Belarus belarus BLR [6,438 × 5]

## 6 Canada canada CAN [10,101 × 5]

## 7 Switzerland switzerland CHE [15,651 × 5]

## 8 Chile chile CHL [1,887 × 5]

## 9 Czechia czechia CZE [7,437 × 5]

## 10 East Germany east\_germany DEUTE [6,660 × 5]

## # … with 39 more rows

And we’re done. Forty nine tables of data smoothly imported and bundled together. Each of the country-level data tables is a row in data that we can take a look at as we like:

mortality %>%

filter(country == "Austria") %>%

unnest(cols = c(data))

## # A tibble: 7,881 x 8

## country cname ccode year age female male total

##

## 1 Austria austria AUT 1947 0 0.0798 0.0994 0.0899

## 2 Austria austria AUT 1947 1 0.00657 0.00845 0.00753

## 3 Austria austria AUT 1947 2 0.00425 0.00469 0.00447

## 4 Austria austria AUT 1947 3 0.00337 0.00340 0.00339

## 5 Austria austria AUT 1947 4 0.00235 0.00270 0.00253

## 6 Austria austria AUT 1947 5 0.00174 0.00195 0.00184

## 7 Austria austria AUT 1947 6 0.00131 0.00152 0.00142

## 8 Austria austria AUT 1947 7 0.00132 0.00169 0.00151

## 9 Austria austria AUT 1947 8 0.00115 0.00149 0.00132

## 10 Austria austria AUT 1947 9 0.000836 0.000997 0.000918

## # … with 7,871 more rows

Now you can get on with the actual analysis.

There isn’t anything especially unusual in the steps shown here. It’s just a pretty common operation that’s worth knowing how to do cleanly. One nice thing about this approach is that it’s immediately applicable to, say, a folder containing the 5-year mortality estimates rather than the 1 year estimates. You don’t have to do anything new, and there’s no mucking around with manually naming files and so on.