**Looking at COVID-19 Data from the European Centers for Disease Control**

Each day, the [ECDC](https://www.ecdc.europa.eu/) publishes a [a summary spreadsheet of global case and death counts](https://www.ecdc.europa.eu/en/publications-data/download-todays-data-geographic-distribution-covid-19-cases-worldwide) since the beginning of the epidemic. This is good data collated by an EU-wide agency, and it’s what I’ve been using to keep up with the trends. Here’s what I’ve been doing to get it into R. Again my principal reason for sharing this code is *not* to add much of anything on the public side. It’s much more of a pedagogical exercise. If you want to look at this data, here’s one way to do that. Along the way I’ll talk about a few of the things needed to work with the data in a reasonably clean way. Then I’ll end up drawing the plot that everyone draws—showing cumulative trends by country in deaths, counted in days since a threshold level of fatalities.

**Preparation**

First we load some libraries to help us out.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | library(tidyverse)  library(lubridate)  library(here)  library(janitor)  library(socviz)  library(ggrepel)  library(paletteer) |

Next, we set things up by writing some functions that will help us grab and clean the data. In reality, of course, these functions got written piecemeal and were then cleaned up and moved to the front of the file. I didn’t sit down and write them off the top of my head.

The first one is going to grab the spreadsheet from the ECDC and both save the .xlsx file to our data/ folder and create a tibble of the results.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | ## Download today's excel file, saving it to data/ and reading it in  get\_ecdc\_data <- function(url = "<https://www.ecdc.europa.eu/sites/default/files/documents/>",  fname = "COVID-19-geographic-distribution-worldwide-",  date = lubridate::today(),  ext = "xlsx",  dest = "data") {    target <- paste0(url, fname, date, ".", ext)  message("target: ", target)  destination <- fs::path(here::here("data"), paste0(fname, date), ext = ext)  message("saving to: ", destination)    tf <- tempfile(fileext = ext)  curl::curl\_download(target, tf)  fs::file\_copy(tf, destination)    switch(ext,  xls = janitor::clean\_names(readxl::read\_xls(tf)),  xlsx = janitor::clean\_names(readxl::read\_xlsx(tf))  )  } |

Things to notice: We have to use curl\_download() to get the file, because read\_xls cannot directly grab an Excel file from a URL in the way that e.g. read\_csv() can for a .csv file. So we create a temporary file handle and use curl to download the data file to it. Then we copy the file to its permanent home in our data/ folder, and we read the target file into R with the appropriate readxl function.

As we’ll see in a moment, the country codes contained in the ECDC data are not quite standard. It will be useful in the long run to make sure that every country has standardized two- and three-letter abbreviations. Some of the countries in the ECDC’s geo\_id variable are missing these. This is a very common situation in data cleaning, where we have a big table with some data we know is missing (e.g., a country code), *and* we know for sure which cases the data are missing for, *and* we have a little lookup table that can fill in the blanks. The operation we will need to perform here is called a *coalescing join*. Before I knew that’s what it was called, I used to do this manually (I’ll show you below):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | coalesce\_join <- function(x, y,  by = NULL, suffix = c(".x", ".y"),  join = dplyr::full\_join, ...) {  joined <- join(x, y, by = by, suffix = suffix, ...)  # names of desired output  cols <- union(names(x), names(y))    to\_coalesce <- names(joined)[!names(joined) %in% cols]  suffix\_used <- suffix[ifelse(endsWith(to\_coalesce, suffix[1]), 1, 2)]  # remove suffixes and deduplicate  to\_coalesce <- unique(substr(  to\_coalesce,  1,  nchar(to\_coalesce) - nchar(suffix\_used)  ))    coalesced <- purrr::map\_dfc(to\_coalesce, ~dplyr::coalesce(  joined[[paste0(.x, suffix[1])]],  joined[[paste0(.x, suffix[2])]]  ))  names(coalesced) <- to\_coalesce    dplyr::bind\_cols(joined, coalesced)[cols]  } |

Next we set up some country codes using ISO2 and ISO3 abbreviations.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62 | iso3\_cnames <- read\_csv("data/countries\_iso3.csv")  iso2\_to\_iso3 <- read\_csv("data/iso2\_to\_iso3.csv")  cname\_table <- left\_join(iso3\_cnames, iso2\_to\_iso3)  cname\_table  # A tibble: 249 x 3  iso3 cname iso2  <chr> <chr> <chr>  1 AFG Afghanistan AF  2 ALA Åland Islands AX  3 ALB Albania AL  4 DZA Algeria DZ  5 ASM American Samoa AS  6 AND Andorra AD  7 AGO Angola AO  8 AIA Anguilla AI  9 ATA Antarctica AQ  10 ATG Antigua and Barbuda AG  # … with 239 more rows  eu <- c("AUT", "BEL", "BGR", "HRV", "CYP", "CZE", "DNK", "EST", "FIN", "FRA",  "DEU", "GRC", "HUN", "IRL", "ITA", "LVA", "LTU", "LUX", "MLT", "NLD",  "POL", "PRT", "ROU", "SVK", "SVN", "ESP", "SWE", "GBR")  europe <- c("ALB", "AND", "AUT", "BLR", "BEL", "BIH", "BGR", "HRV", "CYP", "CZE",  "DNK", "EST", "FRO", "FIN", "FRA", "DEU", "GIB", "GRC", "HUN", "ISL",  "IRL", "ITA", "LVA", "LIE", "LTU", "LUX", "MKD", "MLT", "MDA", "MCO",  "NLD", "NOR", "POL", "PRT", "ROU", "RUS", "SMR", "SRB", "SVK", "SVN",  "ESP", "SWE", "CHE", "UKR", "GBR", "VAT", "RSB", "IMN", "MNE")  north\_america <- c("AIA", "ATG", "ABW", "BHS", "BRB", "BLZ", "BMU", "VGB", "CAN", "CYM",  "CRI", "CUB", "CUW", "DMA", "DOM", "SLV", "GRL", "GRD", "GLP", "GTM",  "HTI", "HND", "JAM", "MTQ", "MEX", "SPM", "MSR", "ANT", "KNA", "NIC",  "PAN", "PRI", "KNA", "LCA", "SPM", "VCT", "TTO", "TCA", "VIR", "USA",  "SXM")  south\_america <- c("ARG", "BOL", "BRA", "CHL", "COL", "ECU", "FLK", "GUF", "GUY", "PRY",  "PER", "SUR", "URY", "VEN")  africa <- c("DZA", "AGO", "SHN", "BEN", "BWA", "BFA", "BDI", "CMR", "CPV", "CAF",  "TCD", "COM", "COG", "DJI", "EGY", "GNQ", "ERI", "ETH", "GAB", "GMB",  "GHA", "GNB", "GIN", "CIV", "KEN", "LSO", "LBR", "LBY", "MDG", "MWI",  "MLI", "MRT", "MUS", "MYT", "MAR", "MOZ", "NAM", "NER", "NGA", "STP",  "REU", "RWA", "STP", "SEN", "SYC", "SLE", "SOM", "ZAF", "SHN", "SDN",  "SWZ", "TZA", "TGO", "TUN", "UGA", "COD", "ZMB", "TZA", "ZWE", "SSD",  "COD")  asia <- c("AFG", "ARM", "AZE", "BHR", "BGD", "BTN", "BRN", "KHM", "CHN", "CXR",  "CCK", "IOT", "GEO", "HKG", "IND", "IDN", "IRN", "IRQ", "ISR", "JPN",  "JOR", "KAZ", "PRK", "KOR", "KWT", "KGZ", "LAO", "LBN", "MAC", "MYS",  "MDV", "MNG", "MMR", "NPL", "OMN", "PAK", "PHL", "QAT", "SAU", "SGP",  "LKA", "SYR", "TWN", "TJK", "THA", "TUR", "TKM", "ARE", "UZB", "VNM",  "YEM", "PSE")  oceania <- c("ASM", "AUS", "NZL", "COK", "FJI", "PYF", "GUM", "KIR", "MNP", "MHL",  "FSM", "UMI", "NRU", "NCL", "NZL", "NIU", "NFK", "PLW", "PNG", "MNP",  "SLB", "TKL", "TON", "TUV", "VUT", "UMI", "WLF", "WSM", "TLS") |

**Now Actually Get the Data**

The next step is to read the data. The file *should* be called COVID-19-geographic-distribution-worldwide- with the date appended and the extension .xlsx. But as it turns out there is a typo in the filename. The distribution part is misspelled disbtribution. I think it must have been introduced early on in the data collection process and so far—possibly by accident, but also possibly so as not to break a thousand scripts like this one—they have not been fixing the typo.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | covid\_raw <- get\_ecdc\_data(url = "<https://www.ecdc.europa.eu/sites/default/files/documents/>",  fname = "COVID-19-geographic-disbtribution-worldwide-",  ext = "xlsx")  covid\_raw  # A tibble: 6,012 x 8  date\_rep day month year cases deaths countries\_and\_t…  <dttm> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>  1 2020-03-21 00:00:00 21 3 2020 2 0 Afghanistan  2 2020-03-20 00:00:00 20 3 2020 0 0 Afghanistan  3 2020-03-19 00:00:00 19 3 2020 0 0 Afghanistan  4 2020-03-18 00:00:00 18 3 2020 1 0 Afghanistan  5 2020-03-17 00:00:00 17 3 2020 5 0 Afghanistan  6 2020-03-16 00:00:00 16 3 2020 6 0 Afghanistan  7 2020-03-15 00:00:00 15 3 2020 3 0 Afghanistan  8 2020-03-11 00:00:00 11 3 2020 3 0 Afghanistan  9 2020-03-08 00:00:00 8 3 2020 3 0 Afghanistan  10 2020-03-02 00:00:00 2 3 2020 0 0 Afghanistan  # … with 6,002 more rows, and 1 more variable: geo\_id |

That’s our base data. The get\_ecdc\_data() function uses file\_copy() from the fs library to move the temporary file to the data/ folder. It will not overwrite a file if it finds one with that name already there. So if you grab the data more than once a day, you’ll need to decide what to do with the file you already downloaded.

The geo\_id country code column isn’t visible here. We’re going to duplicate it (naming it iso2) and then join our table of two- and three-letter country codes. It has an iso2 column as well.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | covid <- covid\_raw %>%  mutate(date = lubridate::ymd(date\_rep),  iso2 = geo\_id)  ## merge in the iso country names  covid <- left\_join(covid, cname\_table)  covid  # A tibble: 6,012 x 12  date\_rep day month year cases deaths countries\_and\_t…  <dttm> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>  1 2020-03-21 00:00:00 21 3 2020 2 0 Afghanistan  2 2020-03-20 00:00:00 20 3 2020 0 0 Afghanistan  3 2020-03-19 00:00:00 19 3 2020 0 0 Afghanistan  4 2020-03-18 00:00:00 18 3 2020 1 0 Afghanistan  5 2020-03-17 00:00:00 17 3 2020 5 0 Afghanistan  6 2020-03-16 00:00:00 16 3 2020 6 0 Afghanistan  7 2020-03-15 00:00:00 15 3 2020 3 0 Afghanistan  8 2020-03-11 00:00:00 11 3 2020 3 0 Afghanistan  9 2020-03-08 00:00:00 8 3 2020 3 0 Afghanistan  10 2020-03-02 00:00:00 2 3 2020 0 0 Afghanistan  # … with 6,002 more rows, and 5 more variables: geo\_id ,  # date , iso2 , iso3 , cname |

At this point we can notice a couple of things about the dataset. For example, not everything in the dataset is a country. This one’s a cruise ship:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | ## Looks like a missing data code  covid %>%  filter(cases == -9)  # A tibble: 1 x 12  date\_rep day month year cases deaths countries\_and\_t…  <dttm> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>  1 2020-03-10 00:00:00 10 3 2020 -9 1 Cases\_on\_an\_int…  # … with 5 more variables: geo\_id , date , iso2 ,  # iso3 , cname |

We can also learn, using an anti\_join() that not all the ECDC’s geo\_id country codes match up with the ISO codes:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | anti\_join(covid, cname\_table) %>%  select(geo\_id, countries\_and\_territories, iso2, iso3, cname) %>%  distinct()  # A tibble: 7 x 5  geo\_id countries\_and\_territories iso2 iso3 cname  <chr> <chr> <chr> <chr> <chr>  1 JPG11668 Cases\_on\_an\_international\_conveyance\_J… JPG116… <NA> <NA>  2 PYF French\_Polynesia PYF <NA> <NA>  3 EL Greece EL <NA> <NA>  4 XK Kosovo XK <NA> <NA>  5 NA Namibia NA <NA> <NA>  6 AN Netherlands\_Antilles AN <NA> <NA>  7 UK United\_Kingdom UK <NA> <NA> |

Let’s fix this. I made a small crosswalk file that can be coalesced into the missing values. In an added little wrinkle, we need to specify the na argument in read\_csv explicity because the missing country codes include Namibia, which has an ISO country code of “NA”! This is different from the missing data code NA but read\_csv() won’t know this by default.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | cname\_xwalk <- read\_csv("data/ecdc\_to\_iso2\_xwalk.csv",  na = "")  cname\_xwalk  # A tibble: 4 x 3  geo\_id iso3 cname  <chr> <chr> <chr>  1 UK GBR United Kingdom  2 EL GRC Greece  3 NA NAM Namibia  4 XK XKV Kosovo |

I used to do coalescing like this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | # covid <- covid %>%  # left\_join(cname\_xwalk, by = "geo\_id") %>%  # mutate(iso3 = coalesce(iso3.x, iso3.y),  # cname = coalesce(cname.x, cname.y)) %>%  # select(-iso3.x, -iso3.y, cname.x, cname.y) |

Actually, I *used* to do it using match() and some index vectors, like an animal. But now I can use Edward Visel’s handy function instead.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | covid <- coalesce\_join(covid, cname\_xwalk,  by = "geo\_id", join = dplyr::left\_join)  ## Take a look again  anti\_join(covid, cname\_table) %>%  select(geo\_id, countries\_and\_territories, iso2, iso3, cname) %>%  distinct()  # A tibble: 7 x 5  geo\_id countries\_and\_territories iso2 iso3 cname  <chr> <chr> <chr> <chr> <chr>  1 JPG11668 Cases\_on\_an\_international\_convey… JPG116… <NA> <NA>  2 PYF French\_Polynesia PYF <NA> <NA>  3 EL Greece EL GRC Greece  4 XK Kosovo XK XKV Kosovo  5 NA Namibia NA NAM Namibia  6 AN Netherlands\_Antilles AN <NA> <NA>  7 UK United\_Kingdom UK GBR United Kin… |

Looks like a couple of new territories have been added to the ECDC file since I made the crosswalk file. I’ll have to update that soon.

**Calculate and Plot Cumulative Mortality**

Now we can actually analyze the data (in the privacy of our own home). Let’s draw the plot that everyone draws, looking at cumulative counts. I think it’s better at this point to plot cumulative deaths rather than cumulative reported cases, given that there’s so much unevenness in case reporting. The mortality counts aren’t free of that, but it’s not as much of a problem. We’ll take an arbitrary threshold for number of deaths, let’s say ten, start every country from zero days when they hit ten deaths, and count the cumulative deaths since that day.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | cov\_curve <- covid %>%  select(date, cname, iso3, cases, deaths) %>%  drop\_na(iso3) %>%  group\_by(iso3) %>%  arrange(date) %>%  mutate(cu\_cases = cumsum(cases),  cu\_deaths = cumsum(deaths)) %>%  filter(cu\_deaths > 9) %>%  mutate(days\_elapsed = date - min(date),  end\_label = ifelse(date == max(date), cname, NA))  cov\_curve  # A tibble: 245 x 9  # Groups: iso3 [21]  date cname iso3 cases deaths cu\_cases cu\_deaths days\_elapsed  <date> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <drtn>  1 2020-01-22 China CHN 140 11 526 17 0 days  2 2020-01-23 China CHN 97 0 623 17 1 days  3 2020-01-24 China CHN 259 9 882 26 2 days  4 2020-01-25 China CHN 441 15 1323 41 3 days  5 2020-01-26 China CHN 665 15 1988 56 4 days  6 2020-01-27 China CHN 787 25 2775 81 5 days  7 2020-01-28 China CHN 1753 25 4528 106 6 days  8 2020-01-29 China CHN 1466 26 5994 132 7 days  9 2020-01-30 China CHN 1740 38 7734 170 8 days  10 2020-01-31 China CHN 1980 43 9714 213 9 days  # … with 235 more rows, and 1 more variable: end\_label |

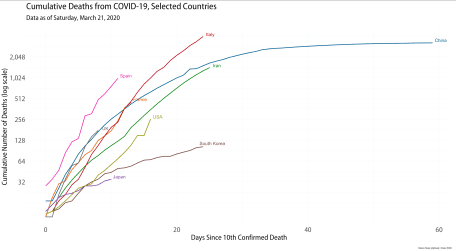
See how at the end there we create an end\_label variable for use in the plot. It only has values for the most recent day in the dataset (i.e. the country name if date is max(date), otherwise NA).

Now we’ll narrow our focus to a few countries and make the plot.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | focus\_cn <- c("CHN", "GBR", "USA", "IRN", "JPN",  "KOR", "ITA", "FRA", "ESP")  cov\_curve %>%  filter(iso3 %in% focus\_cn) %>% ## focus on just a few countries, defined above  mutate(end\_label = recode(end\_label, `United States` = "USA",  `Iran, Islamic Republic of` = "Iran",  `Korea, Republic of` = "South Korea",  `United Kingdom` = "UK")) %>%  ggplot(mapping = aes(x = days\_elapsed, y = cu\_deaths,  color = cname, label = end\_label,  group = cname)) +  geom\_line(size = 0.8) +  geom\_text\_repel(nudge\_x = 1.1,  nudge\_y = 0.1,  segment.color = NA) +  guides(color = FALSE) +  scale\_color\_manual(values = prismatic::clr\_darken(paletteer\_d("ggsci::category20\_d3"), 0.2)) +  scale\_y\_continuous(labels = scales::comma\_format(accuracy = 1),  breaks = 2^seq(4, 11),  trans = "log2") +  labs(x = "Days Since 10th Confirmed Death",  y = "Cumulative Number of Deaths (log scale)",  title = "Cumulative Deaths from COVID-19, Selected Countries",  subtitle = paste("Data as of", format(max(cov\_curve$date), "%A, %B %e, %Y")),  caption = "Kieran Healy @kjhealy / Data: ECDC") +  theme(plot.title = element\_text(size = rel(2), face = "bold"),  plot.subtitle = element\_text(size = rel(1.5)),  axis.text.y = element\_text(size = rel(2)),  axis.title.x = element\_text(size = rel(1.5)),  axis.title.y = element\_text(size = rel(1.5)),  axis.text.x = element\_text(size = rel(2)),  legend.text = element\_text(size = rel(2))  ) |

Again, a few small details polish the plot. We do a quick bit of recoding on the end\_label to shorten some country names, and use geom\_text\_repel() to put the labels at the end of the line. We get our y-axis breaks with 2^seq(4, 11), which (as case numbers rise) will be easier to extend than manually typing all the numbers. It’s useful to look at the doubling time, which base 2 helps you see, rather than powers of ten. (The graphs won’t look any different.) Finally on the thematic side we can date-stamp the title of the graph using the opaque with paste("Data as of", format(max(cov\_curve$date), "%A, %B %e, %Y")).

And here’s our figure.



Cumulative COVID-19 Deaths