General Decomposition Practice 1

Tim Riffe

21 Oct 2020

Setup

We will use some different packages. First let's install `DemoDecomp`

```
library(remotes)
install_github("timriffe/DemoDecomp")
```

And some other packages we'll need for this or that

```
library(DemoDecomp)
library(osfr)
library(tidyverse)
library(lubridate)
library(wpp2019)
```

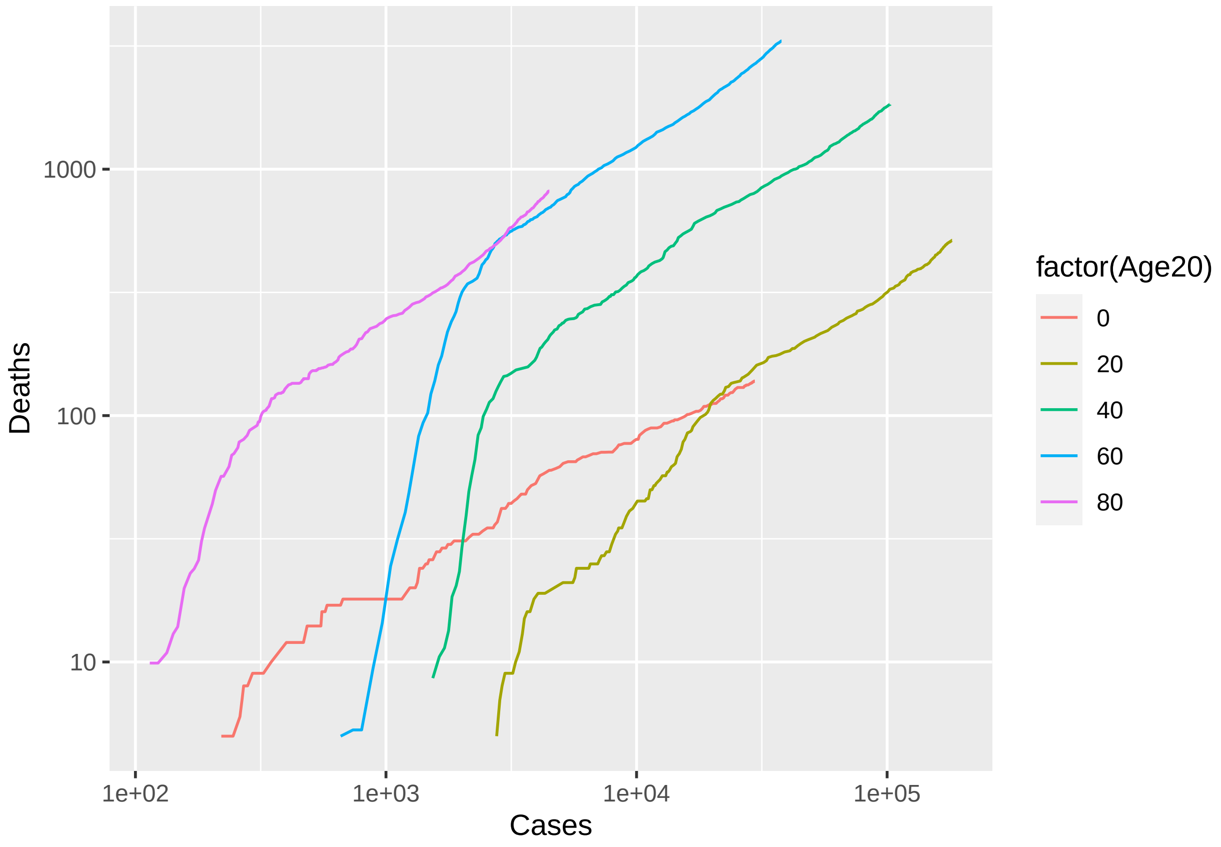
First let's test equivalency between Kitagawa and the general methods. We'll use the example of decomposing differences in crude case fatality ratios of COVID-19, then we'll extend the decomposition creatively (if we can).

```
osf_retrieve_file("7tnfh") %>%
  osf_download(conflicts = "overwrite")
COVerAGEDB5 <-  read_csv("Output_5.zip",
                     skip = 3,
                     col_types = "cccccciiddd")
# convert to date class
COV5 <-
  COVerAGEDB5 %>%
  filter(Country %in% c("Philippines","Germany"),
         Region == "All",
         Sex == "b") %>%
  select(-Tests) %>%
  mutate(Date = dmy(Date),
         CFRx = Deaths / Cases)

rm(COVerAGEDB5)
gc()
```

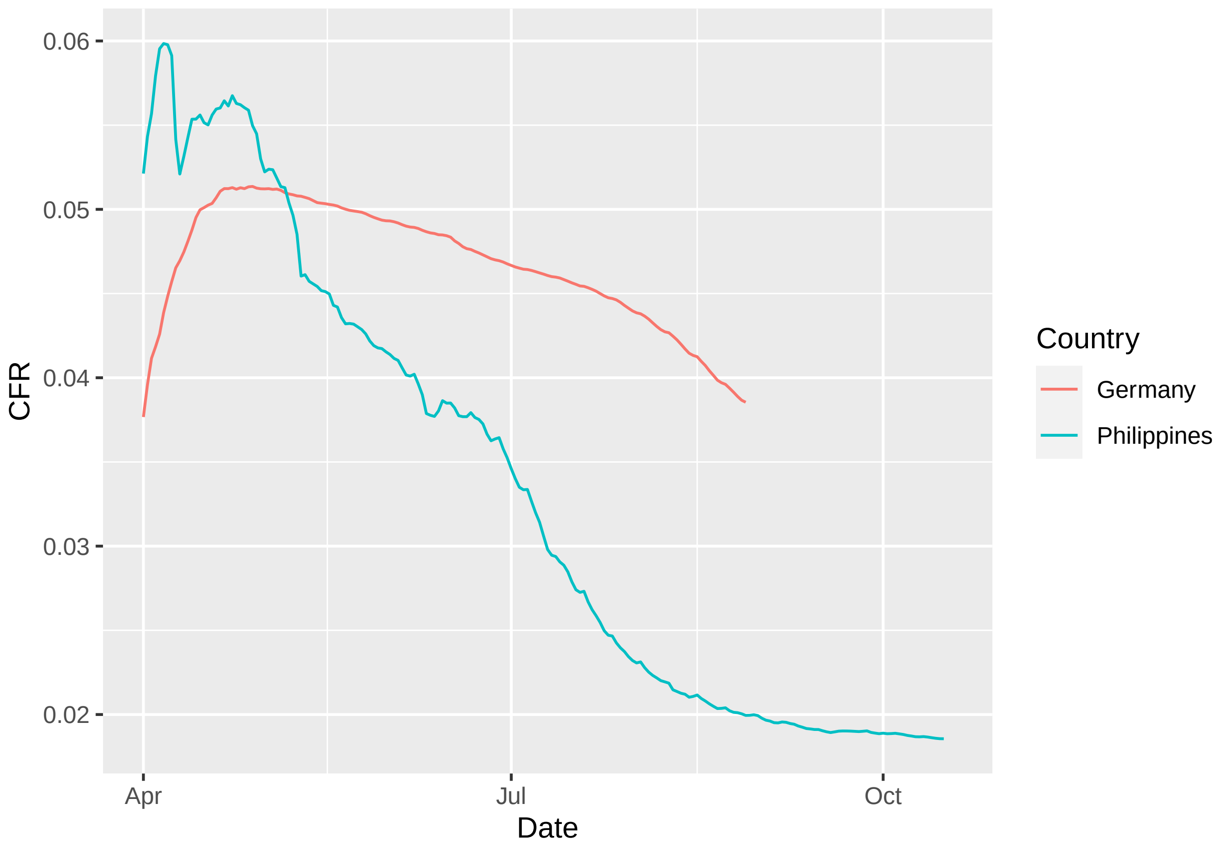
Let's examine cases and deaths by age in Germany. You can switch it to Philippines.

```
COV5 %>%
  filter(Country == "Philippines") %>%
  mutate(Age20 = Age - Age %% 20) %>%
  group_by(Date,Age20) %>%
  summarize(Cases = sum(Cases),
            Deaths = sum(Deaths)) %>%
  filter(Cases >= 20,
         Deaths >= 5) %>%
  ggplot(aes(x = Cases,
             y = Deaths,
             color = factor(Age20),
             group = Age20)) +
  geom_line() +
  scale_x_log10()+
  scale_y_log10()
```



And what about time series of CFR?

```
CFR <-
COV5 %>%
  filter(Date >= dmy("01.04.2020")) %>%
  group_by(Country,Date) %>%
  summarize(Deaths = sum(Deaths),
            Cases = sum(Cases)) %>%
  ungroup() %>%
  mutate(CFR = Deaths / Cases)
CFR %>%
  ggplot(aes(x=Date,y=CFR,color=Country,group=Country)) +
  geom_line()
```



Let's decompose CFR at a particular date. Nevermind the question of aligning deaths and cases properly in calendar time. Ignore other measurement issues at play for this exercise. Let's take these data at face value as ask how much of the CFR difference on July 15 was due to the age-specific CFR itself and how much due to the age structure of cases and how much is due to age-structured rate differences.

```
# These are the CFR values on July 15
CFR %>%
  filter(Date == dmy("15.07.2020"))
```

```
## # A tibble: 2 x 5
##   Country     Date        Deaths   Cases    CFR
##   <chr>       <date>       <dbl>   <dbl>  <dbl>
## 1 Germany     2020-07-15   9171. 200517. 0.0457
## 2 Philippines 2020-07-15   2668.  93700  0.0285
```

```
# This is the difference we want to explain
zeta <-
  CFR %>%
  filter(
    Date == dmy("15.07.2020")) %>%
  pull(CFR) %>%
  diff()
zeta
```

```
## [1] -0.01726446
```

Make a CFR calculating function

We need to design the function in a particular way. It should take all parameters as a single vector. There are different ways to get from this data to CFR. We're interested in the age distribution of cases:

$$c_x = \frac{Cases_x}{\sum(Cases)}$$

In this case CFR is the c_x-weighted average of CFR_x:

$$CFR = \sum c_x \cdot CFR_x = \frac{\sum Deaths_x}{\sum Cases_x}$$

We prefer to decompose using c_x because it is a distribution summing to 1, and can therefore be used as a weight without further ado. Indeed, this setup allows for Kitagawa decomposing directly.

Practically, we extract `c_x` and `CFRx`, and stack them in a single vector.

```r
DE <- COV5 %>%
  filter(Country == "Germany",
         Date == dmy("15.07.2020")) %>%
  mutate(c = Cases / sum(Cases)) %>%
  select(CFRx, c) %>%
  as.matrix() %>% # convert to matrix
  c()             # cheap vec operator

PH <- COV5 %>%
  filter(Country == "Philippines",
         Date == dmy("15.07.2020")) %>%
  mutate(c = Cases / sum(Cases)) %>%
  select(CFRx, c) %>%
  as.matrix() %>%
  c()
```

Now we just need to write a function that knows what to do with this to calculate crude CFR. One way might look like this. Before calculating the result, it is convenient to reshape the parameters from a vector to somethig easier to work with. In practice this function might actually be a wrapper around some other code you've written (step 2).

```r
calc_CFR <- function(pars){

  # step 1: convert from vec into something we can use
  N        <- length(pars)
  dim(pars)<- c(N / 2, 2) # rows, columns

  # step 2: calculate the result
  sum(pars[, 1] * pars[, 2])
}
calc_CFR(DE) - calc_CFR(PH)
```

```
## [1] 0.01726446
```

Now, for exposition and comparison, let's do a Kitagawa decomp. Usually we'd write a straightforward function like this:

```r
kitagawa <- function(r1,s1,r2,s2){
  rd <- r2 - r1
  sd <- s2 - s1
  rm <- (r1 + r2) / 2
  sm <- (s1 + s2) / 2

  reff <- rd * sm
  seff <- sd * rm
  list(r = reff, s = seff)
}
```

Here parameters are separate by type. However, for compatibility with the general methods, we'll write a tiny wrapper that anticipates θ as a stacked vector, sorry if this confuses things a bit:

```
kitagawa_vec <- function(pars1, pars2){

  # 1) turn back into something we can feed to kitagawa()
  N <- length(pars1)
  dim(pars1) <- c(N / 2, 2)
  dim(pars2) <- c(N / 2, 2)

  # 2) calcualte result and return as vec organized in same way
  kitagawa(r1 = pars1[, 1],
           s1 = pars1[, 2],
           r2 = pars2[, 1],
           s2 = pars2[, 2]) %>%
    unlist()
}
```

Now let's see what we get

```
# perform the decomp
dec_kit      <- kitagawa_vec(DE,PH)

# organize results
dim(dec_kit) <- c(21,2)
colnames(dec_kit) <- c("r","s")
# rate vs structure effects
colSums(dec_kit)
```

```
##           r           s
##  0.01389780 -0.03116226
```

```
# Compare

sum(dec_kit) # exact :-)
```

```
## [1] -0.01726446
```

```
             # but not unique :-()
```

Now let's do the same with the three general approaches, we use `calc_CFR()`

```
dec_h <- horiuchi(calc_CFR, DE, PH, N = 20)
dec_s <- stepwise_replacement(calc_CFR, DE, PH, symmetrical = TRUE)
dec_c <- ltre(calc_CFR, DE, PH)

zeta
```

```
## [1] -0.01726446
```

```r
sum(dec_kit)
```

```
## [1] -0.01726446
```

```r
sum(dec_h)
```

```
## [1] -0.01726446
```

```r
sum(dec_s)
```

```
## [1] -0.01726446
```

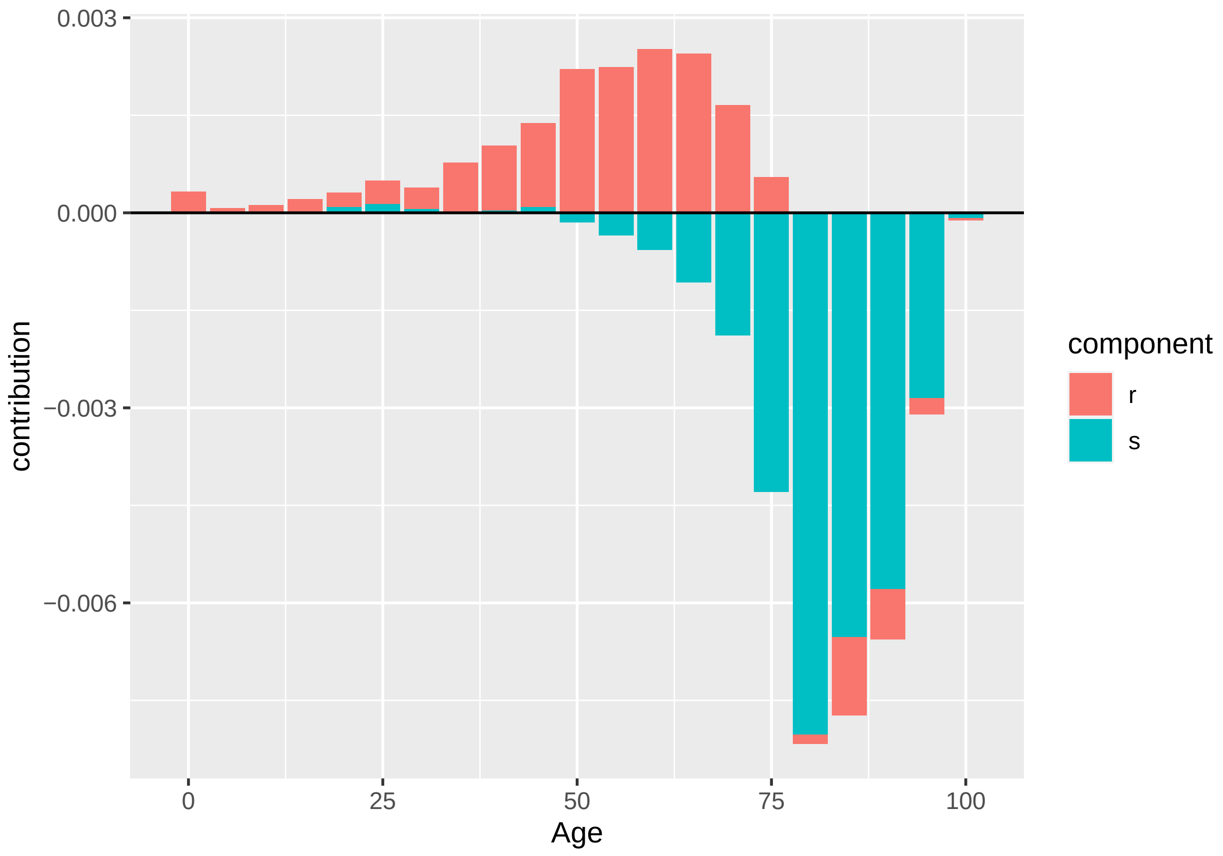
```r
sum(dec_c) # approximate
```

```
## [1] -0.0187029
```

Visualize results?

```r
colnames(dec_kit) <- c("r","s")
dec_kit %>%
  as.tibble() %>%
  mutate(Age = seq(0, 100, by = 5)) %>%
  pivot_longer(r:s,
               names_to = "component",
               values_to = "contribution") %>%
  ggplot(aes(x = Age,
             y = contribution,
             fill = component,
             group = component)) +
  geom_bar(stat = "identity") +
  geom_hline(yintercept = 0)
```

```
## Warning: `as.tibble()` is deprecated as of tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```



They will all look the same visually in this case.

Increase the complexity

Note CFR does NOT take into account population structure. It just looks at structure in terms of the age pattern of cases. Different transmission pathways could hypothetically give the same age structure of cases in two populations with very different age structures. Let's see if we can net out all three effects. For this, we'll need to merge in population structure from another source, and we'll need to reframe the CFR calc somewhat. Pardon this bit of wrangling in order to get the data mergeable.

```r
library(wpp2019)
data(popF)
data(popM)
fem <-
  popF %>%
  select(name, age, `2020`) %>%
  filter(name %in% c("Germany", "Philippines")) %>%
  mutate(Sex = "f")
mal <-
  popM %>%
  select(name, age, `2020`) %>%
  filter(name %in% c("Germany", "Philippines")) %>%
  mutate(Sex = "m")

pop <- fem %>%
  bind_rows(mal) %>%
  mutate(Country = as.character(name)) %>%
  group_by(Country, age) %>%
  summarize(Population = sum(`2020`)) %>%
  ungroup() %>%
  mutate(age = as.character(age),
    Age = case_when(
    age == "0-4" ~ 0,
    age == "5-9" ~ 5,
    age == "10-14" ~ 10,
    age == "15-19" ~ 15,
    age == "20-24" ~ 20,
    age == "25-29" ~ 25,
    age == "30-34" ~ 30,
    age == "35-39" ~ 35,
    age == "40-44" ~ 40,
    age == "45-49" ~ 45,
    age == "50-54" ~ 50,
    age == "55-59" ~ 55,
    age == "60-64" ~ 60,
    age == "65-69" ~ 65,
    age == "70-74" ~ 70,
    age == "75-79" ~ 75,
    age == "80-84" ~ 80,
    age == "85-89" ~ 85,
    age == "90-94" ~ 90,
    age == "95-99" ~ 95,
    age == "100+" ~ 100
  ),
         Population = Population * 1000,
         Sex = "b") %>%
  select(-age) %>%
  arrange(Country, Age)
```

Now this data is ready to merge with `COV5`

```r
COV <-
  COV5 %>%
  filter(Country %in% c("Germany","Philippines"),
         Date == dmy("15.07.2020")) %>%
  left_join(pop)
```

```
## Joining, by = c("Country", "Sex", "Age")
```

To net out population structure, let's say that cases are function of transmission probability (transmission * detection probability), ergo that cases are $Cases_x = c_x * P_x$, so we need to calculate c_x:

```r
COV <-
  COV %>%
  mutate(cx = Cases / Population)
```

Now *Population* is at base a weighting function, so we can reduce it to a distribution for clean (scale-free!) decomposing:

```r
COV <-
  COV %>%
  group_by(Country) %>%
  mutate(Px = Population/ sum(Population)) %>%
  ungroup()
```

Then we should be able to get back the same CFR we did before using P_x, c_x, and CFR_x, right?

```r
  COV %>%
  mutate(Cases_test = cx * Px,
         Deaths_test = cx * Px * CFRx) %>%
  group_by(Country) %>%
  summarize(CFR = sum(Deaths_test) / sum(Cases_test)) %>%
  pull(CFR) %>%
  diff() # looks the same to me!!
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## [1] -0.01726446
```

```r
zeta
```

```
## [1] -0.01726446
```

So, let's make our θs and a new CFR function:

```r
# This is how a normal human would set up the
# function arguments. This is fine. Do this!
# Especially because your case will be more complex
calc_CFR3 <- function(Px,cx,CFRx){
  sum(Px * cx * CFRx) /
    sum(Px * cx)
}

# but DemoDecomp needs parameters as a vector, for silly
# practicality reasons. That might be relaxed, though
# at least for parameters that can be expressed as arrays.
calc_CFR3_vec <- function(pars){

  # 1) reshape!
  # assume in order Px,cx,CFRx
  N <- length(pars)
  dim(pars) <- c(N/3,3)

  # 2) then calculate, using the normal-looking function
  calc_CFR3(Px = pars[, 1],
            cx = pars[, 2],
            CFRx = pars[, 3])
}
```

Now Das Gupta could do this using a Kitagawa approach. There's even an R package for it, but I've not had time to test it and understand it well enough to present. It's here: https://github.com/sadatnfs/dgdecomp

We will use the generalized methods.

```r
DE3 <-
  COV %>%
  filter(Country == "Germany") %>%
  select(Px,cx,CFRx) %>%
  as.matrix() %>%
  c()
```

```r
PH3 <-
  COV %>%
  filter(Country == "Philippines") %>%
  select(Px,cx,CFRx) %>%
  as.matrix() %>%
  c()
```

Now we're ready:

```r
dec3_h <- horiuchi(calc_CFR3_vec, DE3, PH3, N = 50)
dec3_c <- ltre(calc_CFR3_vec, DE3, PH3)
dec3_s <- stepwise_replacement(calc_CFR3_vec, DE3, PH3)

# assign dimensions (bookkeeping works out here)
N           <- length(DE3)
dim(dec3_h) <- c(N/3,3)
dim(dec3_c) <- c(N/3,3)
dim(dec3_s) <- c(N/3,3)

# assign contribution names
colnames(dec3_h) <- c("Px","cx","CFRx")
colnames(dec3_c) <- c("Px","cx","CFRx")
colnames(dec3_s) <- c("Px","cx","CFRx")

# compare components from three approaches
rbind(
colSums(dec3_h),
colSums(dec3_c),
colSums(dec3_s)
)
```

```
##               Px           cx       CFRx
## [1,] -0.03694185 0.0058388836 0.01383862
## [2,] -0.03137691 0.0009007724 0.01317266
## [3,] -0.03629622 0.0051339573 0.01389780
```

```r
# compare sums
sum(dec3_h);sum(dec3_c);sum(dec3_s)
```

```
## [1] -0.01726435
```

```
## [1] -0.01730348
```

```
## [1] -0.01726446
```

And from here out I'll accept challenges, suggestions, questions, etc.