Code and Notes on Cleaning AidData Core Research Release, Version 3.1

Miles D. Williams

Setup

For data wrangling, I'll be using the `{tidyverse}`. You can see from the message below my code for opening the `{tidyverse}` what versions of packages I'm working with. I'm also using the `{here}` package to help me navigate my files.

```
library(tidyverse)   # for syntax
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.2     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.4     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.0
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become er
```

```
library(here)        # navigating files
```

```
here() starts at C:/Users/Miles/Documents/Research Projects/aiddata_database
```

```
library(countrycode) # consistent country codes
```

Donor-Recipient-Year

The below code reads in the donor-recipient-year release of the data. It reports for every donor (whether a bilateral donor country or other multilateral donor) its aid disbursements to recipients in current dollar amounts.

```
data <- read_csv(
  here(
    "_data",
    "aiddata_core_3.1",
    "AidDataCoreDonorRecipientYear_ResearchRelease_Level1_v3.1.csv"
  )
)
```

If you look at the first five rows of the data, you can see the datasets basic structure.

```
data |>
  slice_head(n = 5)
```

```
# A tibble: 5 x 4
  donor                                    recipient  year commitment_amount_us~1
  <chr>                                    <chr>     <dbl>                  <dbl>
1 African Capacity Building Foundation (~ Africa, ~  2000                6239039
2 African Capacity Building Foundation (~ Africa, ~  2001                2440107
3 African Capacity Building Foundation (~ Africa, ~  2004                4371929
4 African Capacity Building Foundation (~ Africa, ~  2005               17006982
5 African Capacity Building Foundation (~ Africa, ~  2006               28517280
# i abbreviated name: 1: commitment_amount_usd_constant_sum
```

There are a few things I want to add to the data. The first is a set of country codes for bilateral donors and recipient countries. Note that for non-state donors, there will not be valid values returned. These codes are most useful for populating these data with other variables specific to certain donor countries or recipient countries.

```
data |>
  mutate(
    ## cow codes for donors and recipients
    ccode_d = countrycode(
      donor, "country.name", "cown"
    ),
    ccode_r = countrycode(
```

```
      recipient, "country.name", "cown"
    ),
    ## gw codes for donors and recipients
    gwcode_d = countrycode(
      donor, "country.name", "gwn"
    ),
    gwcode_r = countrycode(
      recipient, "country.name", "gwn"
    ),
    ## iso codes for donors and recipients
    isocode_d = countrycode(
      donor, "country.name", "iso3n"
    ),
    isocode_r = countrycode(
      recipient, "country.name", "iso3n"
    )
  ) -> data
```

Now, I'll save this additional version of the data:

```
data |>
  select(donor, ccode_d, gwcode_d, isocode_d,
         recipient, ccode_r, gwcode_r, isocode_r,
         year, commitment_amount_usd_constant_sum) |>
  rename(
    commitment_2011_constant = commitment_amount_usd_constant_sum
  ) |>
  write_csv(
    here(
      "_data",
      "aiddata_core_3.1",
      "clean_donor-recipient-year.csv"
    )
  )
```

Donor-Recipient-Purpose-Year

The below code performs a similar set of operations, but it does so for data that's been aggregated to the level of donor-recipient-purpose-years.

```r
## read in the data
data <- read_csv(
  here(
    "_data",
    "aiddata_core_3.1",
    "AidDataCoreDonorRecipientYearPurpose_ResearchRelease_Level1_v3.1.csv"
  )
)

## add country codes
data |>
  mutate(
    ## cow codes for donors and recipients
    ccode_d = countrycode(
      donor, "country.name", "cown"
    ),
    ccode_r = countrycode(
      recipient, "country.name", "cown"
    ),
    ## gw codes for donors and recipients
    gwcode_d = countrycode(
      donor, "country.name", "gwn"
    ),
    gwcode_r = countrycode(
      recipient, "country.name", "gwn"
    ),
    ## iso codes for donors and recipients
    isocode_d = countrycode(
      donor, "country.name", "iso3n"
    ),
    isocode_r = countrycode(
      recipient, "country.name", "iso3n"
    )
  ) -> data

## save
data |>
  select(donor, ccode_d, gwcode_d, isocode_d,
         recipient, ccode_r, gwcode_r, isocode_r,
         year, coalesced_purpose_code, coalesced_purpose_name,
         commitment_amount_usd_constant_sum) |>
```

```
  rename(
    crs_purpose_code = coalesced_purpose_code,
    crs_purpose_name = coalesced_purpose_name,
    commitment_2011_constant = commitment_amount_usd_constant_sum
  ) |>
  write_csv(
    here(
      "_data",
      "aiddata_core_3.1",
      "clean_donor-recipient-purpose-year.csv"
    )
  )
```

Donor-Recipient-Sector-Year

Based on the above data, it is possible to further aggregate by sector. One thing that I've found helpful on this front is creating a reference data table of every sector and purpose code. The below code constructs one based on the values included in the full research release data file:

```
full_data <- read_csv(
  here(
    "_data",
    "aiddata_core_3.1",
    "AidDataCoreFull_ResearchRelease_Level1_v3.1.csv"
  )
)
full_data |>
  select(
    crs_sector_code,
    crs_sector_name,
    crs_purpose_code,
    crs_purpose_name
  ) |>
  distinct() -> sec_purp_codes

## in some cases, if a purpose code is missing the
## sector is used in its place. This is redundant
## so I want to drop these rows
sec_purp_codes |>
```

```
  filter(
    crs_sector_code != crs_purpose_code
  ) -> sec_purp_codes

## there are also some alternate spellings of
## sector and purpose names. Let's fix that
sec_purp_codes |>
  mutate(
    crs_purpose_name = str_to_title(
      crs_purpose_name
    )
  ) |>
  distinct() -> sec_purp_codes
sec_purp_codes |>
  distinct(crs_purpose_code, .keep_all = T) -> sec_purp_codes
```

With values in `sec_purp_codes` will help me cross walk the purpose codes in the previous dataset with the relevant sector codes which I can then merge into the data and then use in aggregation. As you'll see below, however, I discovered some inconsistencies with how some of the purpose codes align with sector codes. I've take steps to fix this issue to ensure the sector totals are accurate.

```
## read in the data
data <- read_csv(
  here(
    "_data",
    "aiddata_core_3.1",
    "clean_donor-recipient-purpose-year.csv"
  )
)

## cross walk using sec_purp_data
data |>
  left_join(
    sec_purp_codes |> select(crs_sector_code, crs_sector_name,
                             crs_purpose_code),
    by = c("crs_purpose_code")
  ) -> data

## found some cases that don't align. I need to document
## these and include them in the sec_purp_code table bc
```

```r
## they're project, they're just somewhat misc.
data |>
  filter(
    is.na(crs_sector_code)
  ) |>
  select(
    crs_purpose_code,
    crs_purpose_name
  ) |>
  distinct() |>
  mutate(
    crs_sector_code =
      floor(crs_purpose_code/100)
  ) -> ext_codes

## merge with sec_purp_codes
sec_purp_codes |>
  full_join(
    ext_codes
  ) |>
  group_by(crs_sector_code) |>
  ## these are some lingering codes
  ## after looking at the data, I've
  ## concluded that the following sector
  ## codes needed to be corrected
  mutate(
    crs_sector_code = case_when(
      crs_sector_code == 110 ~ 111,
      crs_sector_code == 120 ~ 121,
      crs_sector_code == 150 ~ 151,
      crs_sector_code == 200 ~ 210,
      crs_sector_code == 310 ~ 311,
      crs_sector_code == 320 ~ 321,
      crs_sector_code == 100 ~ 160,
      crs_sector_code == 200 ~ 210,
      crs_sector_code == 300 ~ 321,
      TRUE ~ crs_sector_code
    ),

    ## this one is a group all of its
    ## own, so I made a new sector name
```

```
    ## for it
    crs_sector_name = ifelse(
      crs_sector_code == 420,
      "Women in development",
      crs_sector_name
    )
  ) |>
  ## make sure we don't have any lingering NAs
  mutate(
    crs_sector_name = ifelse(
      is.null(crs_sector_name |>
        unique() |>
        na.omit()),
      NA,
      crs_sector_name |>
       unique() |>
       na.omit()
    )
  ) |>
  distinct() -> sec_purp_codes

## save it for later use
write_csv(
  sec_purp_codes |> arrange(crs_purpose_code),
  here(
    "_data",
    "aiddata_core_3.1",
    "sec_purp_codes.csv"
  )
)
```

Okay, now we can do the cross-walking again

```
## read in the data
data <- read_csv(
  here(
    "_data",
    "aiddata_core_3.1",
    "clean_donor-recipient-purpose-year.csv"
  )
)
```

```r
## cross walk using sec_purp_data
data |>
  left_join(
    sec_purp_codes |> select(crs_sector_code, crs_sector_name,
                             crs_purpose_code),
    by = c("crs_purpose_code")
  ) -> data

## aggregate by to sectors
data |>
  group_by(
    donor, ccode_d, gwcode_d, isocode_d,
    recipient, ccode_r, gwcode_r, isocode_r,
    crs_sector_code,
    crs_sector_name,
    year
  ) |>
  summarize(
    commitment_2011_constant = sum(
      commitment_2011_constant
    )
  ) -> sm_data

## any duplicates?
sm_data |>
  select(
    donor, recipient, year, crs_sector_code
  ) |>
  distinct() |>
  nrow() -> rows_i_want
sm_data |>
  nrow() -> rows_i_have
rows_i_want == rows_i_have ## we're good!
```

```
[1] TRUE
```

```r
## save
write_csv(
  sm_data,
  here(
```

```
    "_data",
    "aiddata_core_3.1",
    "clean_donor-recipient-sector-year.csv"
  )
)
```