Data Importing and Wrangling

April 26, 2021

# Install and load libraries

We need to first install all the required packages for the data cleaning process. If you have never used the packages below, it is more likely that you have not installed them on your machine either. Please make sure you install each of the packages below using the following command:

install.packages("here")  
install.packages("openxlsx")  
install.packages("janitor")  
install.packages("tidyverse")  
install.packages("countrycode")  
install.packages("fuzzyjoin")  
install.packages("knitr")

Then you need to load the following packages:

library(here)  
library(openxlsx)  
library(janitor)  
library(tidyverse)  
library(countrycode)  
library(fuzzyjoin)  
library(knitr)

* openxlsx: a package that reads excel files and imports them into R. There are other packages that do the same thing, such as readxl, but we’ve chosen this package because it does a better job at dealing with merged cells within excel sheets.
* tidyverse: a suite of R packages that deal with data management and general wrangling. In this document we will be using mostly dplyr, tidyr, and stringr.
* janitor: a package that cleans and renames variable or columns names into something more consistent and coherent.
* knitr: a package that works with R markdown and exports documents into different formats such as Word, PDF, or HTML. In the case of this document we use knitr to convert all the code chunks here into a full-length R script.
* countrycode: a package that has dataset of country names with their associated legal geographic information such as the U.N. System region names and continent names. We will use this package to merge a region column to our final clean dataset.
* fuzzyjoin: a package to merge cells when string names are not fully matched, i.e. fuzzy.

# Data Import and Wrangling

This section is the bulk of the data cleaning process.

## Import data

### Annex I parties

annex\_i\_emissions <- openxlsx::read.xlsx(here("data",   
 "annual-net-emissions-removals-annex-i-raw.xlsx"),   
 startRow = 5, fillMergedCells = TRUE, rows = 5:96) %>%  
 janitor::clean\_names()

If you open the excel sheet associated with this document, you will find that the first five rows look a bit funny and we need to find a way to import this data without losing any of the information. We’ve also noticed that there is some text at the bottom of the excel sheet which we shouldn’t import into R. This is why we specified that R only imports rows 5 to 96.

Importing excel files into R can sometimes be more tedious because it uses a lot of formatting functions such as merging cells in this case. R does not know how to read merged cells and would view them as a single cell. Therefore, we use the openxlsx package because it has a function that deals with this issue.

We then use janitor to clean the variable names by converting all the names to lowercase and replace empty spaces with underscores. That way it is easier to work with these variables because R cannot read spaces when working with objects’ names.

### Non-Annex I parties

non\_annex\_i\_emissions <- openxlsx::read.xlsx(here("data",   
 "annual-net-emissions-removals-non-annex-i-raw.xlsx"),   
 startRow = 5, fillMergedCells = TRUE, rows = 5:302) %>%  
 janitor::clean\_names()

## Clean variables

Now we need to do some further cleaning and renaming of the dataset because not everything got imported in a perfect fashion. For example, the first row of the dataset is a whole row of only units of measurement. We therefore do not need it and should just remove it. Another example is that the member parties’ column has an empty cell as a header within the excel sheet. As a result, R named the empty header X. As such we renamed that column to party. Finally, the GHG variable did not import correctly and to save time in coding, we decided to remove aggregate part of the variable name. As such we renamed all the variables from aggregate\_gh\_gs to ghg.

### Annex I parties

annex\_i\_emissions <- annex\_i\_emissions[-c(1), ]  
  
colnames(annex\_i\_emissions)[1] <- "party"  
  
annex\_i\_emissions <- annex\_i\_emissions %>%  
 dplyr::rename\_all(funs(stringr::str\_replace\_all(.,   
 "aggregate\_gh\_gs", "ghg")))

### Non-Annex I parties

non\_annex\_i\_emissions <- non\_annex\_i\_emissions[-c(1),   
 ]  
  
colnames(non\_annex\_i\_emissions)[1] <- "party"  
  
non\_annex\_i\_emissions <- non\_annex\_i\_emissions %>%  
 dplyr::rename\_all(funs(stringr::str\_replace\_all(.,   
 "aggregate\_gh\_gs", "ghg")))

## Disaggreagate data by gas type

Right now the dataset is in wide format, which means that the values are spread across countries and years instead of being associated with their true variable which is gas type. The wide is helpful if we want to view the dataset as whole, but can get problematic when we need it to conduct analysis or produce visuals within R.

Another issue we had is that years were not imported into R in the correct way. If you take look at the imported dataset, you will see that co2 and aggregate\_gh\_gs are repeated multiple times throughout the dataset. What we do know, however is that the years begin after column gas and are from “Base year” to 2018 for Annex I and from 1990 to 2018 for Non-Annex I. Keeping that in mind, we first need to split the dataset by gas type, then rename columns four to 32 (Annex I) and three to 31 (Non-Annex I) to their associated years.

### Annex I parties

annex\_i\_emissions\_co2 <- annex\_i\_emissions %>%  
 dplyr::select("party", "gas", contains("co2")) %>%  
 dplyr::rename(base\_year = "co2")  
  
colnames(annex\_i\_emissions\_co2)[4:32] <- 1990:2018  
  
annex\_i\_emissions\_ghg <- annex\_i\_emissions %>%  
 dplyr::select("party", "gas", contains("ghg")) %>%  
 dplyr::rename(base\_year = "ghg")  
  
colnames(annex\_i\_emissions\_ghg)[4:32] <- 1990:2018

### Non-Annex I parties

non\_annex\_i\_emissions\_co2 <- non\_annex\_i\_emissions %>%  
 dplyr::select("party", "gas", contains("co2"))  
  
colnames(non\_annex\_i\_emissions\_co2)[3:31] <- 1990:2018  
  
non\_annex\_i\_emissions\_ghg <- non\_annex\_i\_emissions %>%  
 dplyr::select("party", "gas", contains("ghg"))  
  
colnames(non\_annex\_i\_emissions\_ghg)[3:31] <- 1990:2018

## Convert datasets to long

Here we convert the dataset to long. We also created a new column that specifies each party’s group name. That way it’s easier to conduct analyses by groups.

### Annex I parties

annex\_i\_emissions\_co2 <- annex\_i\_emissions\_co2 %>%  
 tidyr::gather(year, co2, base\_year:`2018`) %>%  
 mutate(group = "Annex I")  
  
annex\_i\_emissions\_ghg <- annex\_i\_emissions\_ghg %>%  
 tidyr::gather(year, ghg, base\_year:`2018`) %>%  
 mutate(group = "Annex I")

### Non-Annex I parties

non\_annex\_i\_emissions\_co2 <- non\_annex\_i\_emissions\_co2 %>%  
 tidyr::gather(year, co2, `1990`:`2018`) %>%  
 mutate(group = "Non-Annex I")  
  
non\_annex\_i\_emissions\_ghg <- non\_annex\_i\_emissions\_ghg %>%  
 tidyr::gather(year, ghg, `1990`:`2018`) %>%  
 mutate(group = "Non-Annex I")

## Merge datasets back

We merge the long datasets back by columns so we can have one column for total GHG emissions and one column for CO2 only. We will also remove the split datasets since we won’t be needing them any longer.

### Annex I parties

annex\_i\_emissions <- annex\_i\_emissions\_ghg %>%  
 dplyr::full\_join(annex\_i\_emissions\_co2, by = c("party",   
 "gas", "year", "group"))  
  
rm(annex\_i\_emissions\_co2, annex\_i\_emissions\_ghg)

### Non-Annex I parties

non\_annex\_i\_emissions <- non\_annex\_i\_emissions\_ghg %>%  
 dplyr::full\_join(non\_annex\_i\_emissions\_co2, by = c("party",   
 "gas", "year", "group"))  
  
rm(non\_annex\_i\_emissions\_co2, non\_annex\_i\_emissions\_ghg)

## Merge Annex I and Non-Annex I

Next we will merge Annex I and Non-Annex I by rows, and rename the values for gas the same for both Annex I and Non-Annex I to keep things consistent.

unfccc\_emissions <- dplyr::bind\_rows(annex\_i\_emissions,   
 non\_annex\_i\_emissions) %>%  
 dplyr::rename(type = "gas") %>%  
 dplyr::mutate(type = stringr::str\_replace(type,   
 "Total GHG emissions excluding LULUCF/LUCF",   
 "Total GHG emissions without LULUCF")) %>%  
 dplyr::mutate(type = stringr::str\_replace(type,   
 "Total GHG emissions including LULUCF/LUCF",   
 "Total GHG emissions with LULUCF")) %>%  
 readr::type\_convert() %>%  
 dplyr::mutate\_if(is.character, as.factor)

## Merge Region data

No we will merge region dataset using the countrycode so we can analyze data deprivations by region.

region <- countrycode::codelist %>%  
 select(country.name.en, un.region.name, iso3c) %>%  
 mutate(country.name.en = stringr::str\_replace\_all(country.name.en,   
 c(`Antigua & Barbuda` = "Antigua and Barbuda",   
 `Bosnia & Herzegovina` = "Bosnia and Herzegovina",   
 `Cape Verde` = "Cabo Verde", `Congo - Kinshasa` = "Congo",   
 `Côte d’Ivoire` = "Cote d'Ivoire", `Congo - Brazzaville` = "Democratic Republic of Congo",   
 `North Korea` = "Democratic People's Republic of Korea")))  
  
unfccc\_emissions <- unfccc\_emissions %>%  
 fuzzyjoin::regex\_left\_join(region, by = c(party = "country.name.en")) %>%  
 dplyr::rename(region = un.region.name, iso = iso3c) %>%  
 select(-country.name.en)  
unfccc\_emissions$region[unfccc\_emissions$party %in%   
 c("European Union (Convention)", "European Union (KP)")] <- "Europe"

## Export as a CSV file

Now we will export our clean dataset as an CSV file so we can conduct our analyses in the Analysis folder.

write.csv(unfccc\_emissions, "unfccc-emissions-clean.csv")

## Export as an R script for future use

Only run this chunk manually once within the .Rmd file. It produces an error when knitting it as a whole because of chunk label duplicates. As of April 26, 2021, there hasn’t been a viable solution to run the code below when as part of the knitting process.

knitr::purl("unfccc-emissions-clean.Rmd", "unfccc-emissions-clean.R")