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("dplyr")
install.packages("tidyr")
install.packages("stringr")
install.packages("readr")
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(dplyr)
library(tidyr)
library(stringr)
library(readr)
library(countrycode)
library(fuzzyjoin)
library(knitr)
```

- `here`: a package that reads path directories efficiently. It's best to use when sharing scripts or loading them in other files.

- `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.

- `janitor`: a package that cleans and renames variable or columns names into something more consistent and coherent.

- `tidyverse`: a suite of R packages that deal with data management and general

wrangling. In this document we will be using mostly `dplyr`, `tidyr`, `stringr`, and `reader`.

- `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.

- `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.

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::here("data/unfccc-emissions",
    "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::here("data/unfccc-emissions",
    "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] <- "country"

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] <- "country"

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("country", "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("country", "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("country", "gas", contains("co2"))

colnames(non_annex_i_emissions_co2)[3:31] <- 1990:2018

non_annex_i_emissions_ghg <- non_annex_i_emissions %>%
    dplyr::select("country", "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`) %>%
    dplyr::mutate(group = "Annex I")

annex_i_emissions_ghg <- annex_i_emissions_ghg %>%
    tidyr::gather(year, ghg, base_year:`2018`) %>%
    dplyr::mutate(group = "Annex I")
```

Non-Annex I parties

```
non_annex_i_emissions_co2 <- non_annex_i_emissions_co2 %>%
    tidyr::gather(year, co2, `1990`:`2018`) %>%
    dplyr::mutate(group = "Non-Annex I")

non_annex_i_emissions_ghg <- non_annex_i_emissions_ghg %>%
    tidyr::gather(year, ghg, `1990`:`2018`) %>%
    dplyr::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 CO_2 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("country", "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("country", "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

Now we will merge region dataset using the `countrycode` so we can analyze data deprivations by region.

```
region <- countrycode::codelist %>%
    dplyr::select(country.name.en, region,
        iso3c) %>%
    dplyr::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",
            Laos = "Lao People's Democratic Republic",
            `Micronesia \\(Federated States of\\)` = "Micronesia",
            `Myanmar \\(Burma\\)` = "Myanmar",
            `St. Lucia` = "Saint Lucia",
            `St. Vincent & Grenadines` = "Saint Vincent and the Grenadines",
            `São Tomé & Príncipe` = "Sao Tome and Principe",
            `Palestinian Territories` = "State of Palestine",
            `Trinidad & Tobago` = "Trinidad and Tobago",
            `North Korea` = "Democratic People's Republic of Korea",
            `South Korea` = "Republic of Korea",
            `St. Kitts & Nevis` = "Saint Kitts and Nevis",
            Vietnam = "Viet Nam")))

unfccc_emissions <- unfccc_emissions %>%
    fuzzyjoin::regex_left_join(region, by = c(country = "country.name.en")) %>%
    dplyr::rename(iso = iso3c) %>%
    dplyr::filter(!(country == "United Kingdom of Great Britain and Northern Ireland" &
        iso == "IRL"), !(country == "Guinea-Bissau" &
        iso == "GIN"), !(country == "Papua New Guinea" &
        iso == "GIN"), !(country == "Democratic People's Republic of Korea" &
        iso == "KOR"), !(country == "Dominican Republic" &
        iso == "DMA"), !(country == "Nigeria" &
        iso == "NER"), !(country == "South Sudan" &
        iso == "SDN")) %>%
    dplyr::select(-country.name.en) %>%
```

```
    dplyr::relocate(iso, .before = country) %>%
    dplyr::relocate(c(region, group, year),
        .after = country)


unfccc_emissions$region[unfccc_emissions$country %in%
    c("European Union (Convention)", "European Union (KP)")] <- "Europe & Central Asia"
unfccc_emissions$iso[unfccc_emissions$country ==
    "Democratic Republic of the Congo"] <- "COG"

rm(region)
```

Over here we needed to conduct some string manipulation because the official UNFCCC dataset does not provide ISO codes, which would have made it easier to merge the two datasets. Therefore, we needed to rename some of the countries so we can easily join the two datasets by country name.

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.

```
utils::write.csv(unfccc_emissions, "unfccc-emissions-clean.csv",
    row.names = FALSE)

rm(unfccc_emissions, non_annex_i_emissions,
    annex_i_emissions)
```

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 May 12, 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")
knitr::write_bib(.packages(), "packages.bib")
```

Software used

Arel-Bundock, Vincent. *Countrycode: Convert Country Names and Country Codes*, 2020. https://github.com/vincentarelbundock/countrycode.

Arel-Bundock, Vincent, Nils Enevoldsen, and CJ Yetman. "Countrycode: An r Package to Convert Country Names and Country Codes." *Journal of Open Source Software* 3, no. 28

(2018): 848. https://doi.org/10.21105/joss.00848.

Firke, Sam. *Janitor: Simple Tools for Examining and Cleaning Dirty Data*, 2021. https://github.com/sfirke/janitor.

Müller, Kirill. *Here: A Simpler Way to Find Your Files*, 2020. https://CRAN.R-project.org/package=here.

R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2021. https://www.R-project.org/.

Robinson, David. *Fuzzyjoin: Join Tables Together on Inexact Matching*, 2020. https://github.com/dgrtwo/fuzzyjoin.

Schauberger, Philipp, and Alexander Walker. *Openxlsx: Read, Write and Edit Xlsx Files*, 2020. https://CRAN.R-project.org/package=openxlsx.

Wickham, Hadley. *Stringr: Simple, Consistent Wrappers for Common String Operations*, 2019. https://CRAN.R-project.org/package=stringr.

———. *Tidyr: Tidy Messy Data*, 2021. https://CRAN.R-project.org/package=tidyr.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. *Dplyr: A Grammar of Data Manipulation*, 2021. https://CRAN.R-project.org/package=dplyr.

Wickham, Hadley, and Jim Hester. *Readr: Read Rectangular Text Data*, 2020. https://CRAN.R-project.org/package=readr.

Xie, Yihui. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC, 2015. https://yihui.org/knitr/.

———. "Knitr: A Comprehensive Tool for Reproducible Research in R." In *Implementing Reproducible Computational Research*, edited by Victoria Stodden, Friedrich Leisch, and Roger D. Peng. Chapman; Hall/CRC, 2014. http://www.crcpress.com/product/isbn/9781466561595.

———. *Knitr: A General-Purpose Package for Dynamic Report Generation in r*, 2021. https://yihui.org/knitr/.