Example

How to read GDX files in R

There are many libraries to read GDX files in R, starting from the official one provided by GAMS itself. All of them require GAMS, which can be downloaded and installed for free from the official website (https://www.gams.com/download/).

However, we prefer to use the `gdxtools` library, which can be installed from the official repository (https://github.com/lolow/gdxtools).

```r
library(gdxtools)
```

We also load the `tidyverse` package (<tidyverse.org/>) for manipulation and plotting.

```r
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.0     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x tidyr::extract() masks gdxtools::extract()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

Loading a GDX file

Before loading any GDX file you need to tell the library where the GAMS libraries are located. You need the command `igdx` providing the right path (this is the one I have on MacOS).

```r
gdxtools::igdx("/Applications/GAMS30.3/Resources/sysdir")
```

```
## GDX API loaded from gamsSysDir=/Applications/GAMS30.3/Resources/sysdir
## The GDX library has been loaded
## GDX library load path: /Applications/GAMS30.3/Resources/sysdir
```

Now we can load a GDX file.

```r
gdx_file <- gdx("../GDX/EnVarClim_1990.gdx")
print(gdx_file)
```

```
## <gdx: ../GDX/EnVarClim_1990.gdx, 21 symbols>
```

The `gdx_file` contains only the metadata. To show the list of the variables included in the GDX you need to check the `parameters` field.

```r
print(gdx_file$parameters$name)
```

```
##  [1] "OutputCommitted"      "OutputFlow"           "OutputPower"
##  [4] "OutputHeat"           "OutputHeatSlack"      "OutputStorageInput"
##  [7] "OutputStorageLevel"   "OutputSystemCost"     "OutputSpillage"
## [10] "OutputShedLoad"       "OutputCurtailedPower" "OutputGenMargin"
## [13] "LostLoad_MaxPower"    "LostLoad_MinPower"    "LostLoad_2D"
## [16] "LostLoad_2U"          "LostLoad_3U"          "LostLoad_RampUp"
## [19] "LostLoad_RampDown"    "ShadowPrice"          "status"
```

Now, we try to load the `OutputPower` variable, containing the hourly generation for all the units in the simulation. We also convert the output data frame into a tibble and then converting the hour field (`h`) from character to numeric.

```r
power_out <- gdx_file["OutputPower"] %>%
  as_tibble() %>%
  mutate(
    h = as.numeric(h)
  )
head(power_out)
```

```
## # A tibble: 6 x 3
##   u                            h value
##   <chr>                    <dbl> <dbl>
## 1 [0] - ES_Hydro reservoir     1 9717.
## 2 [0] - ES_Hydro reservoir     2 9717.
## 3 [0] - ES_Hydro reservoir     3 9717.
## 4 [0] - ES_Hydro reservoir     4 9717.
## 5 [0] - ES_Hydro reservoir     5 9717.
## 6 [0] - ES_Hydro reservoir     6 9717.
```

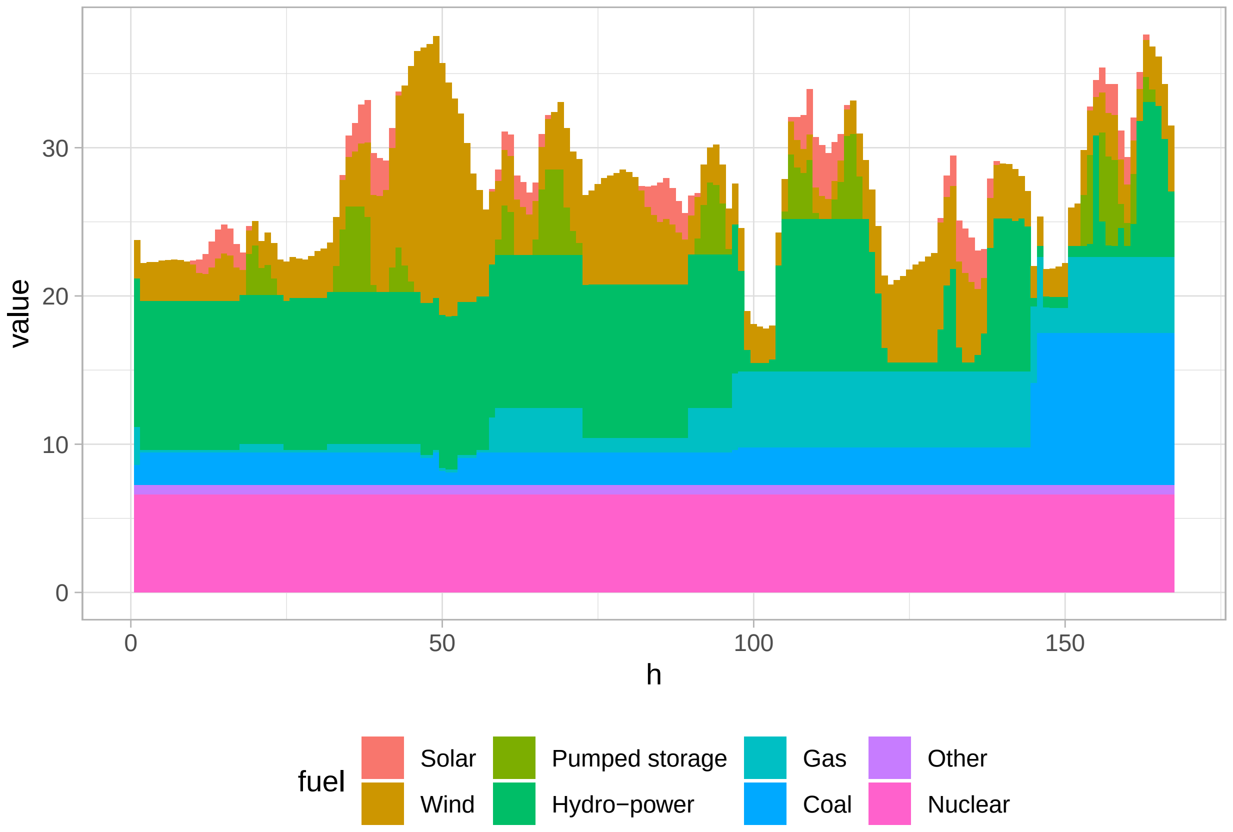
Example plot

Let's plot the first week of generation for Spain (ES). In the unit field (`u`) the first two characters contain the country code. Then we can filter easily the tibble with the `filter` function provided by `dplyr` (part of the tidyverse). We also group the fuels to visualise the dispatching in a nicer way, finally converting MW to GW.

```r
example <- power_out %>%
  dplyr::filter(
    h < 24 * 7,
    str_detect(u, "ES_")
  ) %>%
  mutate(
    fuel = case_when(
      str_detect(u, "Hydro") ~ "Hydro-power",
      str_detect(u, "Coal") ~ "Coal",
      str_detect(u, "CGT") ~ "Gas",
      str_detect(u, "Pumped") ~ "Pumped storage",
      str_detect(u, "Nuclear") ~ "Nuclear",
      str_detect(u, "Solar") ~ "Solar",
      str_detect(u, "Wind") ~ "Wind",
      TRUE ~ "Other"
    ) %>%
      as.factor() %>%
      fct_relevel(
        "Solar", "Wind", "Pumped storage", "Hydro-power", "Gas", "Coal", "Other", "Nuclear"
      )
  ) %>%
  group_by(h, fuel) %>%
  summarise(
    value = sum(value) / 1e3
  )
```

Finally, we can plot the data with `ggplot2`.

```
ggplot(example, aes(x = h, y = value, fill = fuel)) +
  geom_bar(stat = "identity", width = 1) +
  theme_light() +
  theme(
    legend.position = 'bottom'
  )
```


value
h
fuel
Solar
Wind
Pumped storage
Hydro−power
Gas
Coal
Other
Nuclear