GAMS-R Vizualization Lecture

Laurent Drouet

Lara Aleluia Reis

2022-04-26

Presentation

Laurent Drouet

- Researcher at RFF-CMCC EIEE. WITCH model developer.
- Several **R** packages, in particular, `gdxtools`.
- GAMS extension for the editor `Visual Code Studio`.

Lara Aleluia Reis

- She developed these slides.
- Environmental engineer
- Air pollution modeling (**R**)
- WITCH model developer at RFF-CMCC EIEE (**GAMS**+**R**).
- More than 10 years of **R** experience.

This lecture

In this lecture, you will learn how to use **R** with GAMS and for visualization.

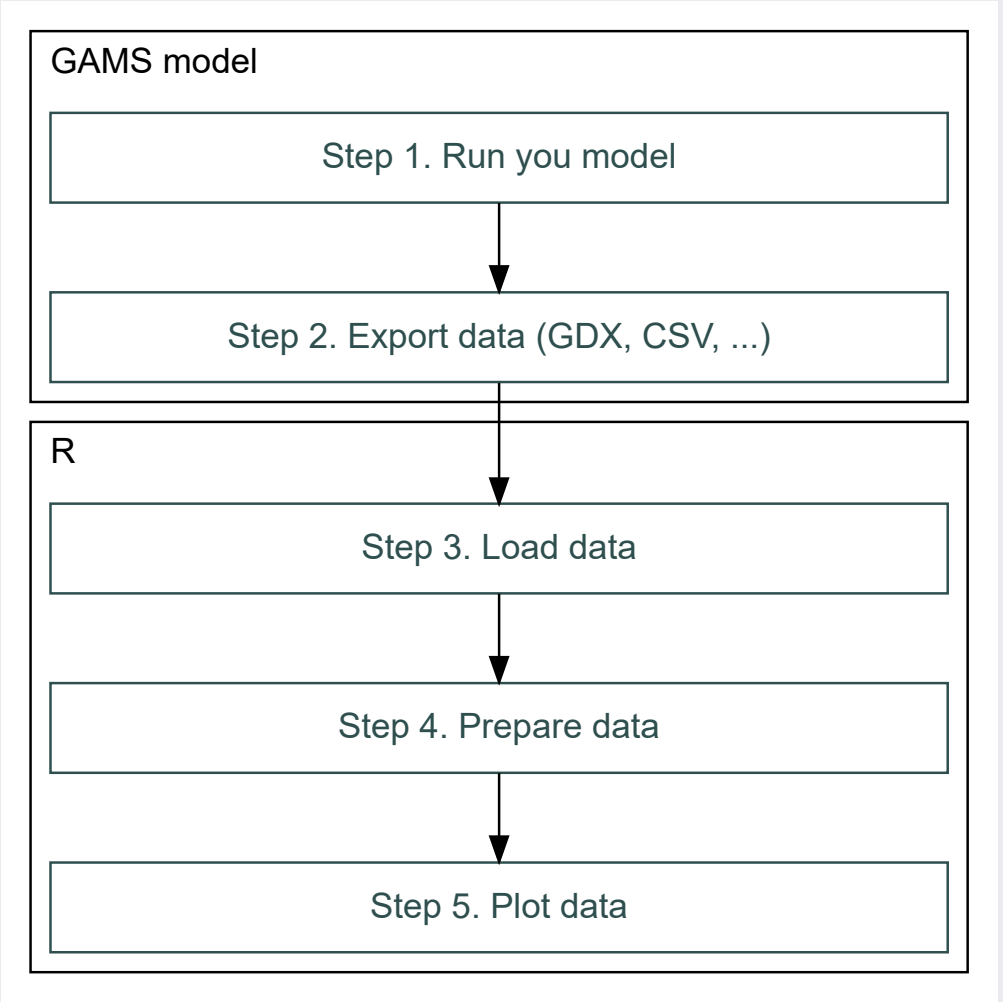
GAMS

- Read GAMS output through the GDX format.

Visualization

- Plot the model outputs
- Check the behavior of the model
- Understand the model results

Workflow


GAMS model
Step 1. Run you model
Step 2. Export data (GDX, CSV, ...)
R
Step 3. Load data
Step 4. Prepare data
Step 5. Plot data

Outline

1. Introduction to **R**
2. `GDX`: The GAMS data exchange format
3. `gdxtools`: an **R** package to read GDX
4. Data wrangling in **R**
5. Data visualization in **R** with `ggplot2`
6. Advanced data visualization
7. Visualization tips (for the curious)

How to Start: Requirements

> R, RStudio and R packages are required for the rest of the lecture

1. Install R: https://cloud.r-project.org
2. Install Rstudio IDE: https://www.rstudio.com/products/rstudio/download
3. Install Rtools (for `gdxtools`): https://cran.r-project.org/bin/windows/Rtools/rtools40.html

We need to install the following R packages for the lecture:

```
'tidyverse'
'gdxtools' # Installation instructions → https://github.com/lolow/gdxtools
```

Open RStudio and paste the following command in the console:

```
install.packages('tidyverse')
```

The GAMS data exchange GDX files

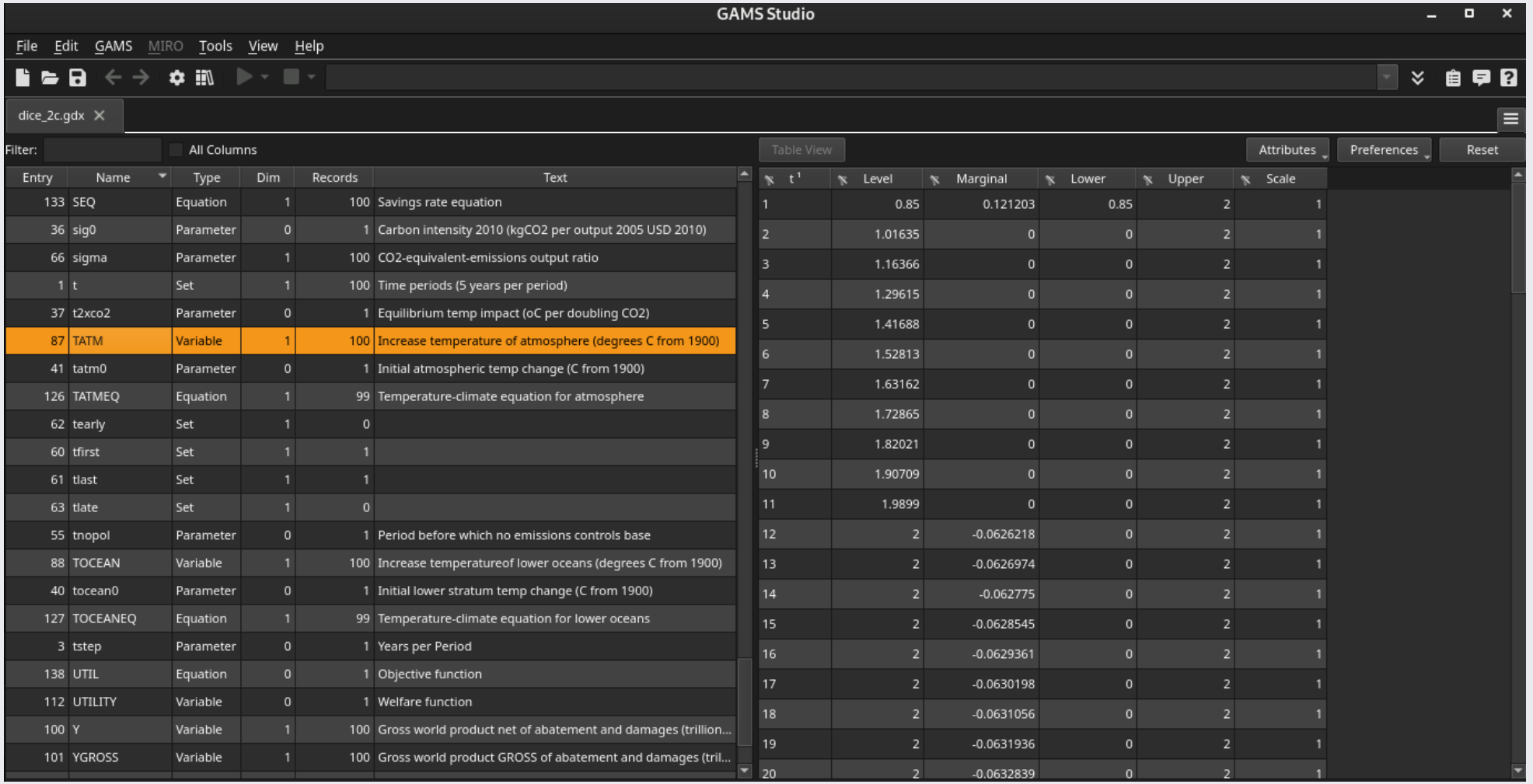
The GAMS data exchange files

The GAMS data exchange format (in short GDX) is a proprietary file format to store the various elements of a GAMS model.

- It is a database which contains a list of data from your program:
  - the sets
  - the parameters
  - the variables
  - the equations

Many information is stored in the GDX file. For example, for the variables, the GDX contain the values (`level`), the bounds (`lower`,`upper`), the marginal value (`marginal`) and the scaling factors (`scale`).

Browsing a GDX in GAMS studio



Writing a GDX (in GAMS)

You can write a GDX file with the command `execute_unload`:

```
execute_unload "results.gdx";
```

This will create a file containing all sets, parameters, variables and equations that are present when the line is executed.

You can also specify which elements you want to store in the GDX file:

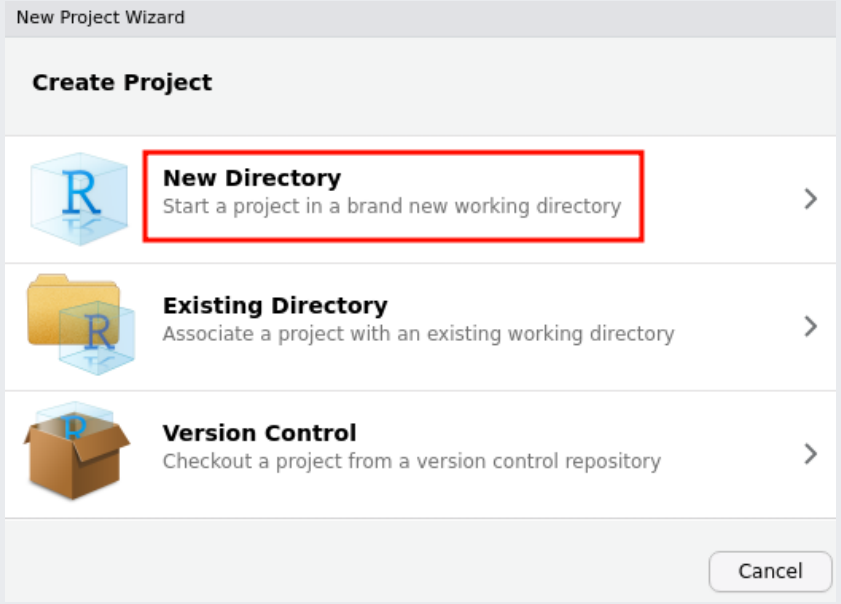
```
execute_unload "results.gdx", VAR1, VAR2, parameter1;
```

> Some models, like WITCH, already produce automatically a GDX of their results.

R Basics

Create an R project in RStudio

First, create a new R project (`File > New Project ...`) then `New Directory`



Choose your subdirectory and enter the name of the R project (for example, **`RGAMS_viz`**).

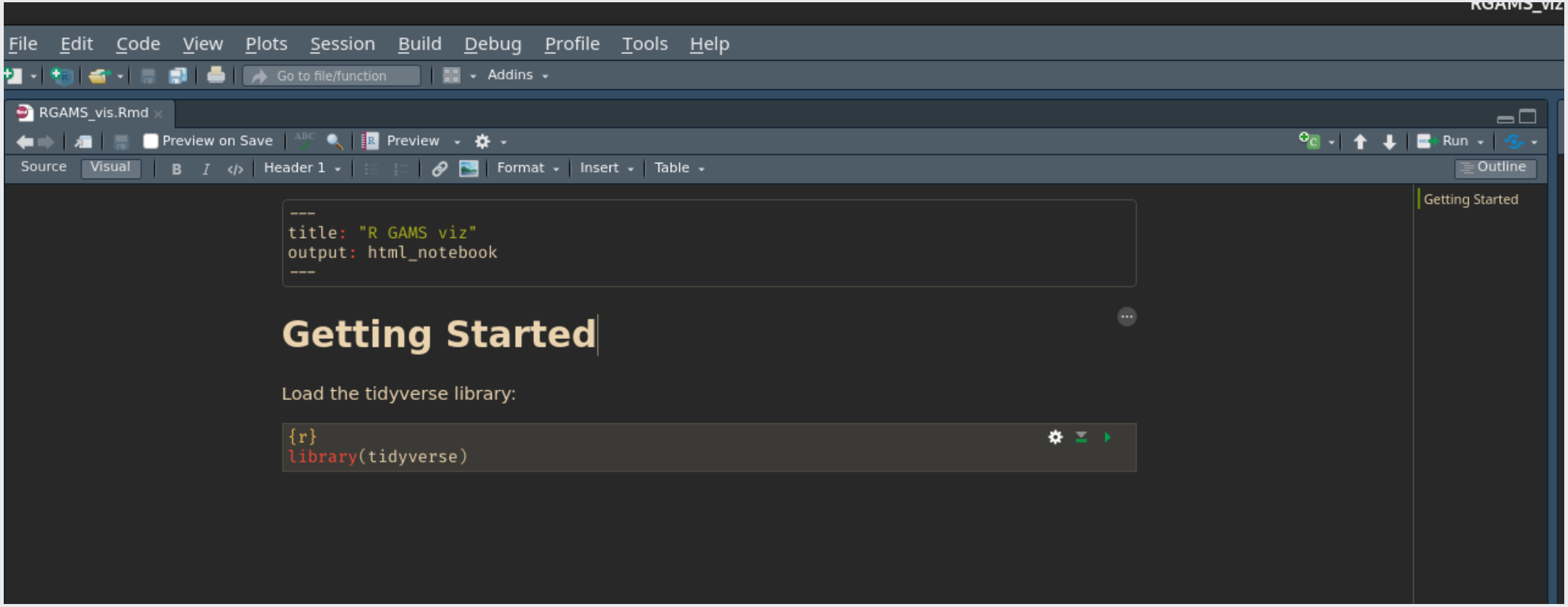
This will create a directory with the project name. Copy the `Material` folder inside.

Create an R notebook

Create a new R Markdown file (`File > New File > R Markdown ...`). Click on `Visual` to access the visual editor.

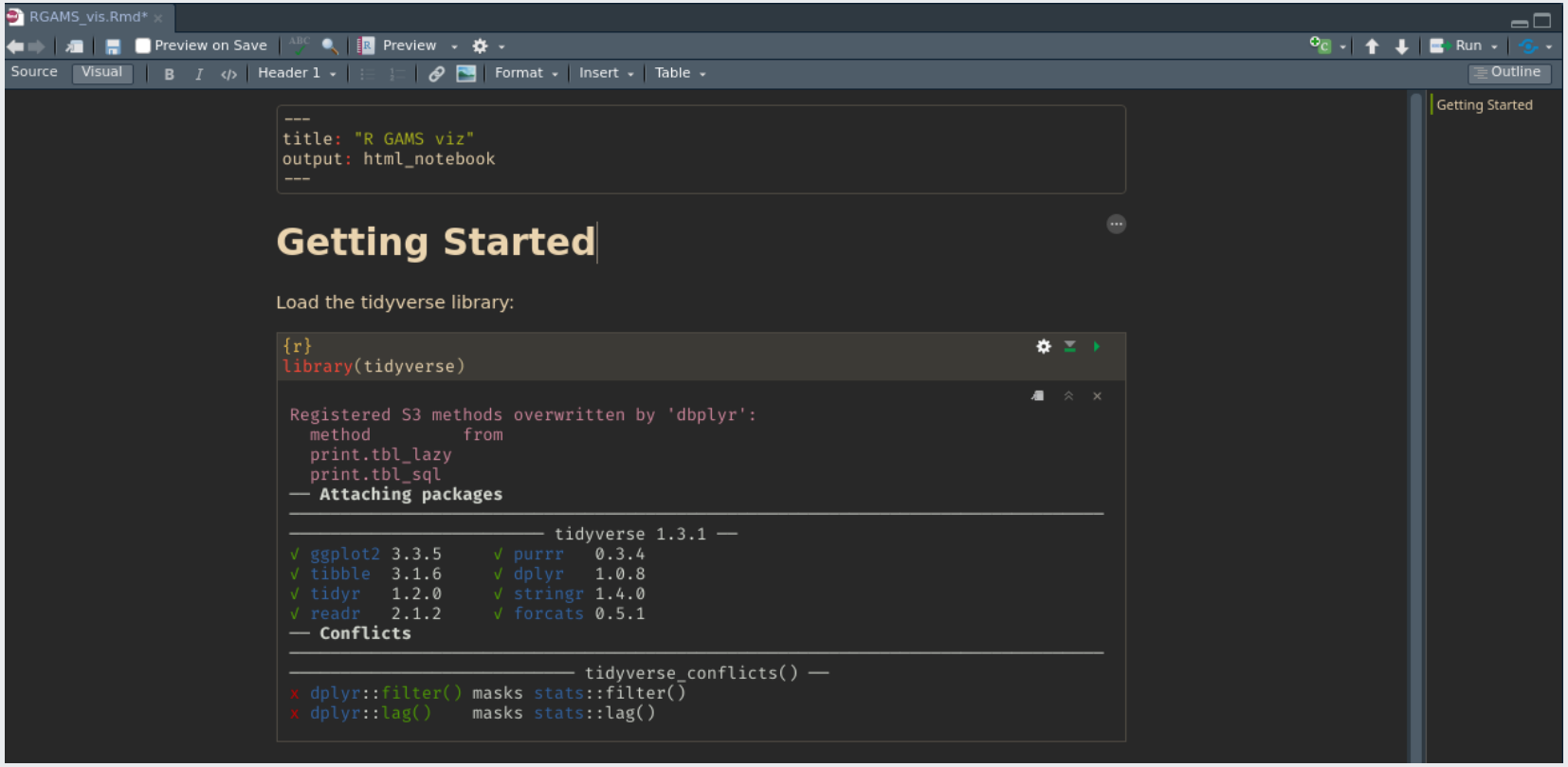
Update the notebook

Update the title, Remove the text and add a **R** chunk loading the `tidyverse` library.



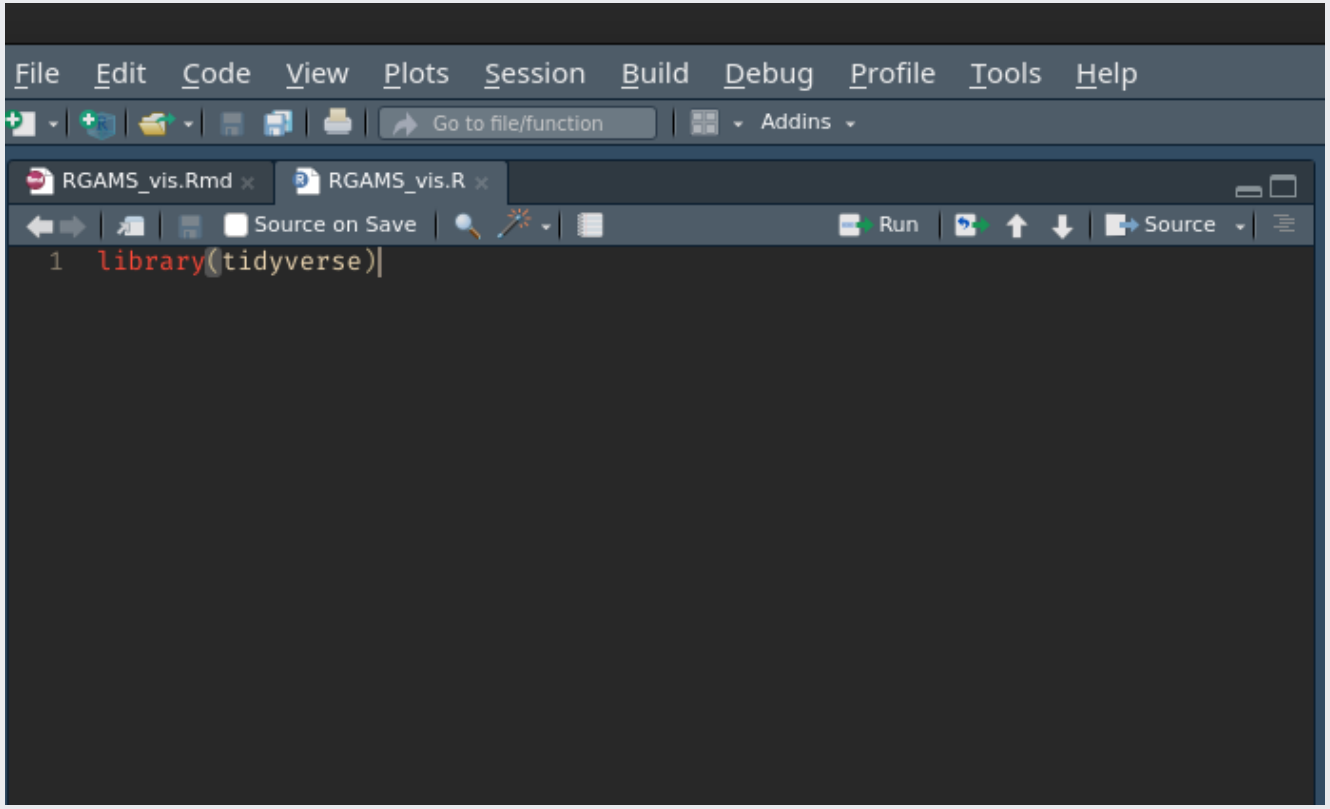
Execute the R chunk

Click on the *green* arrow on the top of the chunk.



R script

This is equivalent to create a proper Rscript (executable with `Source`):



gdxtools R Package

gdxtools

`gdxtools` is an **R** package which loads the content of a GDX file into R.

- It requires a GAMS installation to work.
- It is not available in the `CRAN` repository, so it needs to be installed manually.

Installation

In the console:

```
if (!requireNamespace("remotes"))
  install.packages("remotes")

remotes::install_github("lolow/gdxtools")
```

On windows, you need to install `Rtools` to be able to install the package. See the procedure at https://cran.r-project.org/bin/windows/Rtools

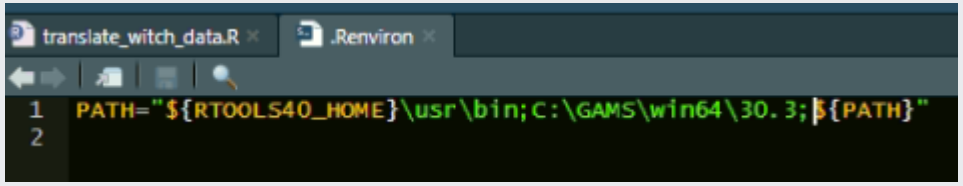
Usage

Load the package with

```
library(gdxtools)
```

`gdxtools` will try to find your `GAMS` installation. If it does not find it, choose one of the following solution:

- Tell windows about your GAMS path: https://helpdeskgeek.com/windows-10/add-windows-path-environment-variable
- Tell R about your GAMS path by adding it in your `.Renviron` file with `usethis::edit_r_environ()`



- Tell directly `gdxtools`:

```
igdx('C:\\GAMS\\win64\\38.1')
```

Usage

Connexion to a GDX file

You define a GDX file with the command `gdx`. Here a result file from the WITCH model.

```
mygdx ← gdx('Material/results_ssp2_bau.gdx')
```

`mygdx` can provide information on the GDX content.

```
mygdx
```

```
## <gdx: Material/results_ssp2_bau.gdx, 2048 symbols>
```

The GDX contains *2048 symbols* (variables, parameters, sets and equations)

Usage

List of variables

```
mygdx$variables$name
```

```
##  [1] "UTILITY"          "PROB"             "I"              "K"              "Q"              "BAU_Q"
##  [8] "COST_EN"          "COST_FUEL"        "FPRICE"         "I_EN"           "K_EN"           "MCOST_FUEL"
## [15] "QEL_OUT"          "QNEL_OUT"         "Q_EN"           "Q_FUEL"         "Q_IN"           "BAU_Q_EMI"
## [22] "CPRICE"           "Q_EMI"            "I_RD"           "K_RD"           "SPILL"          "Q_EL_FLEX"
## [29] "I_EN_GRID"        "CUM_Q_CCS"        "MCOST_EMI"      "Q_CCS"          "I_EN_WINDOFF"   "I_EN_WINDON"
## [36] "K_EN_WINDON"      "MCOST_INV_WINDOFF" "Q_EN_WINDOFF"  "Q_EN_WINDON"    "I_EN_PV"        "I_EN_CSP"
## [43] "K_EN_CSP"         "Q_EN_PV"          "Q_EN_CSP"       "ADDOILCAP"      "COST_OIL"       "CUM_OIL"
## [50] "I_OUT"            "OILCAP"           "OILPROD"        "Q_EMI_OUT"      "Q_OUT"          "RF"
## [57] "TRF"              "W_EMI"            "WCUM_EMI"       "OMEGA"          "ELMOTOR_COST"
```

Usage

List of parameters

```
mygdx$parameters$name
```

```
##   [1] "nb_clt_infes"                "nb_clt_noopt"                "nb_tot_infes"                "allerr"
##   [5] "allinfoiter"                 "delta_price"                 "errtrade"                    "infoiter"
##   [9] "m_consumption"               "mbalance"                    "mprofit"                     "price_iter"
##  [13] "solrep"                      "timer"                       "trust_region"                "errtol"
##  [17] "stop_nash"                   "stop_run"                    "tlen"                        "tperiod"
##  [21] "year"                        "yeoh"                        "eta"                         "gamma"
##  [25] "srtp"                        "stpf"                        "utility_cebge_global"        "utility_cebge_pc_globa
##  [29] "utility_cebge_pc_regional"   "utility_cebge_regional"      "w_negishi"                   "alpha"
##  [33] "delta"                       "delta0"                      "l"                           "mer2ppp"
##  [37] "m_eqq_y"                     "ppp2mer"                     "q0"                          "rho"
##  [41] "tfpn"                        "tfpy"                        "ykali"                       "gdppc_kali"
##  [45] "cexs"                        "csi"                         "csi0"                        "cum_prodpp"
##  [49] "delta_en"                    "delta_en0"                   "delta_lcost"                 "lifetime"
##  [53] "k_en0"                       "localpll"                    "mcost_inv0"                  "mu"
##  [57] "mu0"                         "oem"                         "oem0"                        "p_mkup"
##  [61] "p_mkup0"                     "prodpp"                      "twh2ej"                      "wcum"
##  [65] "tpes"                        "noeloil"                     "c2co2"                       "carbonprice"
##  [69] "emi_bio_harv"                "emi_cap"                     "emi_st"                      "m_eqq_emi_lim"
##  [73] "m_eqq_emi_tree"              "rd_coef"                     "rd_delta"                    "krd0"
##  [77] "rd_crowd_out"                "krd_lead"                    "krd_oth_lead_krd"            "wkrd0"
##  [81] "wlspill_k_rd"                "ctax"                        "cv_coeff"                    "cv_exp"
##  [85] "el_free_cap"                 "flex_coeff"                  "grid_coeff"                  "grid_cost"
##  [89] "grid_delta"                  "peak_load_fraction"          "mkt_growth_rate"             "cwaste"
##  [93] "cwaste_reg"                  "wastemgm"                    "twh2tonU"                    "waste_emi_alpha"
```

Usage

List of sets

```
mygdx$sets$name[1:60]
```

```
##  [1] "all_feasible"   "all_optimal"    "irep"           "iterrep"        "tcheck"         "ierr"           "siter"          "
##  [9] "t"              "tfix"           "tfirst"         "tlast"          "pre"            "n"              "oecd"           "
## [17] "clt"            "map_clt_n"      "negishi_coop"   "g"              "iq"             "extract"        "fuel"           "
## [25] "s"              "f_mkt"          "ices_el"        "j"              "j_to_scale"     "jel"            "jel_own_mu"     "
## [33] "jfed"           "jinv"           "jinv_own_k"     "jinv_to_scale"  "jmcost_inv"     "jnel"           "jnel_ren"       "
## [41] "jpenalty"       "jreal"          "jreal_to_scale" "map_ices_el"    "map_j"          "cce"            "e"              "
## [49] "sink"           "c_mkt"          "map_e"          "jrd"            "jrd_lbd"        "leadrd"         "rd"             "
## [57] "rd_tgap"        "run"            "to_run"         "internal"
```

Usage

To get a vector of the global regions from the set `n`:

```
mygdx["n"][[1]]
```

```
##  [1] "brazil"      "canada"      "china"       "europe"      "india"       "indonesia"   "jpnkor"      "laca"        "mena"
## [11] "oceania"     "sasia"       "seasia"      "southafrica" "ssa"         "te"          "usa"
```

This command extract `"n"` from the variable `mygdx`

> In a case of a set, you are only interested in the first column accessible with `[[1]]`

Usage

To get a data.frame of the parameter `wemi`:

```
mygdx["wemi"] ▷ as_tibble() # use ▷ if R > 4.1 otherwise %>%
```

```
## # A tibble: 437 × 3
##    e     t     value
##    <chr> <chr> <dbl>
##  1 co2   1      8.56
##  2 co2   2      9.31
##  3 co2   3     10.1
##  4 co2   4     10.7
##  5 co2   5     11.4
##  6 co2   6     12.1
##  7 co2   7     12.5
##  8 co2   8     12.8
##  9 co2   9     12.7
## 10 co2   10    12.8
## # … with 427 more rows
```

The parameter `wemi` has 2 dimensions the emission type `e` and the time period `t`.

Usage

The definition of the symbol can be available with its attribute "gams".

```
print(attr(mygdx["wemi"],"gams"))
```

```
## [1] "World GHG emissions"
```

Usage

To get a data.frame of the variable `Q_EMI` (its level by default):

```
mygdx["Q_EMI"] ▷ as_tibble()
```

```
## # A tibble: 16,830 × 4
##    e     t     n         value
##    <chr> <chr> <chr>     <dbl>
##  1 nip   1     brazil        0
##  2 nip   1     canada        0
##  3 nip   1     china         0
##  4 nip   1     europe        0
##  5 nip   1     india         0
##  6 nip   1     indonesia     0
##  7 nip   1     jpnkor        0
##  8 nip   1     laca          0
##  9 nip   1     mena          0
## 10 nip   1     mexico        0
## # … with 16,820 more rows
```

Usage

To load the variable's upper bound:

```
mygdx["Q_EMI",field = "up"] ▷ as_tibble()
```

```
## # A tibble: 16,830 × 4
##    e     t     n         value
##    <chr> <chr> <chr>     <dbl>
##  1 nip   1     brazil        0
##  2 nip   1     canada        0
##  3 nip   1     china         0
##  4 nip   1     europe        0
##  5 nip   1     india         0
##  6 nip   1     indonesia     0
##  7 nip   1     jpnkor        0
##  8 nip   1     laca          0
##  9 nip   1     mena          0
## 10 nip   1     mexico        0
## # … with 16,820 more rows
```

Usage

To load many gdx at once in one data.frame, use the function `batch_extract`

```
myfiles = file.path('Material',c("results_ssp2_bau.gdx","results_ssp2_ctax50.gdx"))
qemi = batch_extract("Q_EMI",files = myfiles)[[1]]
qemi ▷ as_tibble()
```

```
## # A tibble: 33,711 × 5
##    e     t     n         value gdx
##    <chr> <chr> <chr>     <dbl> <chr>
##  1 nip   1     brazil        0 Material/results_ssp2_bau.gdx
##  2 nip   1     canada        0 Material/results_ssp2_bau.gdx
##  3 nip   1     china         0 Material/results_ssp2_bau.gdx
##  4 nip   1     europe        0 Material/results_ssp2_bau.gdx
##  5 nip   1     india         0 Material/results_ssp2_bau.gdx
##  6 nip   1     indonesia     0 Material/results_ssp2_bau.gdx
##  7 nip   1     jpnkor        0 Material/results_ssp2_bau.gdx
##  8 nip   1     laca          0 Material/results_ssp2_bau.gdx
##  9 nip   1     mena          0 Material/results_ssp2_bau.gdx
## 10 nip   1     mexico        0 Material/results_ssp2_bau.gdx
## # … with 33,701 more rows
```

Data loading and wrangling

R Basics

Let's load all the libraries that we need...

```
library(tidyverse)
library(gdxtools)
```

The command `library()` loads all the functions from a library.

Whenever you need to know what a function does, how to use it, or to know more about it use **?** before the function name (e.g. `?library()`,`?sum`)

Load a GDX file into R

```
# If necessary, tell where GAMS is located in your machine.
igdx(dirname(Sys.which('gams')))
```

```
# Define a gdx
dice_gdx ← gdx('Material/dice_2c.gdx')

# Print a summary of the GDX content
dice_gdx
```

```
## <gdx: Material/dice_2c.gdx, 139 symbols>
```

```
# Obtain a variable and see it as a tibble
dice_gdx["TATM"] ▷ as_tibble()
```

```
## # A tibble: 100 × 2
##    t     value
##    <chr> <dbl>
##  1 1      0.85
##  2 2      1.02
##  3 3      1.16
##  4 4      1.30
##  5 5      1.42
##  6 6      1.53
##  7 7      1.63
##  8 8      1.73
##  9 9      1.82
## 10 10     1.91
## # … with 90 more rows
```

Load a CSV file into R

```
dice_csv ← read_csv('Material/Dice2016R-091916ap.csv', skip = 6) # returns already a tibble
dice_csv
```

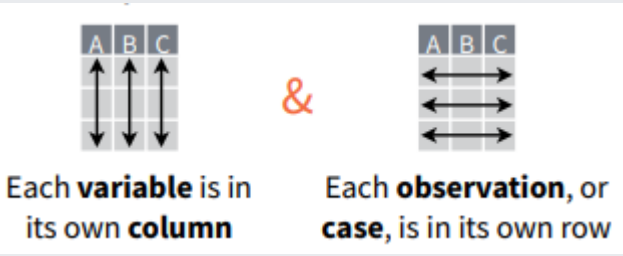
```
## # A tibble: 37 × 101
##    Year       `2015`  `2020`  `2025`  `2030`  `2035`  `2040`  `2045`  `2050`  `2055`  `2060`  `2065`  `2070`  `2075`  `2080`
##    <chr>       <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1 Industri… 3.57e+1 3.94e+1 4.29e+1 4.63e+1 4.96e+1 5.27e+1 5.55e+1 5.82e+1 6.06e+1 6.27e+1 6.46e+1 6.63e+1 6.77e+1 6.88e+1
##  2 Atmosphe… 4.00e+2 4.18e+2 4.38e+2 4.59e+2 4.81e+2 5.04e+2 5.28e+2 5.52e+2 5.78e+2 6.04e+2 6.31e+2 6.58e+2 6.86e+2 7.14e+2
##  3 Atmosphe… 8.5 e-1 1.02e+0 1.19e+0 1.37e+0 1.55e+0 1.74e+0 1.93e+0 2.13e+0 2.32e+0 2.52e+0 2.72e+0 2.92e+0 3.13e+0 3.32e+0
##  4 Output N… 1.05e+2 1.25e+2 1.47e+2 1.72e+2 1.98e+2 2.27e+2 2.59e+2 2.92e+2 3.28e+2 3.67e+2 4.07e+2 4.51e+2 4.96e+2 5.44e+2
##  5 Climate … 1.71e-3 2.44e-3 3.34e-3 4.42e-3 5.68e-3 7.15e-3 8.81e-3 1.07e-2 1.28e-2 1.50e-2 1.75e-2 2.02e-2 2.31e-2 2.61e-2
##  6 Consumpt… 1.05e+1 1.18e+1 1.33e+1 1.49e+1 1.66e+1 1.85e+1 2.05e+1 2.26e+1 2.49e+1 2.73e+1 2.98e+1 3.26e+1 3.54e+1 3.85e+1
##  7 Carbon P… 2   e+0 2.21e+0 2.44e+0 2.69e+0 2.97e+0 3.28e+0 3.62e+0 4.00e+0 4.42e+0 4.88e+0 5.38e+0 5.94e+0 6.56e+0 7.25e+0
##  8 Emission… 2.99e-2 3.23e-2 3.49e-2 3.77e-2 4.08e-2 4.41e-2 4.76e-2 5.15e-2 5.56e-2 6.01e-2 6.50e-2 7.02e-2 7.59e-2 8.21e-2
##  9 Social c… 3.12e+1 3.73e+1 4.40e+1 5.16e+1 6.00e+1 6.93e+1 7.94e+1 9.05e+1 1.02e+2 1.15e+2 1.29e+2 1.44e+2 1.60e+2 1.77e+2
## 10 Interest… 5.10e-2 4.98e-2 4.87e-2 4.76e-2 4.66e-2 4.56e-2 4.46e-2 4.36e-2 4.27e-2 4.18e-2 4.10e-2 4.02e-2 3.94e-2 3.86e-2
## # … with 27 more rows, and 83 more variables: `2100` <dbl>, `2105` <dbl>, `2110` <dbl>, `2115` <dbl>, `2120` <dbl>, `2125` <d
## #   `2135` <dbl>, `2140` <dbl>, `2145` <dbl>, `2150` <dbl>, `2155` <dbl>, `2160` <dbl>, `2165` <dbl>, `2170` <dbl>, `2175` <d
## #   `2185` <dbl>, `2190` <dbl>, `2195` <dbl>, `2200` <dbl>, `2205` <dbl>, `2210` <dbl>, `2215` <dbl>, `2220` <dbl>, `2225` <d
## #   `2235` <dbl>, `2240` <dbl>, `2245` <dbl>, `2250` <dbl>, `2255` <dbl>, `2260` <dbl>, `2265` <dbl>, `2270` <dbl>, `2275` <d
## #   `2285` <dbl>, `2290` <dbl>, `2295` <dbl>, `2300` <dbl>, `2305` <dbl>, `2310` <dbl>, `2315` <dbl>, `2320` <dbl>, `2325` <d
## #   `2335` <dbl>, `2340` <dbl>, `2345` <dbl>, `2350` <dbl>, `2355` <dbl>, `2360` <dbl>, `2365` <dbl>, `2370` <dbl>, `2375` <d
## #   `2385` <dbl>, `2390` <dbl>, `2395` <dbl>, `2400` <dbl>, `2405` <dbl>, `2410` <dbl>, `2415` <dbl>, `2420` <dbl>, `2425` <d
```

```
# Correct column name
dice_csv ← dice_csv ▷ rename(Variable=Year)
```

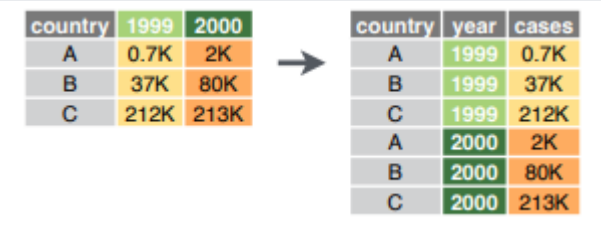
Data wrangling

The raw loaded data are often not directly usable for plotting. It needs to be prepared:

- Transform the data



- Tidy the data (ids and measures columns). If you are not sure, prefer the long version.



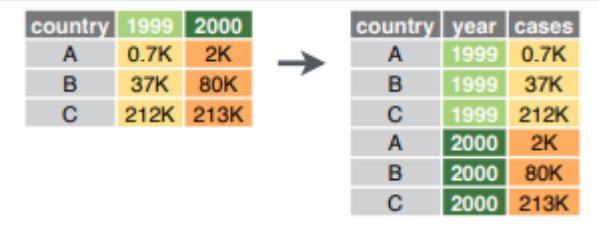
Later, filter the rows and select the columns for your plot.

Reshaping to a long table

```
# Preview table
head(dice_csv)
```

```
## # A tibble: 6 × 101
##   Variable    `2015`  `2020`  `2025`  `2030`  `2035`  `2040`  `2045`  `2050`  `2055`  `2060`  `2065`  `2070`  `2075`  `2080`
##   <chr>        <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 Industria… 3.57e+1 3.94e+1 4.29e+1 4.63e+1 4.96e+1 5.27e+1 5.55e+1 5.82e+1 6.06e+1 6.27e+1 6.46e+1 6.63e+1 6.77e+1 6.88e+1
## 2 Atmospher… 4.00e+2 4.18e+2 4.38e+2 4.59e+2 4.81e+2 5.04e+2 5.28e+2 5.52e+2 5.78e+2 6.04e+2 6.31e+2 6.58e+2 6.86e+2 7.14e+2
## 3 Atmospher… 8.5 e-1 1.02e+0 1.19e+0 1.37e+0 1.55e+0 1.74e+0 1.93e+0 2.13e+0 2.32e+0 2.52e+0 2.72e+0 2.92e+0 3.13e+0 3.32e+0
## 4 Output Ne… 1.05e+2 1.25e+2 1.47e+2 1.72e+2 1.98e+2 2.27e+2 2.59e+2 2.92e+2 3.28e+2 3.67e+2 4.07e+2 4.51e+2 4.96e+2 5.44e+2
## 5 Climate D… 1.71e-3 2.44e-3 3.34e-3 4.42e-3 5.68e-3 7.15e-3 8.81e-3 1.07e-2 1.28e-2 1.50e-2 1.75e-2 2.02e-2 2.31e-2 2.61e-2
## 6 Consumpti… 1.05e+1 1.18e+1 1.33e+1 1.49e+1 1.66e+1 1.85e+1 2.05e+1 2.26e+1 2.49e+1 2.73e+1 2.98e+1 3.26e+1 3.54e+1 3.85e+1
## # … with 83 more variables: `2100` <dbl>, `2105` <dbl>, `2110` <dbl>, `2115` <dbl>, `2120` <dbl>, `2125` <dbl>, `2130` <dbl>
## #   `2140` <dbl>, `2145` <dbl>, `2150` <dbl>, `2155` <dbl>, `2160` <dbl>, `2165` <dbl>, `2170` <dbl>, `2175` <dbl>, `2180` <d
## #   `2190` <dbl>, `2195` <dbl>, `2200` <dbl>, `2205` <dbl>, `2210` <dbl>, `2215` <dbl>, `2220` <dbl>, `2225` <dbl>, `2230` <d
## #   `2240` <dbl>, `2245` <dbl>, `2250` <dbl>, `2255` <dbl>, `2260` <dbl>, `2265` <dbl>, `2270` <dbl>, `2275` <dbl>, `2280` <d
## #   `2290` <dbl>, `2295` <dbl>, `2300` <dbl>, `2305` <dbl>, `2310` <dbl>, `2315` <dbl>, `2320` <dbl>, `2325` <dbl>, `2330` <d
## #   `2340` <dbl>, `2345` <dbl>, `2350` <dbl>, `2355` <dbl>, `2360` <dbl>, `2365` <dbl>, `2370` <dbl>, `2375` <dbl>, `2380` <d
## #   `2390` <dbl>, `2395` <dbl>, `2400` <dbl>, `2405` <dbl>, `2410` <dbl>, `2415` <dbl>, `2420` <dbl>, `2425` <dbl>, `2430` <d
```

We want to convert the measure columns into a id column `Year`



Reshaping to a long table

```
#Let's make a long table we will need it for plotting
mdat_2c = dice_csv ▷
  pivot_longer(!Variable, names_to = "Year")

mdat_2c
```

```
## # A tibble: 3,700 × 3
##    Variable                              Year  value
##    <chr>                                 <chr> <dbl>
##  1 Industrial Emissions GTCO2 per year 2015   35.7
##  2 Industrial Emissions GTCO2 per year 2020   39.4
##  3 Industrial Emissions GTCO2 per year 2025   42.9
##  4 Industrial Emissions GTCO2 per year 2030   46.3
##  5 Industrial Emissions GTCO2 per year 2035   49.6
##  6 Industrial Emissions GTCO2 per year 2040   52.7
##  7 Industrial Emissions GTCO2 per year 2045   55.5
##  8 Industrial Emissions GTCO2 per year 2050   58.2
##  9 Industrial Emissions GTCO2 per year 2055   60.6
## 10 Industrial Emissions GTCO2 per year 2060   62.7
## # … with 3,690 more rows
```

Note on data wrangling

Function	data.table	tidyverse
From columns to row	`melt()`	`pivot_longer()`
From row to columns	`dcast()`	`pivot_wider()`
merge 2 data.tables	`merge()`	`full_join()`

In this course we will use `tidyverse`, but you can use also functions from the `data.table` package.

For tips in `tidyverse`:

- Transform data: https://raw.githubusercontent.com/rstudio/cheatsheets/main/dplyr.pdf
- Tidy data: https://raw.githubusercontent.com/rstudio/cheatsheets/main/tidyr.pdf

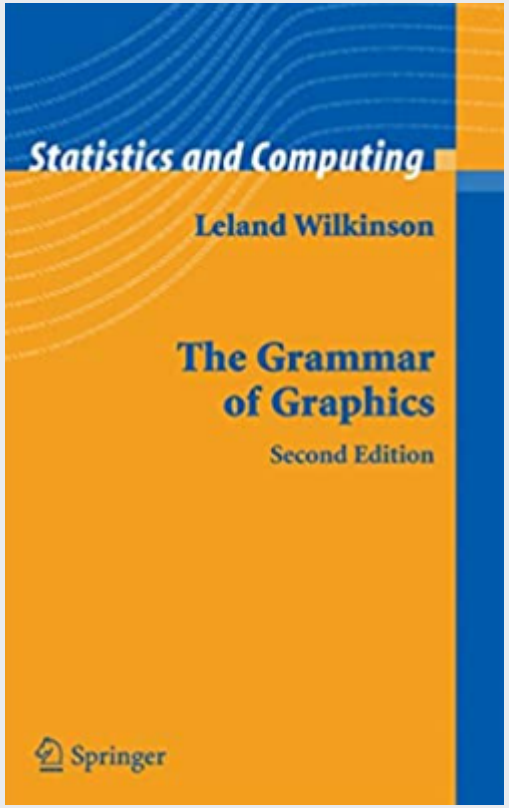
For tips in `data.table`: https://raw.githubusercontent.com/rstudio/cheatsheets/master/datatable.pdf

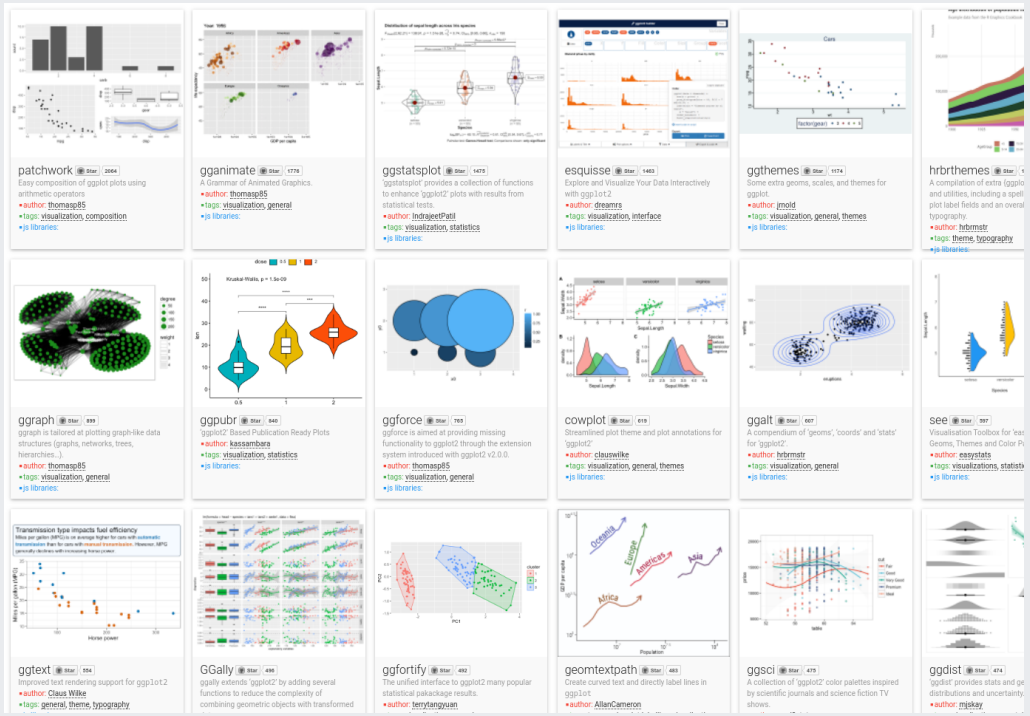
> There are several ways of obtaining the same results.

Data visualization

The `ggplot2` package (included in `tidyverse`)

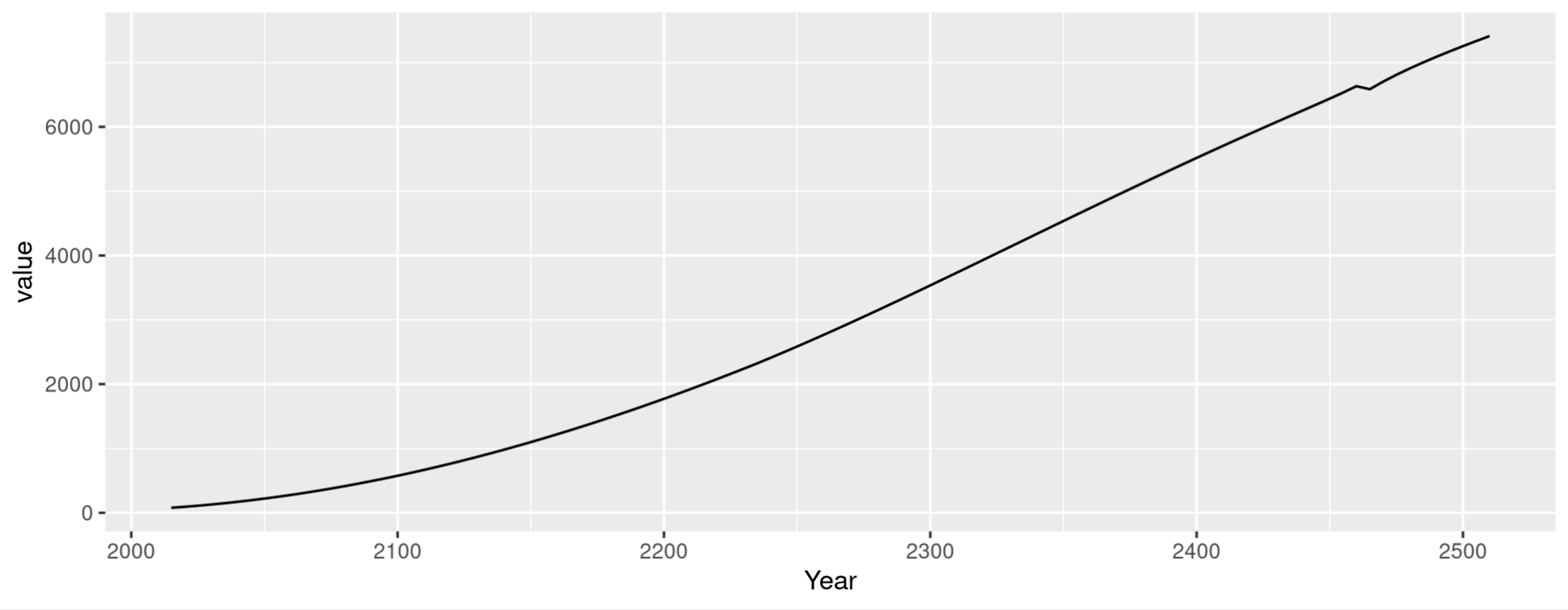
- `ggplot2` implements the grammar of graphics which makes very easy to build a large variety of graphs to explore your data.





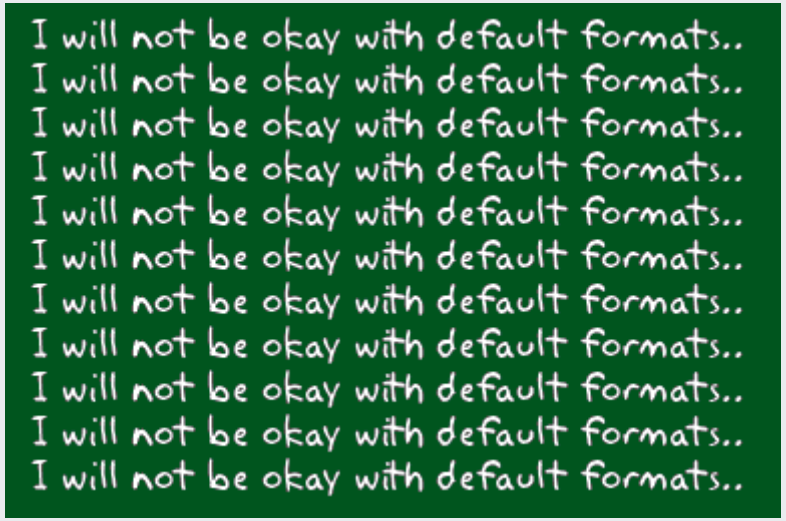
Let's make our first plot (time series)

```
# Year should be numerical
mdat_2c ← mdat_2c ▷ mutate(Year = as.numeric(Year))

# Filter only the variable of interest
ggplot(data = mdat_2c ▷ filter(Variable == 'Consumption'), # DATA
       mapping = aes(x = Year,y = value)) +                # MAPPING
geom_line()                                                # GEOMETRY
```



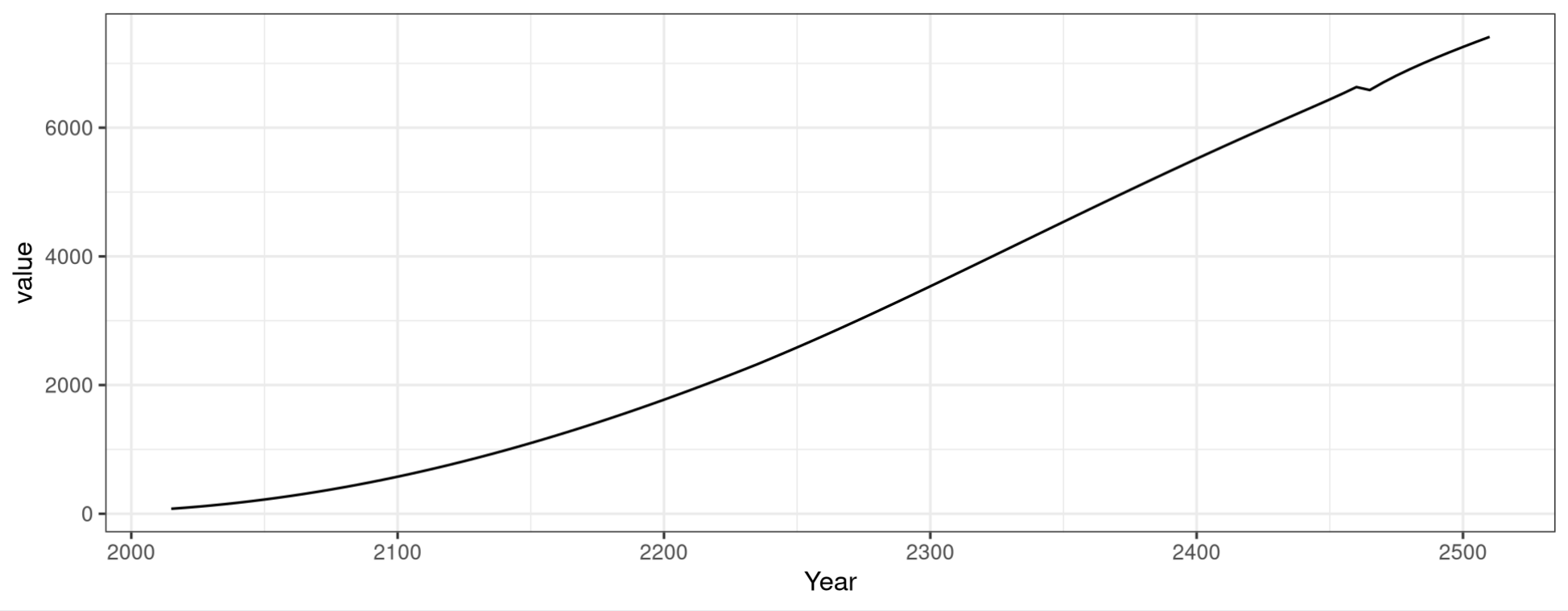
Let's make our first plot (time series)

Take time to design and format your figure



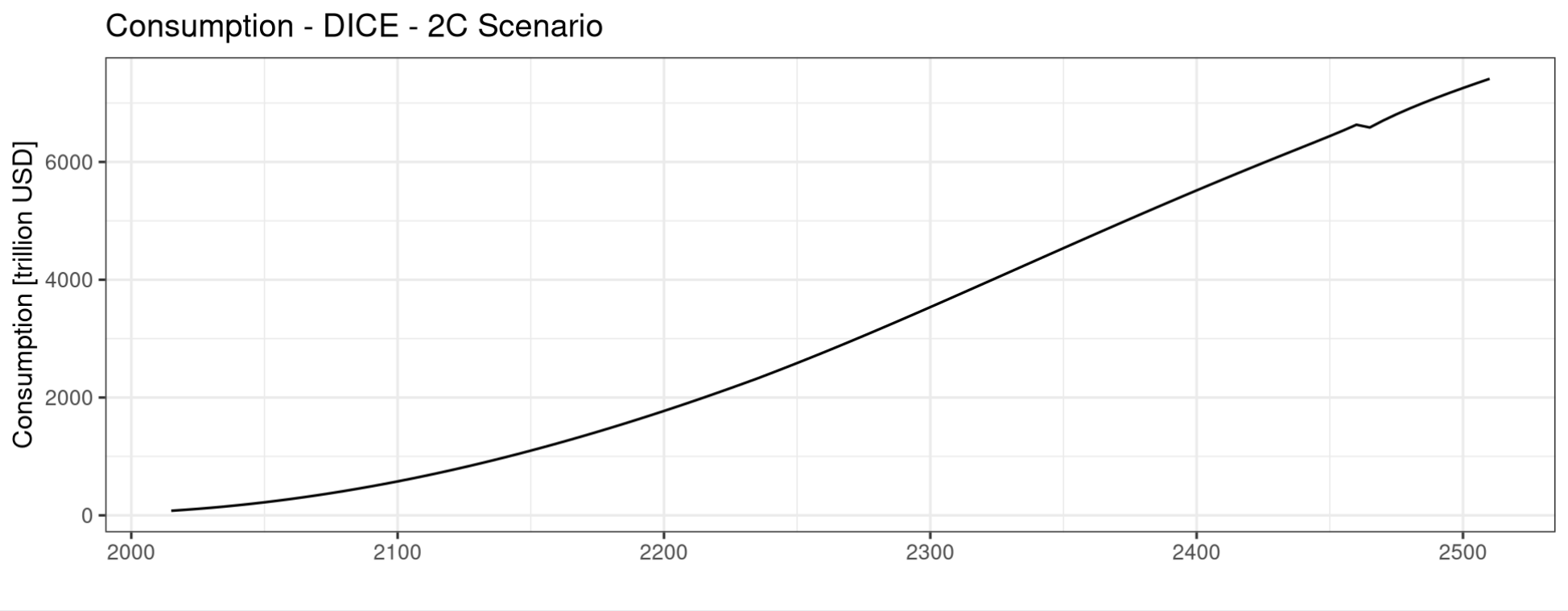
Change the theme

```
ggplot(data = mdat_2c ▷ filter(Variable == 'Consumption'),
       mapping = aes(x = Year,y = value)) +
geom_line() +
theme_bw() # Change overall theme to a more sober and less noisy background
```



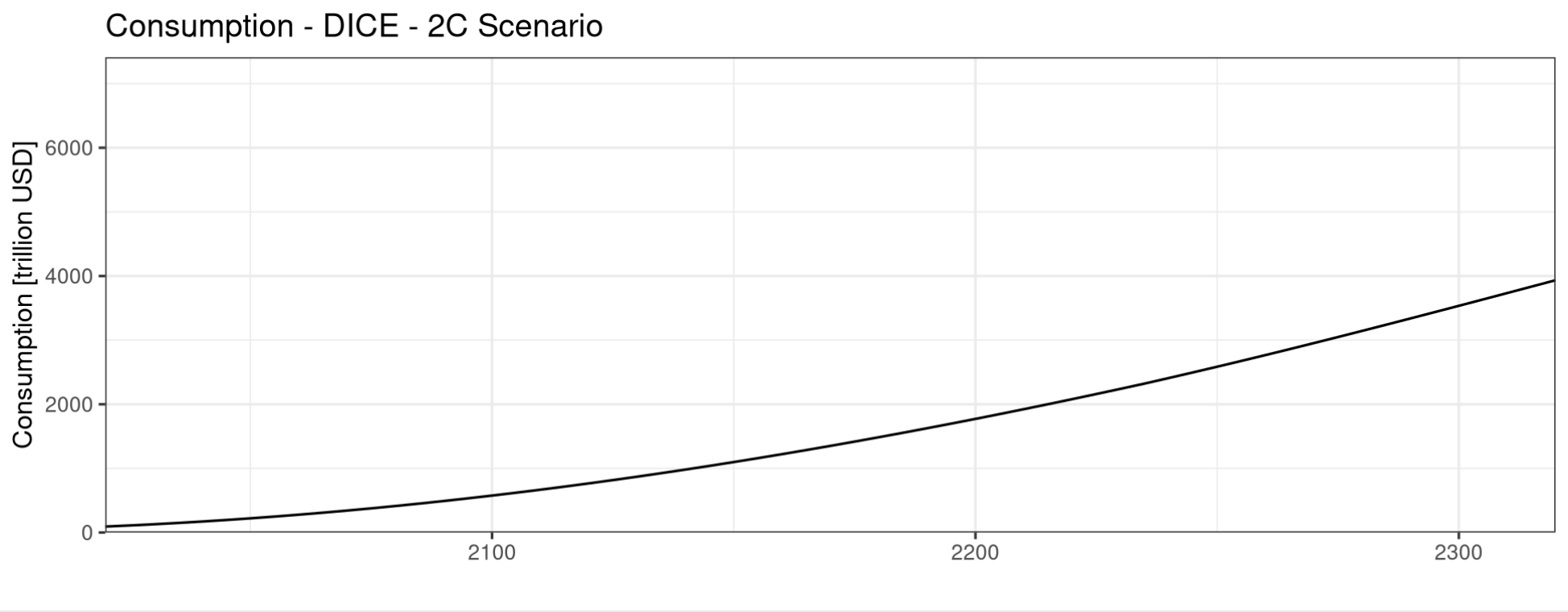
Add a title and labels

```
ggplot(data = mdat_2c ▷ filter(Variable == 'Consumption'),
       mapping = aes(x = Year,y = value)) +
geom_line() +
labs(title = "Consumption - DICE - 2C Scenario", # labs can add subtitle or caption
     x = "",
     y = "Consumption [trillion USD]") +
theme_bw()
```



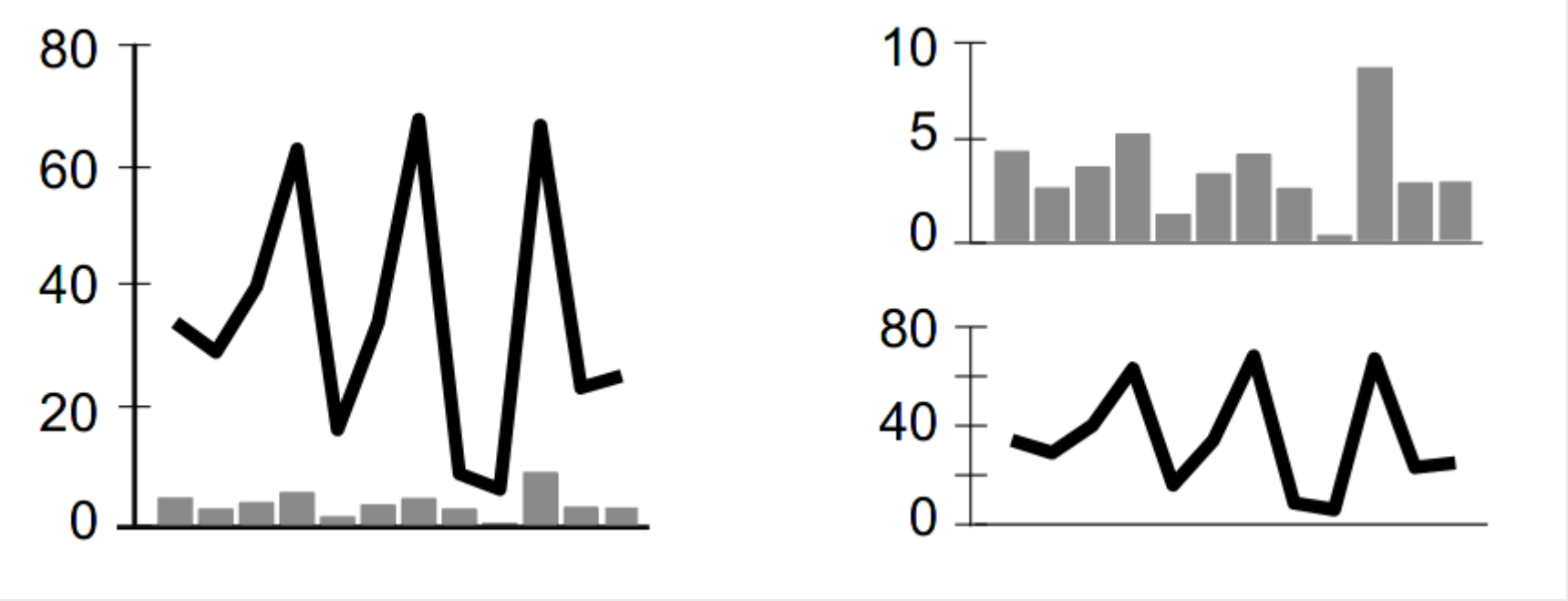
Scale the data

```
ggplot(data = mdat_2c ▷ filter(Variable == 'Consumption'),
       mapping = aes(x = Year,y = value)) +
geom_line() +
labs(title = "Consumption - DICE - 2C Scenario", x = "", y = "Consumption [trillion USD]") +
theme_bw() +
scale_y_continuous(expand = c(0,0), limits = c(0,NA)) +     # Includes 0 in the y-axis
scale_x_continuous(expand = c(0,0), limits = c(2020,2320))  # Zoom-in in the x-axis
```



Let's add more variables

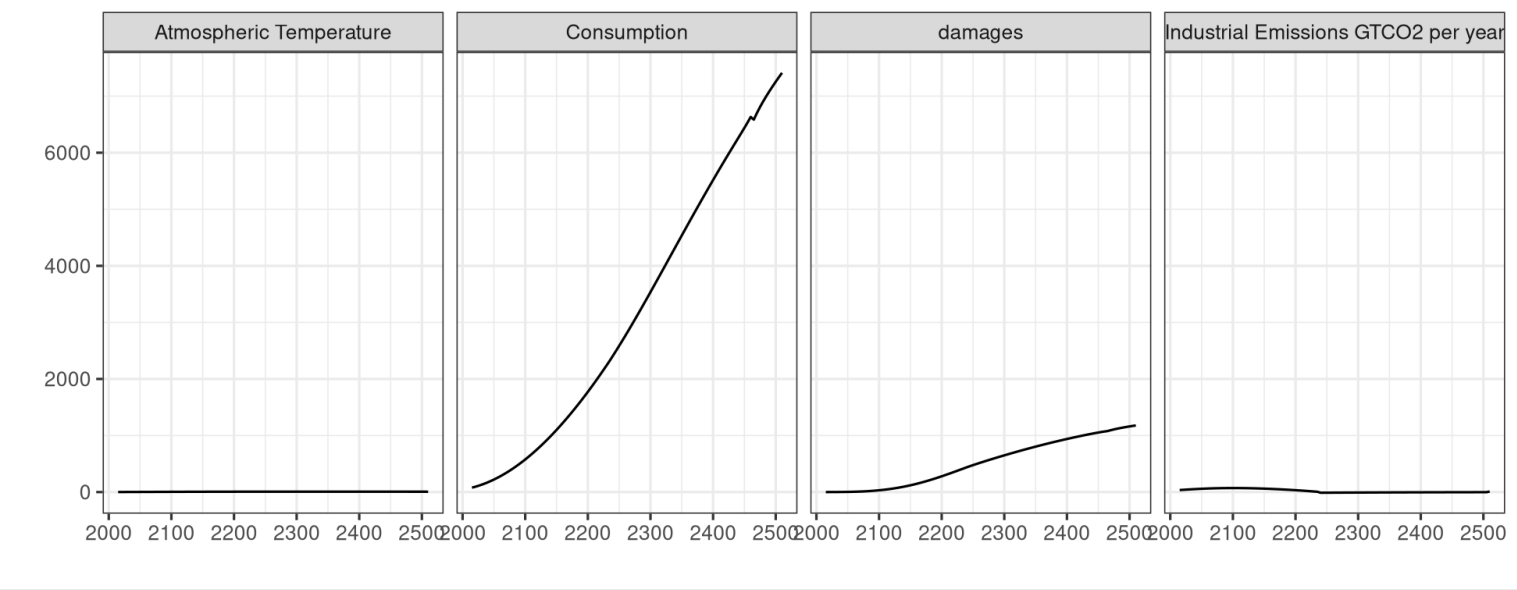
Visualization tip: Do not use mix variables with different scales in the same graph! Separate them into subplots!



source: Kelleher *et al.* (2011)

Use facets

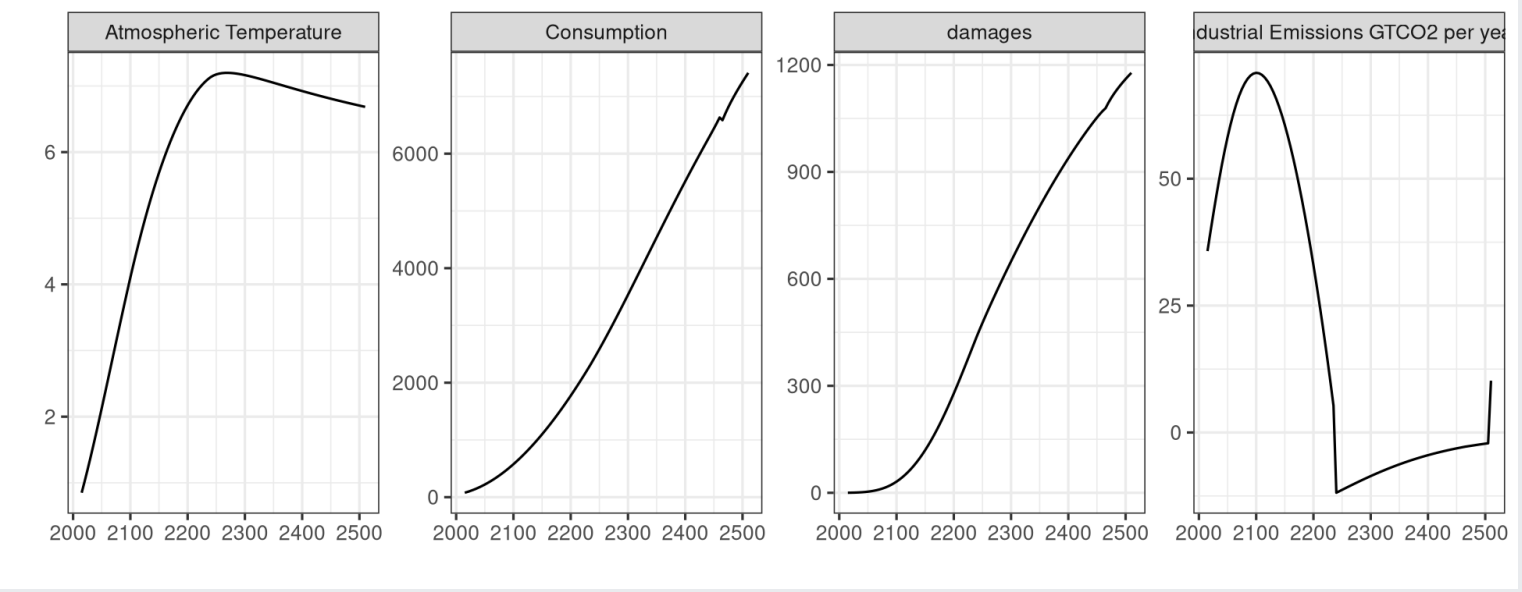
```
vars = c('Consumption','Atmospheric Temperature',
         'Industrial Emissions GTCO2 per year',
         'damages')
ggplot(mdat_2c ▷ filter(Variable %in% vars),
       aes(x = Year, y = value)) +
  geom_line() +
  labs(x = "", y = "") +
  facet_wrap(~Variable, ncol = 4) +
  theme_bw()
```



Use facets, different scales

What do you want to show? Magnitude or relative magnitude?

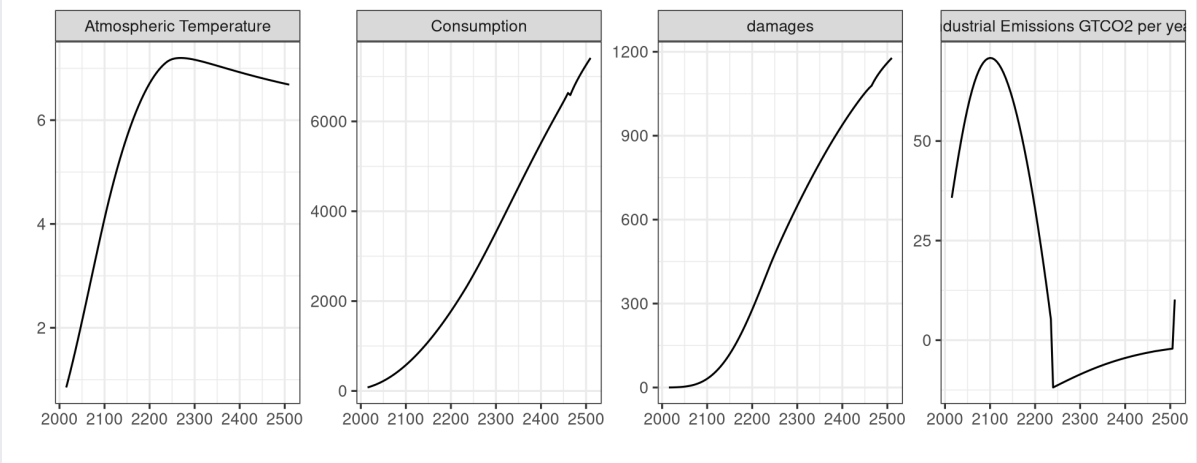
```
vars = c('Consumption','Atmospheric Temperature',
         'Industrial Emissions GTCO2 per year',
         'damages')
ggplot(mdat_2c ▷ filter(Variable %in% vars),
       aes(x = Year, y = value)) +
  geom_line() +
  labs(x = "", y = "") +
  facet_wrap(~Variable, ncol = 4, scales='free') + # ← free scales
  theme_bw()
```



Save your graph

You can store the graph in a variable!

```
p ← ggplot(mdat_2c ▷ filter(Variable %in% vars), aes(x = Year, y = value)) +
  geom_line() +
  labs(x = "", y = "") +
  facet_wrap(~Variable, ncol = 4, scales='free') + # ← free scales
  theme_bw()
p
```




```
ggsave(filename = "Fig1.pdf", plot = p, width = 21, height = 29.7, units = "cm") # A4 format
ggsave(filename = "Fig1.png", plot = p, width = 8, height = 5) # smaller png file
```

Advanced techniques

2D graphs - lines plots

First, read 3 scenarios and organize your data

```
# read several GDX (several scenarios with different carbon taxes)
myfiles ← file.path("Material", c("results_ssp2_bau.gdx","results_ssp2_ctax50.gdx","results_ssp2_ctax100.gdx"))
res ← batch_extract("Q_EMI", files = myfiles) # load GHG emissions
# have a look at the structure
str(res) # a list of a data frame
```

```
## List of 1
##  $ Q_EMI:'data.frame':    50592 obs. of  5 variables:
##   ..$ e    : chr [1:50592] "nip" "nip" "nip" "nip" ...
##   ..$ t    : chr [1:50592] "1" "1" "1" "1" ...
##   ..$ n    : chr [1:50592] "brazil" "canada" "china" "europe" ...
##   ..$ value: num [1:50592] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ gdx  : chr [1:50592] "Material/results_ssp2_bau.gdx" "Material/results_ssp2_bau.gdx" "Material/results_ssp2_bau.gdx" "M
##   ..- attr(*, "gams")= chr ""
```

```
# this grabs the data.frame inside the list GHG$Q_EMI
qemi = res$Q_EMI ▷ as_tibble()
# The element in a list can also be accessed with res[[1]] or res[['Q_EMI']]
```

2D graphs - lines plots

```
# Have a look now
qemi
```

```
## # A tibble: 50,592 × 5
##    e     t     n         value gdx
##    <chr> <chr> <chr>     <dbl> <chr>
##  1 nip   1     brazil        0 Material/results_ssp2_bau.gdx
##  2 nip   1     canada        0 Material/results_ssp2_bau.gdx
##  3 nip   1     china         0 Material/results_ssp2_bau.gdx
##  4 nip   1     europe        0 Material/results_ssp2_bau.gdx
##  5 nip   1     india         0 Material/results_ssp2_bau.gdx
##  6 nip   1     indonesia     0 Material/results_ssp2_bau.gdx
##  7 nip   1     jpnkor        0 Material/results_ssp2_bau.gdx
##  8 nip   1     laca          0 Material/results_ssp2_bau.gdx
##  9 nip   1     mena          0 Material/results_ssp2_bau.gdx
## 10 nip   1     mexico        0 Material/results_ssp2_bau.gdx
## # … with 50,582 more rows
```

```
qemi ← qemi ▷
  mutate(year = as.numeric(t) * 5 + 2000) ▷ # Transform time period into year
  mutate(scen = basename(gdx)) ▷            # Transform the filename into a scenario name
  mutate(scen = str_replace(scen, ".gdx", "")) ▷
  mutate(scen = str_replace(scen, "results_ssp2_", ""))

# basename returns the filename without the path
# str_replace replace a given string pattern
```

2D graphs - lines plots

```
# Update the scenario as factors (ordered set)
qemi ← qemi ▷
  mutate(scen = factor(scen, levels = c('bau','ctax50','ctax100')))

# let's have a look again
qemi
```

```
## # A tibble: 50,592 × 7
##    e     t     n         value gdx                          year scen
##    <chr> <chr> <chr>     <dbl> <chr>                       <dbl> <fct>
##  1 nip   1     brazil        0 Material/results_ssp2_bau.gdx  2005 bau
##  2 nip   1     canada        0 Material/results_ssp2_bau.gdx  2005 bau
##  3 nip   1     china         0 Material/results_ssp2_bau.gdx  2005 bau
##  4 nip   1     europe        0 Material/results_ssp2_bau.gdx  2005 bau
##  5 nip   1     india         0 Material/results_ssp2_bau.gdx  2005 bau
##  6 nip   1     indonesia     0 Material/results_ssp2_bau.gdx  2005 bau
##  7 nip   1     jpnkor        0 Material/results_ssp2_bau.gdx  2005 bau
##  8 nip   1     laca          0 Material/results_ssp2_bau.gdx  2005 bau
##  9 nip   1     mena          0 Material/results_ssp2_bau.gdx  2005 bau
## 10 nip   1     mexico        0 Material/results_ssp2_bau.gdx  2005 bau
## # … with 50,582 more rows
```

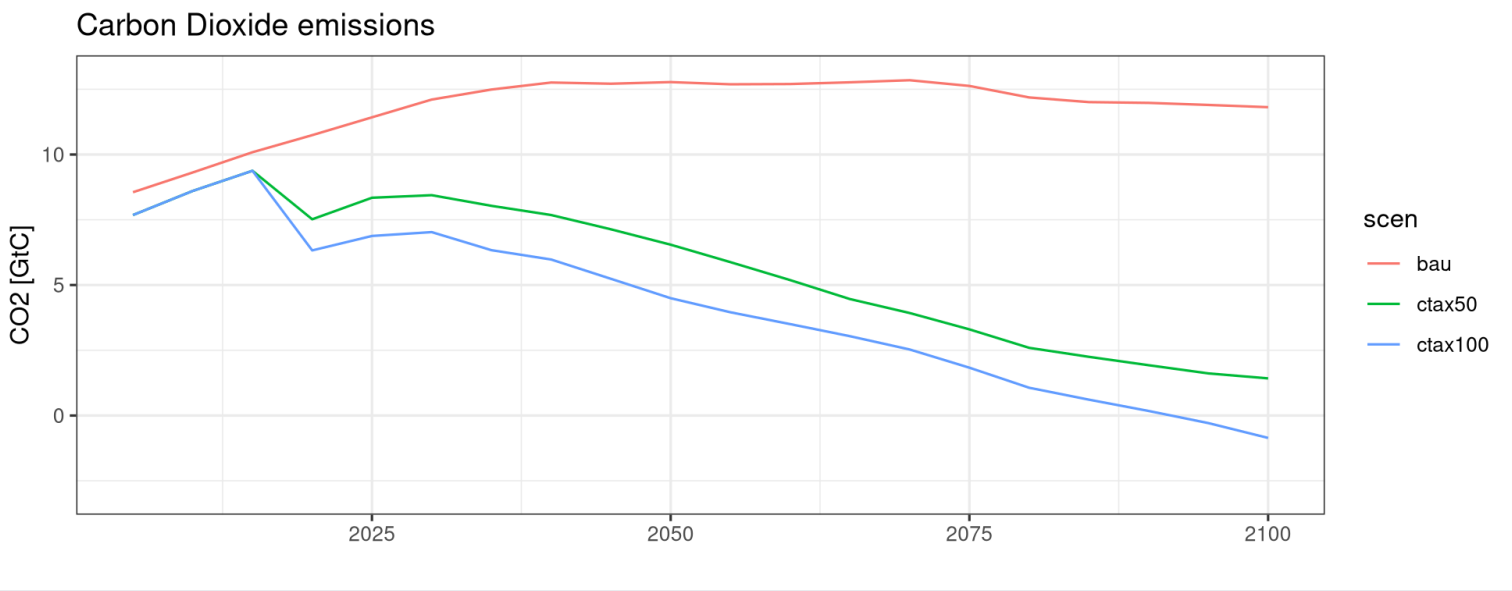
2D graphs - lines plots

```
# We have 4 id variables, let's reduce it to 3 by aggregation regions into world
wemi ← qemi ▷
  group_by(scen,e,year) ▷
  summarise(value = sum(value))

# have a look
wemi
```

```
## # A tibble: 2,976 × 4
## # Groups:   scen, e [101]
##    scen  e      year   value
##    <fct> <chr> <dbl>   <dbl>
##  1 bau   c2f6   2005 0.00689
##  2 bau   c2f6   2010 0.00923
##  3 bau   c2f6   2015 0.0107
##  4 bau   c2f6   2020 0.0113
##  5 bau   c2f6   2025 0.0113
##  6 bau   c2f6   2030 0.0108
##  7 bau   c2f6   2035 0.00986
##  8 bau   c2f6   2040 0.00870
##  9 bau   c2f6   2045 0.00742
## 10 bau   c2f6   2050 0.00616
## # … with 2,966 more rows
```

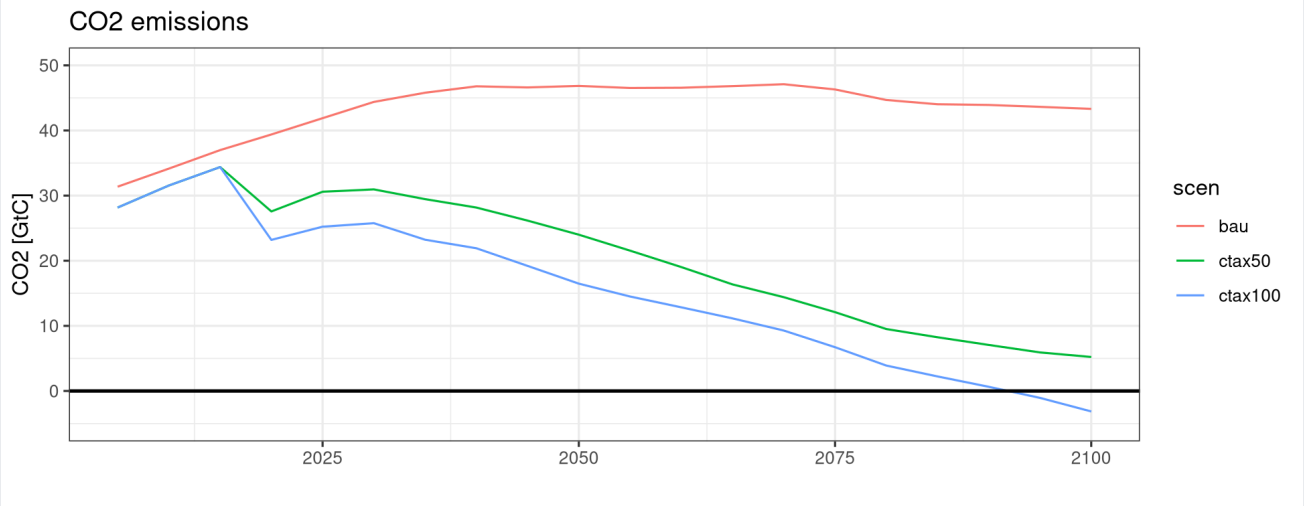
2D graphs - lines plots

```
ggplot(wemi ▷ filter(e == 'co2' & year ⩽ 2100),
       aes(x = year, y = value, color = scen)) +
  geom_line() +
  labs(x = "", y = "CO2 [GtC]", title = "Carbon Dioxide emissions") +
  theme_bw() +
  scale_x_continuous(limits=c(2005, 2100)) +
  scale_y_continuous(limits=c(-3, 13))
```



lines plots - meaningful messages

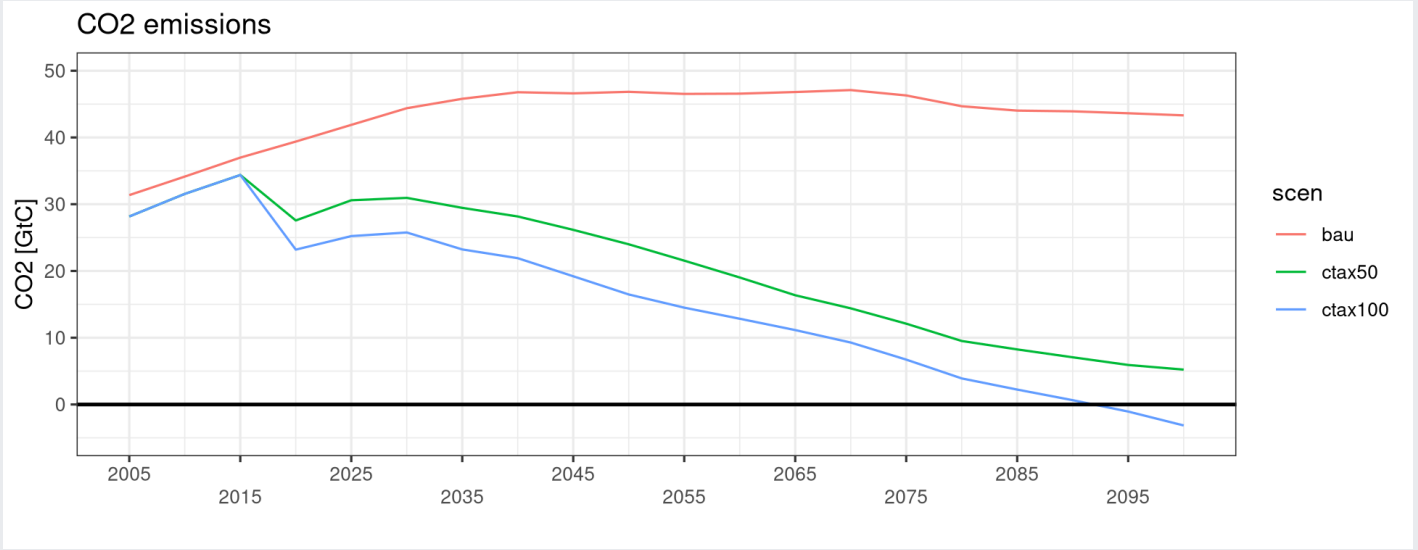
```r
# Let's change unit from GtC to GtCO2
wemi ← wemi ▷
  mutate(value = value * 44 / 12)

# Now let's have a look at our data
ggplot(wemi ▷ filter(e == 'co2' & year ⩽ 2100),
       aes(x = year, y = value, color = scen)) +
  geom_line() +
  geom_hline(yintercept = 0, color = 'Black', size = 0.8) +
  labs(x = "", y = "CO2 [GtC]", title = "CO2 emissions") +
  theme_bw() +
  scale_x_continuous(limits=c(2005, 2100)) +
  scale_y_continuous(limits=c(-5, 50))
```



2D graphs - lines plots - adjust axis

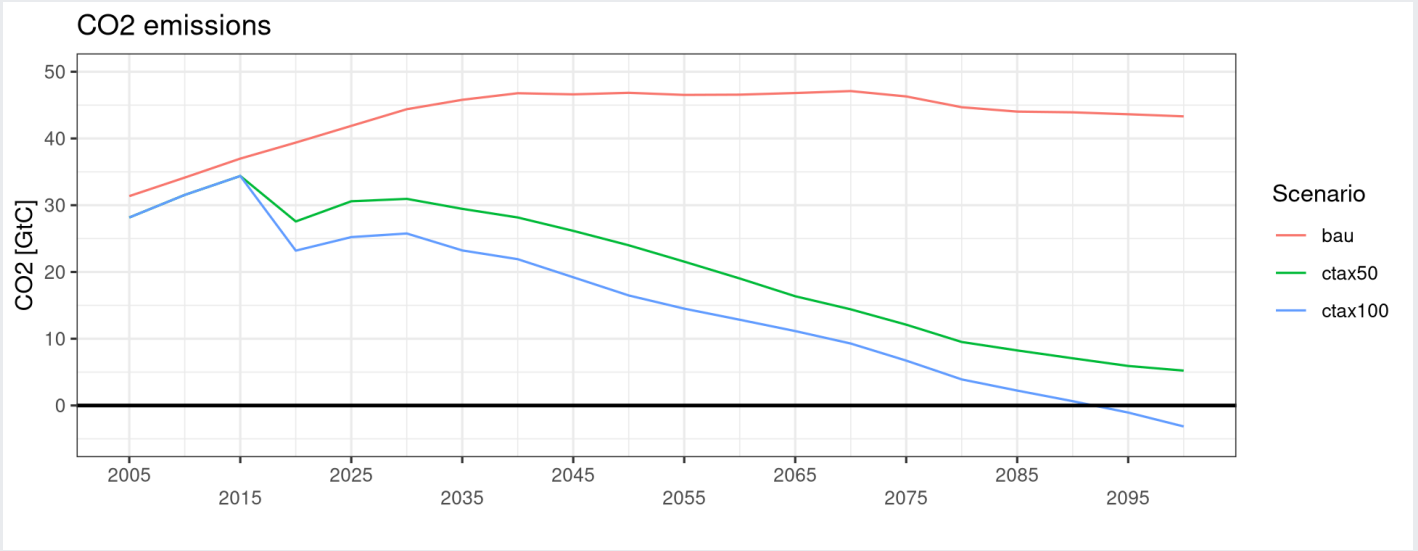
Ease the eye

```
ggplot(wemi ▷ filter(e == 'co2' & year ⩽ 2100),
       aes(x = year, y = value, color = scen)) +
  geom_line() +
  geom_hline(yintercept = 0, color = 'Black', size = 0.8) +
  labs(x = "", y = "CO2 [GtC]", title = "CO2 emissions") +
  theme_bw() +
  scale_x_continuous(limits=c(2005, 2100),
                     breaks=seq(2005, 2100,by = 10),    # Define the axis tiks breaks
                     guide = guide_axis(n.dodge = 2)) + # avoid overlapping
  scale_y_continuous(limits=c(-5, 50))
```



Better label for colors

```
ggplot(wemi ▷ filter(e == 'co2' & year ⩽ 2100),
        aes(x = year, y = value, color = scen)) +
  geom_line() +
  geom_hline(yintercept = 0, color = 'Black', size = 0.8) +
  labs(x = "", y = "CO2 [GtC]", title = "CO2 emissions") +
  theme_bw() +
  scale_x_continuous(limits=c(2005, 2100),
                     breaks=seq(2005, 2100,by = 10),    # Define the axis tiks breaks
                     guide = guide_axis(n.dodge = 2)) + # avoid overlapping
  scale_y_continuous(limits=c(-5, 50)) +
  scale_color_discrete(name = "Scenario")
```

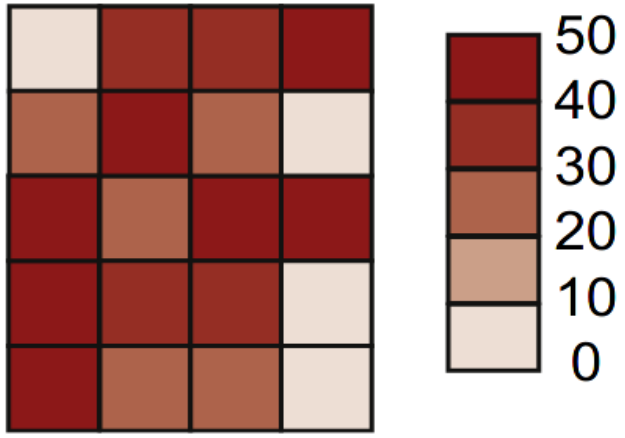


Choose your colors

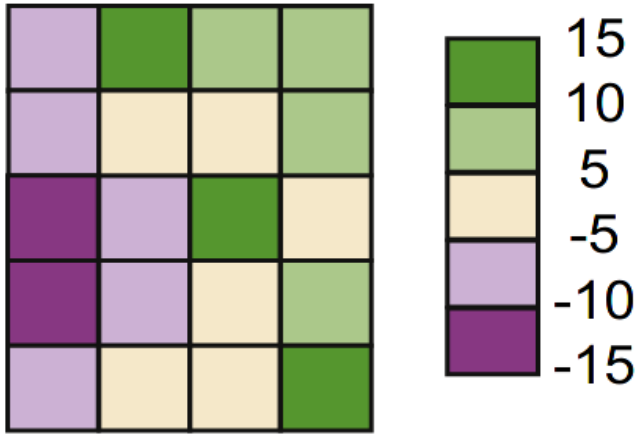
Here are some libraries with color palettes

```
#install them first if you have not yet done it
#install.packages('viridis')
library(ggsci) # scientific colors
library(viridis) #looks good and includes many options
library(RColorBrewer) # wide choice
library(wesanderson) # for the nostalgic days
```

Same magnitude



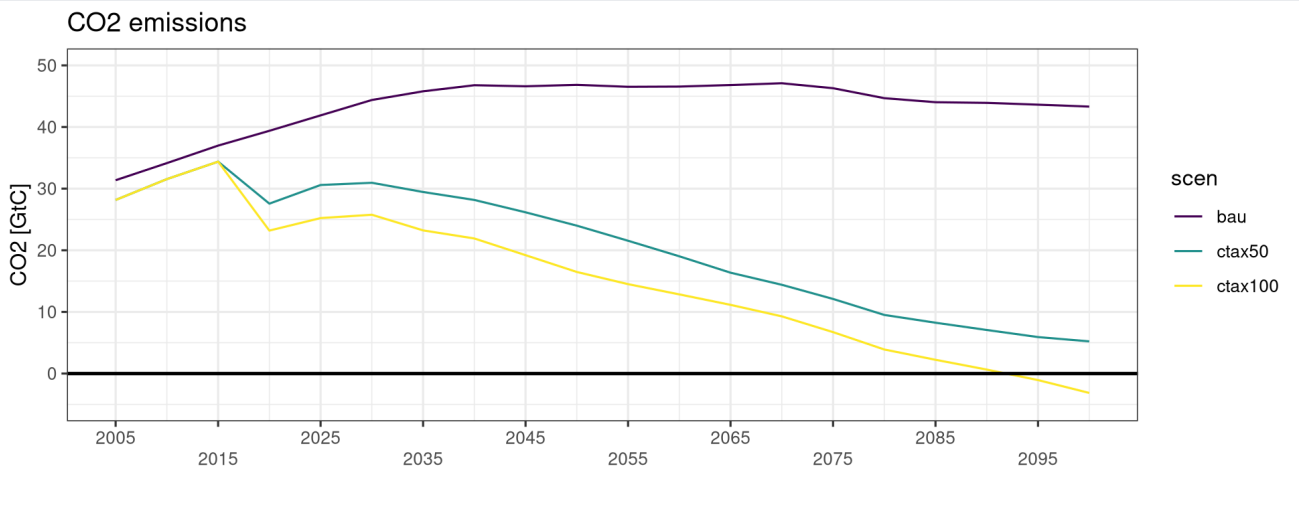
Divergent (zero in the scale)



source: Kelleher *et al.* (2011)

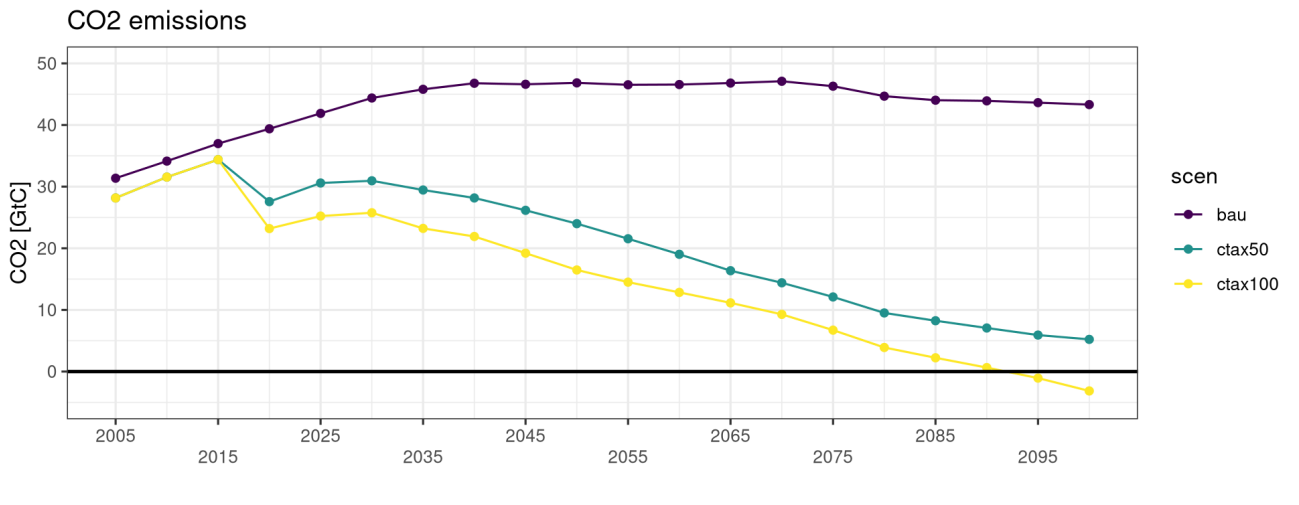
Line plot - Colors

```
ggplot(wemi |> filter(e == 'co2' & year <= 2100),
       aes(x = year, y = value, color = scen)) +
  geom_line() +
  geom_hline(yintercept = 0, color = 'Black', size = 0.8) +
  labs(x = "", y = "CO2 [GtC]", title = "CO2 emissions") +
  theme_bw() +
  scale_x_continuous(limits=c(2005, 2100),
                     breaks=seq(2005, 2100,by = 10),
                     guide = guide_axis(n.dodge = 2)) +
  scale_y_continuous(limits=c(-5, 50)) +
  scale_color_viridis(discrete = TRUE, option = "D") # use "?" to check the options
```



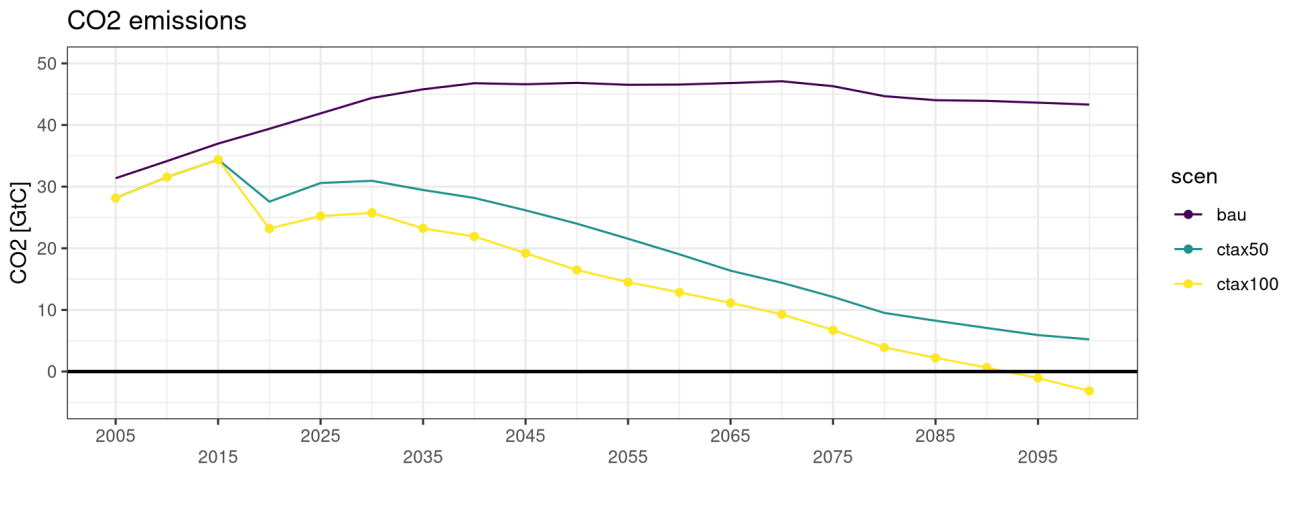
Line plot - Add points

```
ggplot(wemi |> filter(e == 'co2' & year <= 2100),
       aes(x = year, y = value, color = scen)) +
  geom_line() +
  geom_point() +  # NEW GEOMETRY, SAME DATA AND MAPPING
  geom_hline(yintercept = 0, color = 'Black', size = 0.8) +
  labs(x = "", y = "CO2 [GtC]", title = "CO2 emissions") +
  theme_bw() +
  scale_x_continuous(limits=c(2005, 2100),
                     breaks=seq(2005, 2100,by = 10),
                     guide = guide_axis(n.dodge = 2)) +
  scale_y_continuous(limits=c(-5, 50)) +
  scale_color_viridis(discrete = TRUE, option = "D")
```



Line plot - Subset different data

```
ggplot(wemi |> filter(e == 'co2' & year <= 2100),
       aes(x = year, y = value, color = scen)) +
  geom_line() +
  geom_point(data = wemi |> filter(e == 'co2' & year <= 2100 & scen == "ctax100")) +# NEW GEOMETRY, SUBSET BUT SAME MAPI
  geom_hline(yintercept = 0, color = 'Black', size = 0.8) +
  labs(x = "", y = "CO2 [GtC]", title = "CO2 emissions") +
  theme_bw() +
  scale_x_continuous(limits=c(2005, 2100),
                     breaks=seq(2005, 2100,by = 10),
                     guide = guide_axis(n.dodge = 2)) +
  scale_y_continuous(limits=c(-5, 50)) +
  scale_color_viridis(discrete = TRUE, option = "D")
```



Scatter plots - load more data

Let's get another variable and merge it with our regional Emissions (GHG)

```
pop ← batch_extract("l",files =  myfiles)[[1]] ▷ as_tibble() # Load population

# let's transform time periods into Years
pop ← pop ▷
  mutate(year = as.numeric(t) * 5 + 2000) ▷ # Transform time period into year
  mutate(scen = basename(gdx)) ▷            # Transform the filename into a scenario name
  mutate(scen = str_replace(scen, ".gdx", "")) ▷
  mutate(scen = str_replace(scen, "results_ssp2_", ""))
```

Scatter plots - Merging two data.table objects

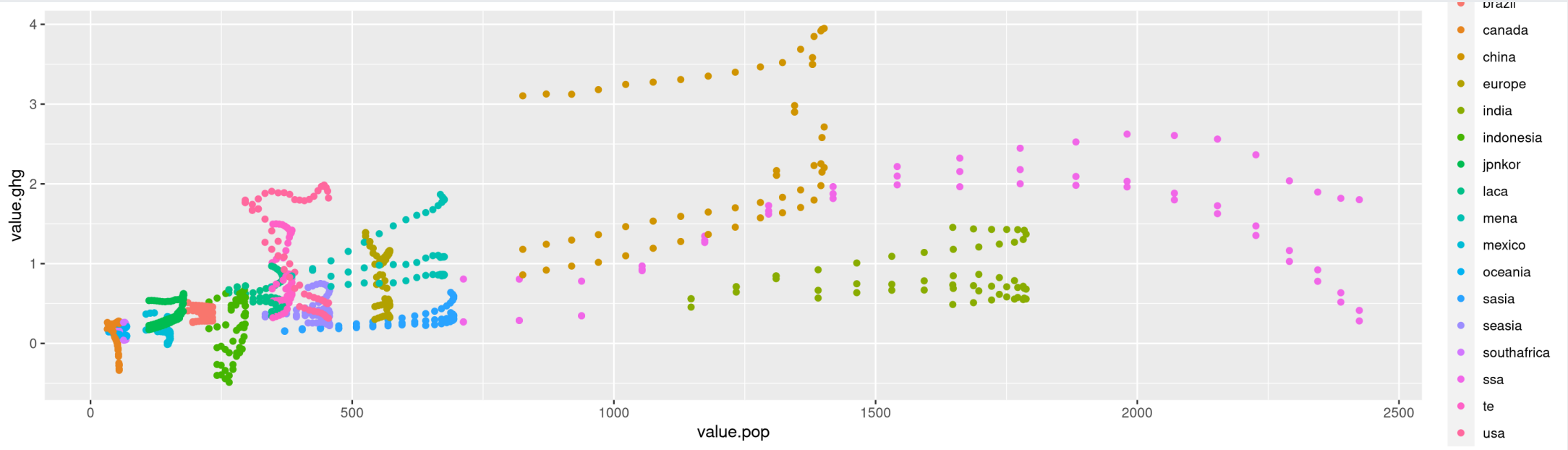
Let's get another variable and merge it with our regional Emissions (GHG)

```
# let's keep 2 emission parameters
 dat = full_join(qemi  ▷ filter(e %in% c('kghg','ccs')) ▷ select(-gdx,-t),
                 pop  ▷ select(-gdx,-t),
                 by = c("scen","year","n"),
                 suffix = c(".ghg",".pop"))
 # have a look, it's magical
 dat
```

```
## # A tibble: 3,060 × 6
##    e     n         value.ghg  year scen  value.pop
##    <chr> <chr>         <dbl> <dbl> <chr>     <dbl>
##  1 kghg  brazil        0.512  2005 bau       186.
##  2 kghg  canada        0.263  2005 bau        32.3
##  3 kghg  china         2.17   2005 bau      1311.
##  4 kghg  europe        1.34   2005 bau       526.
##  5 kghg  india         0.455  2005 bau      1147.
##  6 kghg  indonesia     0.520  2005 bau       226.
##  7 kghg  jpnkor        0.540  2005 bau       176.
##  8 kghg  laca          0.634  2005 bau       264.
##  9 kghg  mena          0.706  2005 bau       346.
## 10 kghg  mexico        0.167  2005 bau       106.
## # … with 3,050 more rows
```

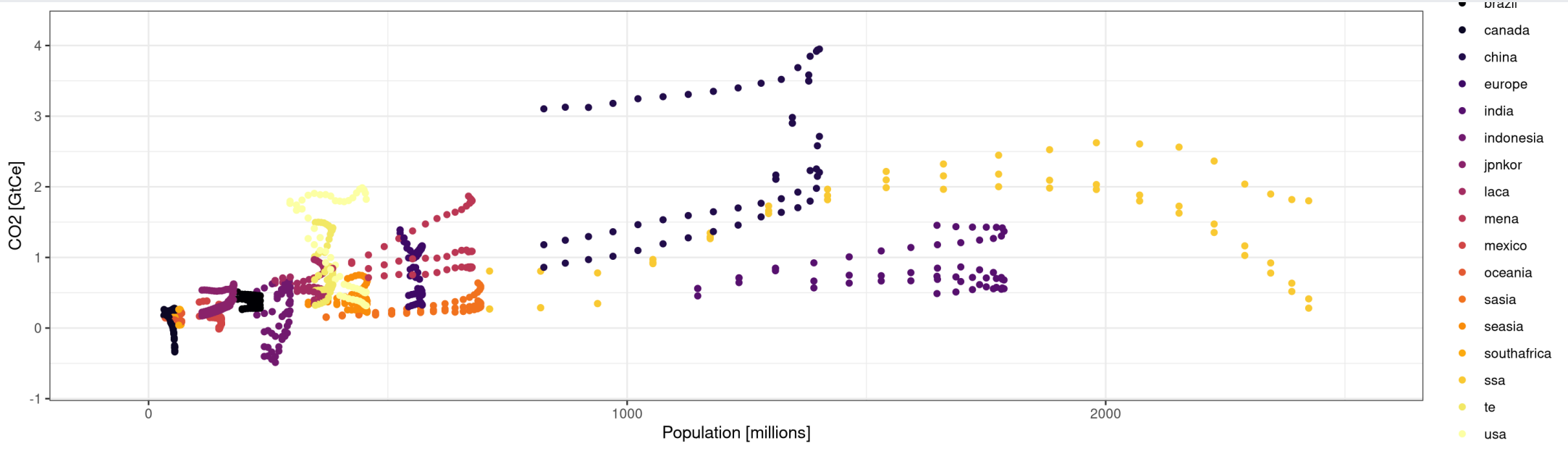
Scatter plots - Adding dimentions

```
ggplot(dat ▷ filter(year < 2100 & e == 'kghg'),
       aes(x = value.pop, y = value.ghg, color = n)) +
 geom_point()
```



Scatter plots - Refuse the defaut options!

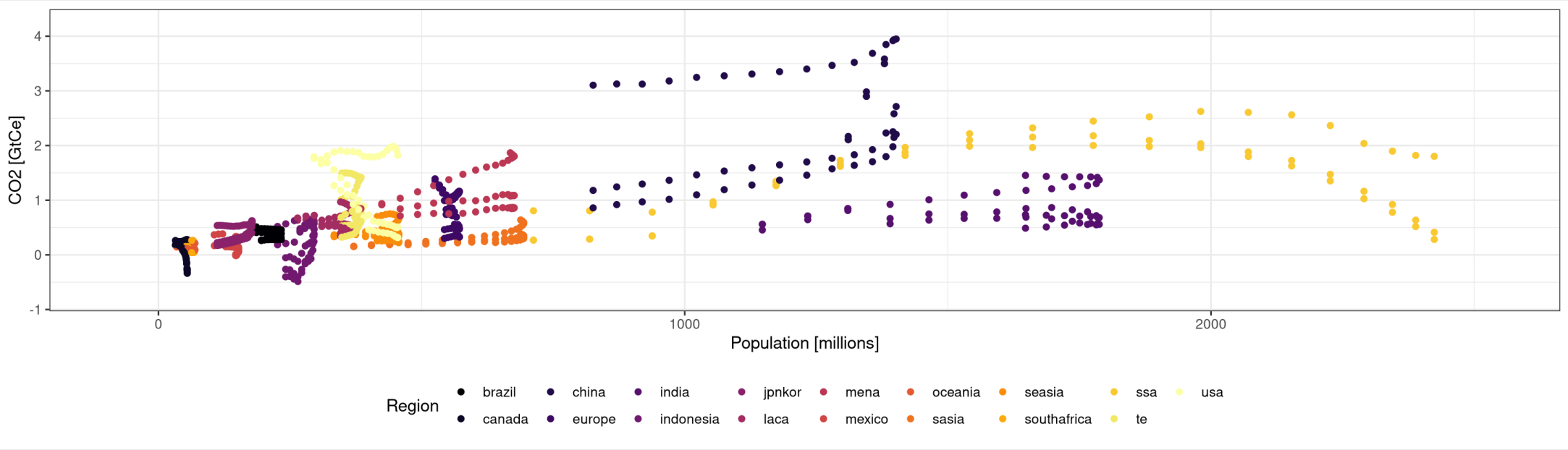
```
ggplot(dat ▷ filter(year < 2100 & e == 'kghg'),
       aes(x = value.pop,y = value.ghg, color = n)) +
  geom_point() +
  scale_x_continuous(name = "Population [millions]",expand = c(0.1, 0.1)) +
  scale_y_continuous(name = "CO2 [GtCe]",expand = c(0.1, 0.1))+
  scale_color_viridis(discrete = TRUE, option = "B", name = "Region") +
  theme_bw()
```



For nice and ready to used scientific themes, check the package `ggpubr`

Scatter plots - Adjust legend

```
ggplot(dat |> filter(year < 2100 & e == 'kghg'),
       aes(x = value.pop,y = value.ghg, color = n)) +
  geom_point() +
  scale_x_continuous(name = "Population [millions]",expand = c(0.1, 0.1)) +
  scale_y_continuous(name = "CO2 [GtCe]",expand = c(0.1, 0.1))+
  scale_color_viridis(discrete = TRUE, option = "B", name = "Region") +
  guides(col = guide_legend(ncol = 9)) + # Legend in 9 columns
  theme_bw() +
  theme(legend.position = "bottom")      # Legend in the bottom
```



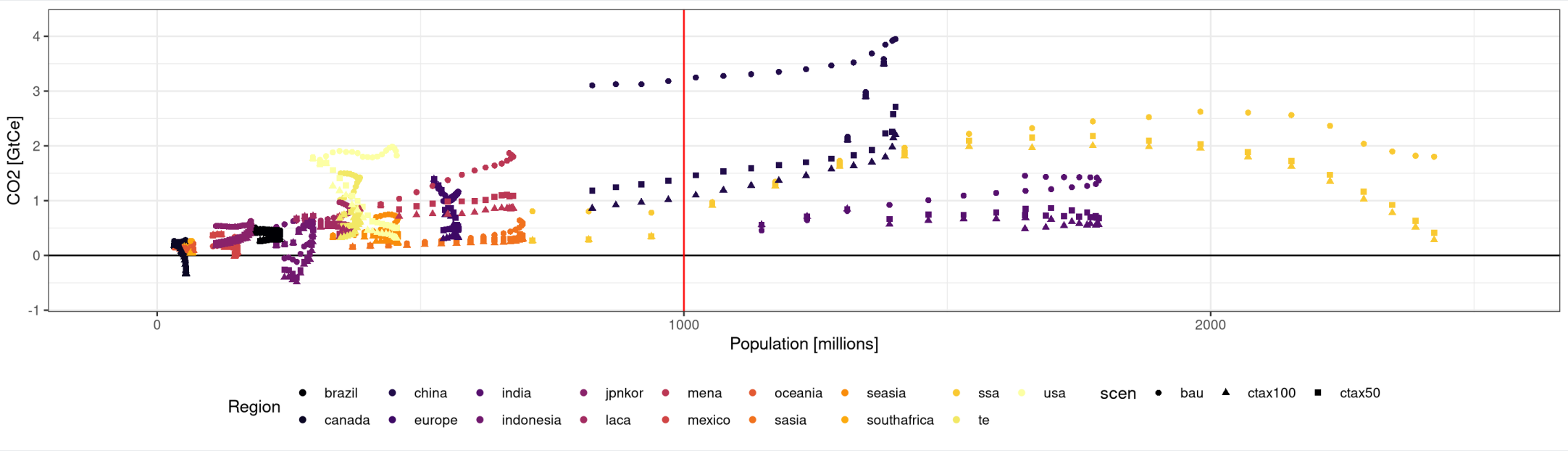
Scatter plots - Details are important

Change names of legend and add guide lines

```
ggplot(dat |> filter(year < 2100 & e == 'kghg'),
       aes(x = value.pop,y = value.ghg, color = n)) +
  geom_hline(yintercept = 0) +                  # add a horizontal line at 0
  geom_vline(xintercept = 1000, color = 'red') + # Separate low-high population regions
  geom_point() +
  scale_x_continuous(name = "Population [millions]",expand = c(0.1, 0.1)) +
  scale_y_continuous(name = "CO2 [GtCe]",expand = c(0.1, 0.1))+
  scale_color_viridis(discrete = TRUE, option = "B", name = "Region") +
  guides(col = guide_legend(ncol = 9)) +
  theme_bw() +
  theme(legend.position = "bottom")
```

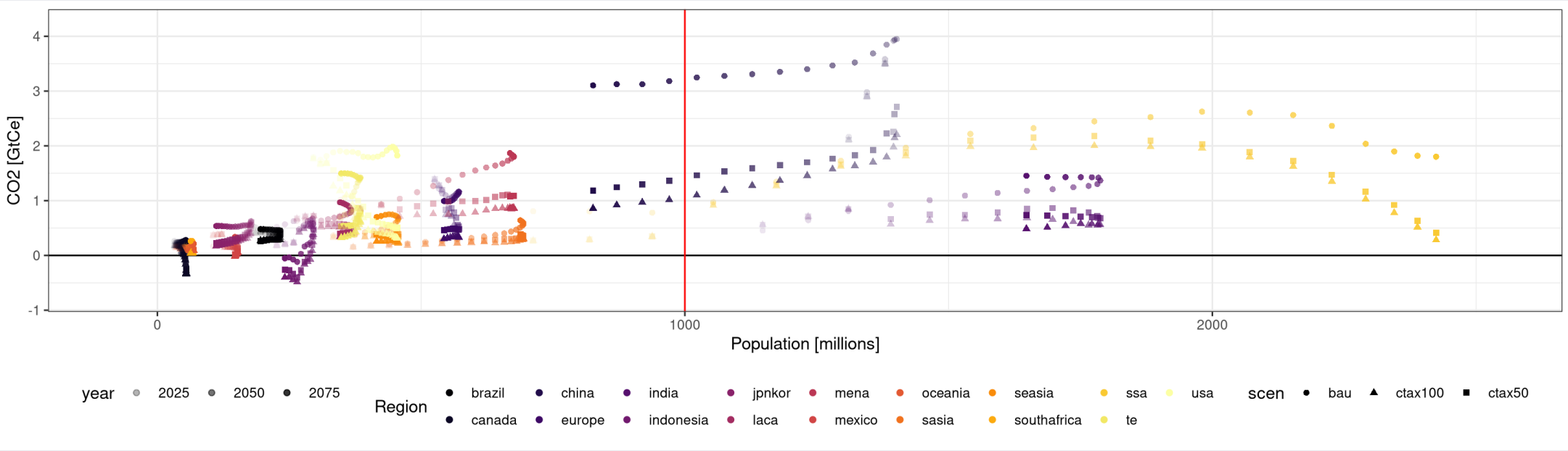
Scatter plots - Add one more dimention

```
ggplot(dat ▷ filter(year < 2100 & e == 'kghg'),
       aes(x = value.pop,y = value.ghg, color = n, shape = scen)) + # add shape
  geom_hline(yintercept = 0) + geom_vline(xintercept = 1000, color = 'red') +
  geom_point() + theme_bw() +
  scale_x_continuous(name = "Population [millions]",expand = c(0.1, 0.1)) +
  scale_y_continuous(name = "CO2 [GtCe]",expand = c(0.1, 0.1))+
  scale_color_viridis(discrete = TRUE, option = "B", name = "Region") +
  guides(col = guide_legend(ncol = 9)) +
  theme(legend.position = "bottom")
```



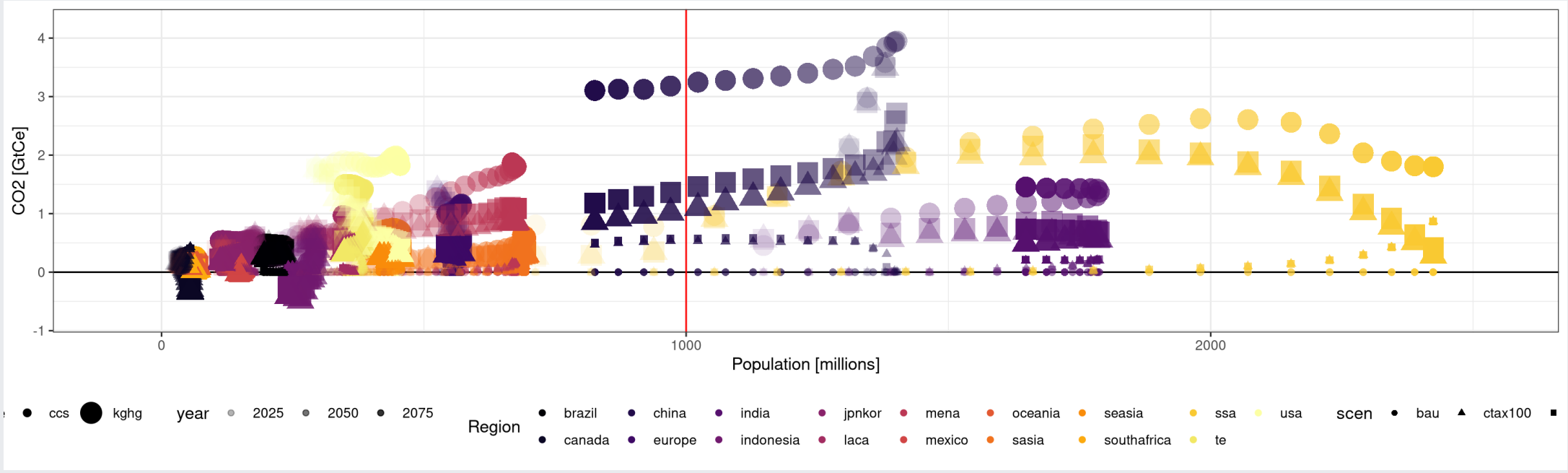
Scatter plots - Add yet another dimention

```
ggplot(dat ▷ filter(year < 2100 & e == 'kghg'),
       aes(x = value.pop,y = value.ghg, color = n, shape = scen, alpha = year))+ # add transparency
  geom_hline(yintercept = 0) + geom_vline(xintercept = 1000, color = 'red') +
  geom_point() +theme_bw() +
  scale_x_continuous(name = "Population [millions]",expand = c(0.1, 0.1)) +
  scale_y_continuous(name = "CO2 [GtCe]",expand = c(0.1, 0.1))+
  scale_color_viridis(discrete = TRUE, option = "B", name = "Region") +
  guides(col = guide_legend(ncol = 9)) +
  theme(legend.position = "bottom")
```



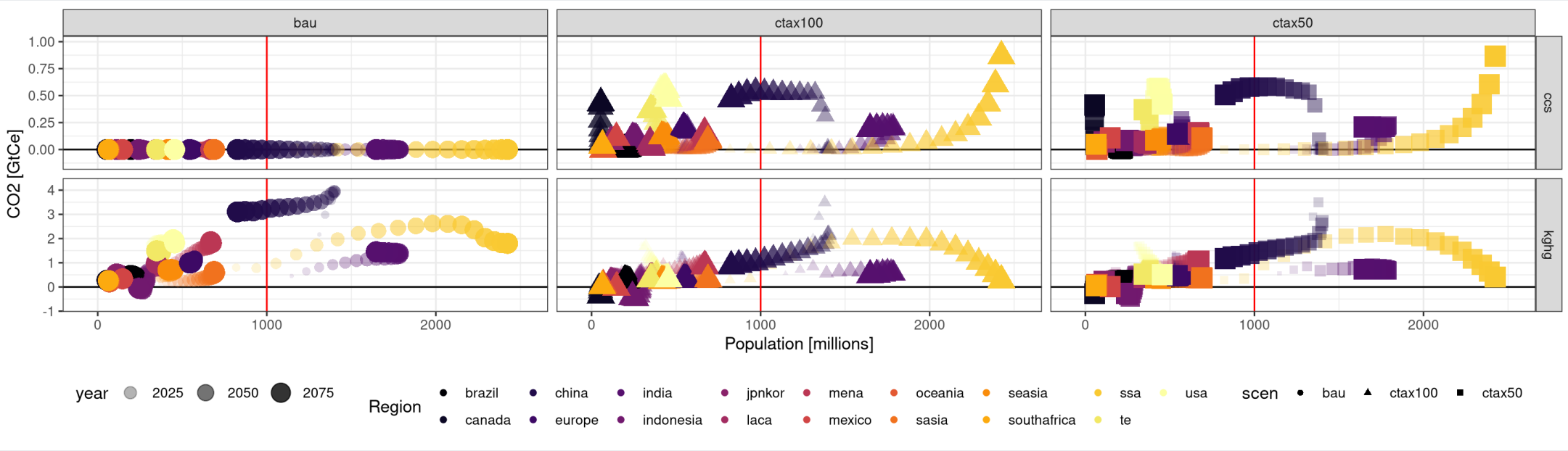
Scatter plots - and one more ... too many!

```
ggplot(dat ▷ filter(year < 2100),
       aes(x = value.pop,y = value.ghg, color = n, shape = scen, alpha = year, size = e)) + # add size with 'e'
  geom_hline(yintercept = 0) + geom_vline(xintercept = 1000, color = 'red') +
  geom_point() + theme_bw() +
  scale_x_continuous(name = "Population [millions]",expand = c(0.1, 0.1)) +
  scale_y_continuous(name = "CO2 [GtCe]",expand = c(0.1, 0.1))+
  scale_color_viridis(discrete = TRUE, option = "B", name = "Region") +
  guides(col = guide_legend(ncol = 9)) +
  theme(legend.position = "bottom")
```



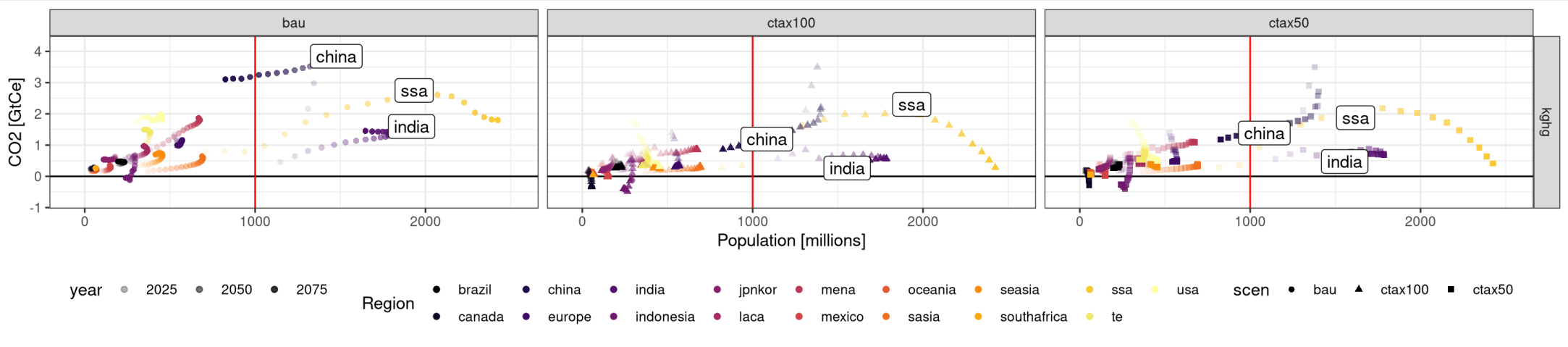
Scatter plots - Let's exagerate

```
ggplot(dat ▷ filter(year < 2100),
       aes(x = value.pop,y = value.ghg, color = n, shape = scen, alpha = year, size = year)) + # add size with 'e'
  geom_hline(yintercept = 0) + geom_vline(xintercept = 1000, color = 'red') +
  geom_point() + theme_bw() +
  scale_x_continuous(name = "Population [millions]",expand = c(0.1, 0.1)) +
  scale_y_continuous(name = "CO2 [GtCe]",expand = c(0.1, 0.1))+
  scale_color_viridis(discrete = TRUE, option = "B", name = "Region") +
  facet_grid(e~scen, scales='free') + # Add facets
  guides(col = guide_legend(ncol = 9)) +
  theme(legend.position = "bottom")
```



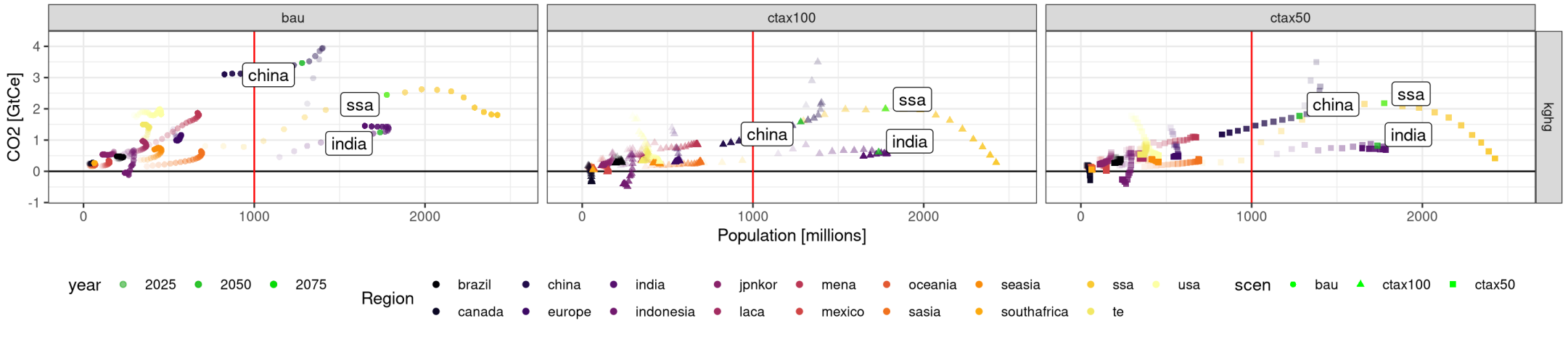
Scatter plots - help the reader

```
library(ggrepel)
ggplot(dat ▷ filter(year < 2100 & e == "kghg"),
       aes(x = value.pop,y = value.ghg, color = n, shape = scen, alpha = year)) +
  geom_hline(yintercept = 0) + geom_vline(xintercept = 1000, color = 'red') + geom_point() + theme_bw() +
  geom_label_repel(aes(label = n),
                   dat ▷ filter(year == 2050 & value.pop > 1000 & e == 'kghg'),
                   color = 'black', alpha = 1) + # Add labels
  scale_x_continuous(name = "Population [millions]",expand = c(0.1, 0.1)) +
  scale_y_continuous(name = "CO2 [GtCe]",expand = c(0.1, 0.1))+
  scale_color_viridis(discrete = TRUE, option = "B", name = "Region") +
  facet_grid(e~scen, scales='free') + guides(col = guide_legend(ncol = 9)) + theme(legend.position = "bottom")
```



Scatter plots - Draw attention to ...

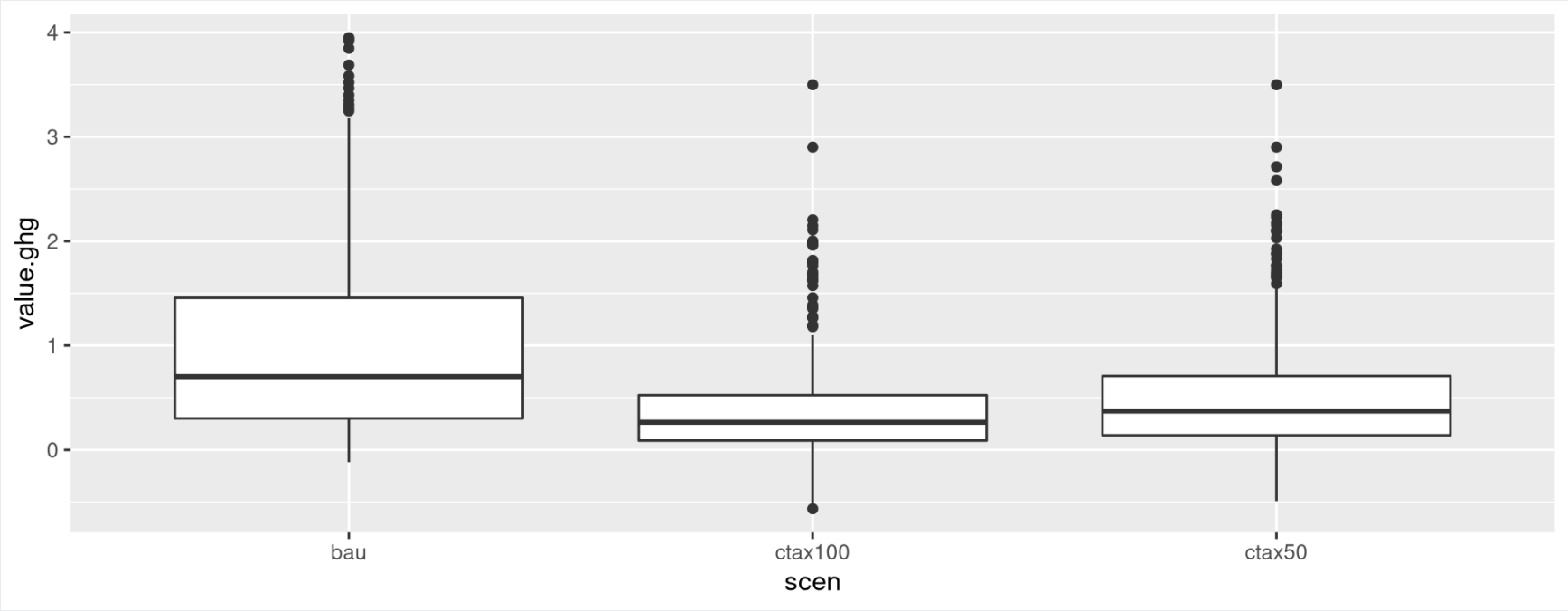
```
ggplot(dat ▷ filter(year < 2100 & e == "kghg"),
       aes(x = value.pop,y = value.ghg, color = n, shape = scen, alpha = year)) +
  geom_hline(yintercept = 0) + geom_vline(xintercept = 1000, color = 'red') + geom_point() + theme_bw() +
  geom_point(data = dat ▷ filter(year == 2050 & value.pop > 1000 & e == 'kghg'),
             color='green')+ # Emphasize a few points
  geom_label_repel(aes(label = n), dat ▷ filter(year == 2050 & value.pop > 1000 & e == 'kghg'),
                   color = 'black', alpha = 1) +
  scale_x_continuous(name = "Population [millions]",expand = c(0.1, 0.1)) +
  scale_y_continuous(name = "CO2 [GtCe]",expand = c(0.1, 0.1))+
  scale_color_viridis(discrete = TRUE, option = "B", name = "Region") +
  facet_grid(e~scen, scales='free') +
  guides(col = guide_legend(ncol = 9)) +
  theme(legend.position = "bottom")
```



Box plots - Plotting distributions

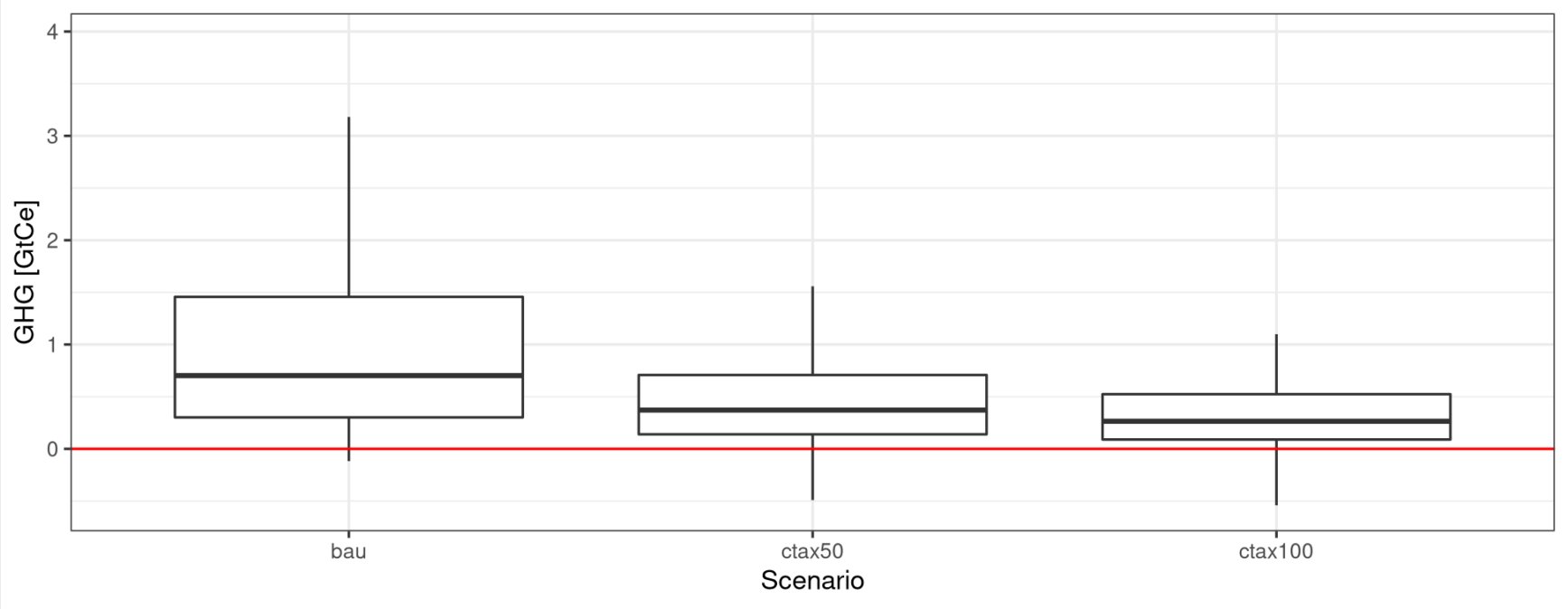
When mean or median are not enough

```
ggplot(dat ▷ filter(e=='kghg'),
       aes(x = scen, y = value.ghg)) +
  geom_boxplot()
```



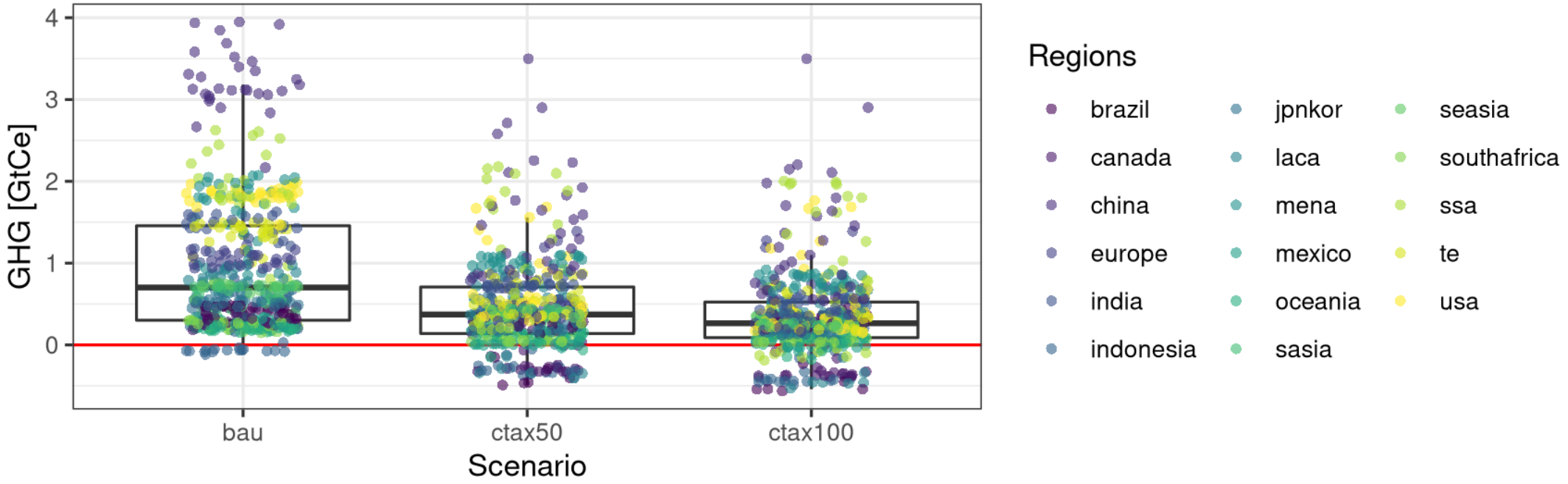
Box plots - typical retouch..

```r
# Update the scenario as factors (ordered set)
dat ← dat ▷
  mutate(scen = factor(scen, levels = c('bau','ctax50','ctax100')))

ggplot(dat ▷ filter(e == 'kghg'), aes(x = scen, y = value.ghg)) +
  geom_boxplot(outlier.shape = NA) + theme_bw() +
  geom_hline(yintercept = 0, color = "red") +
  labs(x = "Scenario", y = "GHG [GtCe]")
```



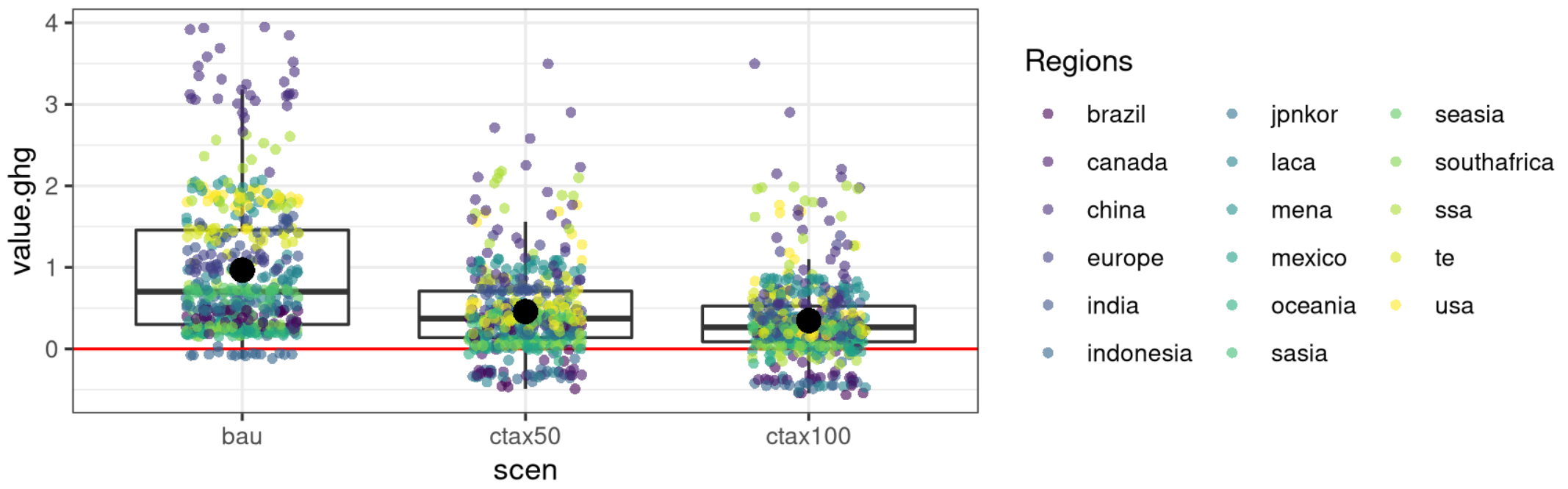
Box plots - add the underlying observations

```
ggplot(dat ▷ filter(e == 'kghg'), aes(x = scen, y = value.ghg)) +
  geom_boxplot(outlier.shape = NA) + theme_bw() +
  geom_hline(yintercept = 0, color = "red") +
  geom_jitter(aes(color = n), shape = 16,
              position = position_jitter(0.2), alpha = 0.6) + # Add observations
  labs(x = "Scenario", y = "GHG [GtCe]") +
  scale_color_viridis(name="Regions",discrete = TRUE, option = "D")+
  guides(col = guide_legend(ncol = 3))# color per region
```

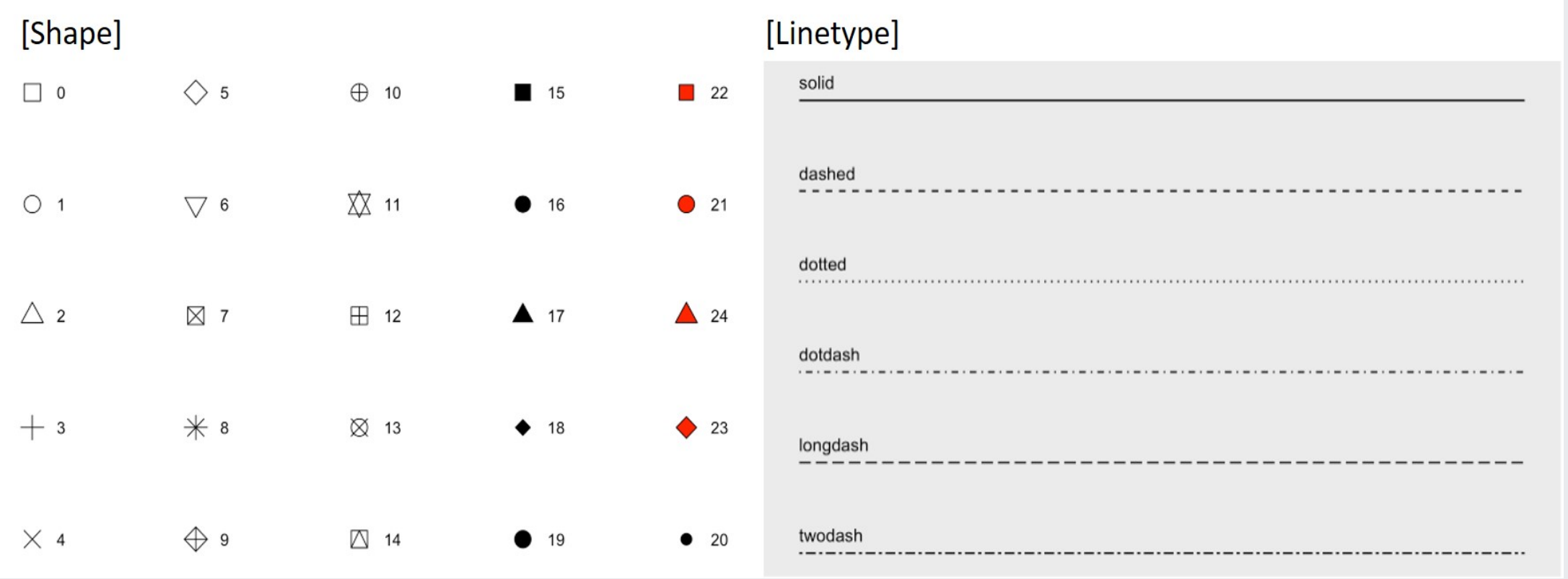


Box plots - add more infomation

```
ggplot(dat ▷ filter(e == 'kghg'), aes(x = scen, y = value.ghg)) +
  geom_boxplot(outlier.shape = NA) + theme_bw()+
  geom_boxplot(outlier.shape = NA) + theme_bw() +
  geom_hline(yintercept = 0, color = "red") +
  geom_jitter(aes(color = n), shape = 16, position = position_jitter(0.2), alpha = 0.6) +
  stat_summary(fun = mean, geom = "point", shape = 16, size = 4) + # Add mean distribution
  scale_color_viridis(name="Regions",discrete = TRUE, option = "D")+
  guides(col = guide_legend(ncol = 3))
```



Shapes and linetypes



Maps - Showing regional information

We need to prepare the data first

```
# import the WITCH model regional definition
wreg ← read_csv('Material/mapwitch17.csv')

# Country name tools
library(countrycode)

# transforms ISO codes into country names
wreg ← wreg ▷
  mutate(region = countrycode(ISO, origin = 'iso3c', destination = 'country.name'))

# Correcting some mismatches
#wreg[ISO=='GBR']$region = 'UK'
#wreg[ISO=='USA']$region = 'USA'
wreg
```

```
## # A tibble: 250 × 3
##    n           ISO   region
##    <chr>       <chr> <chr>
##  1 canada      CAN   Canada
##  2 canada      SPM   St. Pierre & Miquelon
##  3 jpnkor      JPN   Japan
##  4 jpnkor      KOR   South Korea
##  5 oceania     NZL   New Zealand
##  6 oceania     AUS   Australia
##  7 indonesia   IDN   Indonesia
##  8 southafrica ZAF   South Africa
##  9 brazil      BRA   Brazil
## 10 mexico      MEX   Mexico
```

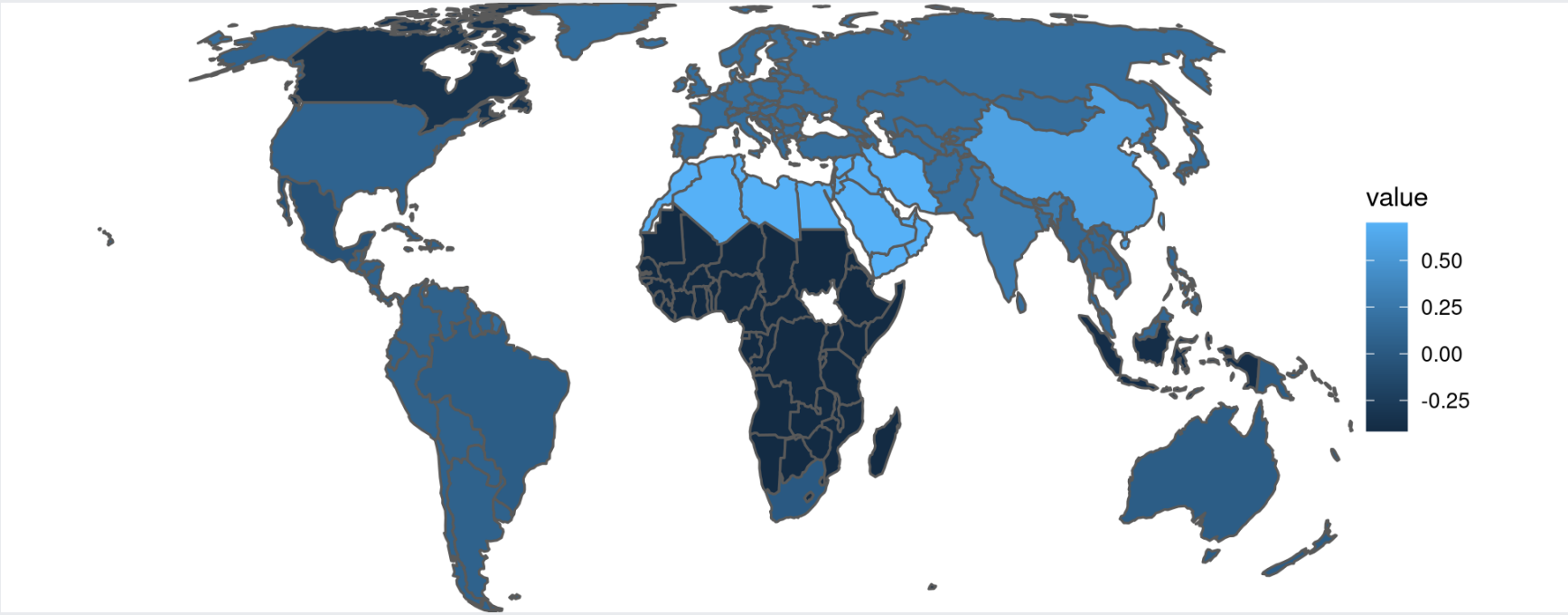
Maps - load world map

```
library(sf)
library(rnaturalearth)
library(rnaturalearthdata)

world ← ne_countries(scale = "small", returnclass = "sf")
world ← subset(world,!adm0_a3 %in% c("ATA","FJI"))

# merge the WITCH regional definition with the world map
world ← merge(world,wreg, by.x = "adm0_a3", by.y = "ISO")

# merge with the emission data.frame
world0 ← merge(world,qemi ▷ filter(year == 2100 & e == "co2"), by = "n", allow.cartesian=TRUE)

# Use a better projection 'equal projection'
target_crs ← '+proj=eqearth +wktext'
world1 ← st_transform(world0, crs = target_crs)
```

Ready to plot the map now!

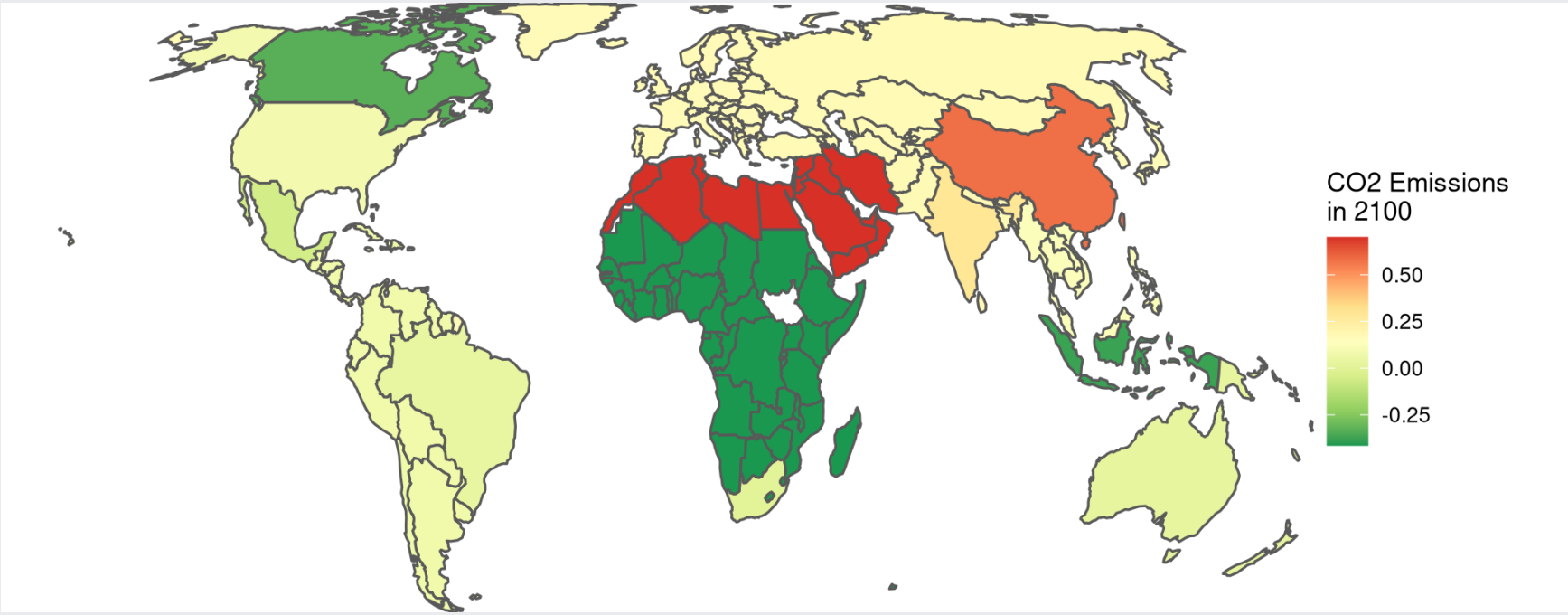
Maps - Showing regional information

```
ggplot(data = world1 |> filter(scen == "ctax50")) +
  geom_sf(aes(fill = value)) +
  coord_sf(datum = target_crs, expand = FALSE, clip = "off") +
  theme_void()
```



Maps - color palettes are important

```
ggplot(data = world1 ▷ filter(scen == "ctax50")) +
  geom_sf(aes(fill = value)) +
  coord_sf(datum = target_crs, expand = FALSE, clip = "off") +
  scale_fill_distiller(name = "CO2 Emissions\nin 2100", palette ="RdYlGn", direction=-1) +
  theme_void()
```



Maps - Facets are also possible

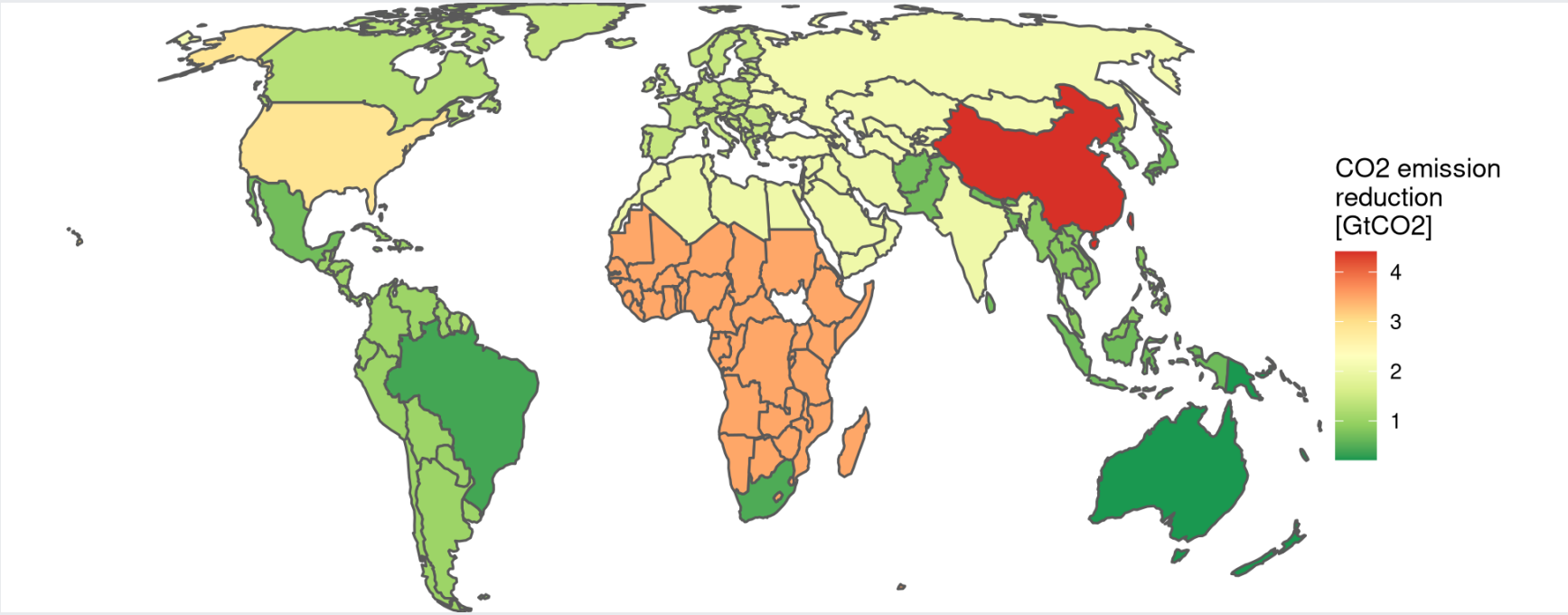
```
ggplot(data = world0) +
  geom_sf(aes(fill = value)) +
  coord_sf(datum = target_crs, expand = FALSE, clip = "off") +
  scale_fill_distiller(name = "CO2 Emissions\nin 2100", palette ="RdYlGn", direction=-1) +
  theme_void() +
  theme(legend.position = "bottom",
        strip.text.x = element_text(size = 12, face="bold"))+
  facet_grid(~scen)
```

Transforming variables for deeper analysis

```
# Put the scenarios values in columns
dqemi ← qemi ▷
  select(-t,-gdx) ▷
  filter(e == "co2" & year == 2100) ▷
  pivot_wider(names_from = "scen", values_from = "value")

# Let's create a new variable/column
dqemi ← dqemi ▷ mutate(dif_wrtBAU = bau - ctax100)

# merge with the emission data.frame
world2 ← merge(world,dqemi, by = "n", allow.cartesian=TRUE)

# Use a better projection 'equal projection'
target_crs ← '+proj=eqearth +wktext'
world3 ← st_transform(world2, crs = target_crs)
```

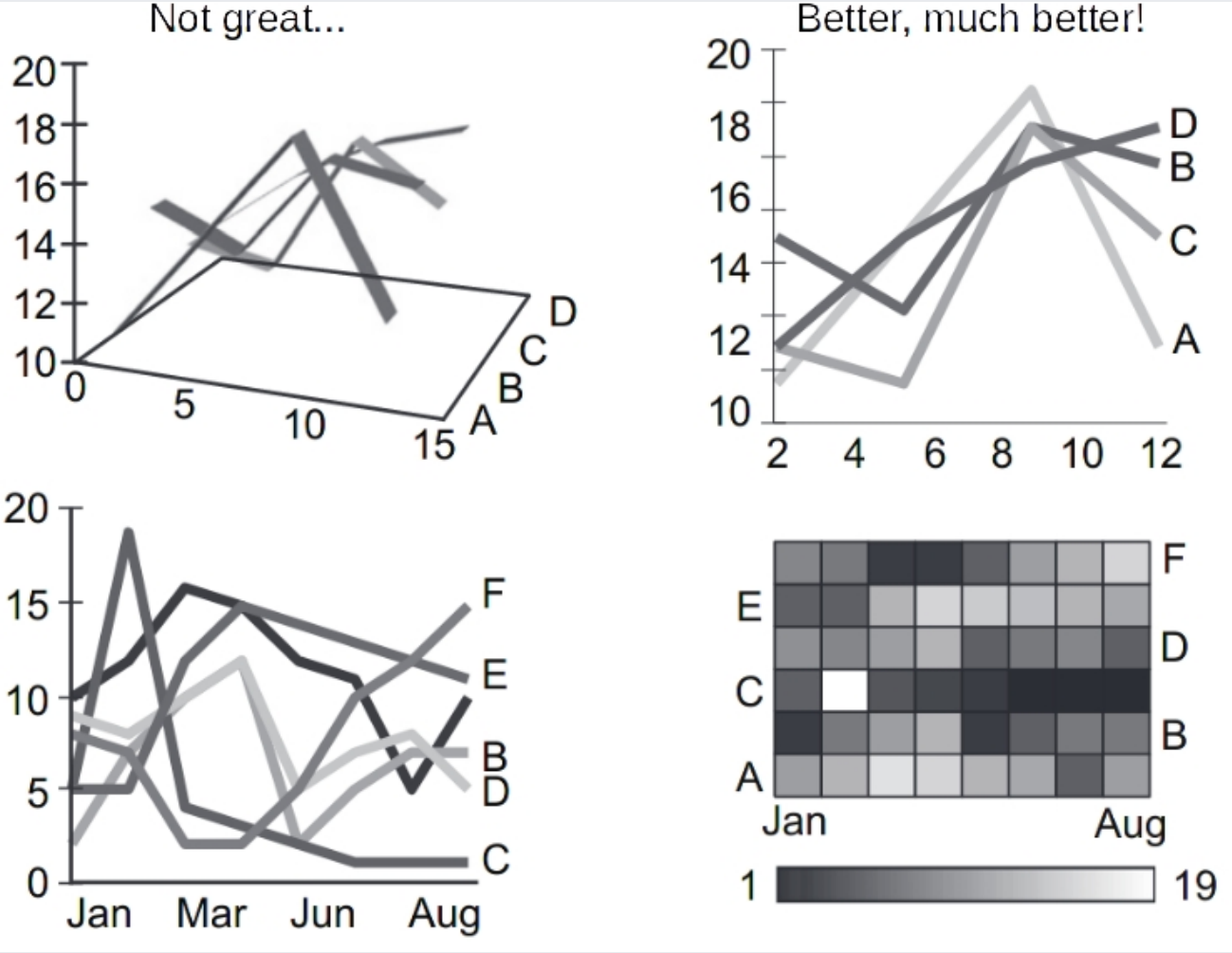
Transforming variables

```
ggplot(data = world3) +
  geom_sf(aes(fill = dif_wrtBAU * 44 / 22)) +
  coord_sf(datum = target_crs, expand = FALSE, clip = "off") +
  scale_fill_distiller(name = "CO2 emission\nreduction\n[GtCO2]",
                       palette ="RdYlGn", direction = -1) +
  theme_void()
```



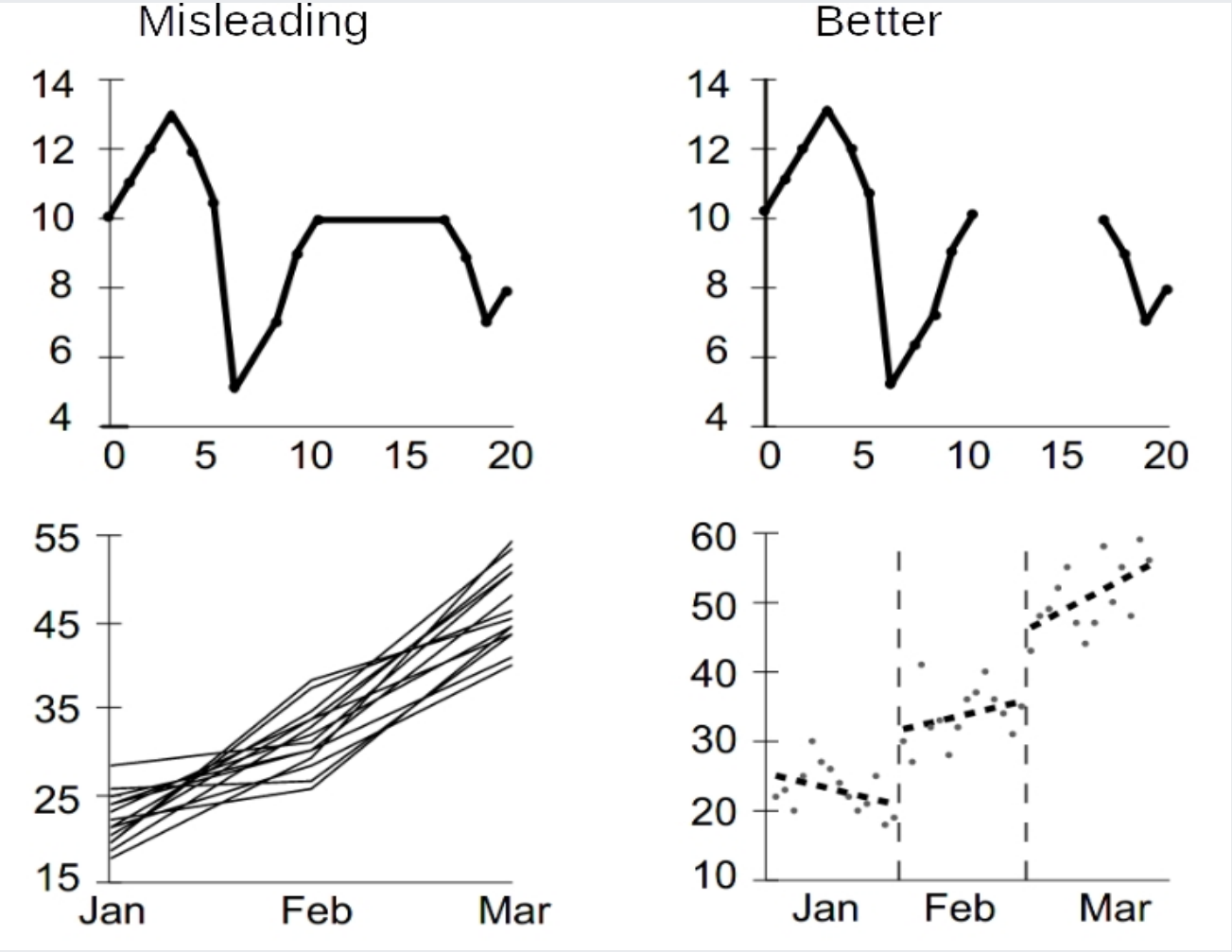
Tips for good visualization

Tips for good visualization



source: Kelleher et al. 2011

Tips for good visualization



source: Kelleher et al. 2011

Further Readings

GAMS

- GAMS documentation: https://www.gams.com/latest/docs

R

- RStudio Cheatsheets https://www.rstudio.com/resources/cheatsheets/

ggplot2

- ggplot2 book https://ggplot2-book.org/
- ggplot2 reference documentation https://ggplot2.tidyverse.org/reference/index.html
- ggplot2 extensions: https://exts.ggplot2.tidyverse.org/

Questions

Laurent Drouet | laurent.drouet@eiee.org | https://lolow.github.io

Lara Reis | lara.aleluia@eiee.org | https://laleluia.github.io/page