

GAMS-R Vizualization lecture

Lara Aleluia Reis

Laurent Drouet

2021-04-15

Presentation

Lara Aleluia Reis

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

Laurent Drouet (contributor)

- author of several R packages, in particular, `gdxtools` and a GAMS extension for the code editor `Visual Code Studio`.

Purpose of the lecture

You will learn how to interface GAMS and R

Through GDX file format

To visualize data sets for the diagnostic and the analysis of GAMS models.

Ultimate Goal: Learn how to convey your message without words

Outline

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

How to Start: Requirements

| Assumption: R, RStudio and R packages are already installed

1. Install R ([Here](#))
2. Install Rstudio ([Here](#))

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

```
'data.table'  
'ggplot2'  
'tidyverse'  
'stringr'  
'gdxtools'* (how to install (https://github.com/lolow/gdxtools) for the brave ;) )
```

Open RStudio and paste the following command in the console:

```
install.packages('data.table','ggplot2','tidyverse','stringr')
```

The GAMS data exchange files

The GAMS data exchange files

The GAMS data exchange format (in short GDX) is a proprietary file format to store all elements of your GAMS programs.

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

A lot of information is stored in the GDX file. For example, for the variables, you can obtain, their values (`level`) but also their bounds (`lower,upper`), their marginal value (`marginal`) and scaling factors (`scale`).

Browsing a GDX in GAMS studio

GAMS Studio Mon Apr 12 12:08

GAMS Studio

File Edit GAMS MIRO Tools View Help

dice_2c.gdx

All Columns

Entry	Name	Type	Dim	Records	Text
14	a0	Parameter	0	1	Initial level of total factor productivity
48	a1	Parameter	0	1	Damage intercept
46	a10	Parameter	0	1	Initial damage intercept
49	a2	Parameter	0	1	Damage quadratic term
47	a20	Parameter	0	1	Initial damage quadratic term
50	a3	Parameter	0	1	Damage exponent
105	ABATECOST	Variable	1	100	Cost of emissions reductions (trillions 2005 USD per year)
120	ABATEEQ	Equation	1	100	Cost of emissions reductions equation
65	a1	Parameter	1	100	Level of total factor productivity
83	atfrac	Parameter	1	0	Atmospheric share since 1850
84	atfrac2010	Parameter	1	0	Atmospheric share since 2010
31	b11	Parameter	0	1	Carbon cycle transition matrix
29	b12	Parameter	0	1	Carbon cycle transition matrix
32	b21	Parameter	0	1	Carbon cycle transition matrix
33	b22	Parameter	0	1	Carbon cycle transition matrix
30	b23	Parameter	0	1	Carbon cycle transition matrix
34	b32	Parameter	0	1	Carbon cycle transition matrix
35	b33	Parameter	0	1	Carbon cycle transition matrix
94	C	Variable	1	100	Consumption (trillions 2005 US dollars per year)
42	c1	Parameter	0	1	Climate equation coefficient for upper level

Table View

t	Level	Marginal	Lower	Upper	Scale
1	0.000846957	0	-INF	+INF	1
2	0.000208633	0	-INF	+INF	1
3	0.00025839	0	-INF	+INF	1
4	0.000317974	0	-INF	+INF	1
5	0.000389075	0	-INF	+INF	1
6	0.000473656	0	-INF	+INF	1
7	0.000573999	0	-INF	+INF	1
8	0.000692753	0	-INF	+INF	1
9	0.000832992	0	-INF	+INF	1
10	0.000998285	0	-INF	+INF	1
11	0.00119277	0	-INF	+INF	1
12	0.00142122	0	-INF	+INF	1
13	0.0016892	0	-INF	+INF	1
14	0.00200311	0	-INF	+INF	1
15	0.00237038	0	-INF	+INF	1
16	0.00279958	0	-INF	+INF	1
17	0.0033006	0	-INF	+INF	1
18	0.00388488	0	-INF	+INF	1
19	0.00456561	0	-INF	+INF	1
20	0.00535799	0	-INF	+INF	1

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. In that case, this step is not required.

gdxtools R Package

gdxttools

gdxttools 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

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

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

You might also 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

```
library(gdxtools)
```

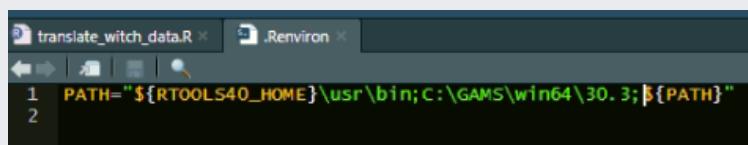
First, `gdxtools` will try to find your GAMS installation. If it does not find it, add your GAMS directory in the PATH environment variable and tell the location to the library with:

```
igdx(dirname(Sys.which('gams')))
```

Usually, when you install GAMS on Windows, it adds automatically the GAMS location to your PATH. Otherwise, follow the instructions from <https://helpdeskgeek.com/windows-10/add-windows-path-environment-variable>

or, alternatively, add the PATH in your `.Renvironment` file with this command

```
usethis::edit_r_environ()
```



A screenshot of the RStudio interface showing the .Renvironment file open. The code in the editor is:

```
1 PATH = "${RTOOLS40_HOME}\usr\bin;c:\GAMS\win64\30.3;${PATH}"
```

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(file.path('Material', 'results_ssp2_bau.gdx'))
```

mygdx can provide information on the GDX content.

```
print(mygdx)  
  
## <gdx: Material/results_ssp2_bau.gdx, 2048 symbols>
```

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

Usage

List of variables

```
print(mygdx$variables$name)

## [1] "UTILITY"           "PROB"          "I"
## [4] "K"                 "Q"             "BAU_Q"
## [7] "COST_Y"            "COST_EN"        "COST_FUEL"
## [10] "FPRICE"            "I_EN"           "K_EN"
## [13] "MCOST_FUEL"        "MCOST_INV"      "QEL_OUT"
## [16] "QNEL_OUT"          "Q_EN"           "Q_FUEL"
## [19] "Q_IN"              "BAU_Q_EMI"      "COST_EMI"
## [22] "CPRICE"            "Q_EMI"          "I_RD"
## [25] "K_RD"              "SPILL"          "Q_EL_FLEX"
## [28] "K_EN_GRID"         "I_EN_GRID"      "CUM_Q_CCS"
## [31] "MCOST_EMI"          "Q_CCS"          "I_EN_WINDOFF"
## [34] "I_EN_WINDON"        "K_EN_WINDOFF"    "K_EN_WINDON"
## [37] "MCOST_INV_WINDOFF" "Q_EN_WINDOFF"    "Q_EN_WINDON"
## [40] "I_EN_PV"            "I_EN_CSP"        "K_EN_PV"
## [43] "K_EN_CSP"           "Q_EN_PV"         "Q_EN_CSP"
## [46] "ADDOILCAP"          "COST_OIL"        "CUM_OIL"
## [49] "I_OIL"              "I_OUT"          "OILCAP"
```

Usage

List of parameters

```
print(mygdx$parameters$name)

## [1] "nb_clt_infes"                      "nb_clt_noopt"
## [3] "nb_tot_infes"                       "allerr"
## [5] "allinfoiter"                        "delta_price"
## [7] "errtrade"                           "infoiter"
## [9] "m_consumption"                     "mbalance"
## [11] "mprofit"                            "price_iter"
## [13] "solrep"                             "timer"
## [15] "trust_region"                      "errtol"
## [17] "stop_nash"                          "stop_run"
## [19] "tlen"                               "tperiod"
## [21] "year"                               "yeoh"
## [23] "eta"                                "gamma"
## [25] "srtp"                               "stpf"
## [27] "utility_cebge_global"               "utility_cebge_pc_global"
## [29] "utility_cebge_pcRegional"           "utility_cebge_global"
## [31] "w_negishi"                          "utility_cebge_pc_global"
## [33] "delta"                              "utility_cebge_pc_global"
## [35] "delta0"                             "utility_cebge_pc_global"
```

Usage

List of sets

```
print(mygdx$sets$name)

## [1] "all_feasible"
## [2] "all_optimal"
## [3] "irep"
## [4] "iterrep"
## [5] "tcheck"
## [6] "ierr"
## [7] "siter"
## [8] "conf"
## [9] "t"
## [10] "tfix"
## [11] "tfirst"
## [12] "tlast"
## [13] "pre"
## [14] "n"
## [15] "oecd"
## [16] "non_oecd"
## [17] "clt"
```

Usage

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

```
mygdx[["n"]][[1]]  
  
## [1] "brazil"      "canada"       "china"        "europe"       "india"  
## [6] "indonesia"   "jpnkor"       "laca"         "mena"         "mexico"  
## [11] "oceania"     "sasia"        "seasia"       "southafrica"  "ssa"  
## [16] "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:

```
head(mygdx["wemi"])
```

	e	t	value
## 1:	co2	1	8.555397e+00
## 2:	co2	2	9.312155e+00
## 3:	co2	3	1.008733e+01
## 4:	co2	4	1.074177e+01
## 5:	co2	5	1.142443e+01
## ---			
## 433:	sf6	26	6.467846e-06
## 434:	sf6	27	6.467689e-06
## 435:	sf6	28	6.467630e-06
## 436:	sf6	29	6.467609e-06
## 437:	sf6	30	6.467601e-06

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"]  
  
##          e   t      n      value  
## 1:      nip  1    brazil 0.000000e+00  
## 2:      nip  1    canada 0.000000e+00  
## 3:      nip  1     china 0.000000e+00  
## 4:      nip  1   europe 0.000000e+00  
## 5:      nip  1    india 0.000000e+00  
## ---  
## 16826: ccs_plant 30    seasia 7.000000e-08  
## 16827: ccs_plant 30 southafrica 7.000000e-08  
## 16828: ccs_plant 30        ssa 4.538060e-06  
## 16829: ccs_plant 30        te 9.142503e-07  
## 16830: ccs_plant 30        usa 6.323520e-06
```

Usage

To load the variable's upper bound:

```
mygdx["Q_EMI",field = "up"]  
  
##          e   t      n value  
## 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  
## ---  
## 16826: ccs_plant 30    seasia Inf  
## 16827: ccs_plant 30 southafrica Inf  
## 16828: ccs_plant 30      ssa Inf  
## 16829: ccs_plant 30      te Inf  
## 16830: ccs_plant 30      usa Inf
```

Usage

To load several gdx 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
```

```
##          e   t       n      value                         gdx
## 1:      nip  1    brazil 0.00000000 Material/results_ssp2_bau.gdx
## 2:      nip  1    canada 0.00000000 Material/results_ssp2_bau.gdx
## 3:      nip  1     china 0.00000000 Material/results_ssp2_bau.gdx
## 4:      nip  1   europe 0.00000000 Material/results_ssp2_bau.gdx
## 5:      nip  1    india 0.00000000 Material/results_ssp2_bau.gdx
## ---
## 33707: ccs_plant 30    seasia 0.09963209 Material/results_ssp2_ctax50.gdx
## 33708: ccs_plant 30 southafrica 0.02860252 Material/results_ssp2_ctax50.gdx
## 33709: ccs_plant 30        ssa 1.07402186 Material/results_ssp2_ctax50.gdx
## 33710: ccs_plant 30         te 0.32548768 Material/results_ssp2_ctax50.gdx
## 33711: ccs_plant 30        usa 0.40049922 Material/results_ssp2_ctax50.gdx
```

R basics

Getting started

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

In RStudio, paste this code in the console

```
library('data.table')
library('ggplot2')
library('tidyverse')
library('stringr')
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`)

Import data into R --- From GDX

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

```
#define a gdx  
dat_2c <- gdx('Material/dice_2c.gdx')  
  
#have a look at the data structure  
str(dat_2c)
```

```
## List of 7  
## $ filename : chr "Material/dice_2c.gdx"  
## $ sets     :'data.frame':   6 obs. of  7 variables:  
##   ..$ name   : chr [1:6] "t" "tfirst" "tlast" "tearly" ...  
##   ..$ index  : int [1:6] 1 60 61 62 63 139  
##   ..$ dim    : int [1:6] 1 1 1 1 1 1  
##   ..$ card   : int [1:6] 100 1 1 0 0 10  
##   ..$ text   : chr [1:6] "Time periods (5 years per period)" "" "" "" ...  
##   ..$ doms   :List of 6  
##     .. .$. : int 0  
##     .. .$. : int 1  
##     .. .$. : int 1
```

Import data into R

Import data from a CSV file:

```
dat_2c <- fread('Material/Dice2016R-091916ap.csv', header=TRUE, skip=6)
dat_2c
```

```
##                                     Year      2015      2020      2025
## 1: Industrial Emissions GTCO2 per year 35.7447137 39.38356371 42.93148531
## 2: Atmospheric concentration C (ppm) 399.5305164 418.46842938 438.31119767
## 3:          Atmospheric Temperature 0.8500000 1.01634518 1.18923663
## 4:          Output Net Net) 104.9972370 125.04045324 147.27653872
## 5: Climate Damages fraction output 0.0017051 0.00243778 0.00333771
## 6:          Consumption Per Capita 10.5017772 11.84233079 13.30423855
##                                     2030      2035      2040      2045      2050
## 1: 4.634970e+01 4.960597e+01 5.267407e+01 5.553314e+01 58.16705630
## 2: 4.591487e+02 4.810062e+02 5.038643e+02 5.276727e+02 552.36004083
## 3: 1.367930e+00 1.551747e+00 1.740021e+00 1.932077e+00 2.12722733
## 4: 1.717445e+02 1.984766e+02 2.274989e+02 2.588316e+02 292.48981054
## 5: 4.416108e-03 5.682685e-03 7.145307e-03 8.809699e-03 0.01067923
## 6: 1.489188e+01 1.660952e+01 1.846120e+01 2.045064e+01 22.58115071
##                                     2055      2060      2065      2070      2075
## 1: 60.56381778 62.71493656 64.61491429 66.26076014 67.65156760
## 2: 577.84087254 604.02077690 630.79984568 658.07525038 685.74306680
## 3: 2.32477390 2.52402262 2.72429207 2.92492408 3.12529213
```

The basics of data handling

```
#dimensions  
dim(dat_2c)  
  
## [1] 37 101  
  
#column names  
names(dat_2c[,1:10])  
  
## [1] "Year" "2015" "2020" "2025" "2030" "2035" "2040" "2045" "2050" "2055"  
  
#reformat column names  
names(dat_2c)[2:101] = as.character(as.numeric(names(dat_2c)[2:101]))  
  
#have a look  
names(dat_2c[,1:10])  
  
## [1] "Year" "2015" "2020" "2025" "2030" "2035" "2040" "2045" "2050" "2055"
```

The basics of data handling

```
#change name of column
setnames(dat_2c, 'Year', 'Variable')

#Let's make a long table we will need it for plotting
mdat_2c = melt(dat_2c, id.vars = c('Variable'), variable.name='Years')

#or
#mdat_2c=melt(dat_2c, measure.vars = names(dat_2c)[2:101]), variable.name = 'Years'

#let's have a look at the result
str(mdat_2c)

## Classes 'data.table' and 'data.frame':    3700 obs. of  3 variables:
## $ Variable: chr  "Industrial Emissions GTCO2 per year" "Atmospheric concentration C (ppm)" "Atmospheric
## $ Years    : Factor w/ 100 levels "2015","2020",...: 1 1 1 1 1 1 1 1 1 ...
## $ value    : num  3.57e+01 4.00e+02 8.50e-01 1.05e+02 1.71e-03 ...
## - attr(*, ".internal.selfref")=<externalptr>
```

Useful data handling functions

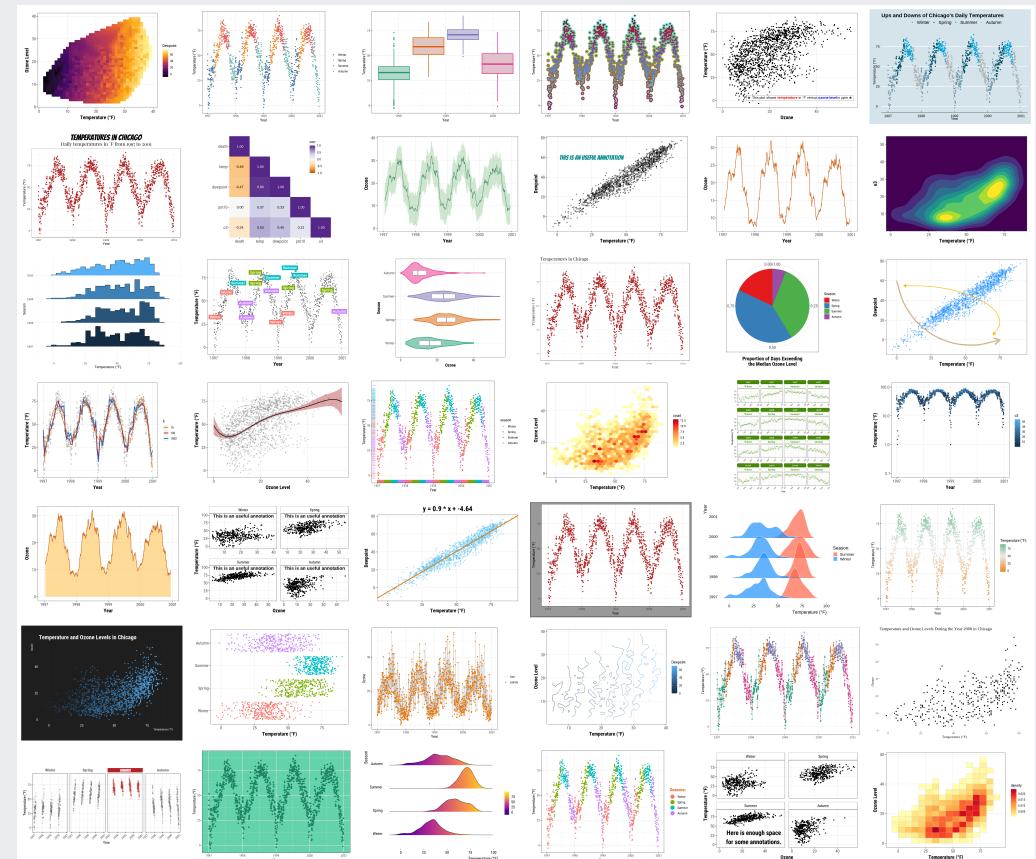
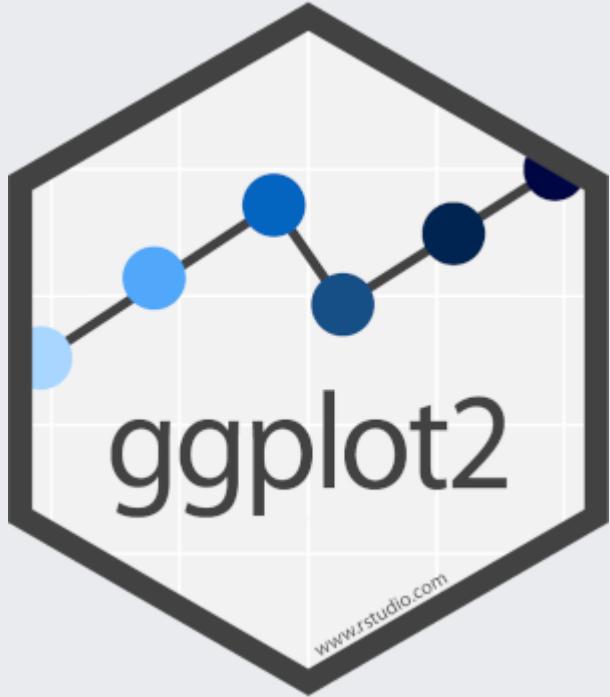
Function	data.table	tidyverse
From columns to row	<code>melt()</code>	<code>pivot_longer()</code>
From row to columns	<code>dcast() or spread()</code>	<code>pivot_longer()</code>
merge 2 data.tables	<code>merge()</code>	<code>full_join()</code>

In this course we will use `data.table` functions, but you can use also functions from the `tidyverse` package. For tips in `data.table` functions: <https://raw.githubusercontent.com/rstudio/cheatsheets/master/datatable.pdf>

| There are several ways of obtaining the same results.

Data visualization

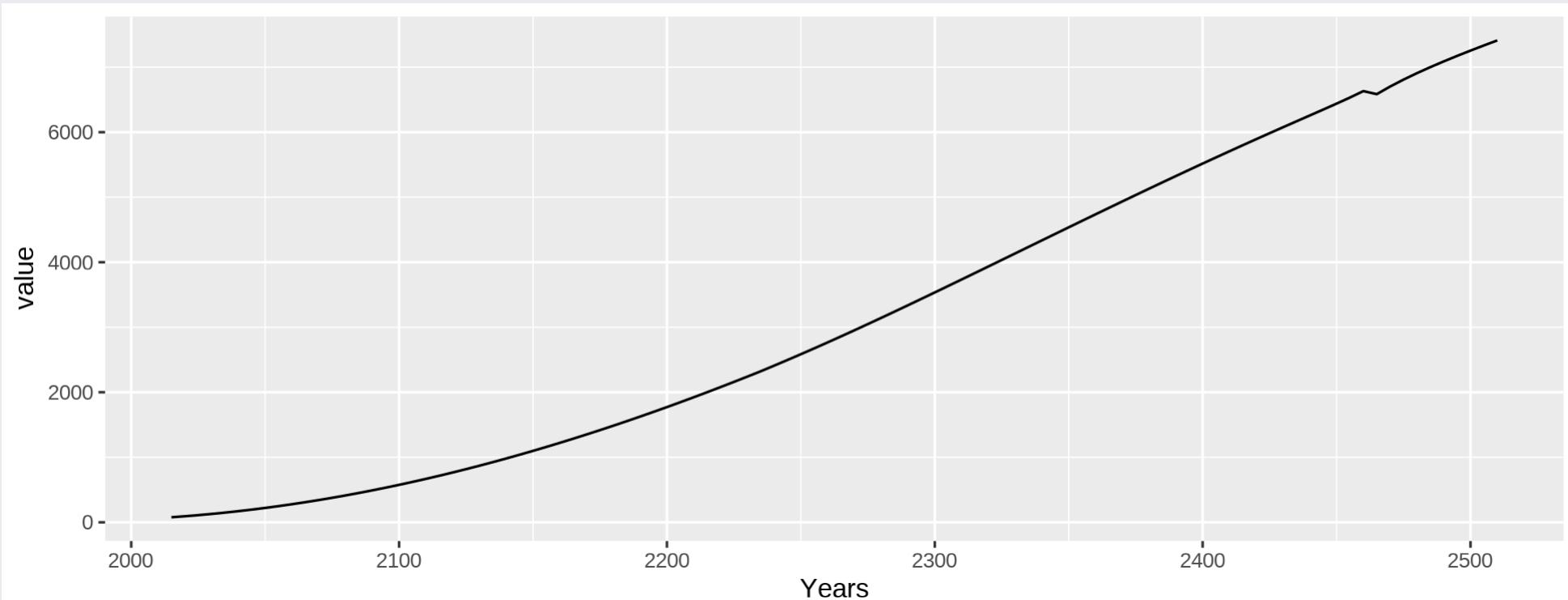
The ggplot2 package



ggplot2 cheatsheets: ([Here](#))

Let's make our first plot (time series)

```
#For time series we need numerical years  
mdat_2c$Years <- as.numeric(as.character(mdat_2c$Years))  
#let's use data.table options to subset our data  
ggplot(data = mdat_2c[Variable == 'Consumption'], aes(x = Years, y = value)) +  
geom_line()
```



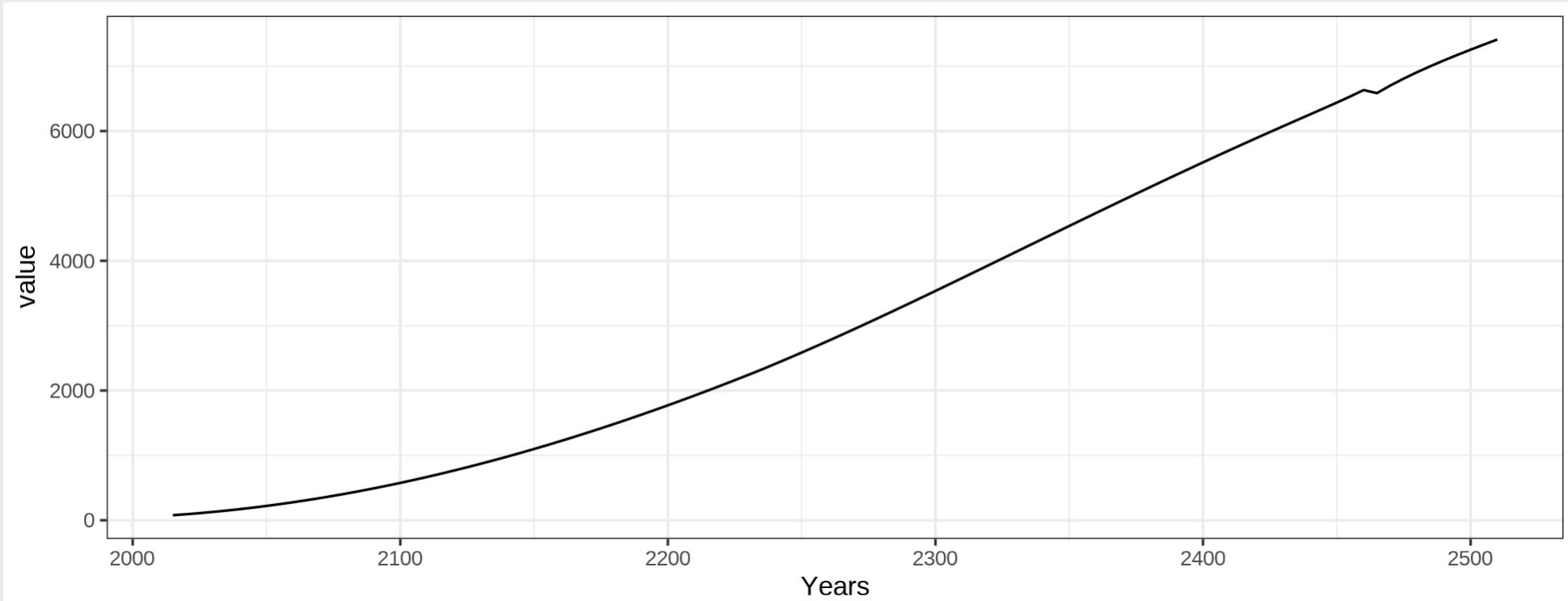
Let's make our first plot (time series)

Take time to design and format your figure

I will not be okay with default formats..
I will not be okay with default formats..

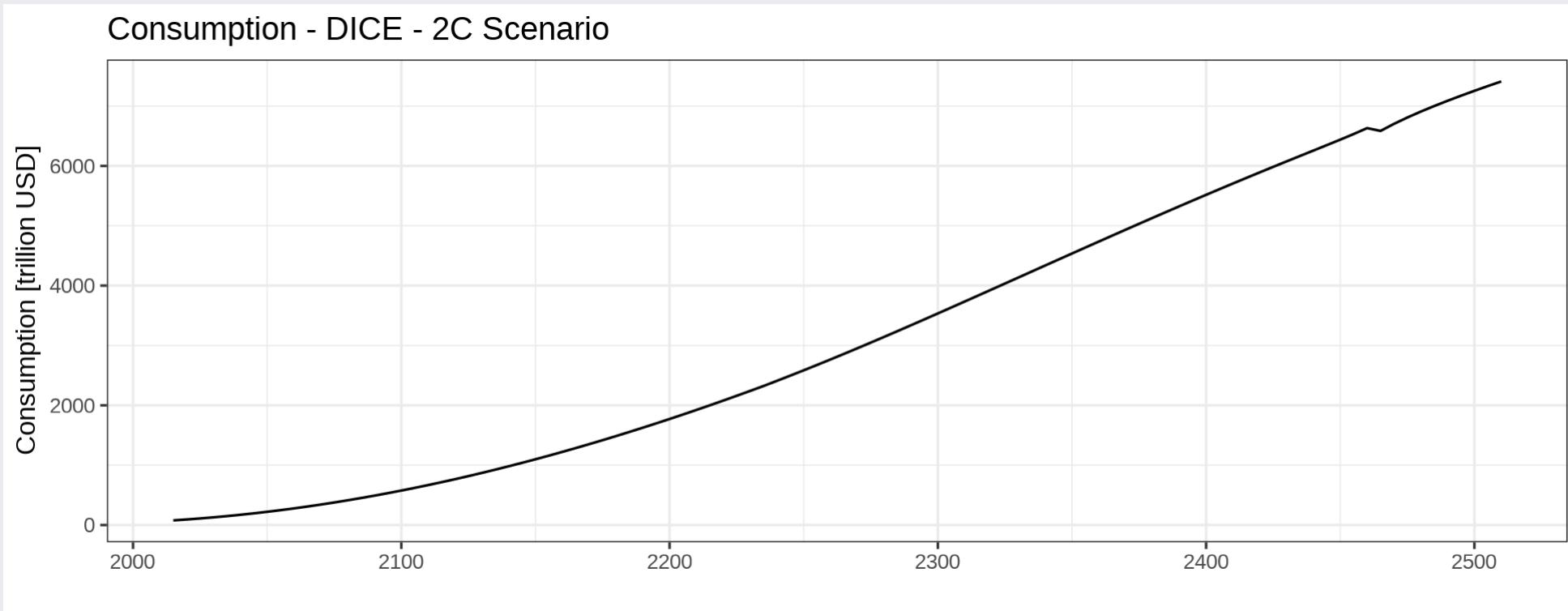
Change the theme

```
ggplot(data = mdat_2c[Variable == 'Consumption'], aes(x = Years, y = value)) +  
  geom_line() +  
  theme_bw() # change theme to a more sober and less noisy background
```



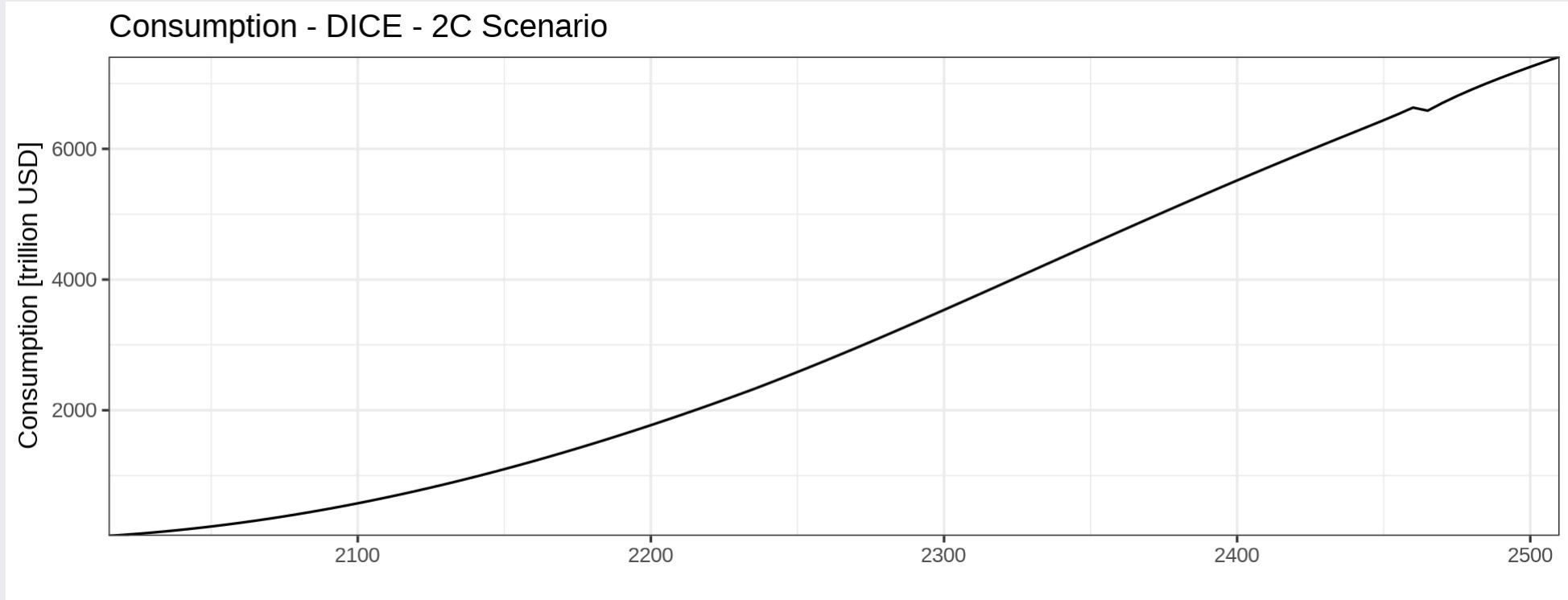
Add a title and labels

```
ggplot(data = mdat_2c[Variable == 'Consumption'], aes(x = Years, y = value)) +  
  geom_line() +  
  labs(title = "Consumption - DICE - 2C Scenario", x = "", y = "Consumption [trillion USD]") +  
  theme_bw()
```



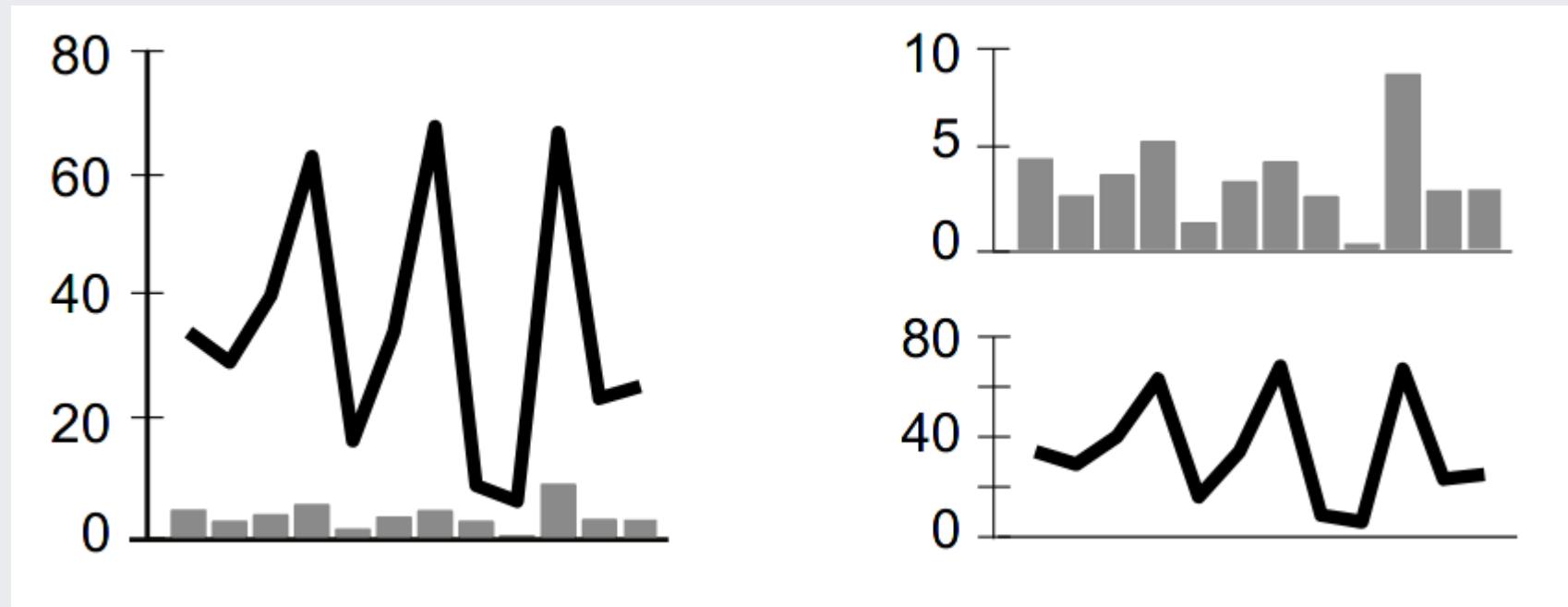
Scale the data

```
ggplot(data=mdat_2c[Variable == 'Consumption'], aes(x=Years,y=value ))+  
  geom_line() +  
  labs(title = "Consumption - DICE - 2C Scenario", x = "", y = "Consumption [trillion USD]") +  
  theme_bw() # change theme to a more sober and less noisy background  
  scale_y_continuous(expand = c(0, 0)) +  
  scale_x_continuous(expand = c(0,0))
```



Let's add more variables

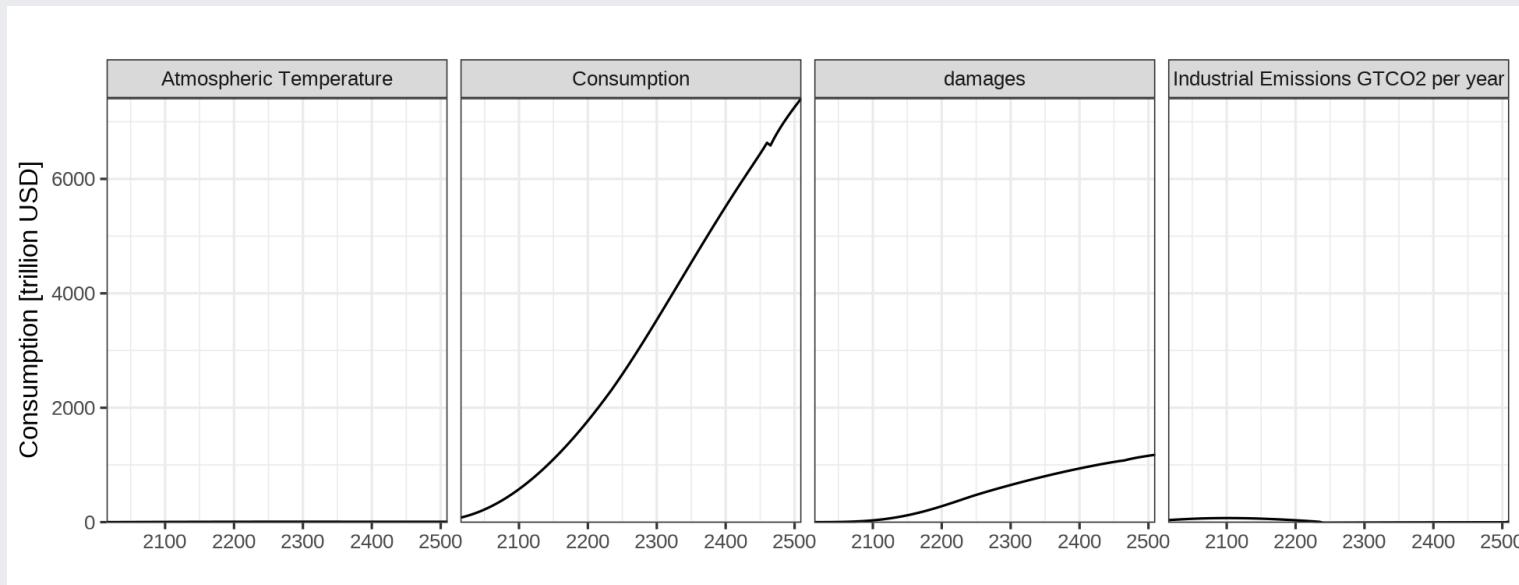
Visualization tip: Do not use different parameters in the same graph! Separate variables with large scale differences into subplots!



source: Kelleher *et al.* (2011)

Let's add more variables

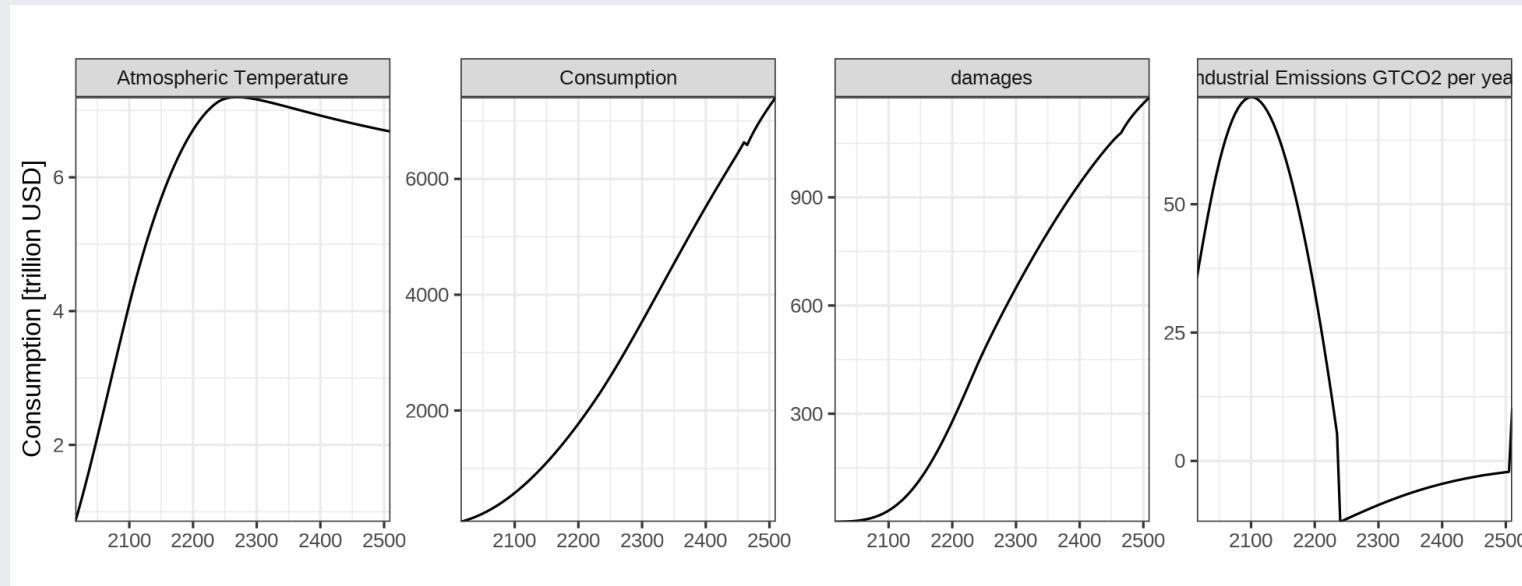
```
vars = c('Consumption', 'Atmospheric Temperature',
        'Industrial Emissions GTCO2 per year',
        'damages')
ggplot(data = mdat_2c[Variable %in% vars], aes(x = Years, y = value)) +
  geom_line() + xlab("") + ylab('Consumption [trillion USD]') + # add axis labels
  scale_y_continuous(expand = c(0, 0)) + # start at 0,0
  scale_x_continuous(expand = c(0,0)) +
  facet_wrap(~Variable, ncol = 4) +
  ggttitle('') + # change add title
  theme_bw() # change theme to a more sober and less noisy background
```



Let's add more variables --- Scale is important

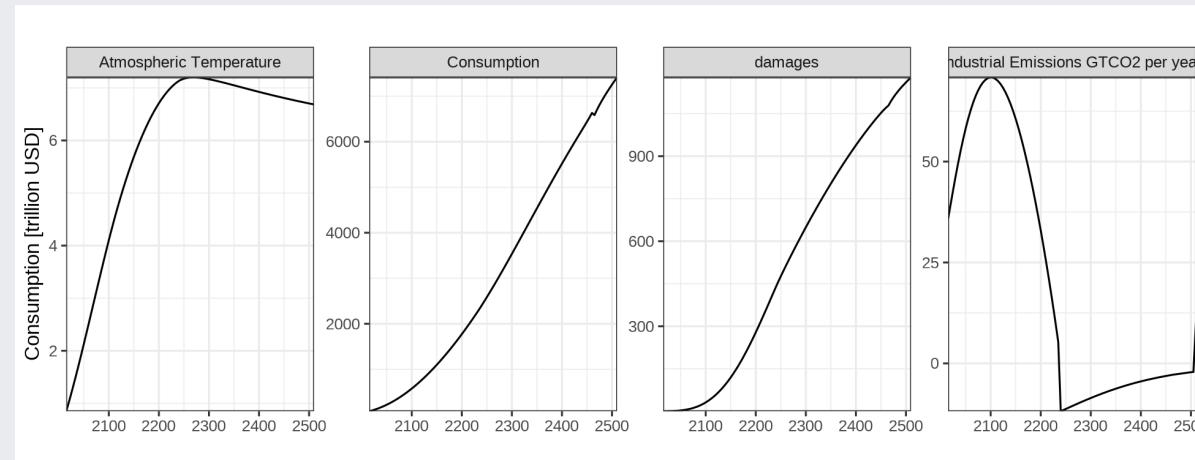
What do you want to show? Magnitude or relative magnitude? Select meaningful axis ranges depending on your purpose!

```
ggplot(data = mdat_2c[Variable %in% vars], aes(x = Years, y = value)) +  
  geom_line() + xlab("") + ylab('Consumption [trillion USD]') +  
  scale_y_continuous(expand = c(0, 0)) +  
  scale_x_continuous(expand = c(0,0)) +  
  facet_wrap(~Variable, ncol = 4, scales='free') + # <- free scales  
  ggtitle('') +  
  theme_bw()
```



Save your graph

```
p <- ggplot(data = mdat_2c[Variable %in% vars], aes(x = Years, y = value)) +  
  geom_line() + xlab("") + ylab('Consumption [trillion USD]') +  
  scale_y_continuous(expand = c(0, 0)) +  
  scale_x_continuous(expand = c(0,0)) +  
  facet_wrap(~Variable, ncol = 4, scales='free') + # <- free scales  
  ggtitle('') +  
  theme_bw()  
  
p
```



```
ggsave("Fig1.png",p)  
ggsave("Fig1.pdf", width = 60, height = 6, units = "cm")
```

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"))
GHG = batch_extract("Q_EMI", files = myfiles) # load GHG emissions
# have a look at the structure
str(GHG) # 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/re
##   ..- attr(*, "gams")= chr ""

# this grabs the data.frame inside the list GHG$Q_EMI
# let's transform it into a data.table for easiness
GHG = setDT(GHG$Q_EMI)
#same as GHG = setDT(GHG[[1]]) or setDT(GHG[['Q_EMI']])
```

2D graphs - lines plots

First, read and organize your data

```
#have a look now
str(GHG)

## Classes 'data.table' and 'data.frame':    50592 obs. of  5 variables:
## $ e    : chr  "nip" "nip" "nip" "nip" ...
## $ t    : chr  "1" "1" "1" "1" ...
## $ n    : chr  "brazil" "canada" "china" "europe" ...
## $ value: num  0 0 0 0 0 0 0 0 0 0 ...
## $ gdx  : chr  "Material/results_ssp2_bau.gdx" "Material/results_ssp2_bau.gdx" "Material/results_ssp2_bau.gdx" ...
## - attr(*, "gams")= chr  ""
## - attr(*, ".internal.selfref")=<externalptr>

#let's transform time periods into Years
GHG$t = as.numeric(GHG$t) * 5 + 2000
#let's transform the file names into meaningful scenario names
#let's use functions of the package stringr
GHG$gdx = str_remove(GHG$gdx, 'Material/results_ssp2_')
GHG$gdx = str_remove(GHG$gdx, '.gdx')
```

2D graphs - lines plots

```
# we can also change the names directly as we wish
# first let's transform into it factors
GHG$gdx = as.factor(GHG$gdx)
levels(GHG$gdx) = c('REF','Ctax100','Ctax50')
# now let's change a columns name
setnames(GHG,'gdx','Scenario')
# let's have a look again
str(GHG)

## Classes 'data.table' and 'data.frame': 50592 obs. of 5 variables:
## $ e      : chr "nip" "nip" "nip" "nip" ...
## $ t      : num 2005 2005 2005 2005 2005 ...
## $ n      : chr "brazil" "canada" "china" "europe" ...
## $ value  : num 0 0 0 0 0 0 0 0 0 0 ...
## $ Scenario: Factor w/ 3 levels "REF","Ctax100",...: 1 1 1 1 1 1 1 1 1 1 ...  
## - attr(*, "gams")= chr ""
## - attr(*, ".internal.selfref")=<externalptr>
```

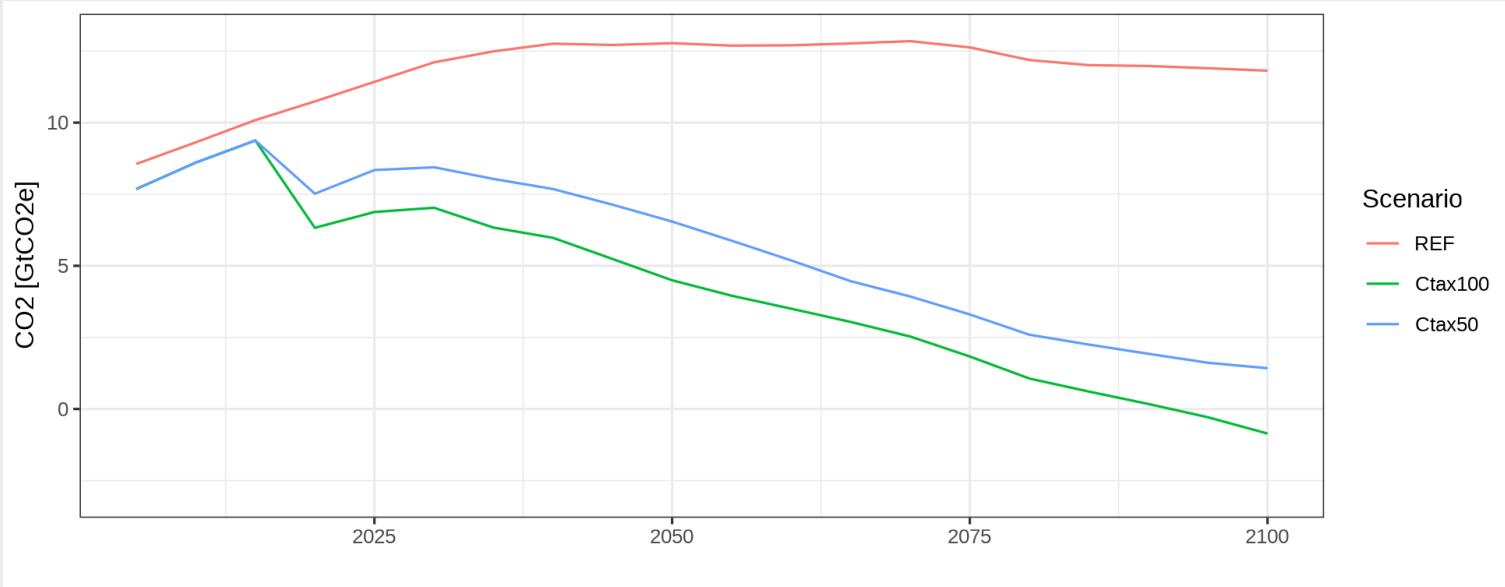
2D graphs - lines plots

```
# We have 4 changing variables, let's reduce it to 3 by aggregation the regions into world
# let's use the functionalities of data.table
wGHG = GHG[,list(value = sum(value)), by = c('e','t','Scenario')]
# have a look
str(wGHG)

## Classes 'data.table' and 'data.frame': 2976 obs. of 4 variables:
## $ e      : chr "nip" "nip" "nip" "nip" ...
## $ t      : num 2005 2010 2015 2020 2025 ...
## $ Scenario: Factor w/ 3 levels "REF","Ctax100",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ value   : num 0 0 0 0 0 0 0 0 0 0 ...
## - attr(*, ".internal.selfref")=<externalptr>
```

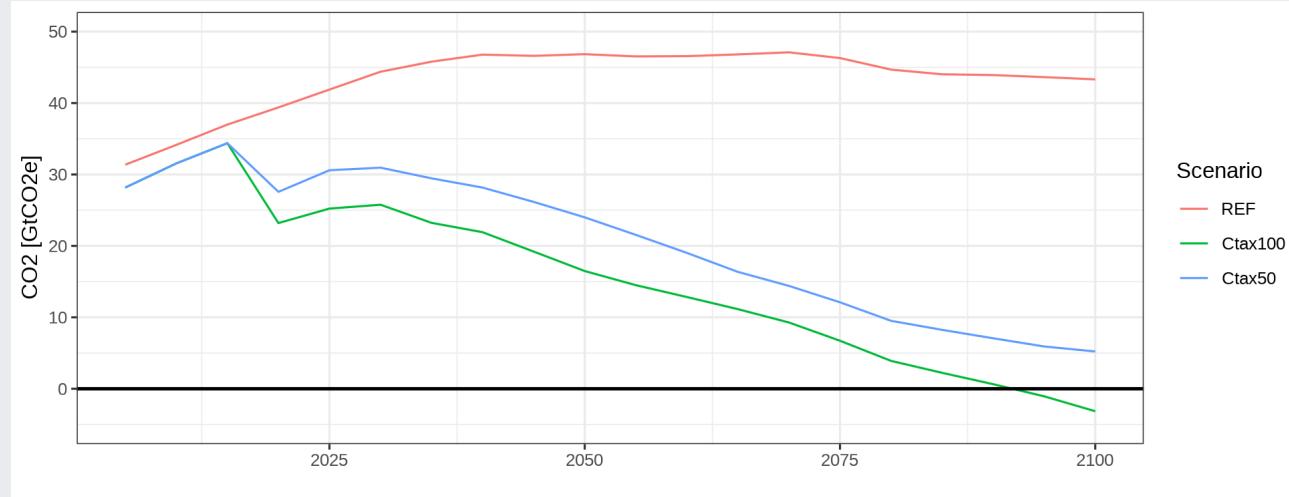
2D graphs - lines plots

```
ggplot(data=wGHG[e=="co2" & t <= 2100],aes(x=t,y=value,color=Scenario))+  
  geom_line() +  
  theme_bw() + # change theme to a more sober and less noisy background  
  scale_x_continuous(name="", limits=c(2005, 2100)) + # add title and axis limits  
  scale_y_continuous(name="CO2 [GtCO2e]", limits=c(-3, 13))
```



lines plots - meaningful messages

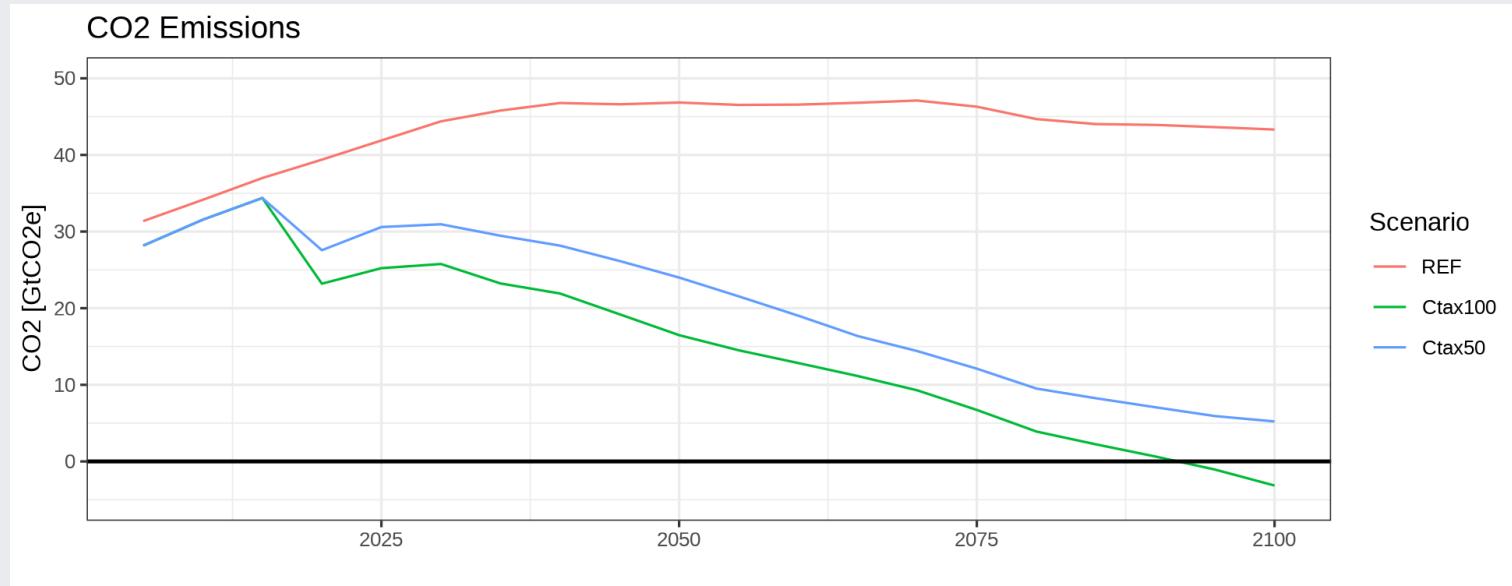
```
# let's change unit of a given Pollutant (CO2) from GtCe to CtCO2
wGHG[e=='co2',]$value = wGHG[e=='co2',]$value * 44 / 12
# Now let's have a look at our data
ggplot(data = wGHG[e=='co2' & t<=2100], aes(x=t,y=value,color=Scenario))+
  geom_line() +
  theme_bw() +
  scale_x_continuous(name = "", limits = c(2005, 2100)) + # add title and axis limits
  scale_y_continuous(name = "CO2 [GtCO2e]", limits = c(-5, 50))+
  geom_hline(yintercept = 0, color = 'Black', size = 0.8)
```



Now it is easy to see that a carbon tax of 100 \$/tCO₂ is enough to hit negative carbon emissions

2D graphs - add a title

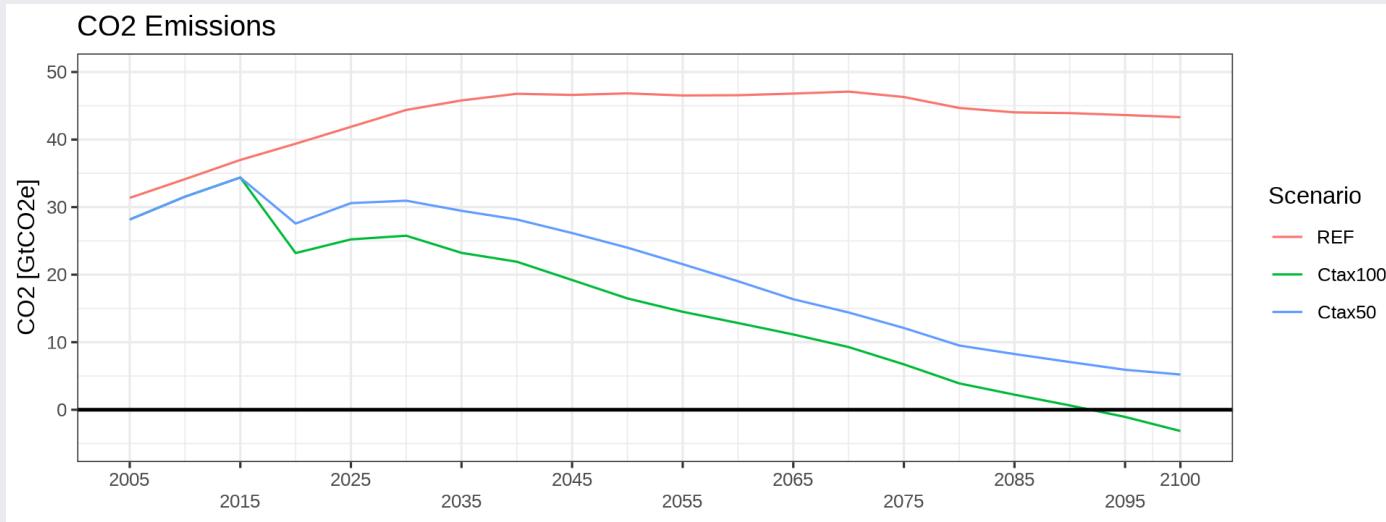
```
ggplot(data=wGHG[e=='co2' & t<=2100], aes(x = t,y = value,color = Scenario)) +  
  geom_line() +  
  theme_bw() + # change theme to a more sober and less noisy background  
  scale_x_continuous(name = "", limits = c(2005, 2100)) + # add title and axis limits  
  scale_y_continuous(name = "CO2 [GtCO2e]", limits=c(-5, 50))+  
  geom_hline(yintercept = 0, color = 'Black', size = 0.8)+  
  ggtitle('CO2 Emissions') # add title
```



2D graphs - lines plots - adjust axis

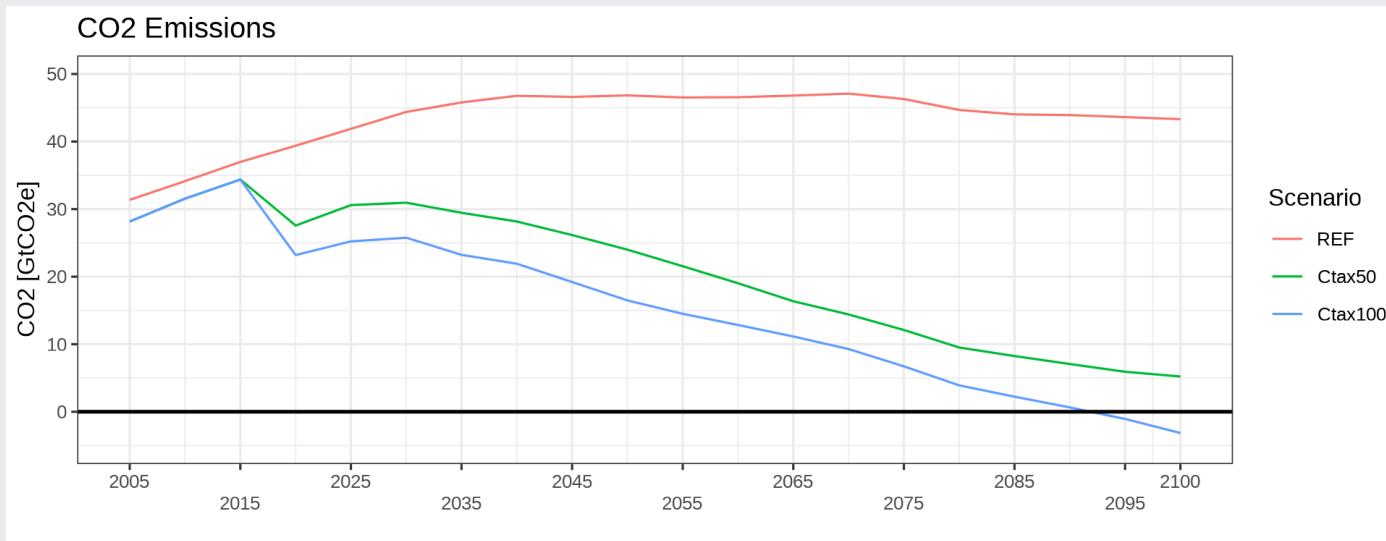
Ease the eye

```
ggplot(data=wGHG[e=="co2" & t<=2100], aes(x = t,y = value,color = Scenario)) +  
  geom_line() +  
  theme_bw() # change theme to a more sober and less noisy background  
  scale_x_continuous(name="", limits=c(2005, 2100),# add title and axis limits  
                     breaks=c(seq(2005,2100,by = 10),2100), #define the axis labels  
                     guide = guide_axis(n.dodge = 2)) # avoid overlapping  
  scale_y_continuous(name = "CO2 [GtCO2e]", limits = c(-5, 50))+  
  geom_hline(yintercept = 0, color = 'Black', size = 0.8)+  
  ggtitle('CO2 Emissions') # add title
```



Help the reader

```
# change the color order to help pass your message
wGHG$Scenario <- factor(wGHG$Scenario, levels = c("REF","Ctax50","Ctax100"))
ggplot(data=wGHG[e=='co2' & t<=2100], aes(x = t,y = value,color = Scenario))+  
  geom_line() +  
  theme_bw() + # change theme to a more sober and less noisy background  
  scale_x_continuous(name = "", limits = c(2005, 2100),# add title and axis limits  
                      breaks = c(seq(2005,2100,by = 10),2100), # define the axis labels  
                      guide = guide_axis(n.dodge = 2)) + # avoid overlapping  
  scale_y_continuous(name = "CO2 [GtCO2e]", limits = c(-5, 50))+  
  geom_hline(yintercept = 0, color = 'Black', size = 0.8)+  
  ggtitle('CO2 Emissions') # add title
```

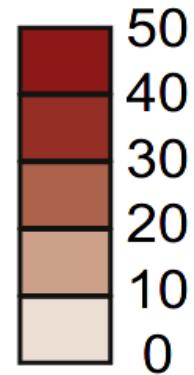
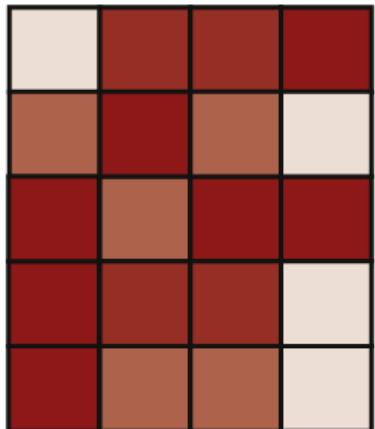


Choose your colors

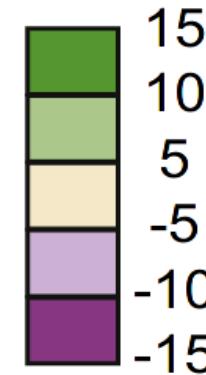
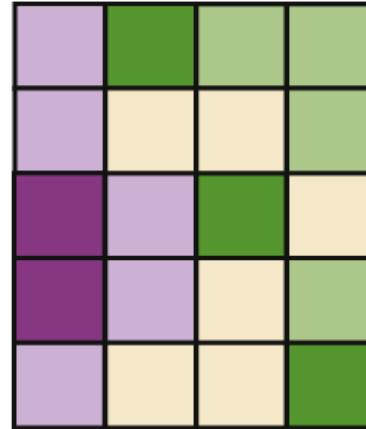
Here are some libraries with color pallets

```
#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



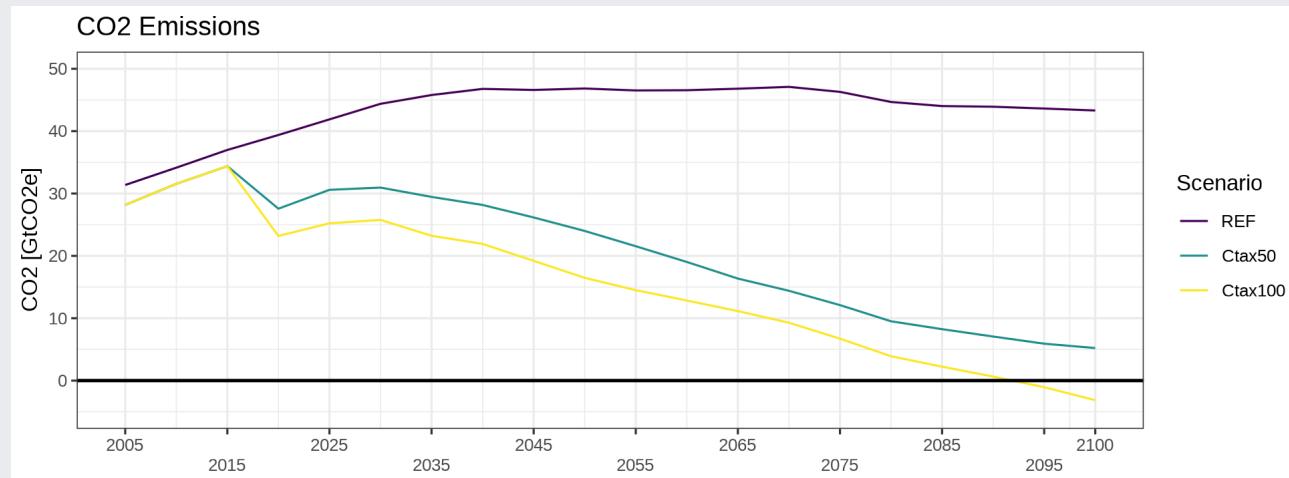
Divergent (zero in the scale)



source: Kelleher *et al.* (2011)

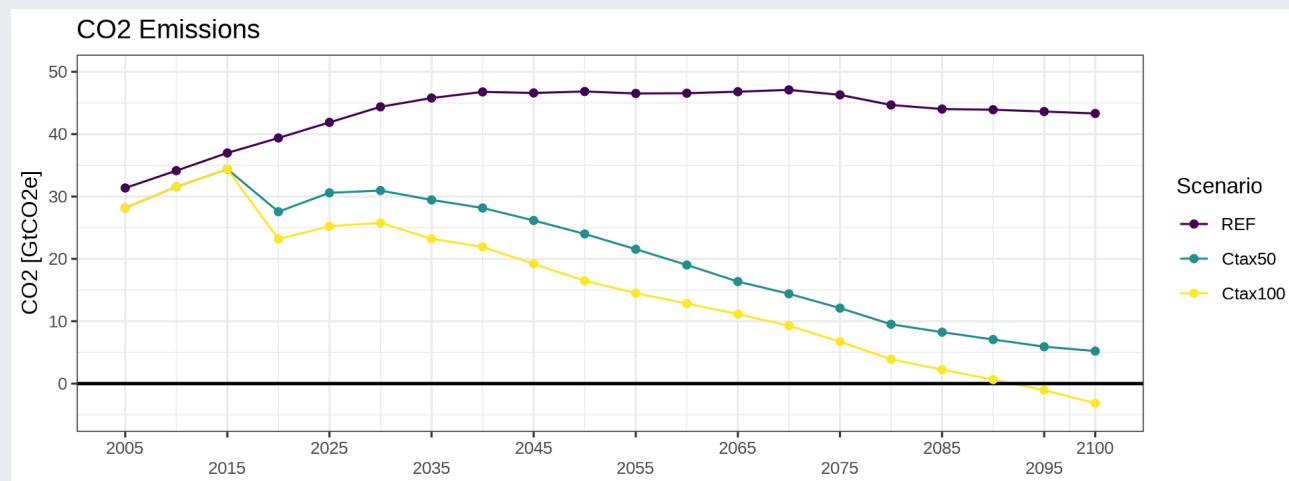
Line plot - Colors

```
# change the color order
ggplot(data=wGHG[e=='co2' & t<=2100],aes(x = t,y = value,color = Scenario))+  
  geom_line() +  
  theme_bw() + # change theme to a more sober and less noisy background  
  scale_x_continuous(name = "", limits = c(2005, 2100),# add title and axis limits  
                     breaks = c(seq(2005,2100,by = 10),2100), # define the axis labels  
                     guide = guide_axis(n.dodge=2)) + #avoid overlapping  
  scale_y_continuous(name="CO2 [GtCO2e]", limits=c(-5, 50))+  
  geom_hline(yintercept = 0,color='Black',size=0.8)+  
  ggtitle('CO2 Emissions') +# add title  
  scale_color_viridis(discrete = TRUE, option = "D") # use "?" to check the options
```



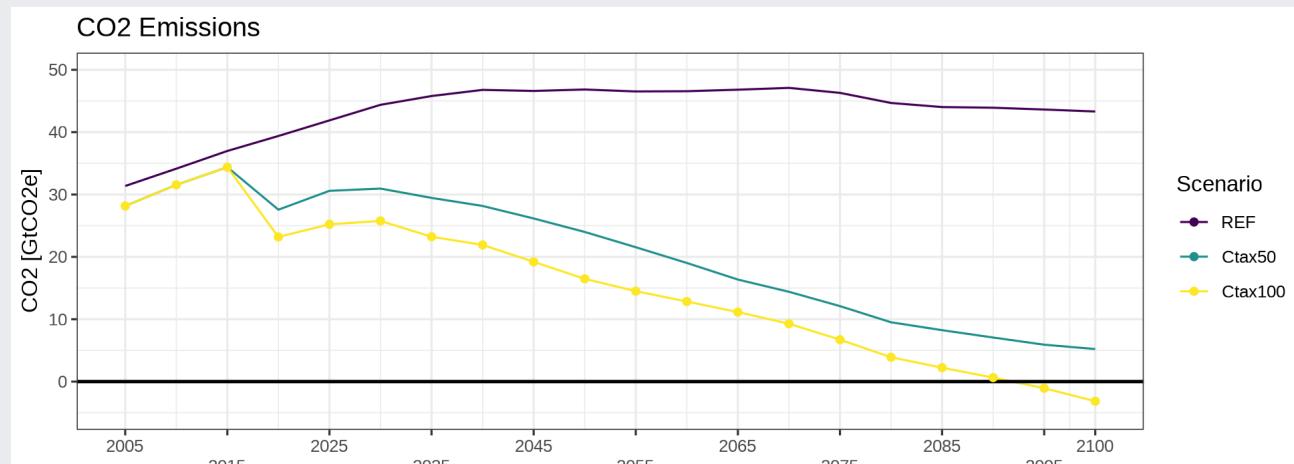
Line plot - Add points

```
ggplot(data=wGHG[e=="co2" & t<=2100],aes(x = t,y = value,color = Scenario))+  
  geom_line() +  
  theme_bw() + # change theme to a more sober and less noisy background  
  scale_x_continuous(name = "", limits = c(2005, 2100), # add title and axis limits  
                     breaks = c(seq(2005,2100,by = 10),2100), # define the axis labels  
                     guide = guide_axis(n.dodge = 2)) + # avoid overlapping  
  scale_y_continuous(name = "CO2 [GtCO2e]", limits = c(-5, 50))+  
  geom_hline(yintercept = 0,color = 'Black',size = 0.8)+  
  ggtitle('CO2 Emissions') + #add title  
  scale_color_viridis(discrete = TRUE, option = "D")+ # use "?" to check the options  
  geom_point()
```



Line plot - subset different data

```
ggplot(data=wGHG[e=='co2' & t<=2100],aes(x = t,y = value,color = Scenario))+  
  geom_line() +  
  theme_bw() + # change theme to a more sober and less noisy background  
  scale_x_continuous(name = "", limits = c(2005, 2100), # add title and axis limits  
                     breaks = c(seq(2005,2100,by = 10),2100), # define the axis labels  
                     guide = guide_axis(n.dodge = 2)) + # avoid overlapping  
  scale_y_continuous(name="CO2 [GtCO2e]", limits = c(-5, 50)) +  
  geom_hline(yintercept = 0,color = 'Black',size = 0.8)+  
  ggtitle('CO2 Emissions') + # add title  
  scale_color_viridis(discrete = TRUE, option = "D")# use "?" to check the options  
  geom_point(data = wGHG[e=='co2' & t<=2100 & Scenario == 'Ctax100'],  
             aes(x = t,y = value,color = Scenario))
```



Scatter plots - load more data

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

```
POP <- batch_extract("l",files = myfiles) # load population
POP <- setDT(POP$l)

# let's transform time periods into Years
POP$t <- as.numeric(POP$t) * 5 + 2000

# let's transform the file names into meaningful scenario names
POP$gdx <- as.factor(POP$gdx)
levels(POP$gdx) <- c('REF','Ctax100','Ctax50')
# now let's change a columns name
setnames(POP,'gdx','Scenario')
```

Scatter plots - Merging two data.table objects

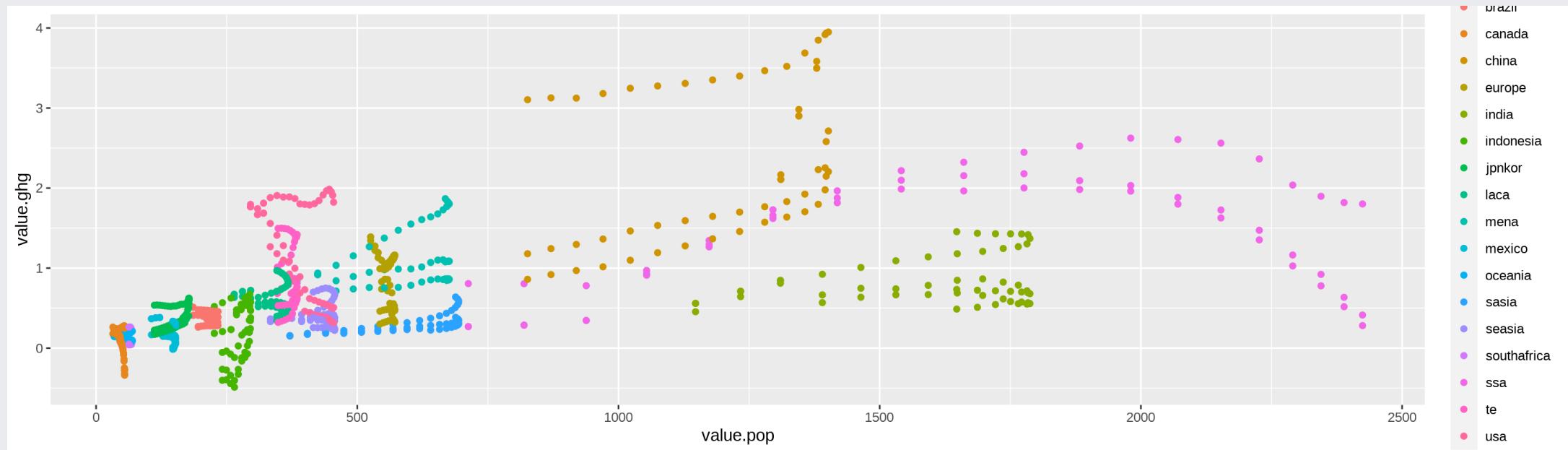
Let's get another variable and merge it with our regional Emissions (GHG)

```
# let's keep 2 emission parameters
dat = merge(GHG[e %in% c('kghg','ccs'),],POP,by = c('t','n','Scenario'),suffixes = c(".ghg",".pop"))
# have a look, it's magical
str(dat)

## Classes 'data.table' and 'data.frame':    3060 obs. of  6 variables:
## $ t      : num  2005 2005 2005 2005 2005 ...
## $ n      : chr  "brazil" "brazil" "brazil" "brazil" ...
## $ Scenario : Factor w/ 3 levels "REF","Ctax100",...: 1 1 2 2 3 3 1 1 2 2 ...
## $ e      : chr  "kghg" "ccs" "kghg" "ccs" ...
## $ value.ghg: num  5.12e-01 7.00e-08 4.10e-01 7.00e-08 4.10e-01 ...
## $ value.pop: num  186 186 186 186 186 ...
## - attr(*, ".internal.selfref")=<externalptr>
## - attr(*, "sorted")= chr [1:3] "t" "n" "Scenario"
```

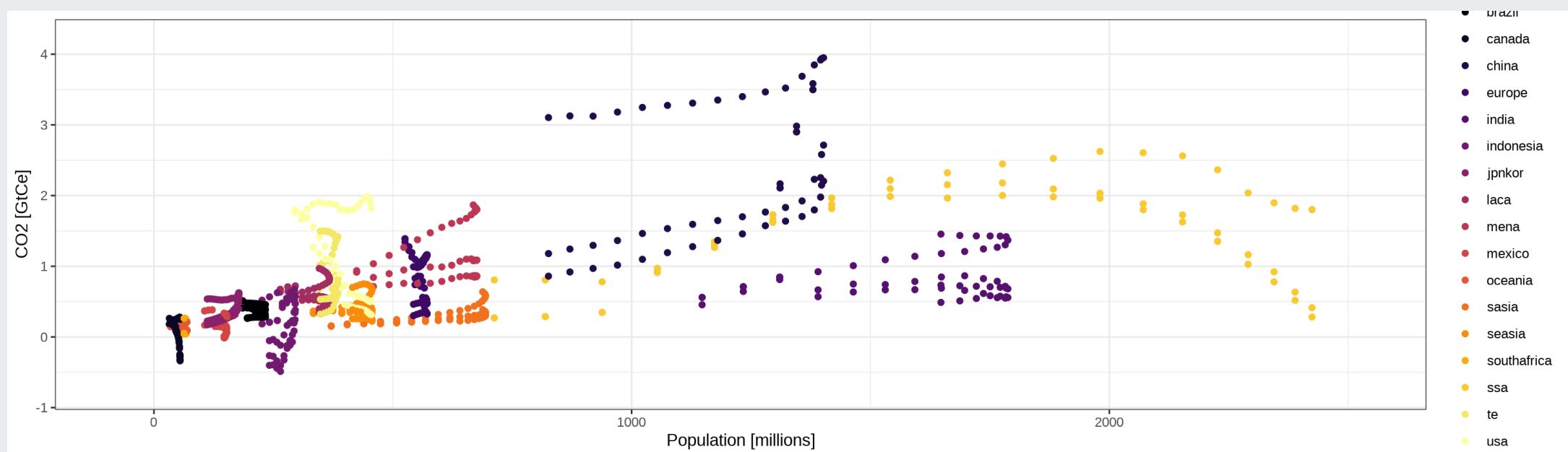
Scatter plots - Adding dimensions

```
ggplot(data = dat[t<2100 & e=='kggh'], aes(x = value.pop,y = value.ghg,color = n)) +  
  geom_point()
```



Scatter plots - Refuse the default options!

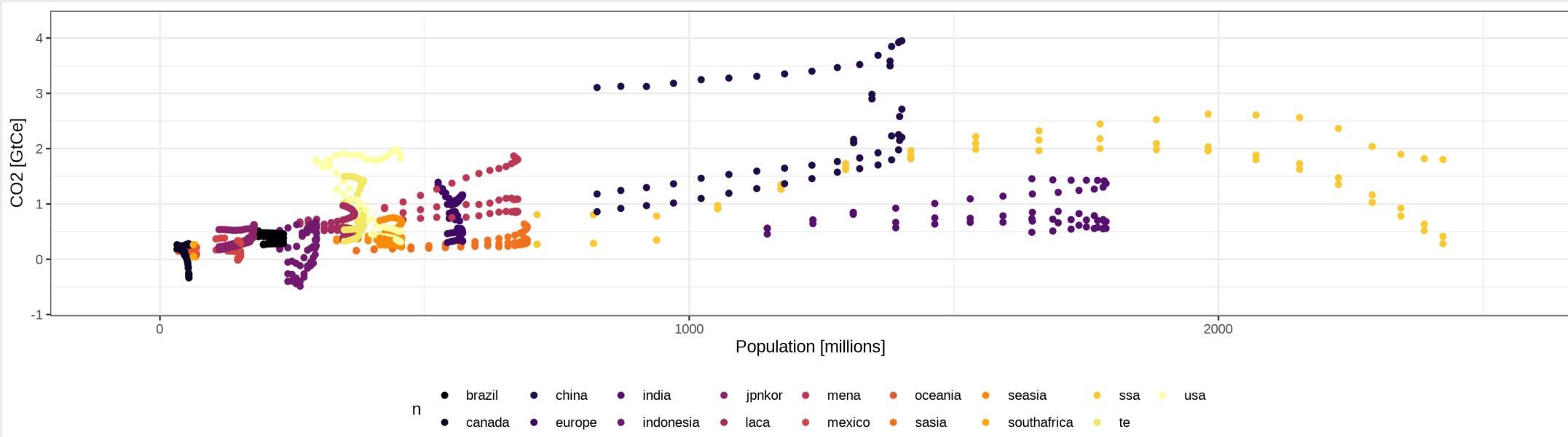
```
ggplot(data = dat[t<2100 & e=='kghg'], aes(x = value.pop,y = value.ghg,color = n)) +  
  geom_point() + theme_bw() +  
  scale_x_continuous(name = "Population [millions]",expand = c(0.1, 0.1)) +  
  scale_y_continuous(name = "CO2 [GtCe]",expand = c(0.1, 0.1))+  
  scale_color_viridis(discrete = TRUE, option = "B") # choosing another option
```



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

Scatter plots - Adjust legend

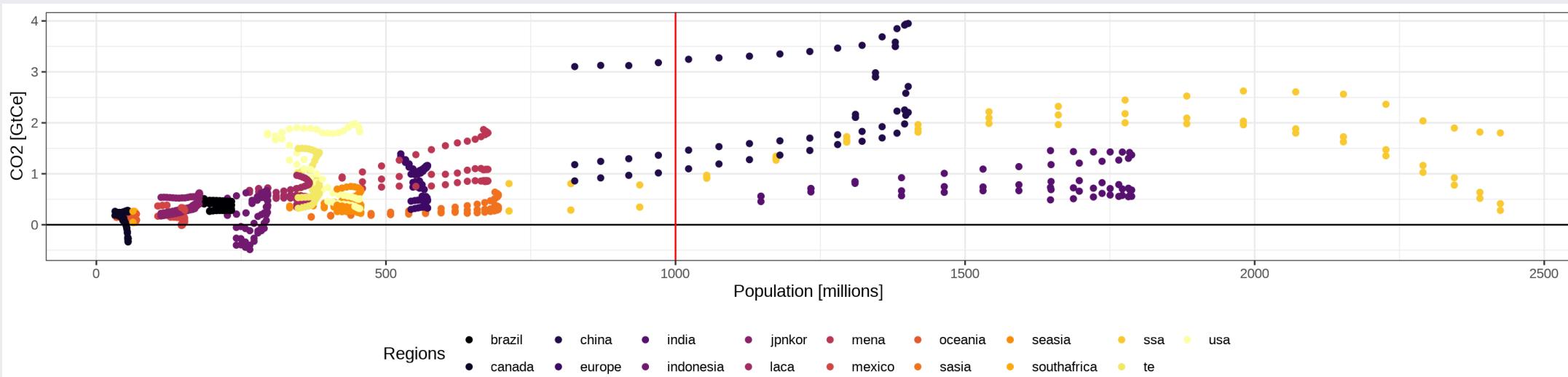
```
ggplot(data = dat[t<2100 & e=='kghg'], aes(x = value.pop,y = value.ghg,color = n)) +  
  geom_point() + theme_bw() +  
  scale_x_continuous(name="Population [millions]",expand = c(0.1, 0.1)) +  
  scale_y_continuous(name="CO2 [GtCe]",expand = c(0.1, 0.1))+  
  scale_color_viridis(discrete = TRUE, option = "B") # choosing another option  
  guides(col = guide_legend(ncol = 9))+ # divide in 9 columns  
  theme(legend.position = "bottom") # put on top
```



Scatter plots - Details are important

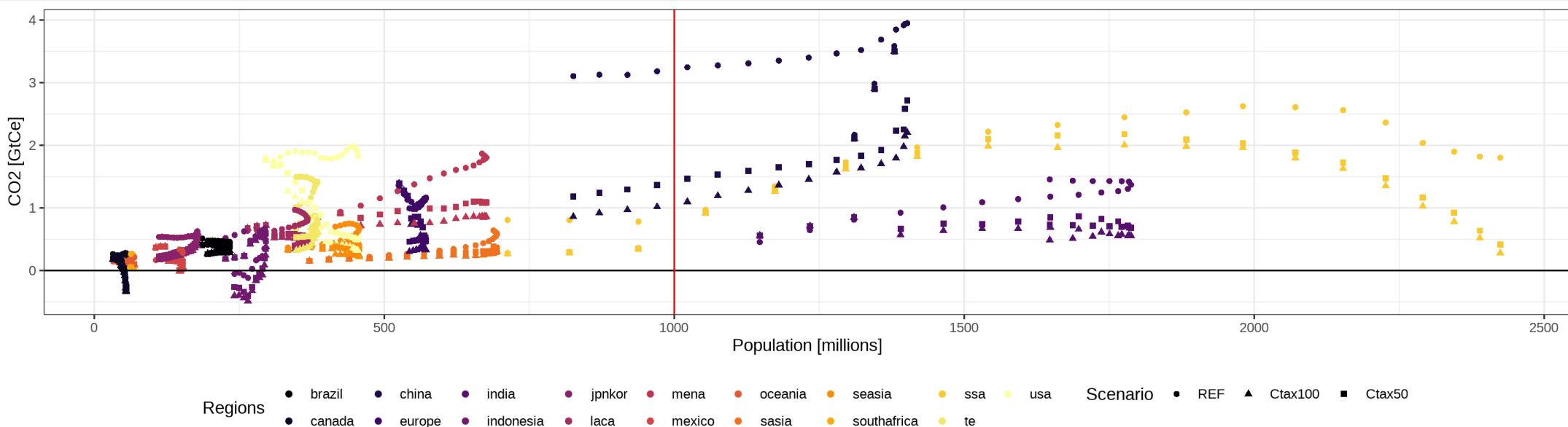
Change names of legend and add guide lines

```
ggplot(data = dat[t<2100 & e=='kgghg'], aes(x = value.pop,y = value.ghg,color = n)) +  
  geom_point() + theme_bw() +  
  scale_x_continuous(name = "Population [millions]") +  
  scale_y_continuous(name = "CO2 [GtCe]") +  
  scale_color_viridis(name = "Regions",discrete = TRUE, option = "B") + # choosing another option  
  guides(col = guide_legend(ncol = 9)) + # divide in 9 columns  
  theme(legend.position="bottom") + # put on top  
  geom_hline(yintercept = 0) + # add a horizontal line  
  geom_vline(xintercept = 1000,color = 'red') # separate high population regions
```



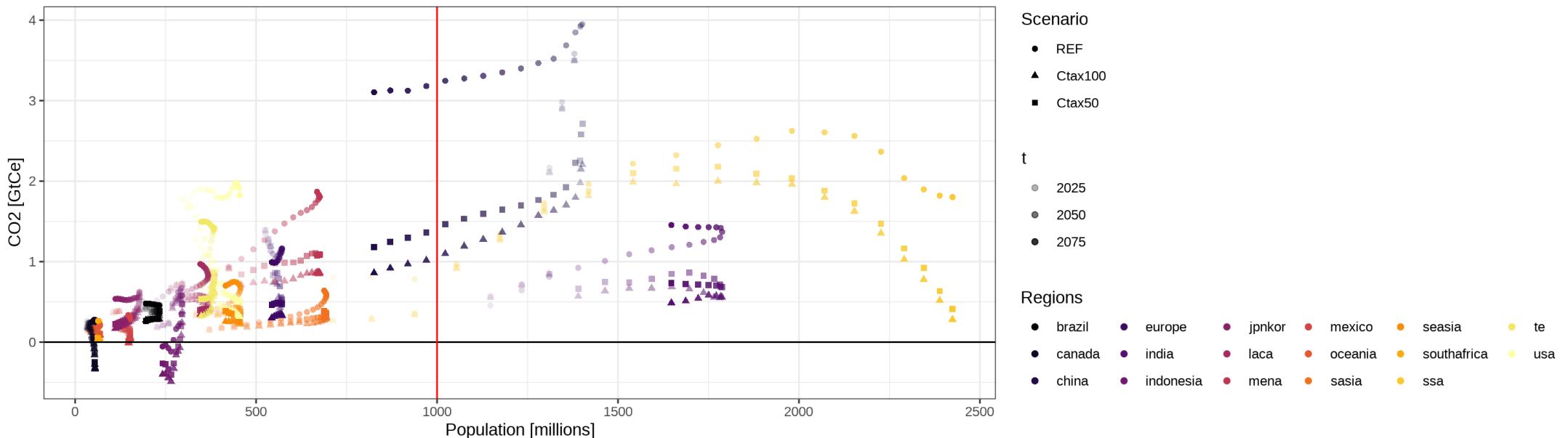
Scatter plots - Add one more dimension

```
ggplot(data=dat[t<2100 & e=="kghg"], aes(x = value.pop,y = value.ghg,color = n,shape = Scenario))+ # add shape  
  geom_point() + theme_bw() +  
  scale_x_continuous(name = "Population [millions]") +  
  scale_y_continuous(name = "CO2 [GtCe]") +  
  scale_color_viridis(name = "Regions",discrete = TRUE, option = "B") + # choosing another option  
  guides(col = guide_legend(ncol = 9)) + # divide in 9 columns  
  theme(legend.position = "bottom") + # put on top  
  geom_hline(yintercept = 0) +  
  geom_vline(xintercept = 1000,color = 'red')
```



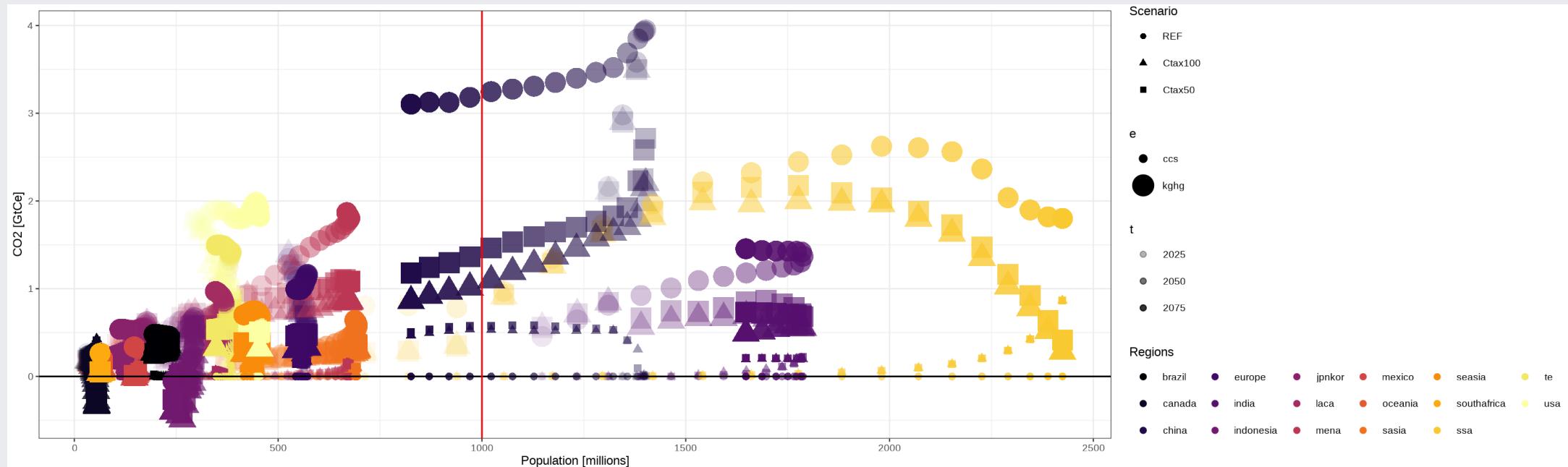
Scatter plots - Add yet another dimension

```
ggplot(data=dat[t<2100 & e=="kghg"], aes(x = value.pop,y = value.ghg,color = n,shape = Scenario,alpha = t))+ # add transparency
  geom_point() + theme_bw() +
  scale_x_continuous(name = "Population [millions]") +
  scale_y_continuous(name = "CO2 [GtCe]") +
  scale_color_viridis(name = "Regions", discrete = TRUE, option = "B") +
  guides(col = guide_legend(ncol = 6)) +
  theme(legend.position = "right") + # put on to right
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 1000,color = 'red')
```



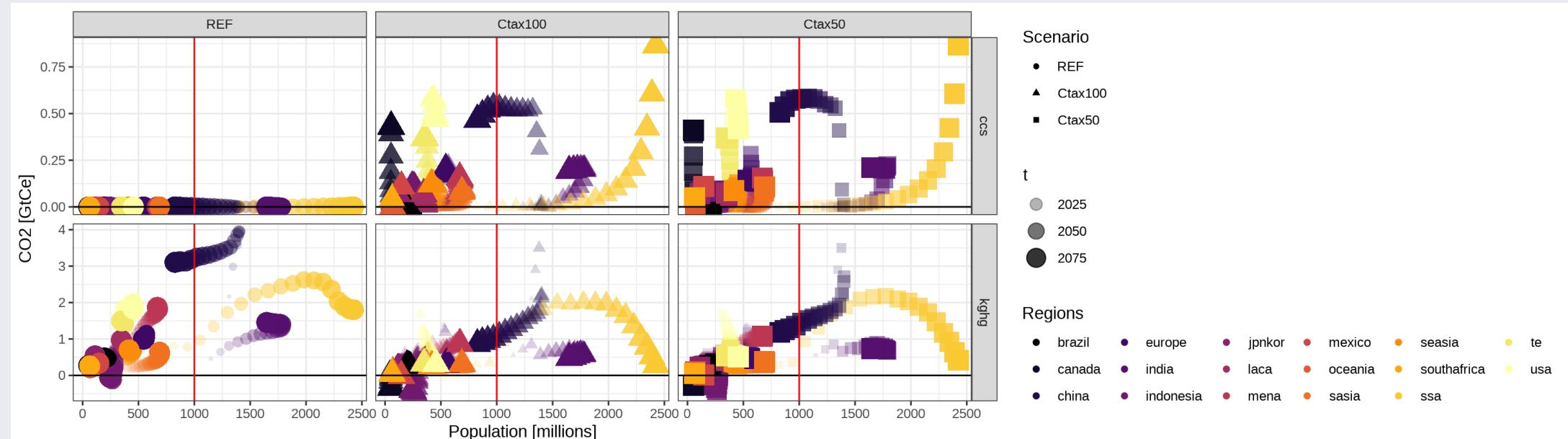
Scatter plots - and one more ... too many!

```
ggplot(data = dat[t<2100 ], aes(x = value.pop,y = value.ghg,color = n,shape = Scenario,alpha = t,size = e)) + # add size with  
geom_point() + theme_bw(base_size = 8) +  
scale_x_continuous(name = "Population [millions]") +  
scale_y_continuous(name = "CO2 [GtCe]") +  
scale_color_viridis(name = "Regions", discrete = TRUE, option = "B") +  
guides(col = guide_legend(ncol = 6)) +  
theme(legend.position = "right") +  
geom_hline(yintercept = 0) +  
geom_vline(xintercept = 1000,color = 'red')
```



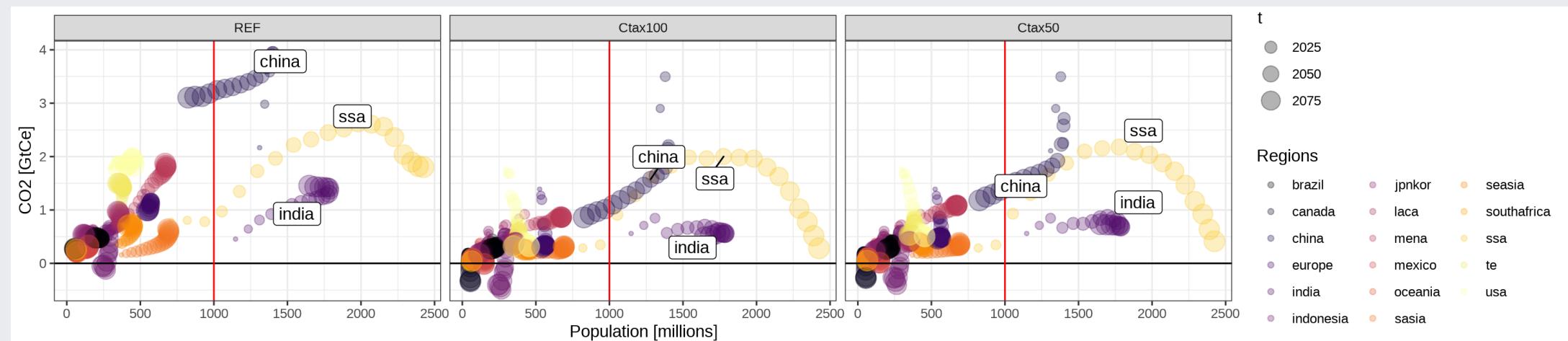
Scatter plots - Let's exaggerate

```
ggplot(data=dat[t<2100 ], aes(x = value.pop,y = value.ghg,color = n,shape = Scenario,alpha = t,size = t)) +  
  geom_point() + theme_bw() +  
  scale_x_continuous(name="Population [millions]") +  
  scale_y_continuous(name="CO2 [GtCe]") +  
  scale_color_viridis(name="Regions",discrete = TRUE, option = "B") +  
  guides(col = guide_legend(ncol = 6)) +  
  theme(legend.position="right") +  
  geom_hline(yintercept=0) +  
  geom_vline(xintercept=1000,color='red') +  
  facet_grid(e~Scenario,scales='free')
```



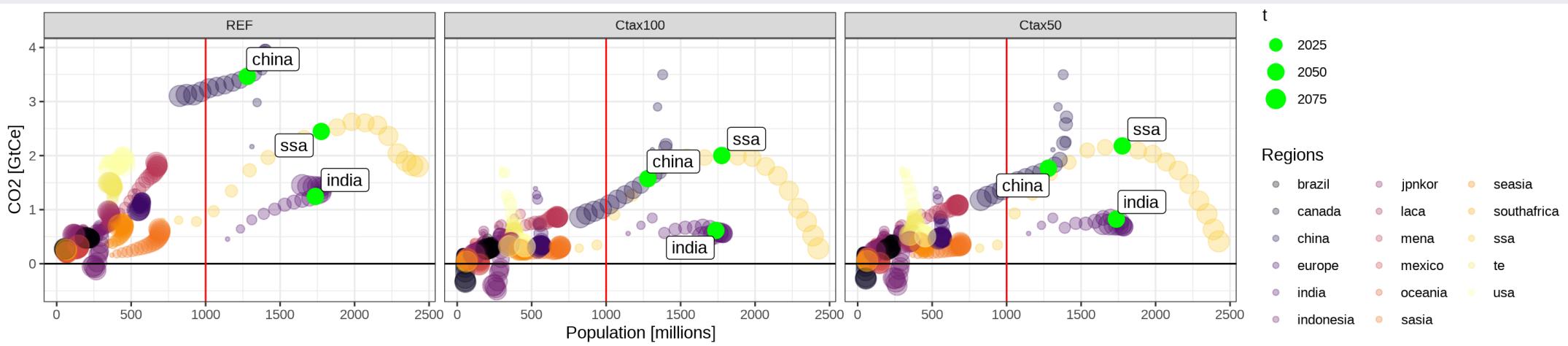
Scatter plots - help the reader

```
library(ggrepel)
ggplot(data=dat[t<2100 & e=='kghg'], aes(x=value.pop,y=value.ghg,color=n))+ 
  geom_point(aes(size=t),alpha=0.3) + theme_bw() + 
  scale_x_continuous(name="Population [millions]") + 
  scale_y_continuous(name="CO2 [GtCe]") + 
  scale_color_viridis(name="Regions",discrete = TRUE, option = "B") + 
  guides(col = guide_legend(ncol = 3)) + # divide in 3 columns
  theme(legend.position="right") + 
  geom_hline(yintercept=0) + 
  geom_vline(xintercept=1000,color='red') + 
  geom_label_repel(data=dat[t==2050 & value.pop>1000 & e=='kghg'],
                    aes(x=value.pop,y=value.ghg,label=n),color='black') + # draw attention to ...
  facet_grid(~Scenario,scales='free')
```



Scatter plots - Draw attention to ...

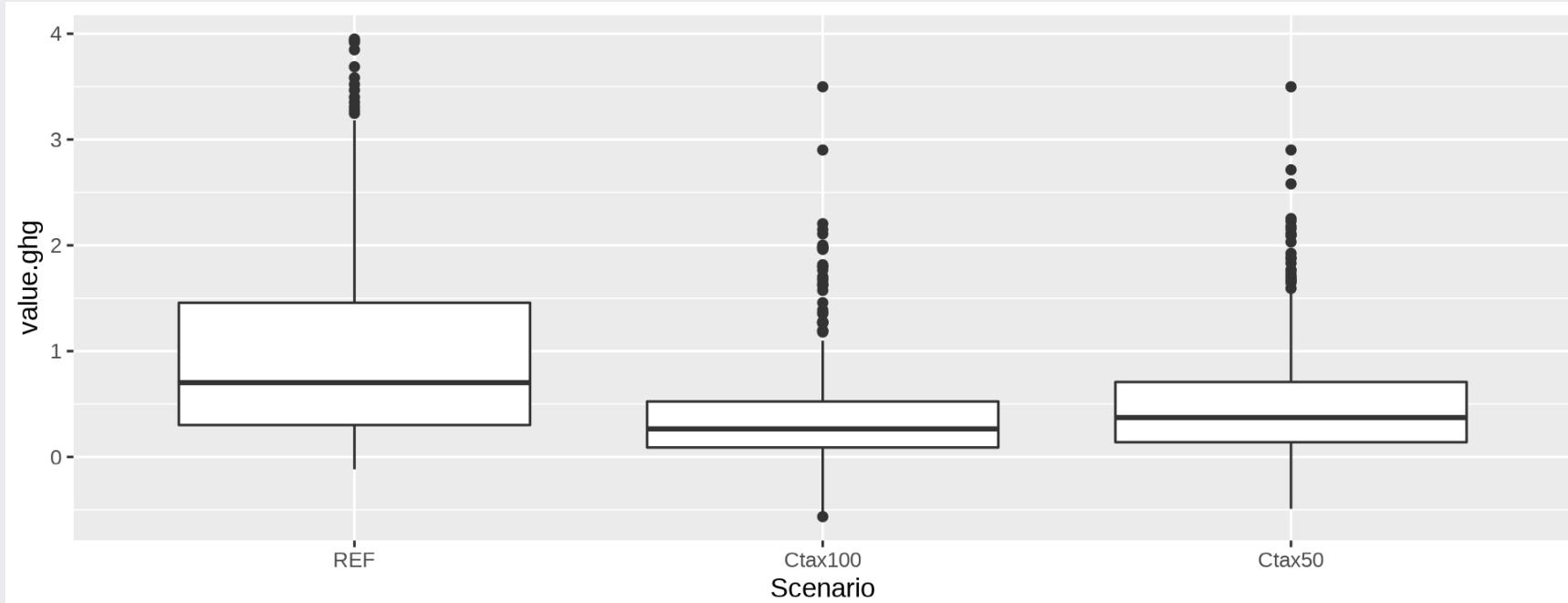
```
ggplot(data=dat[t<2100 & e=="kghg"], aes(x=value.pop,y=value.ghg,color=n))+  
  geom_point(aes(size=t),alpha=0.3) + theme_bw() +  
  scale_x_continuous(name="Population [millions]") +  
  scale_y_continuous(name="CO2 [GtCe]") +  
  scale_color_viridis(name="Regions",discrete = TRUE, option = "B") +  
  guides(col = guide_legend(ncol = 3)) + # divide in 3 columns  
  theme(legend.position="right") +  
  geom_hline(yintercept=0) +  
  geom_vline(xintercept=1000,color='red') +  
  geom_point(data=dat[t==2050 & value.pop>1000 & e=="kghg"],  
             aes(x=value.pop,y=value.ghg,size=t),color='green') + #emphasize a few points  
  geom_label_repel(data=dat[t==2050 & value.pop>1000 & e=="kghg"],  
                    aes(x=value.pop,y=value.ghg,label=n),color='black') + #draw attention to ...  
  facet_grid(~Scenario,scales='free')
```



Box plots - Plotting distributions

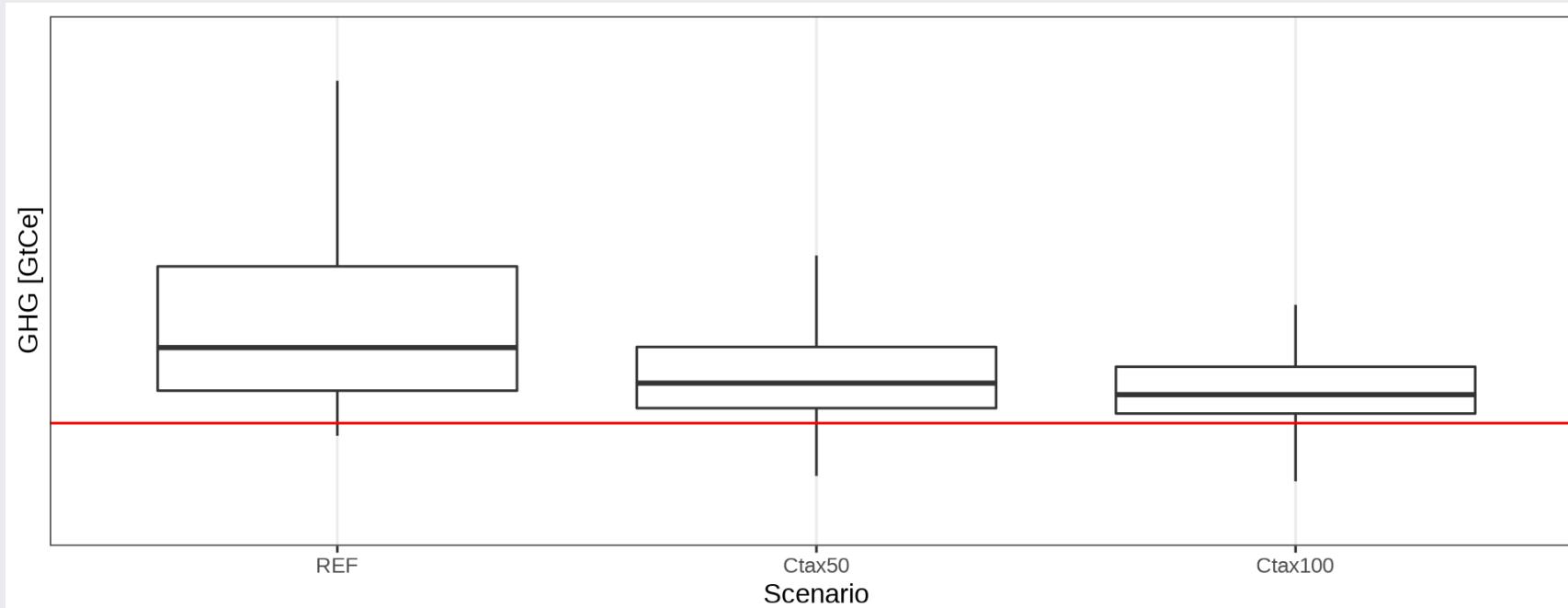
When mean or median are not enough

```
ggplot(data=dat[e=='kghg'], aes(x=Scenario,y=value.ghg))+  
  geom_boxplot()
```



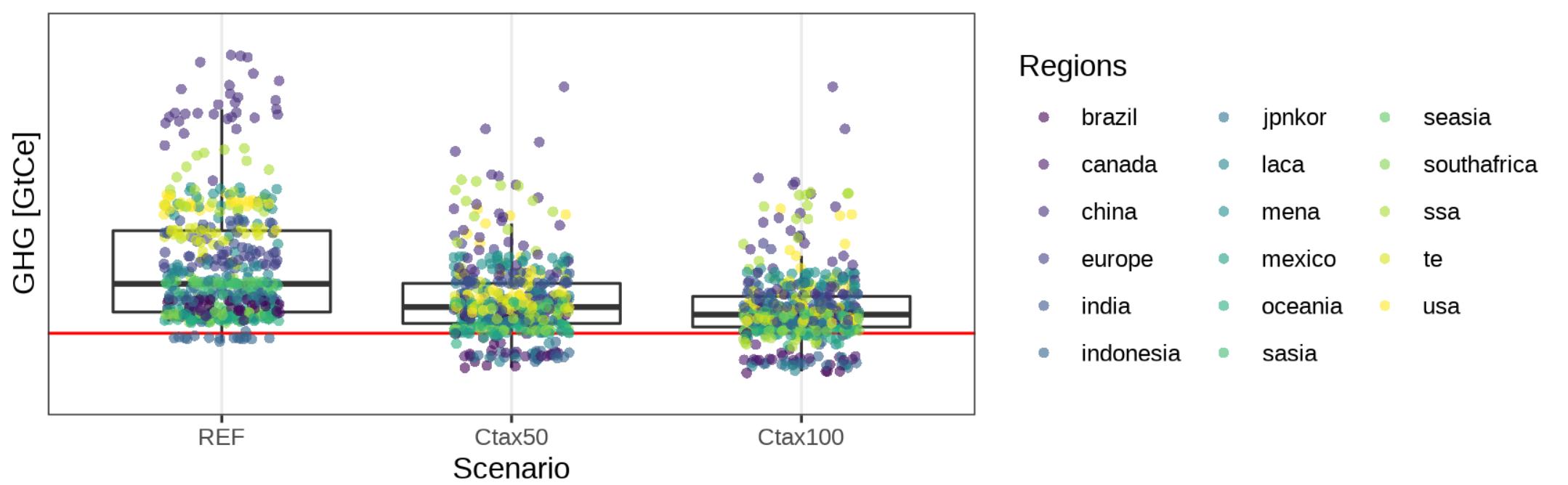
Box plots - typical retouch..

```
# order x by something that makes sense  
dat$Scenario <- factor(dat$Scenario, levels = c("REF","Ctax50","Ctax100"))  
  
ggplot(data=dat[e=='kghg'], aes(x=Scenario,y=value.ghg))+  
  geom_boxplot(outlier.shape = NA) + theme_bw() +  
  geom_hline(yintercept = 0,color="red") +  
  scale_y_discrete(name="GHG [GtCe]")
```



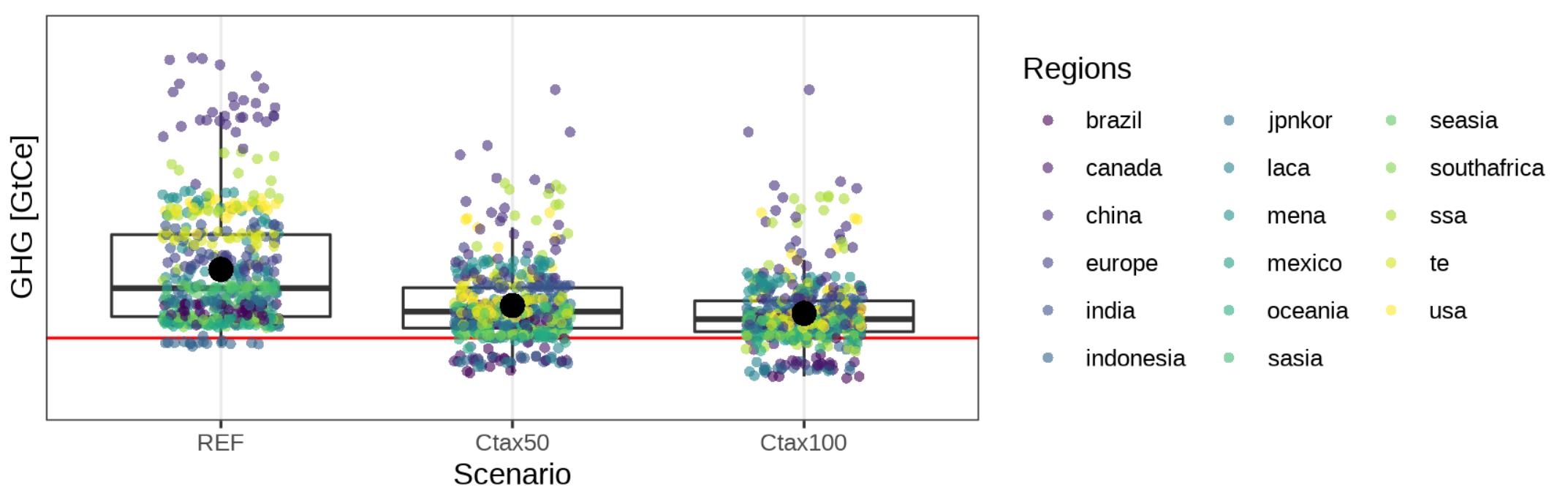
Box plots - add the underlying observations

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



Box plots - add more information

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



Shapes and linetypes

[Shape]

□ 0	◇ 5	⊕ 10	■ 15	■ 22
○ 1	▽ 6	⊗ 11	● 16	● 21
△ 2	⊗ 7	田 12	▲ 17	▲ 24
+ 3	* 8	⊗ 13	◆ 18	◆ 23
× 4	◇ 9	田 14	● 19	● 20

[Linetype]



Maps - Showing regional information

We need to prepare the data first

```
# import the WITCH model regional definition
wreg = fread('Material/mapwitch17.csv',header = TRUE)
# we will need a new library
library(countrycode)

wreg$region = wreg$ISO
# transforms ISO codes into country names
wreg$region = countrycode(wreg$ISO, origin = 'iso3c', destination = 'country.name')
# correcting some mismatches
wreg[grep('Congo',wreg$region)]$region = c("Democratic Republic of the Congo", "Republic of Congo" )
wreg[region=='Bosnia & Herzegovina']$region = 'Bosnia and Herzegovina'
wreg[region=='Myanmar (Burma)']$region = 'Myanmar'
wreg[region=='Côte d'Ivoire']$region = 'Ivory Coast'
wreg[region=='Czechia']$region = 'Czech Republic'
wreg[ISO=='GBR']$region = 'UK'
wreg[ISO=='USA']$region = 'USA'
```

Maps - Showing regional information

```
# Retrieve the world data for the regions of WITCH
world.maps <- map_data("world")

# let's transform it into a data.table
world.maps = setDT(world.maps)

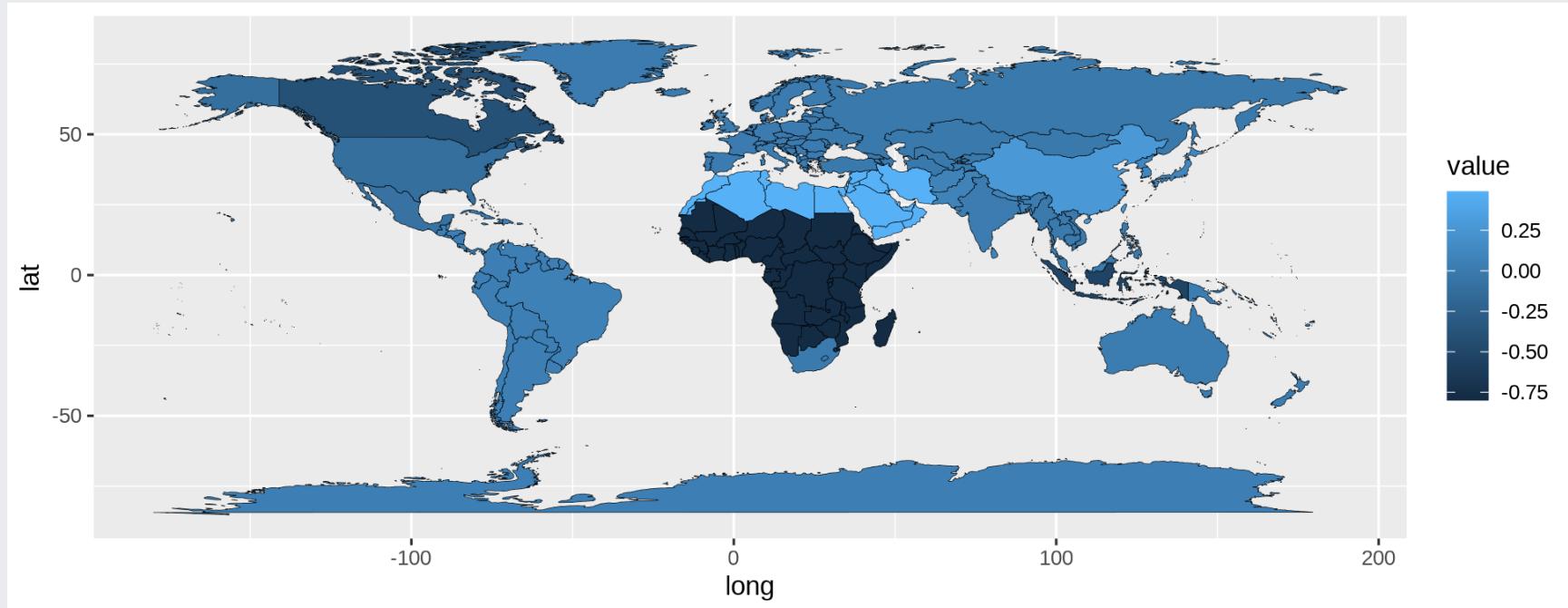
# merge the WITCH regional definition with the world map
world.maps = merge(world.maps,wreg,by='region',allow.cartesian=TRUE)

# merge with our regional GHG data.table
# let's choose an year, a pollutant
mGHG=merge(world.maps,GHG[t==2100 & e=='co2'] ,by=c('n') ,allow.cartesian=TRUE)
```

Ready to plot the map now!

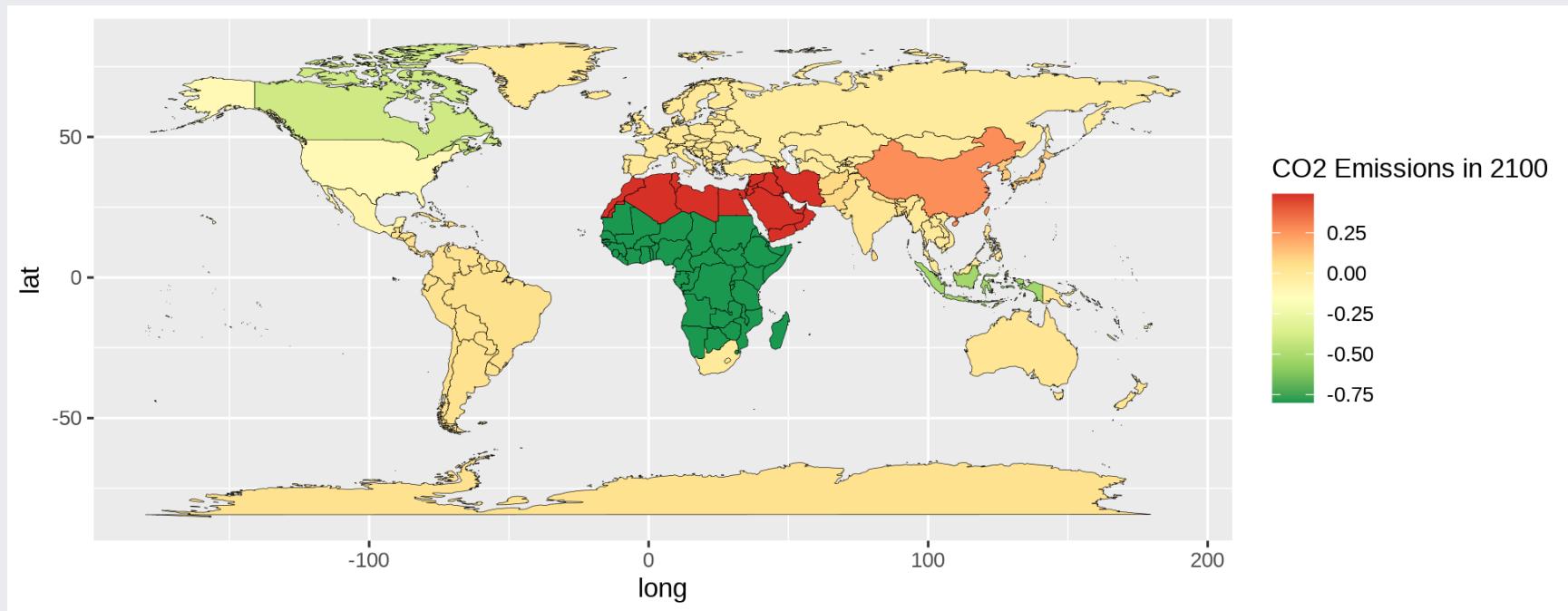
Maps - Showing regional information

```
ggplot(mGHG[ Scenario=='Ctax100',], aes(x = long, y = lat)) +  
  geom_polygon(aes( group = group, fill = value),color='black',size=.1)
```



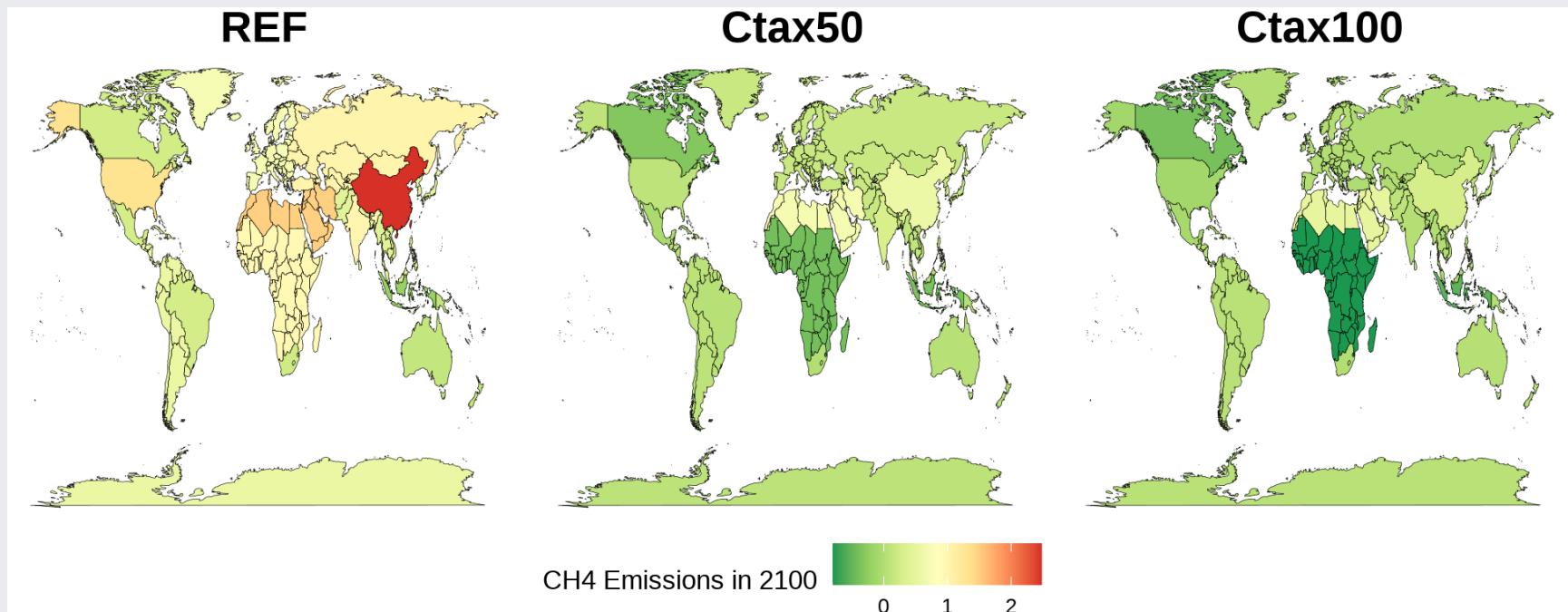
Maps - color pallets are important

```
ggplot(mGHG[ Scenario=='Ctax100'], aes(x = long, y = lat)) +  
  geom_polygon(aes( group = group, fill = value),color='black',size=.1)+  
  scale_fill_distiller(name = "CO2 Emissions in 2100",palette ="RdYlGn",direction=-1) #revert color
```



Maps - Apply the tricks you already know

```
mGHG$Scenario <- factor(mGHG$Scenario, levels = c("REF", "Ctax50", "Ctax100"))
ggplot(mGHG, aes(x = long, y = lat)) +
  geom_polygon(aes( group = group, fill = value),color='black',size=.1) +
  theme_void() + # clean the background
  scale_fill_distiller(name = "CH4 Emissions in 2100",palette ="RdYlGn",direction=-1) + # revert color
  theme(legend.position = "bottom", strip.text.x = element_text(size=18, face="bold")) +
  facet_grid(~Scenario)
```



Transforming variables

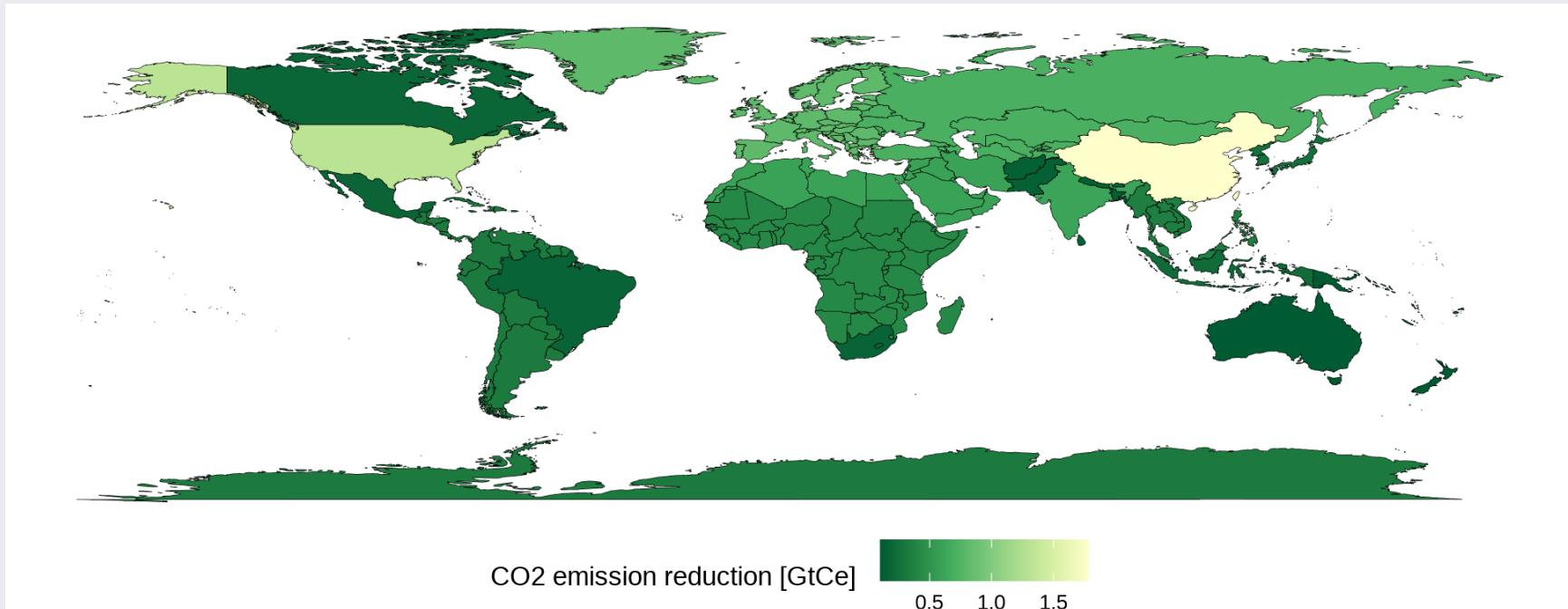
```
# let's transform GHG by having the scenarios as columns while keeping the rest of the structure
dGHG = dcast(GHG[e=='co2',], ... ~ Scenario , value.var='value' )

# Let's create a new variable/column
dGHG$dif_wrtREF = dGHG$REF - dGHG$Ctax100

# merge our regional dGHG data.table to the world polygons as before
mpGHG=merge(world.maps,dGHG[t==2050 & e=='co2' ],by=c('n'),allow.cartesian=TRUE)
```

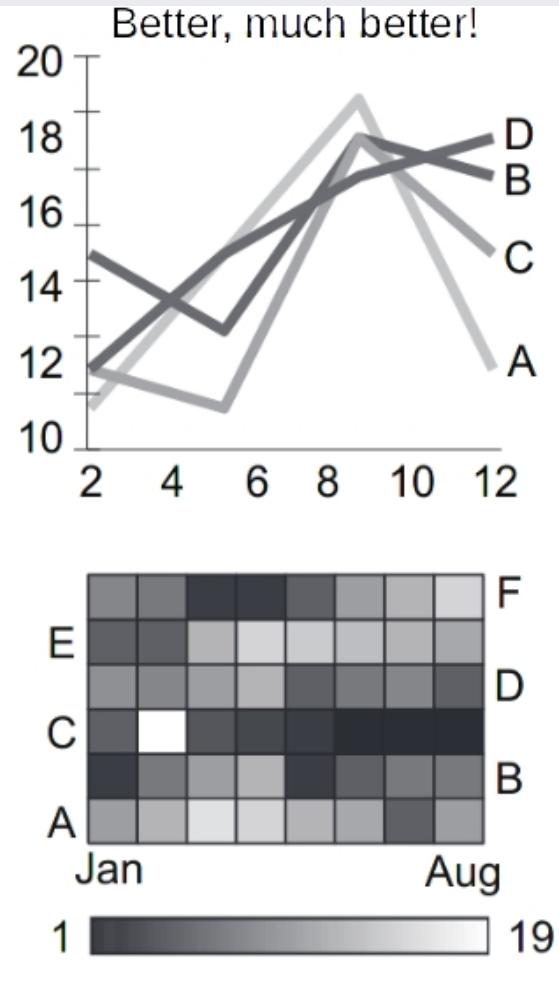
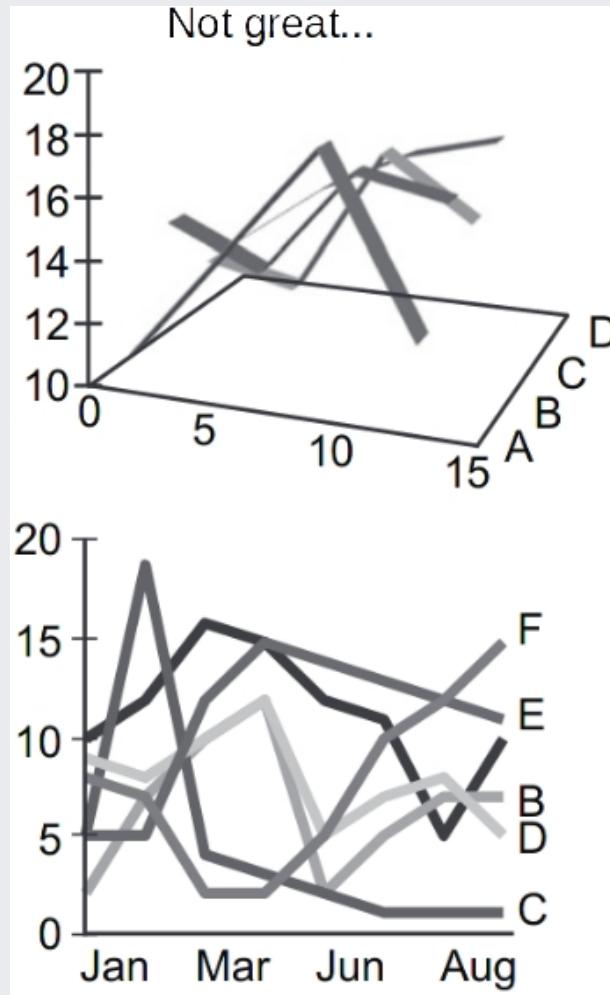
Transforming variables

```
ggplot(mpGHG, aes(x = long, y = lat)) +  
  geom_polygon(aes( group = group, fill = dif_wrtREF),color='black',size=.1)+  
  theme_void() # clean the background  
  scale_fill_distiller(name = "CO2 emission reduction [GtCe]",palette ="YlGn")#revert color  
  theme(legend.position = "bottom", strip.text.x = element_text(size=18, face="bold"))
```



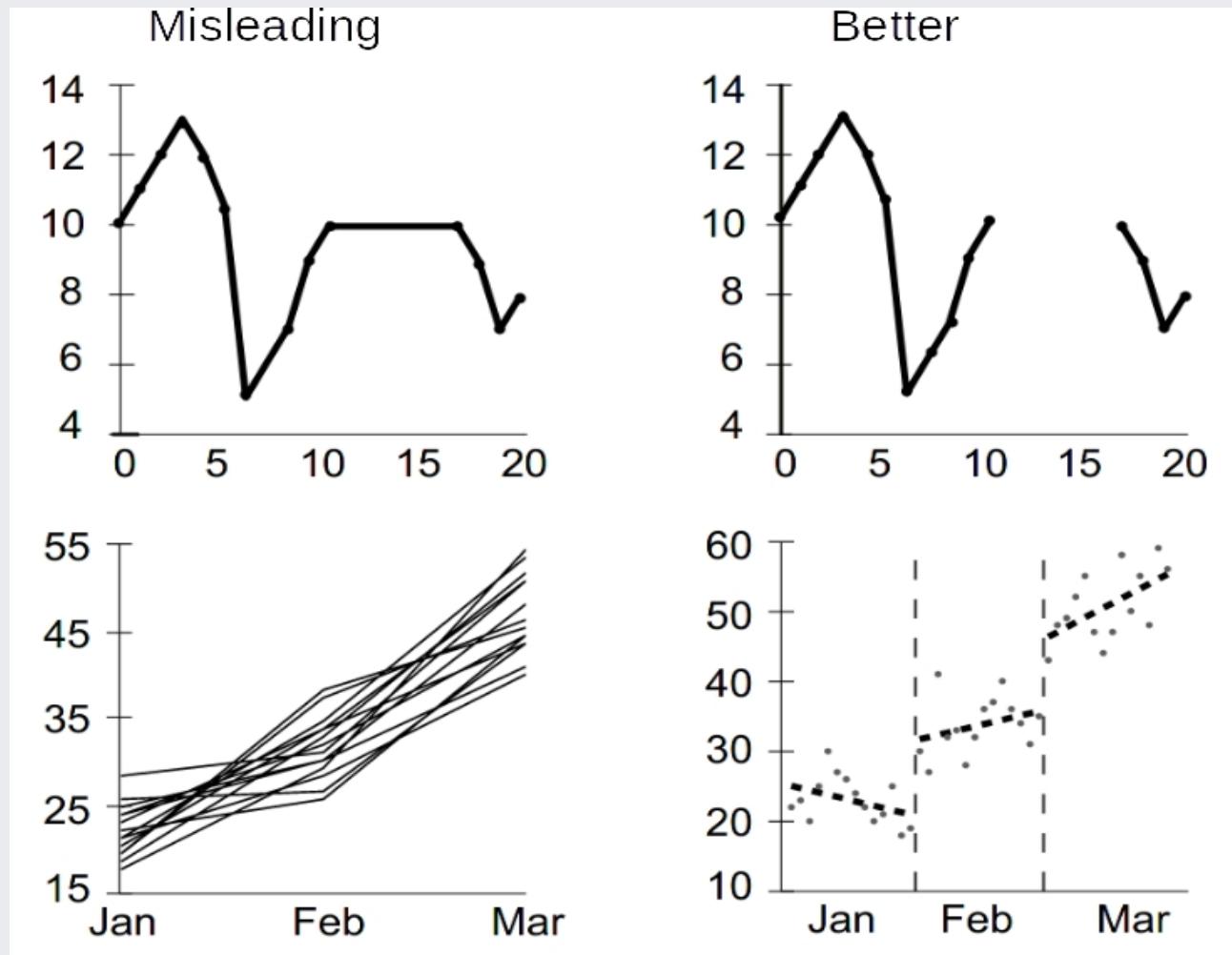
Tips for good visualization

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

- Data Visualization in R: <https://wilkelab.org/SDS375>

ggplot2

- ggplot2 book <https://ggplot2-book.org/>
- ggplot2 reference documentation <https://ggplot2.tidyverse.org/reference/index.html>