Computer Lab 5

In this computer lab we will have to achieve the following tasks/learning outcomes:

- import time-series data
- understand some important features of the dataset
- estimate basic time-series models
- perform Granger causality tests

The data are similar to the ones used in the CORE Doing Economics Project 1 on Measuring Climate Change. Some of the analysis is simlar to that described in *Attanasio, A., Pasini, A. and Triacca, U. (2013) Granger Causality Analyses for Climatic Attribution, Atmospheric and Climate Sciences, 2013, 3, 515-522.*

Preparing your workfile

We add the basic libraries needed for this week's work:

```
library(tidyverse)     # for almost all data handling tasks
library(ggplot2)       # to produce nice graphiscs
library(stargazer)     # to produce nice results tables
library(AER)           # access to HS robust standard errors
library(readxl)        # enable the read_excel function
library(xts)           # to use xts
```

You should also save the separately supplied `stargazer_HAC.r` file in your working directory (see Lecture 9 in BB). This will make it straightforward to estimate and compare regressions with HAC standard errors. Once you have done that you should include the following line into your code which basically makes this function available to you.

```
source("stargazer_HAC.r")  # includes the robust regression
```

Data Import

We will use three dataset

1. Global temperatures
2. CO2 emissions (or more precise radiative forcing) data
3. A measure of the sun's intensity

Global temperature data

Go to https://data.giss.nasa.gov/gistemp/ and scroll down to the subheading "Tables of Global and Hemispheric Monthly Means and Zonal Annual Means", select the CSV version of 'Global-mean monthly, seasonal, and annual means, 1880-present, updated through most recent month'.

```
tempdata <- read.csv("GLB.Ts+dSST.csv",skip=1,na.strings = "***")
```

When using the `read.csv` function, we added two options. The `skip=1` option is there as the real data table only starts in Row 2, so we need to skip one row. The `na.strings = "***"` option informs R how missing observations in the spreadsheet are coded. When looking at the spreadsheet in Excel (and you should always do that before you import data into R), you can see that missing data is coded as `"***"`. It is best to specify this here, as otherwise some of the data is not recognized as numeric data.

Understanding the data you are using is key to all empirical work. You can view the first few rows of the dataset, and confirm that they correspond to the columns in the csv file.

```
head(tempdata)
```

```
##   Year   Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
## 1 1880 -0.17 -0.24 -0.09 -0.16 -0.09 -0.20 -0.17 -0.09 -0.14 -0.23 -0.21 -0.17
## 2 1881 -0.19 -0.13  0.04  0.06  0.07 -0.18  0.01 -0.02 -0.14 -0.21 -0.18 -0.06
## 3 1882  0.17  0.15  0.05 -0.16 -0.14 -0.22 -0.15 -0.06 -0.14 -0.23 -0.15 -0.35
## 4 1883 -0.28 -0.36 -0.12 -0.17 -0.17 -0.07 -0.06 -0.13 -0.20 -0.10 -0.22 -0.10
## 5 1884 -0.12 -0.07 -0.35 -0.39 -0.34 -0.35 -0.31 -0.26 -0.26 -0.24 -0.32 -0.30
## 6 1885 -0.58 -0.32 -0.25 -0.41 -0.44 -0.42 -0.32 -0.29 -0.27 -0.22 -0.22 -0.09
##     J.D   D.N   DJF   MAM   JJA   SON
## 1 -0.16    NA    NA -0.11 -0.15 -0.19
## 2 -0.08 -0.09 -0.16  0.05 -0.06 -0.18
## 3 -0.10 -0.08  0.09 -0.08 -0.14 -0.17
## 4 -0.16 -0.19 -0.33 -0.15 -0.08 -0.18
## 5 -0.28 -0.26 -0.10 -0.36 -0.31 -0.28
## 6 -0.32 -0.34 -0.40 -0.37 -0.35 -0.24
```

We have monthly data (with years in rows and months in columns). Later we will use annual data and they are saved in the column `J.D` (January to December). Also try yourself what happens if you replace `head` with `tail` in the previous command.

Before we go on, it is also important to understand the data formats.

```
str(tempdata)
```

```
## 'data.frame':    141 obs. of  19 variables:
##  $ Year: int  1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 ...
##  $ Jan : num  -0.17 -0.19 0.17 -0.28 -0.12 -0.58 -0.42 -0.7 -0.33 -0.07 ...
##  $ Feb : num  -0.24 -0.13 0.15 -0.36 -0.07 -0.32 -0.49 -0.55 -0.35 0.18 ...
##  $ Mar : num  -0.09 0.04 0.05 -0.12 -0.35 -0.25 -0.42 -0.34 -0.4 0.07 ...
##  $ Apr : num  -0.16 0.06 -0.16 -0.17 -0.39 -0.41 -0.27 -0.33 -0.19 0.1 ...
##  $ May : num  -0.09 0.07 -0.14 -0.17 -0.34 -0.44 -0.23 -0.29 -0.21 0 ...
##  $ Jun : num  -0.2 -0.18 -0.22 -0.07 -0.35 -0.42 -0.33 -0.23 -0.16 -0.09 ...
##  $ Jul : num  -0.17 0.01 -0.15 -0.06 -0.31 -0.32 -0.17 -0.24 -0.09 -0.07 ...
##  $ Aug : num  -0.09 -0.02 -0.06 -0.13 -0.26 -0.29 -0.29 -0.34 -0.14 -0.19 ...
##  $ Sep : num  -0.14 -0.14 -0.14 -0.2 -0.26 -0.27 -0.23 -0.24 -0.11 -0.23 ...
##  $ Oct : num  -0.23 -0.21 -0.23 -0.1 -0.24 -0.22 -0.26 -0.34 0.03 -0.25 ...
##  $ Nov : num  -0.21 -0.18 -0.15 -0.22 -0.32 -0.22 -0.26 -0.25 0.04 -0.32 ...
##  $ Dec : num  -0.17 -0.06 -0.35 -0.1 -0.3 -0.09 -0.24 -0.32 -0.03 -0.28 ...
##  $ J.D : num  -0.16 -0.08 -0.1 -0.16 -0.28 -0.32 -0.3 -0.35 -0.16 -0.1 ...
##  $ D.N : num  NA -0.09 -0.08 -0.19 -0.26 -0.34 -0.29 -0.34 -0.18 -0.08 ...
##  $ DJF : num  NA -0.16 0.09 -0.33 -0.1 -0.4 -0.33 -0.5 -0.33 0.03 ...
##  $ MAM : num  -0.11 0.05 -0.08 -0.15 -0.36 -0.37 -0.31 -0.32 -0.27 0.06 ...
##  $ JJA : num  -0.15 -0.06 -0.14 -0.08 -0.31 -0.35 -0.26 -0.27 -0.13 -0.12 ...
##  $ SON : num  -0.19 -0.18 -0.17 -0.18 -0.28 -0.24 -0.25 -0.28 -0.01 -0.27 ...
```

You can see that all variables are formatted as numerical data, which is helpful. If you don't declare `na.strings = "***"` as you load the data, the variables with missing information would have been loaded as factor (categorical) variables.

Let's extract the Year (`Year`) and the Annual Series (`J.D`) only. Wherever you see `XXXX` it is up to you to replace these with the correct command.

```
tempdata <- tempdata %>%  select(Year,XXXX)
```

And now we will translate this into a `xts` format time-series

```
gt <- xts(tempdata$J.D, order.by=as.Date(paste0("01/07/",tempdata$Year),"%d/%m/%Y"))
```

When you use `xts` as your data format you need to specify that the data belong to a particular day not only a year. Even if you only have annual data. So here we associate the datapoints to the first of July of every year (`paste0("01/07/",tempdata$Year)`). When we translate our series `tempdata$J.D` into an `xts` formatted series we need to tell the `xts` function where the date information is (`order.by =`). In our case it is the 1st of July every year as defined above.

To understand what `paste0("01/07/",tempdata$Year)` you may want to past this bit of code into the command window and execute it. You should see that you have created strings such as "01/07/1880". We then apply the `as.Date` function to tell R that these are no ordinary strings but actuallt date information. We also hand-in the option `%d/%m/%Y"` which declares to the `as.Date` function the day/month/year ordering (important as you will often find a month/day/year ordering) and how the day, month and years are separated (here with a "/").

Greenhouse Gas (GHG)

There are a number of ways to create a variable which would allow to model the impact of greenhouse gases on global temperatures. Let's first import some estimates for **CO2 concentration in the atmosphere** from the `1_CO2-data.xlsx` file.

```
co2cdata <- read_excel("1_CO2-data.xlsx")
```

The data are sourced from the https://www.esrl.noaa.gov/gmd/ccgg/trends/data.html but for your convenience presented in the above Excel file.

The measurements are in parts per million of CO2 (abbreviated as ppm) in every million molecules of (dry) air and come from a single observatory in Hawaii https://www.esrl.noaa.gov/gmd/ccgg/about/co2_measurements.html.

Once you imported the data have a look at this data frame (`str(co2cdata)`) to understand what variables the data frame contains.

Let's look at the data. For a quick plot we use the `plot` function to first plot the `co2edata$Interpolated` series against the series index (`index(co2edata$Year)`, which basically just numbers the observations, R doesn't know yet that these are time-series data). The `lines(index(co2edata$Year),co2edata$Trend, col = "red")` command adds the trend series to this plot (which removes all the seasonality).

```
plot(index(co2cdata$Year), co2cdata$Interpolated, xlab="Observations",
     ylab="CO2 levels (ppm)",type = "l", col="blue")
lines(index(co2cdata$Year),co2cdata$Trend, col = "red")
title("Monthly CO2 concentration, and Trend")
```

We shall use the trend series as our series and define it as an annual series such that we can match it to the temperature data.

An easy way to achieve this is to use the `group_by` feature of the tidyverse combined with `mutate` which creates a new variable. The new variable should be defined as the average of the `Trend` variable in every year:

```
co2cy <- co2cdata %>% group_by(Year) %>%
          XXXX(co2cy = XXXX(Trend)) %>% #calculates annual average
```

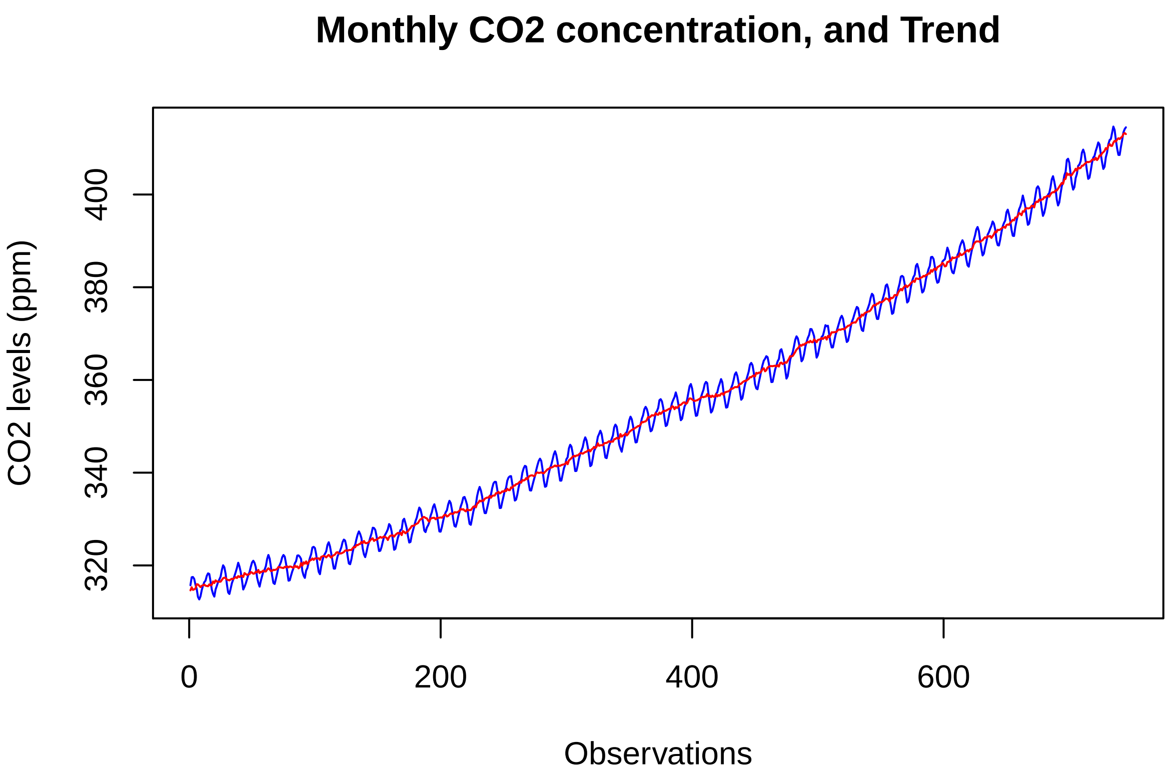


Figure 1: Figure: CO2 concentration

```
    select(Year,co2cy) %>%          # drops all but Year and co2ey
    unique()                        # only keeps one per year
```

Your new series `co2cy$co2cy` has been calculated correctly if its average value is 355.6159471.

Now we shall turn this series into a `xts` series. Complete the following line of code:

```
co2c <- xts(co2cy$co2cy, XXXX=XXXX(paste0("01/07/",XXXX),"XXXX"))
```

This measured the concentration of CO2 in the air at a particular observatory. But the series is not particularly long as it starts in 1958.

An alternative is to use actual estimates of **CO2 emissions**. We obtain these from the https://ourworldindata.org/co2-and-other-greenhouse-gas-emissions#annual-co2-emissions. Click the data tab underneath the world map.

```
co2edata <- read.csv("annual-co2-emissions-per-country.csv")
names(co2edata)
```

```
## [1] "Entity"                              "Code"
## [3] "Year"                                "Annual.COâ...emissions..tonnes."
```

The first thing you can notice that the name of the fourth variable is very long. Let's change that.

```
names(co2edata)[4] <- "co2e"
head(co2edata)
```

```
##        Entity Code Year   co2e
## 1 Afghanistan  AFG 1949  14656
## 2 Afghanistan  AFG 1950  84272
## 3 Afghanistan  AFG 1951  91600
## 4 Afghanistan  AFG 1952  91600
## 5 Afghanistan  AFG 1953 106256
## 6 Afghanistan  AFG 1954 106256
```

The data are organised such that every row represents the CO2 emissions for a country or terretory (`Entity`) in a particular year. Afghanistan is the first country in the alphabet. For this purpose we really only want the data for the entire world. Let's see whether any of the entities looks like delivering these data. Run `unique(co2edata$Entity)` to see a list of all coutries and you will find an entry `World`. Let's filter out these data.

```
co2edata <- co2edata %>%  filter(Entity == "World")
```

The global series is not very long. We could think of aggregating over all the countries, but perhaps we can find another source. After googling something like "CO2 global emissions global data history" I eventually ended up at https://cdiac.ess-dive.lbl.gov/trends/emis/tre_glob_2014.html and downloaded the comma deliminated (csv) file. I saved it as `global.1751_2014.csv`.

```
co2edata2 <- read.csv("global.1751_2014.csv",skip = 1, na.strings = "***")
names(co2edata2)
```

```
## [1] "Year"
## [2] "Total.carbon.emissions.from.fossil.fuel.consumption.and.cement.production..million.metric.tons.o
## [3] "Carbon.emissions.from.gas.fuel.consumption"
## [4] "Carbon.emissions.from.liquid.fuel.consumption"
## [5] "Carbon.emissions.from.solid.fuel.consumption"
## [6] "Carbon.emissions.from.cement.production"
## [7] "Carbon.emissions.from.gas.flaring"
## [8] "Per.capita.carbon.emissions..metric.tons.of.carbon..after.1949.only."
```

```r
summary(co2edata2$Year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1751    1817    1882    1882    1948    2014
```

We can see that the data go from 1751 to 2014. These are estimates and updates can sometimes take a few years. The names are very messy so let's simplify the one we will be using, the total emissions in column 2.

```r
names(co2edata2)[2] <- "co2e"
```

The units of measurement here are millions of metric tons (mmt) of carbon. Let's give this a quick plot

```r
plot(co2edata2$Year, co2edata2$co2e, xlab="Time",
     ylab="CO2 emission (mill metric tons)",type = "l", col="blue")
title("Annual CO2 emission")
```

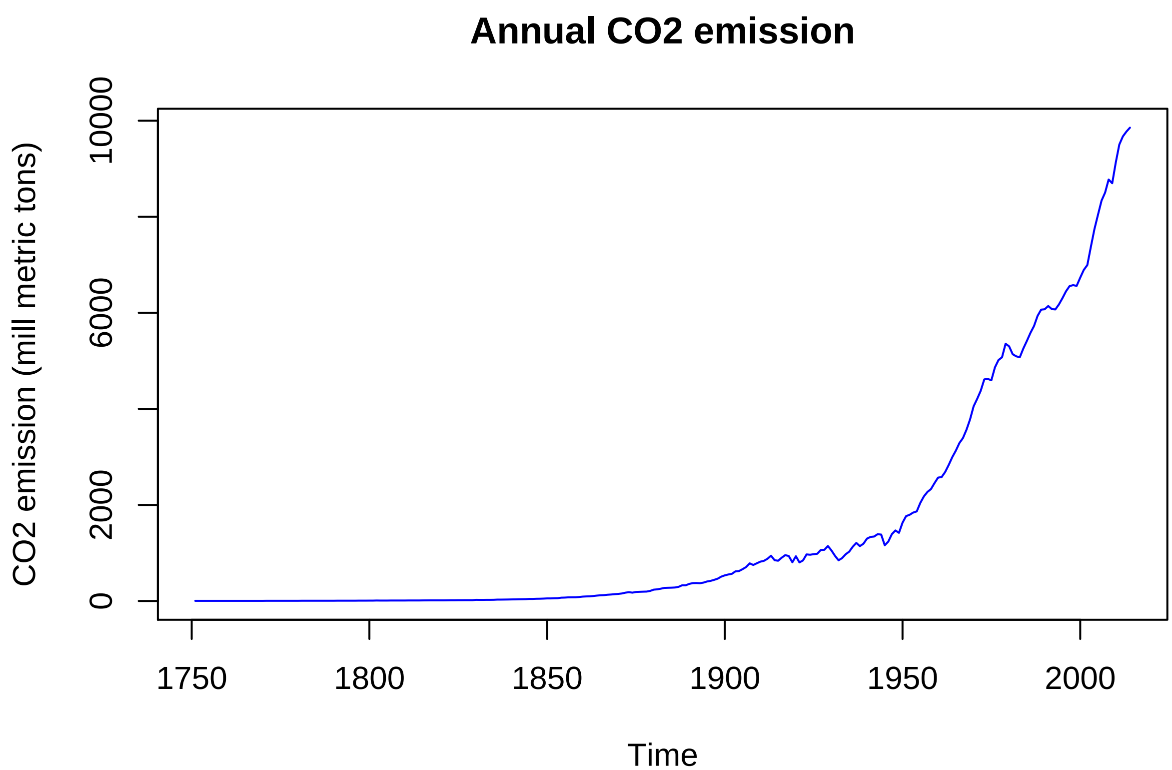


Figure 2: Figure: CO2 emission

Clearly an exponential increase. However, you can see the impact of recessions (e.g. in the 1970s and 2008/9) and perhaps also the beginning of a slowing down of emission growth at the very end of the sample period. Let's convert this to a `xts` series.

```r
co2e <- xts(co2edata2$co2e, XXXX=XXXX(XXXX("01/07/",XXXX),"XXXX"))
```

A somewhat different way of going about quantifying the effect of GHG on global temperatures is to estimate what contribution a particular GHG has made on **radiative forcing** (Definition from https://en.wikipedia.org/wiki/Radiative_forcing: Radiative forcing or climate forcing is the difference between insolation (sunlight) absorbed by the Earth and energy radiated back to space. The influences that cause changes to the Earth's climate system altering Earth's radiative equilibrium, forcing temperatures to rise or fall, are called climate forcings.)

These can be broken down by different contributing GHG and a https://data.giss.nasa.gov/modelforce/Fe_H11_1880-2011.txt is available from the Goddard Institute for Space Science (with data from 1880 to 2011).

```
co2fdata <- read.csv("Fe_H11_1880-2011.csv",na.strings = "***")
names(co2fdata)
```

```
##  [1] "Year"    "WMGHGs"  "O3"      "StrH2O"  "ReflAer" "AIE"     "BC.S"
##  [8] "nowAlb"  "StrAer"  "Solar"   "LandUse"
```

The most interesting variables here are

- `WMGHGs` - Well mixed greenhouse gases
- `Solar` - Solar Irradiance which is the contribution of the Sun

The former man-made (anthropogenic) and the later a natural cause.

```
plot(co2fdata$Year, co2fdata$WMGHGs, xlab="Time",
     ylab="Levels of radiative forcing, 1880 = 0",type = "l", col="blue")
lines(co2fdata$Year,co2fdata$Solar, col = "red")
legend(1880, 2.5, legend=c("Solar", "GHG"),
       col=c("red", "blue"), lty=1, cex=0.8)
title("Radiative Forcing")
```

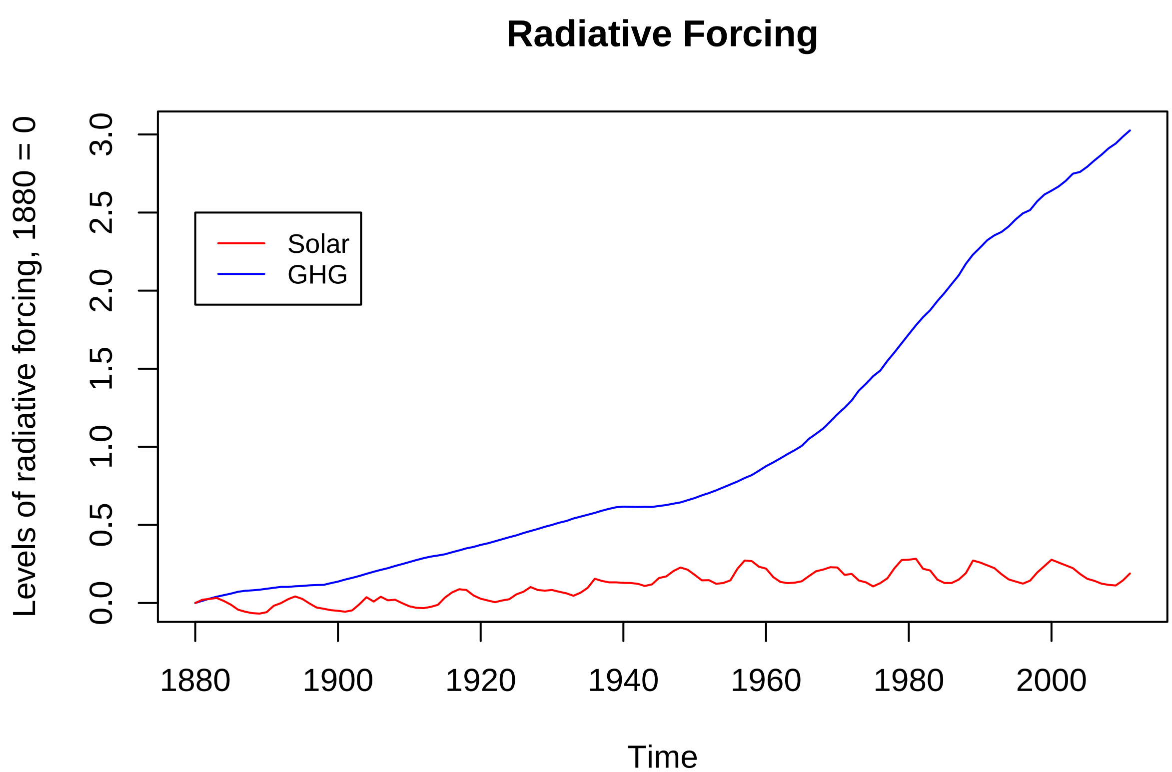


Figure 3: Figure: CO2 emissions.

Without any technical details, the data series are defined relative to the level of radiation in 1880, the first year. You can clearly see that the radiative forcing coming from the sun is basically constant with some regular, cyclical variation. GHGs, however, have significantly increased their contribution to warming.

You can see that the GHG radiative forcing is quite similar to the CO2 emissions.

Let's translate these into `xts` series.

```
Solarf <- xts(co2fdata$Solar, order.by=as.Date(paste0("01/07/",co2fdata$Year),"%d/%m/%Y"))

GHGf <- xts(co2fdata$WMGHGs, order.by=as.Date(paste0("01/07/",co2fdata$Year),"%d/%m/%Y"))
```

Merge all data into one dataframe

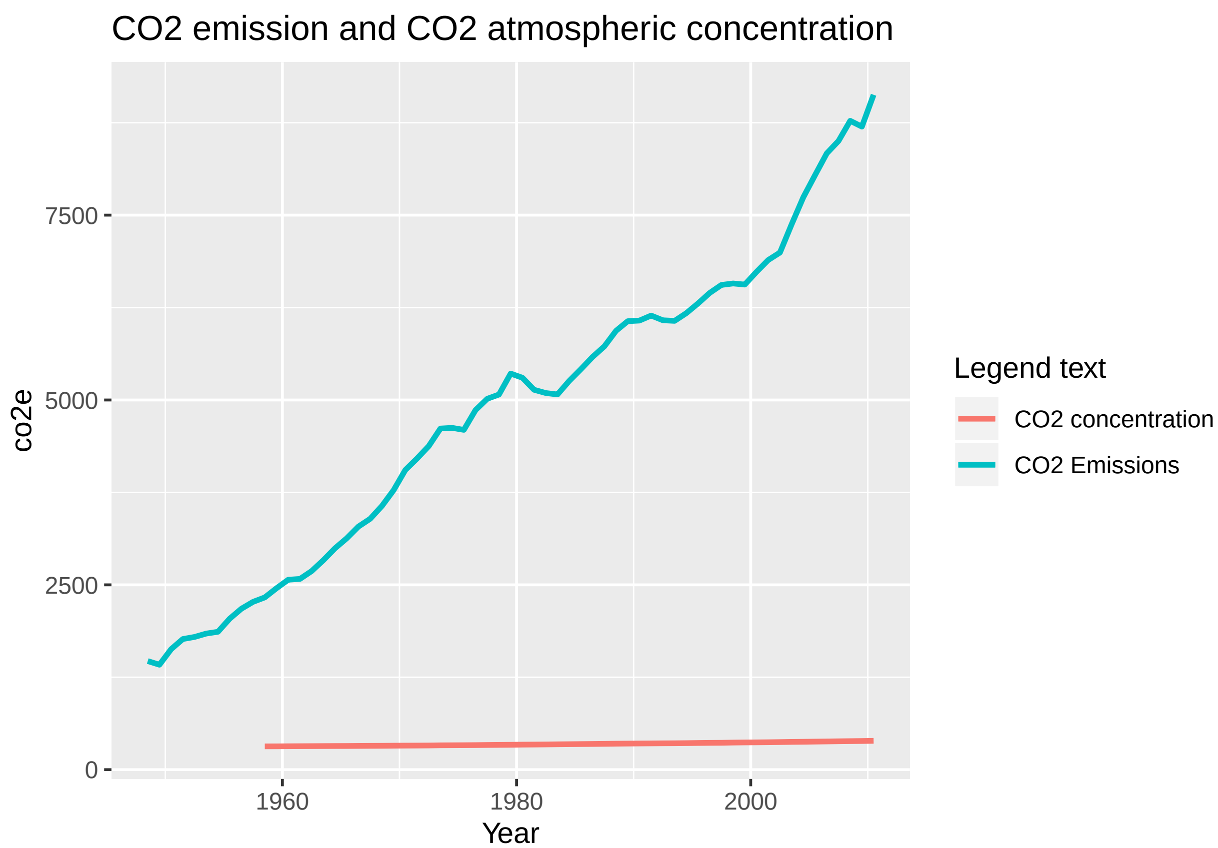
```
climate_data <- merge(gt,co2c,co2e,Solarf,GHGf)
```

Have a look at the resulting new data frame `climate_data` so that you understand its structure.

Some graphical analysis

Let's plot `co2e` and `co2c` in one graph, restricting the plot to the years 1948 to 2010 and plotting it using `ggplot` as this produces visually more pleasing results.

```
ggplot(climate_data["1948/2010"],aes(x=index(climate_data["1948/2010"]))) +
  geom_line(aes(y=co2e, color="CO2 Emissions"), size=1 ) +
  geom_line(aes(y=co2c, color="CO2 concentration"), size=1 ) +
  xlab("Year") +
  ggtitle("CO2 emission and CO2 atmospheric concentration") +
  labs(color="Legend text")
```



What we see is that the different series have very different scales and hence we only really see the variable with the largest scale, here `co2e`. We rescale (standardise) the variables before displaying them. We loose the information about the level of the data, but for comparing the data this is just fine.

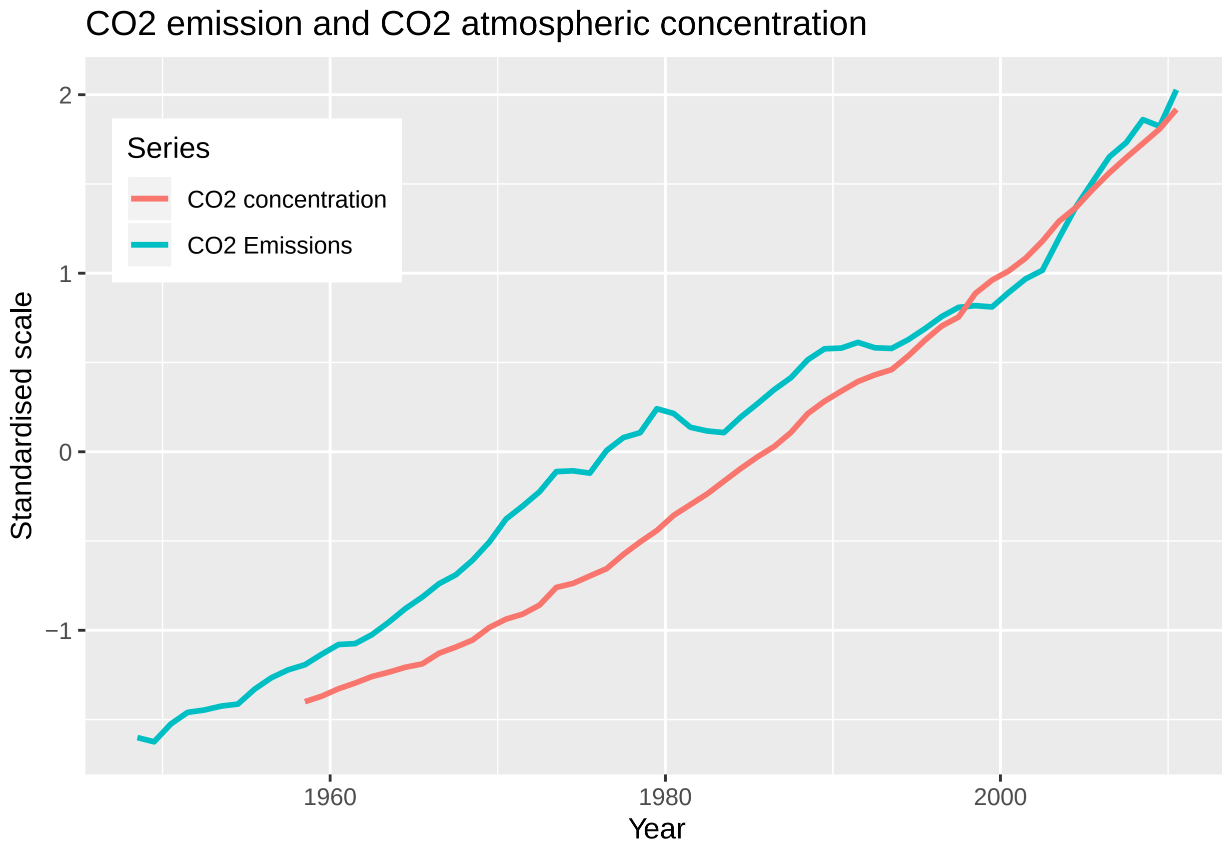
We use the `scale` function to scale the variables in our dataframe. We standardise over the sample period over which we want to print the data.

```
climate_data_s <- scale(climate_data["1948/2010"])
```

Use the help function `?scale` to figure out what that function actually does/

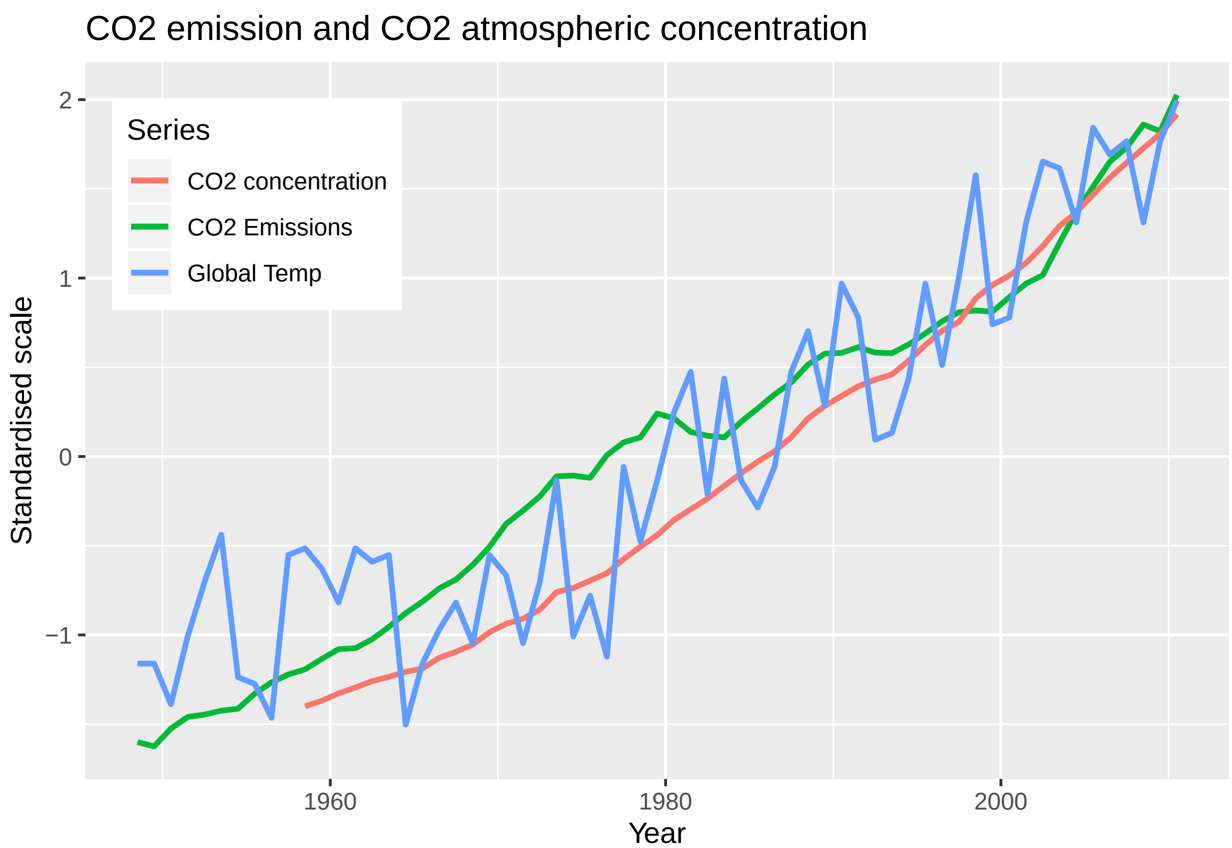
And now we replicate the same code from above:

```
p <- ggplot(climate_data_s["1948/2010"],aes(x=index(climate_data_s["1948/2010"]))) +
  geom_line(aes(y=co2e, color="CO2 Emissions"), size=1 ) +
  geom_line(aes(y=co2c, color="CO2 concentration"), size=1 ) +
  xlab("Year") +
  ylab("Standardised scale") +
  ggtitle("CO2 emission and CO2 atmospheric concentration") +
  labs(color="Series") +           # gives the legend
  theme(legend.position = c(0.15,0.8))  # determines position of legend
print(p)
```



Let's add the temperature to the graph (note that we take the graph we produced and saved before, `p`, and just add the next serious. How good is that!).

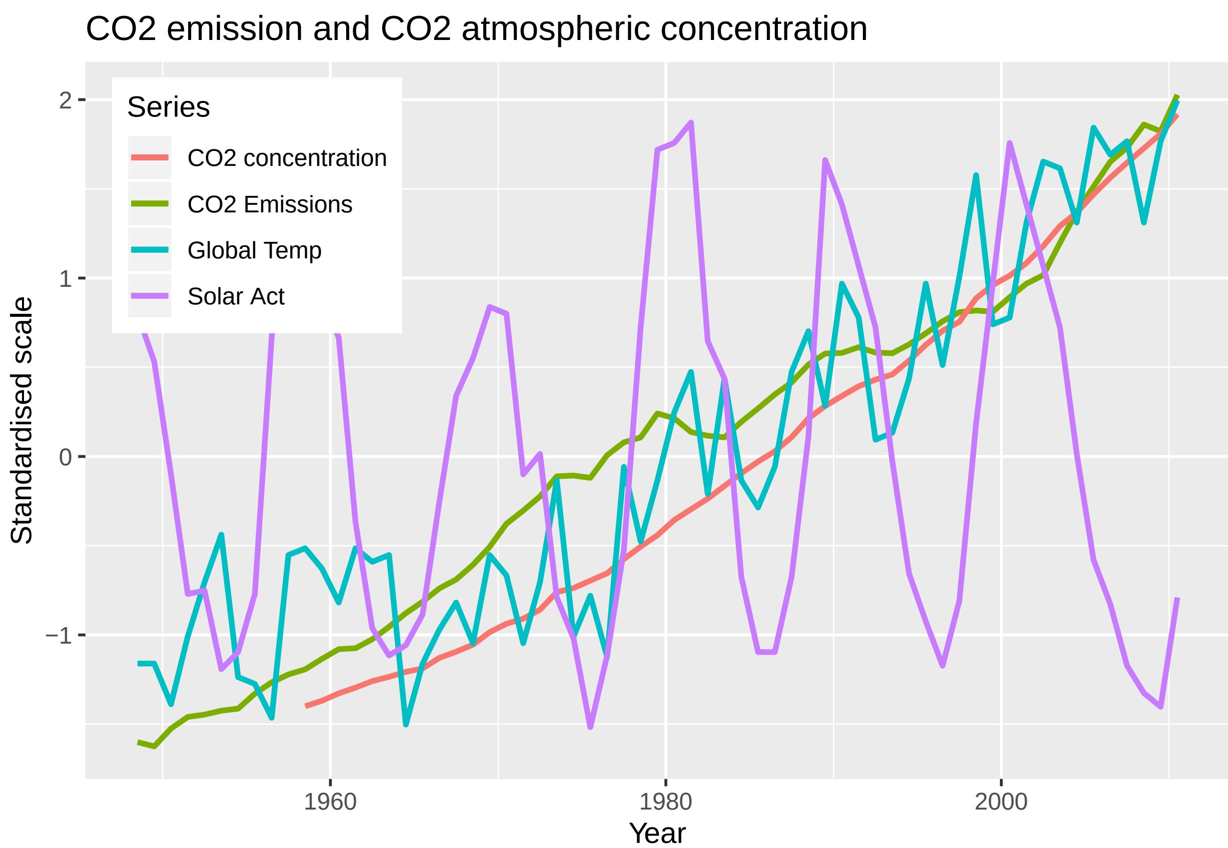
```
p <- p + geom_line(aes(y=gt, color="Global Temp"), size=1 )
print(p)
```



All you can tell from here is that all three series are trending upwards. Be careful to not use plots like this to conclude that one of the series causes the movement in another.

But sometimes graphs can help you in figuring out what is certainly not responsible for global warming. Let's add the series of solar forcing, the variable which measures solar activity.

```
p <- p + geom_line(XXXX)
print(p)
```



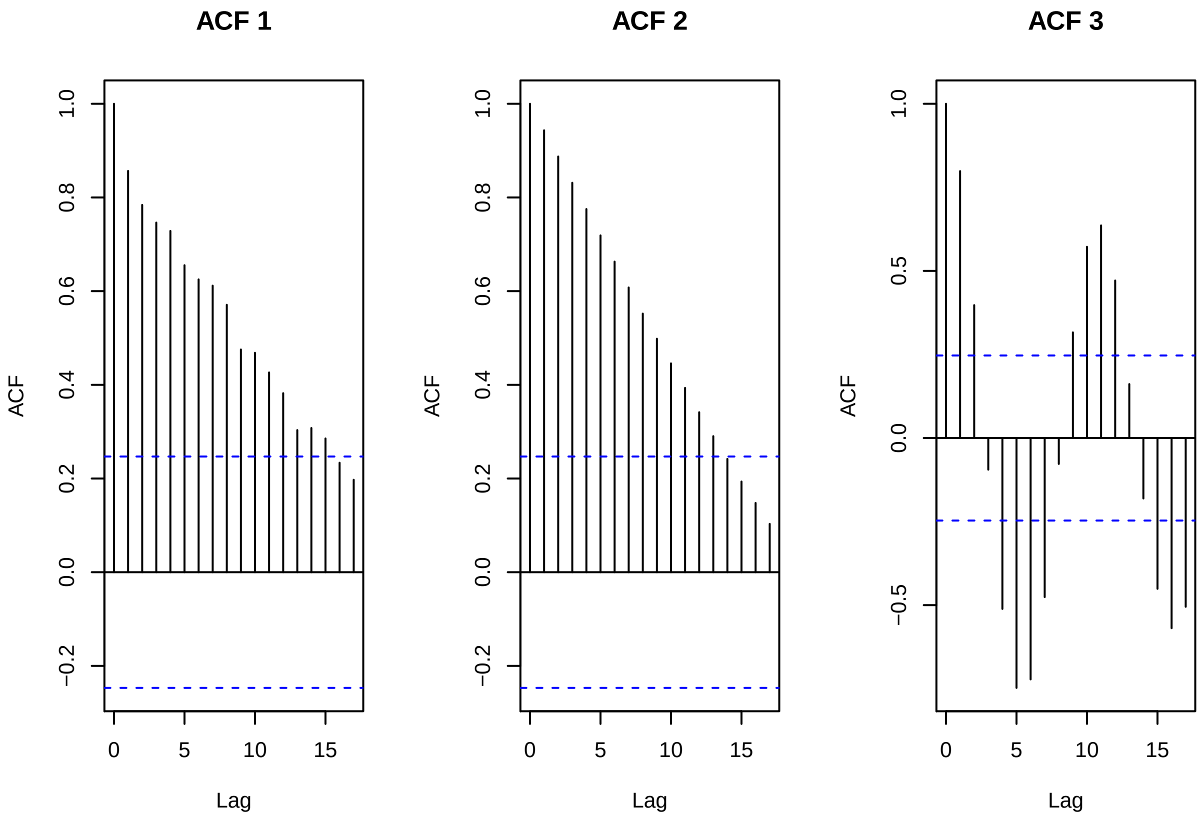
We can see clear cycles of solar acticity, but they are not trending up.

Lets look at the ACF of these series.

```
par(mfrow=c(1,3)) # this plots the next three graphs into a 1x3 array
acf(climate_data_s["1948/2010"]$gt,main = "ACF 1")
acf(climate_data_s["1948/2010"]$co2c,main = "ACF 2")
acf(climate_data_s["1948/2010"]$Solarf,main = "ACF 3")
```

When you run this code you should obtain an error message. Try and identify what the issue is and what to do about it. Googling and looking at the help function `?acf` could be helpful.

If you solve the problem the solution could look like this:



Granger causality testing

Let's run the following regression:

$$gt_t = \alpha + \beta_1 gt_{t-1} + \beta_2 gt_{t-1} + \gamma_1 co2e_{t-1} + \gamma_2 co2e_{t-2} + u_t$$

If `co2e` does not granger-cause `gt`, then we should not be able to reject the null hypoothesis $H_0 : \gamma_1 = \gamma_2 = 0$.

As our data are in the `xts` format R will understand the lag function. Let us first estimate the model above `mod_A` and then the model which would be correct if the null hypothesis was true `mod_0`:

```
mod_A <- lm(gt~lag(gt,1)+lag(gt,2)+lag(co2e,1)+lag(co2e,2),data = climate_data_s)
mod_0 <- lm(gt~lag(gt,1)+lag(gt,2),data = climate_data_s)

stargazer_HAC(mod_0, mod_A)
```

```
##
## ===================================================================
##                                    Dependent variable:
##                       ---------------------------------------------
##                                            gt
##                                 (1)                    (2)
## -------------------------------------------------------------------
## lag(gt, 1)                   0.646***               0.428***
##                               (0.094)                (0.114)
```

```
##
## lag(gt, 2)                    0.306***                 0.107
##                              (0.089)                  (0.108)
##
## lag(co2e, 1)                                           1.316*
##                                                       (0.799)
##
## lag(co2e, 2)                                           -0.875
##                                                       (0.833)
##
## Constant                       0.066                    0.004
##                               (0.054)                  (0.064)
##
## ---------------------------------------------------------------------
## Observations                     61                       61
## R2                             0.818                    0.853
## Adjusted R2                    0.811                    0.843
## Residual Std. Error     0.432 (df = 58)         0.394 (df = 56)
## F Statistic         129.936*** (df = 2; 58) 81.264*** (df = 4; 56)
## =====================================================================
## Note:                                     *p<0.1; **p<0.05; ***p<0.01
##                               Newey-West standard errors in parenthesis
```

Note that we used the `stargazer_HAC` function in order to ensure that standard errors are calculated allowing for autocorrelated error terms

We now use the `lht` function (short for lineat hypothesis testing) which comes from the `car` package which has been imported as part of the `AER` package.

```r
# lht tests linear hypotheses
# vcov = vcovHAC, allows for autocorrelated residuals
# ensure that variable names are EXACTLY as they appear in
# the regression output table!
lht(mod_A, c("lag(co2e, 1)=0","lag(co2e, 2)=0"), vcov = vcovHAC)
```

```
## Linear hypothesis test
##
## Hypothesis:
## lag(co2e,0
## lag(co2e, 2) = 0
##
## Model 1: restricted model
## Model 2: gt ~ lag(gt, 1) + lag(gt, 2) + lag(co2e, 1) + lag(co2e, 2)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df     F    Pr(>F)
## 1     58
## 2     56  2 10.38 0.0001464 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here, the

$$H_0$$

that the two lags of `co2e` are irrelevant is rejected as the p-value is very small (0.0001806).

As we discussed before these data are nonstationary and it would be important to understand that this result is not merely a result of the non-stationary nature of the data. On this occasion it seems not really appropriate to merely include a time trend. It is the (potential) time-trend which is actually the oject of interest here. If we introduced it as an exogenous series we would potentially, und unjustifiably, remove the potential for rising co2 emissions explaining this trend.

We therefore estimate the model in differences instead.

```
mod_Ad <- lm(diff(gt)~lag(diff(gt),1)+lag(diff(gt),2)+lag(diff(co2e),1)+
                lag(diff(co2e),2),data = climate_data_s)
mod_0d <- lm(XXXX~XXXX+XXXX,data = climate_data_s)

stargazer_HAC(XXXX, XXXX)
lht(mod_Ad, c("XXXX","XXXX"), vcov = XXXX)
```

```
##
## ================================================================
##                                  Dependent variable:
##                      -------------------------------------------
##                                       diff(gt)
##                              (1)                   (2)
## ----------------------------------------------------------------
## lag(diff(gt), 1)          -0.432***             -0.454***
##                            (0.100)               (0.102)
##
## lag(diff(gt), 2)          -0.285***             -0.295***
##                            (0.110)               (0.109)
##
## lag(diff(co2e), 1)                                1.013
##                                                  (1.041)
##
## lag(diff(co2e), 2)                                0.933
##                                                  (1.158)
##
## Constant                   0.089*                -0.021
##                            (0.052)               (0.078)
##
## ----------------------------------------------------------------
## Observations                 60                    60
## R2                          0.184                 0.217
## Adjusted R2                 0.155                 0.161
## Residual Std. Error    0.418 (df = 57)       0.417 (df = 55)
## F Statistic         6.411*** (df = 2; 57) 3.820*** (df = 4; 55)
## ================================================================
## Note:                                *p<0.1; **p<0.05; ***p<0.01
##                         Newey-West standard errors in parenthesis
```

```
## Linear hypothesis test
##
## Hypothesis:
## lag(diff(co2e),0
## lag(diff(co2e), 2) = 0
##
## Model 1: restricted model
## Model 2: diff(gt) ~ lag(diff(gt), 1) + lag(diff(gt), 2) + lag(diff(co2e),
##     1) + lag(diff(co2e), 2)
```

```
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F Pr(>F)
## 1     57
## 2     55  2 1.9425 0.1531
```

You can now see that the lagged changes in co2 emissions do not appear to granger cause changes in `gt`. When you look at the two estimated models you can also see that differencing the data resulted in significantly lower R^2 for these regressions. This is a very typical result and is not a concern. In contrast, it is the high R^2 in the models estimated in levels which are a concern. When regressing nonstationary data on each other high R^2 are quite common and often lead inexperienced users to attach too much importance to these results.

Here we tested whether `co2e` granger caused `gt`. We could modify the analysis in a number of directions.

1) You could use the testing procedure proposed by https://www.sciencedirect.com/science/article/abs/pii/030440769401616 as this allows you to test for Granger Causality regardless of the data being stationary or non-stationary.
2) You could extend the number of lags to 4, to acknowledge that the greenhouse effect of co2 emissions may take some time.

3) You could include the `Solarf` variable as a third variable and see whether its inclusion changes the results regarding `co2e` granger causing `gt`.