Lab 7: Climate modelling

COMP130  
Instructor: John MacCormick

In this lab, you will experiment with [SimMod](https://berkeleyearth.org/simmod-a-simple-python-based-climate-model-new/), a climate model created by Zeke Hausfather at the [Berkeley Earth](https://berkeleyearth.org/) project.

Download the zip file of lab materials and unzip them into your folder for this lab. We will refer to this location as your *lab7 folder*.

# Understanding the emissions data

Like most climate models, SimMod relies on data about emissions and atmospheric concentrations of carbon dioxide (CO2), nitrous oxide (N2O) and methane (CH4). This data is in the emissions subfolder of your lab7 folder. We will first examine these files and graph the data they contain. This will better prepare us to understand the results we obtain from running climate model simulations with SimMod.

Check out the five CSV files in the emissions folder. Open historical\_ghgs.csv in a spreadsheet program such as Excel.

As usual, create a responses document using a word processing program. Save it in your lab7 folder, then answer the following questions using the historical\_ghgs.csv file.

**Qu 1.** Please give answers in your responses document.  
(a) How does the first row differ from the remaining lines?   
(b) For what years does this file contain data?  
(c) How much carbon was emitted in 1772?   
(d) How much methane was emitted in 1783?   
(e) How much nitrous oxide was emitted in 1850?

Below is some code that plots the carbon emissions data from the historical\_ghgs.csv file. You will need to install the library matplotlib before this code can be used. Get help from the instructor or the QRA to achieve this. Most likely, you will need to use the command pip install matplotlib from a Command Prompt or Terminal window. If you’re unable to install matplotlib within a reasonable amount of time, you must use an iMac provided in our classroom for this lab. (On the iMacs, use pip3 install matplotlib from a Terminal window.)

import numpy

from matplotlib.pyplot import \*

years, emissions = numpy.loadtxt('emissions/historical\_ghgs.csv', usecols=(0, 1),

delimiter=',', skiprows=1, unpack=True)

plot(years, emissions, 'r-', label='Carbon') # Plot c vs cyear using a red line.

legend(loc='upper left') # Add a legend to the graph.

title("Historical Carbon Emissions") # Set the title for the plot.

xlabel("Year") # Label the x axis.

ylabel("Carbon Emissions (pg)") # Label the y axis.

grid(True) # Turn on the grid lines.

show() # Make the graph show up in the notebook.

**Qu 2.** Create a lab07.py file in your lab7 folder. Encapsulate the above code into a function called plot\_carbon\_emissions() and test that it displays an appropriate graph.

The code above uses several features of Python that we have not studied yet. Nevertheless, we can make use of it by building a partial understanding and then altering just the parts that we need to achieve our goals. Let’s take a careful look at the function call that extracts the emissions data from the CSV file:

years, emissions = numpy.loadtxt('emissions/historical\_ghgs.csv', usecols=(0, 1),

delimiter=',', skiprows=1, unpack=True)

Here is an explanation of that code:

* the variable years will be assigned the first column of data that is read.
* the variable emissions will be assigned the second column of data that is read.
* the function numpy.loadtxt will be called and it will read the data. Its parameters tell it that:
  + the data is to be read from the file historical\_ghgs.csv in the emissions directory.
  + the data should be read from columns 0 and 1 of the file (usecols=(0,1)).
  + the columns are separated (i.e. delimited) by commas (delimiter=',').
  + one row at the top of the file (i.e. the headers) should be skipped (skiprows=1).
  + the data should be unpacked so that:
    - the data from column 0 is assigned to years, and
    - the data from column 1 is assigned to emissions.

**Qu 2.** By copying the previous function and making one very small change, create a new function called plot\_methane\_emissions(). This function should display a graph of methane emissions rather than carbon emissions. Hint: only a single character needs to be changed in the call to numpy.loadtxt. However, you will also need to change some of the graph labels and title.

# Representative Concentration Pathways (RCPs)

In 2014, the Intergovernmental Panel on Climate Change (IPCC) created four different scenarios for future atmospheric concentrations of greenhouse gases. These scenarios are called Representative Concentration Pathways or RCPs. The four RCPs produced by the IPCC range from optimistic (the smallest increases) to pessimistic (the largest increases in greenhouse gases). These RCP scenarios are known by their numbers: 2.6, 4.5, 6.0 and 8.5. The RCP data is in appropriately named files in the emissions folder: rcp\_2.6\_data.csv, and so on.

**Qu 3.** Using your previous functions as a model, write a function plot\_carbon\_rcps(). This function should create a graph showing the projected CO2 concentration (co2\_concentration\_ppm) in the atmosphere for each of the four RCP scenarios. Your program should create a graph with four lines, one for each RCP scenario. It should include a useful legend. Hint: Use a separate call to numpy.loadtxt to read the data from each RCP file. Add a plot statement for each scenario. Adding multiple plot statements will draw all of the lines on the same graph. Guess how to change the color of the lines; alternatively, read the online documentation of the [matplotlib.pyplot.plot()](https://matplotlib.org/2.1.2/api/_as_gen/matplotlib.pyplot.plot.html) function. To make sure your plot is correct, you can compare it to the one on the WikiPedia page for [Representative Concentration Pathways](https://en.wikipedia.org/wiki/Representative_Concentration_Pathway), which shows the same information.

**Qu 4.** (Answer in responses document.) Paste in a screenshot of the graph output by your plot\_carbon\_rcps function. Based on this graph, which is the most pessimistic RCP?

Simulating temperature changes

To run a climate simulation using SimMod, we will invoke the run\_simmod function from the simmod module. This void function has the following interface:

def run\_simmod(start\_year, end\_year, rcp, CS\_param, out\_file):

""" Run a climate simulation for the specified range of years using

\* the given rcp (e.g. '2.6', '4.5', '6.0' or '8.5')

\* the climate sensitivity parameter

\* and the full path and name of the output file into which the

results are written.

"""

Here is an explanation of the parameters:

* start\_year and end\_year: These specify the range of years for which to run the simulation.
* rcp: the IPCC Representative Concentration Pathway to be used, as a string (i.e. one of '2.6', '4.5', '6.0' or '8.5').
* CS\_param: the climate sensitivity parameter, explained below.
* out\_file: the path and filename of a csv file to hold the results (e.g. 'results/rcp8.5.csv' will put the results in a file named rcp8.5.csv inside the results folder.)

The Climate Sensitivity Parameter (CSP) is an estimated value that indicates how many degrees Celsius the temperature is expected to increase if the concentration of CO2 in the atmosphere doubles. Scientists have not been able to determine a precise value for the CSP. In this lab, we will use the estimate provided by the 2013 IPCC assessment report. That report stated the CSP is likely to lie between 1.5 and 4.5 °C. The midpoint of this range is 3°C, so our initial experiments will set the CSP to 3°C.

**Qu 5.** Write a function simulate\_rcp\_2\_6(). This void function uses run\_simmod to run a simulation from 1900-2100 using rcp 2.6. Use the midpoint CSP value decribed above. The results should written to the file results/rcp2.6.csv. You will need to add an appropriate import to your lab07.py file.

**Qu 6.** (Answer in responses document.) Open the CSV results file generated by the previous question, and use it to answer the following questions.   
(a) Which column number (counting from 0) contains the year?  
(b) Which column number (counting from zero) contains the estimated temperature change, which is labeled “t\_s”?  
(c) What is the estimated temperature change for the year 2100?

**Qu 7.** Write a function plot\_rcp\_2\_6(). This function produces a graph of the projected change in temperature for the years in the previous question’s simulation. The graph should have appropriate labels and titles.

**Qu 8.** Write a function simulate\_all\_rcps(). This void function uses run\_simmod to run separate simulations from 1900-2100 for each of the four RCPs. Use the same climate sensitivity parameter as earlier, and save output in the results for using the same naming convention as earlier. After running this function, there should be four CSV files in the results folder.

**Qu 9.** Write a function plot\_all\_rcps(). This function produces a graph of the projected change in temperature for all four RCP’s, plotted on the same graph. The graph should have appropriate labels, legend, and titles.

**Qu 10.** (Answer in responses document.) Paste a screenshot of your graph from the previous question into the responses document. Write three or four sentences interpreting the graph. Feel free to be creative in deciding what to discuss. For example, you could discuss the variation in the outputs or the consequences for our society if this model is correct.

# Optional: Parameter Sensitivity

Create a new Python script named param\_sensitivity.py, in your lab7 folder. Complete the following optional activities in this script.

**Stage 1.** Refactor your code from questions 8 and 9 to remove repeated code and also generalize it. It may be helpful to define one or more new functions in doing this. To generalize the code, allow for a new parameter CS\_param, which represents the climate sensitivity parameter and can be passed to any or all of the simulations. Change the naming convention of your results files so that the climate sensitivity parameter is included in the file name.

**Stage 2.** Generate results files for the lowest and highest climate sensitivity parameters in the range mentioned above, and for most and least pessimistic RCPs. Plot these all on a single graph. Paste this graph into your responses document. Write a few sentences comparing these results to the medium climate sensitivity parameter used earlier.

**Stage 3.** For a single RCP (choose one), generate a *spaghetti plot* that explores the range of possible climate sensitivity parameters. That is, plot the results of 10 or more independent simulations, where each simulation uses a climate sensitivity parameter chosen randomly from the range mentioned above. You will need to read the Python documentation to find out how to generate a random floating-point number in a given range. Paste your spaghetti plot into your responses document and briefly comment on the results.

**Acknowledgment.** This lab was originally authored by Grant Braught. It was adapted and edited by Lev Fruchter and John MacCormick.