Lab 12 – Projected Climate Model Data: Working with Multidimensional Datasets in Python

Today you will work with historical and projected climate data in Python, which would not be possible in Excel due to the size and multidimensionality of the datasets. You will assess both local and global air temperatures, as well as how air temperatures are projected to change in the future according to an ensemble of cmip5 climate models using two RCP emission scenarios.

Python

Today you will be analyzing data using Python. You will be running the program in the Cloud through Binder so you do not need to download any software. To access the Binder, go to https://mybinder.org/v2/gh/ebake310/GEOSC107-Climate_simulation_lab/HEAD. When you first load the Binder, it may take five minutes as it prepares the Python environment. A Python notebook has already been started for you in the Binder. You will work with functions from three different Python libraries (numpy, matplotlib, and xarray) to analyze climate model output. Please carefully read all instructions and avoid typos, as typos will throw Python errors.

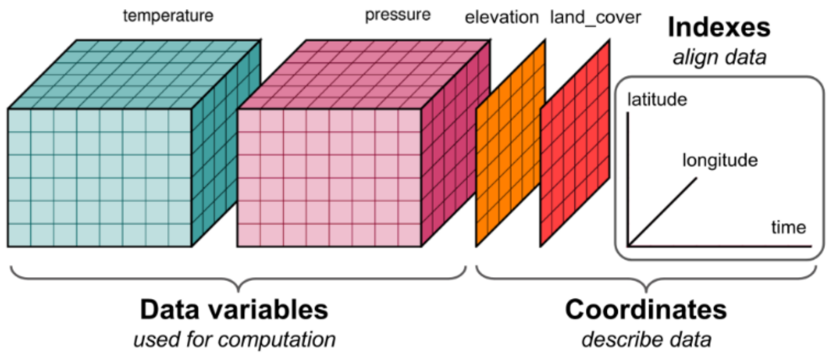
To save your Python Jupyter Notebook, click the 'Download' button. This will allow you to re-import the notebook (and all your coding progress) to the Binder in the future.

Copy each graph/map as you make it and save it to a Word document to hand in via BB.

Part I. Analyzing Baseline Climate Data (1986-2005)

You will first take a look at the annual mean of the monthly mean air temperatures at Earth's surface. The data is a historical average for the years 1986 through 2005. This baseline dataset incorporates both observed and model data to generate a comprehensive record of past climate.

1. Open the link to the Binder that is posted on Blackboard. The Binder may take a few minutes to load as it compiles its environment. An environment is the collection of tools and libraries needed for a programmer to execute their code.
2. Once the Binder has loaded, open the Jupyter notebook located in the left pane. All Jupyter notebooks have .ipynb as their file extension.
3. You will also see a folder in the left pane that contains all the climate model output files we will be using in today's lab. Climate model outputs are typically stored as NetCDF files. NetCDF (network Common Data Form) is a file format for storing multidimensional scientific data (variables) such as temperature, humidity, pressure, wind speed, and direction where each variable is stored in a grid (see figure below). You will see these files have .nc as their file extension.



4. Jupyter Notebooks are organized in cells. In a Jupyter Notebook, the primary cell types are "Code" cells, where you write executable code, and "Markdown" cells, where you write formatted text using Markdown syntax to document your analysis. You can also have "Raw" cells for specific formatting needs, allowing you to switch between cell types depending on the content you want to include in your notebook. Today you will work with "Code" cells.
5. The first cell in the Jupyter Notebook you have been provided is a "Markdown" cell, containing a brief description of the notebook. You do not need to do anything to this cell.
6. The second cell is a "Code" cell. Its first line starts with a # symbol, which allows you to add comments that will not be run by Python. Lines 2-4 contain three import commands that load the three libraries you will work with in today's lab. The as command is used subsequent to the import command to abbreviate the library same so that the full name does not need to be typed in every time we want to use a function from that library. To run this cell, use the ▶ Run button along the top pane.
7. The third cell contains a description, followed by a line of code that opens the NetCDF file that contains the baseline average air temperature data from 1986-2005. The data is stored as a variable named ds. To store the data within ds we must use and equal sign. The code to the right of the equal sign uses the xarray library to open the dataset. The file path (the location of the data file) is written as text within quotation mark. File paths always need to be text format, which means they need to be surrounded by quotes. Forward slashes separate the different folders that the file lives within and the file name comes last, followed by its file extension. Run this cell.
8. Use the next cell to view the information stored in the xarray. You will do this using the print() function. Write the name of the dataset you want to view, in this case ds, within the parentheses of the print() function and then run this cell. This should output a list of information about the baseline dataset.
9. **How are the data organized spatially?**

10. **How many data variables are in the dataset?**

11. In the next cell you need to access the annual temperature data from the xarray you created and store it as a new variable. Name the new variable temp. Therefore you code should follow the format temp = ds['name of variable you want to store from ds']. Run the cell. If you then type out the name of your new variable below that line of code and run the cell again, you will see how the data has been stored in the new variable you created.
12. Next you want to plot the temperature data on a map. Because we are using an xarray which automatically stores the data's coordinates with the data, this is simple. Type in the code:

```
plt.figure(figsize=(16,8))
temp.plot()
plt.title('Baseline Annual Mean of Monthly Air Temperatures')
plt.show()
```

13. **Describe any patterns or interesting areas you see on the map of global mean air temperatures.**

14. Baseline dataset notes: The air temperature variable contains baseline values of the annual mean of monthly air temperature means. The units are degrees Celsius (°C). In the file and variable name, tas means temperature 2 meters above the surface, which represents the temperature people feel. The temperature at or of the ground is more difficult to model given the many types of ground cover. The X Dimension and Y Dimension parameters are longitude and latitude, respectively. The cells data layer are 1 degree of latitude by 1 degree of longitude, or about 4,900 square miles (12,346 square kilometers) at the equator. The north-south extent of the individual cells is 70 miles everywhere, but the east-west extent decreases with the distance from the equator. However, when we view the map we just created, the cells widths get displayed equally, so the projection masks this spatial nuance.

15. In the next cell write the code needed to create a scatterplot of the temperature data across all latitudes for each line of longitude.
    a. First, extract all the latitude values from the xarray ds:
    ```
    lat = ds['latitude'].values
    ```
    b. Next, start your figure by setting up its dimensions:
    ```
    plt.figure(figsize=(10, 6))
    ```
    c. Next, create a numpy array containing every longitude value as integers:
    ```
    long_array = np.arange(-90, 91)
    ```
    d. Create a for loop that extracts the temperature data along every line of longitude and adds it to the plot you have started. The second and third lines of code should start with a single tab indent in order to be within the “for” loop:
```
for longitude in long_array:
    temps_x_long = ds['climatology_tas_annual_mean_of_monthly_means'].sel(longitude=longitude, method='nearest').squeeze()
    plt.plot(lat, temps_x_long,'.')
```
    e. Finally, add a title, x and y axis labels and a grid to the graph. Then display the graph.
    ```
    plt.title('Annual Mean of Monthly Air Temperatures')
    plt.xlabel('Latitude')
    plt.ylabel('Temperature ($\degree$C)')
    plt.grid(True)
    plt.show()
    ```

16. **Looking at the scatterplot you just created, how does temperature vary with latitude? Does the Arctic (North 90) or Antarctic (South -90) have colder temperatures?**

17. Next, make a histogram of global baseline temperatures.
    a. To do this, you will first have to flatten the multidimensional xarray of temperature data:
    ```
    temp_flat = temp.values.flatten()
    ```
    b. Then plot the histogram.
    ```
    plt.figure(figsize=(10, 6))
    plt.hist(temp_flat, bins=30, color='blue', edgecolor='black', alpha=0.7)
    ```
    c. Lastly, add tittles, labels, grid, and display the histogram.
    ```
    plt.title('Histogram of Average Monthly Mean Air Temperatures')
    plt.xlabel('Temperature ($\degree$C)')
    plt.ylabel('Frequency')
    plt.grid(True)
    plt.show()
    ```

18. **What is the most frequent temperature (approximately) on the histogram? Where do these temperatures occur according to the map you made?**

19. Use the functions np.min() and np.max() to calculate the minimum and maximum average temperature values. Use your variable temp. The variable name should go in the parentheses. **What are the maximum and minimum temperatures?**

The equatorial region occupies almost half of the earth's surface area, but this fact isn't obvious on your map because the WGS84 coordinate system stretches the polar regions. In fact, the area of the earth's surface represented by the uppermost row of cells in your map is smaller than the surface area represented by just one cell at the equator. If your data were based on equal-area cells, the skew in this histogram would be even stronger, with a significant majority of cells being warm. The two small spikes on the left side of the histogram are another artifact of the WGS84 coordinate system. These represent the extreme polar values, which are repeated many times when stretched onto a flat rectangle.

Part II. Compare projected climates
In the previous section, you explored baseline climate data, which is historical in nature. In this section, you'll explore climate anomaly data, or models of possible future climates.

The range of future climate change depends on the concentrations of greenhouse gases that will be emitted into the atmosphere over years and decades. Climate science organizations worldwide generate future climate models based on scenarios called representative concentration pathways (RCPs). There are four RCPs. Each represents a different level of greenhouse gases being added to the earth's atmosphere.

- RCP 2.6: Radiative forcing levels not only peak prior to but also decline to 2.6 W/m^2 (watts per square meter) by 2100. Many people no longer consider scenario 2.6 a realistic possibility.
- RCP 4.5: Radiative forcing levels stabilize without exceeding 4.5 W/m^2 by 2100. Scenario 4.5 is still considered a realistic possibility.
- RCP 6.0: Radiative forcing levels stabilize without exceeding 6.0 W/m^2 by 2100. Scenario 6.0 is also considered a realistic possibility.
- RCP 8.5: Radiative forcing levels are modeled with different assumptions than 2.6, 4.5, and 6.0. In scenario 8.5, very high levels of radiative forcing (8.5 W/m^2) are assumed due to high population growth and lower incomes in developing countries. Scenario 8.5 is an extreme scenario, but still considered possible.

Projected future climates are expressed in terms of change from baseline climate conditions. Climate scientists refer to this kind of change as an **anomaly** because it represents a deviation from normal climate conditions.

In this section, you'll combine anomaly temperature values in a future climate scenario (2040–2059) with baseline values to show projected future temperatures for RCP 4.5 and RCP 8.5.

20. Use the xr.open_dataset function again, but this time open the .nc datasets for the RCP 4.5 and RCP8.5 scenarios for the time range 2040-2059. Store these two datasets as variables named ds_45 and ds_85 respectively.

These datasets contains projected average temperatures from a 20-year (2040–2059) time frame. It is based on a model that produces daily temperature anomalies. Twenty-year averages smooth out the extreme events and are useful when exploring long-term climate trends. Real climate is not gradually and smoothly getting warmer every year in every location. The climate models account for many variables to produce daily values that behave like real weather.

21. Use the print() function again to view one of the new variables you just created.

22. **What are the three data variables stored in the dataset?**

23. Extract and store the projected temperature anomaly data for each of the two xarrays. For example for the RCP 4.5 scenario:

    ```
    tas_45 = ds_45['tas']
    ```

24. Make maps of each of the temperature anomaly datasets, similar to how you made the map of baseline air temperatures in Part I. Provide each with a title.

25. **Are air temperatures projected to be warmer everywhere on the planet?**

26. **Where do the greatest increases (anomalies) occur?**

27. **In general, will land areas or water areas have higher temperature increases?**

28. Lastly, calculate the minimum and maximum temperature anomalies for RCP 4.5 and 8.5.

29. **What is the range in projected temperature increases during this period (2040-2059) for each of the two RCP scenarios?**

Part III. Assess Future Climate in Clinton, NY

Because climate model simulation data is gridded according to latitude and longitude, you can pull out location specific data, at least at the resolution of the model grid cell. For this part, we will look at the temperature data for the grid cell in which Clinton, NY is located.

30. First, store the gps coordinates for Hamilton College 43.0484° N, 75.3785° W as two variables, named latitude and longitude. Remember, we are in the western hemisphere so to indicate this the longitude value should be made negative.

31. Next, select the data for the specified location from both the baseline dataset and for the two RCP scenario datasets. Use the code below for the baseline and RCP 4.5. You will need to adapt it for RCP 8.4.

```
clinton_base = ds['climatology_tas_annual_mean_of_monthly_means'].sel(latitude=latitude, longitude =
longitude, method='nearest').squeeze()

clinton_45 = ds_45['tas'].sel(latitude=latitude, longitude = longitude, method='nearest').squeeze()
```

32. Next, store the three temperature values you just extracted in an array. We also want to add the anomalies to the baseline data within this array so we can compare actual projected temperatures to the baseline, rather than just looking at the anomaly data. You may have to modify the below code based on how you named your RCP 8.4 dataset.

```
values = [clinton_base.values, clinton_45.values+clinton_base.values, clinton_85.values+clinton_base.values]
```

33. Next, create an array with the three data categories. We want to create an array containing the text labels ‘Baseline’, ‘RCP 4.5’ and ‘RCP 8.5’. Try to figure this out.

34. Create a bar chart using the two arrays you created in steps 33-34. The general code for madding a bar chart is `plt.bar(categories, values)`.

35. Add a title and x and y labels to the bar chart you just created.

36. **What was the baseline temperature for the Clinton area in the past?**

37. **How are temperatures projected to change by the 2040-2059 period? What is the new mean temperature predicted by the two scenarios?** You can get an exact value for these by typing out the name of the array where these values are stored and then running the cell.

Download the completed Jupyter Notebook using the Download button along the top pane.

Download the completed Jupyter Notebook containing your code and graphics using the Download button along the top pane. Turn in this completed .ipynb file on Blackboard, along with a word document containing the graphs and maps you made. You can turn in this packet with the written responses to the question in class as a paper copy.

Rubric (20 points)

6 formatted & labeled graphs/maps	1	2	3	4	5	6	7
Copy of completed Jupyter Notebook	1	2	3	4			
Responses to the 12 questions	1	2	3	4	5	6	
Organized/easy to follow	1	2	3				