Final_Project_DSC200

December 12, 2024

1 DSC 200 Final project

This project involves the analysis of a climate projection of temperature change across the Earth under a 'middle-of-the-road' future scenario in which global mean temperatures reach more than 2 degress centigrade above the pre-industrial. You will read in the data, analyze it, and visualize it in a variety of ways. You will also write a small command line interface to make the analysis more interactive.

We will be using data created by the NorESM2 climate model and processed and as part of the ClimateBench dataset, described in this paper. **All the files you will need for this project are available in the `public` folder on DataHub though.**

1.0.1 Table of contents:

1. Read in CO2 emissions data for historical + future scenario [5 points]
2. Read in temperature data [5 points]
3. Create a simple regression model [15 points]
4. Extend this to a regional temperature model, by region, and by state [15 points]
5. Plot the regression coefficients for each country [5 points]
6. Do an analysis of your choosing [15 points]
7. Make a command line interface [20 points]

1.0.2 Other requirements:

You will also be graded on Documentation and commenting, coding sytle, and code quality: - Documentation should be in the form of a Jupyter notebook, and should include a description of the data, the analysis, and the results. [10 points] - The code should be well documented, and should follow the PEP8 coding style. [5 points] - The code should be well organized, and should be broken up into functions and classes as appropriate. For full marks try to use no for-loops in your code. [5 points] - Submit your work on both datahub and gradescope.

Be sure to read the question and reach out to the instructor or TA if you have any questions.

1.0.3 Total points: 100 (30% of total), plus midterm makeup

- Note, the midterm grade is still capped at 100%

1.0.4 Deadline: Sunday December 11th 11:59pm

[76]:
```
import numpy as np
import pandas as pd
import xarray as xr
import matplotlib.pyplot as plt
```

1.1 1. Read in historical and future (estimated) CO2 emissions data

Monthly CO2 emissions data is available since 1850 globally for each industrial sector. We want to use annual average totals of the emissions for all sectors for our analysis. Since CO2 has a very long lifetime in the atmosphere (1000's of years), we can assume that the total amount of anthropogenic CO2 in the atmosphere is the cumulative sum of all emissions since 1850. This is what we will use for our analysis.

To read this data do **either** Q1a (to get 5 points plus additional makup points for the midterm) **or** Q1b (to get 5 points for this project)

[77]:
```
input_path = '~/public/Project_data/'
```

1.1.1 1a. OPTIONAL: Create interpolated cumalitve CO2 from the raw data using Pandas

To gain (up to) 50% additional marks for your midterm makeup (capped at 100%), you can create a new column in the CO2 emissions data that is the cumulative CO2 emissions.

[78]:
```
# These input files provide CO2 emissions data for the historical period␣
↪(1850-2014) and the future period (2015-2100). They should all be␣
↪concatenated into a single file.

historical_input_files = [input_path+'CO2_emissions/
↪CO2-em-anthro_input4MIPs_emissions_CMIP_CEDS-2017-05-18_gn_200001-201412.
↪csv',
                          input_path+'CO2_emissions/
↪CO2-em-anthro_input4MIPs_emissions_CMIP_CEDS-2017-05-18_gn_195001-199912.
↪csv',
                          input_path+'CO2_emissions/
↪CO2-em-anthro_input4MIPs_emissions_CMIP_CEDS-2017-05-18_gn_190001-194912.
↪csv',
                          input_path+'CO2_emissions/
↪CO2-em-anthro_input4MIPs_emissions_CMIP_CEDS-2017-05-18_gn_185101-189912.
↪csv']

future_input_file = input_path+'CO2_emissions/
↪CO2-em-anthro_input4MIPs_emissions_ScenarioMIP_IAMC-MESSAGE-GLOBIOM-ssp245-1-1_gn_201501-21
↪csv'
```

First you will need to read, concatenate and process the raw CSV files, and sum over the `sector`

and `month` columns to get an annual total.

Documentation for Q1:

First, we read the historical and future CSV files into DataFrames and combines them into one DataFrame, full_data. Then the data is grouped by year and units, summing the global_total column to obtain annual totals. The data is split into historical data (years <= 2014) and future data (years >= 2015). The future data is then reindexed to include all years between 2015 and 2101. Any missing values are interpolated using linear interpolation. Historical and interpolated future data are combined, and the global_total is converted to gigatonnes of carbon (GtC). A cumulative CO2 column is also calculated. The preprocessed data is compared with previously stored CO2 data by reading a second CSV and aligning both datasets by year. The DataFrames are combined side by side. The columns are renamed for clarity, and the difference between the cumulative CO2 values from both datasets is calculated. A threshold is defined, and if any difference between the datasets exceeds this threshold, a warning message is printed.

[79]:
```
# Concatenate historical and future files into a single DataFrame
all_input_files = historical_input_files + [future_input_file]
dataframes = [pd.read_csv(file) for file in all_input_files]
full_data = pd.concat(dataframes, ignore_index=True)

# Sum over the sector and month columns to get an annual total
annual_totals = full_data.groupby(['year', 'units'])['global_total'].sum().
 ↪reset_index()
```

Note, the future data is only provided every five years so that will need linearly intepolated to get annual values.

[80]:
```
# Separate historical and future data
historical_data = annual_totals[annual_totals['year'] <= 2014]
future_data = annual_totals[annual_totals['year'] >= 2015]

# Interpolate the future data to fill in missing years
future_data = future_data.set_index('year').reindex(range(2015, 2101),␣
 ↪method=None)
future_data['global_total'] = future_data['global_total'].
 ↪interpolate(method='linear')
future_data = future_data.reset_index()

# Combine historical and interpolated future data
preprocessed_data = pd.concat([historical_data, future_data], ignore_index=True)
```

Now, divide by 1e6 to get the units in GtC (Giga tonnes of carbon) and calculate the cumulative sum.

[81]:
```
# Convert to GtC and calculate cumulative emissions
preprocessed_data['global_total'] = preprocessed_data['global_total'] / 1e6
preprocessed_data['cumulative_CO2'] = preprocessed_data['global_total'].cumsum()
```

```
preprocessed_data = preprocessed_data.drop(columns=['units'])
```

[82]:
```
# Reading the existing file
combined_co2 = pd.read_csv(input_path+'combined_co2.csv')
```

Check the data against the existing `combined_co2.csv` and save it to use for the rest of the project.

[83]:
```
combined_co2.set_index('year', inplace=True)
preprocessed_data.set_index('year', inplace=True)

# Concatenating the two DataFrames horizontally
combined = pd.concat([combined_co2, preprocessed_data], axis=1, keys=['df1',␣
 ↪'df2'])


# Renaming columns for clarity
combined.columns = ['_'.join(col).strip() for col in combined.columns]
combined.rename(columns={
    'df1_global_total': 'cumulative_CO2_combined_co2',
    'df2_cumulative_CO2': 'cumulative_CO2_preprocessed_data'
}, inplace=True)

# Adding a comparison column
combined['difference'] = combined['cumulative_CO2_combined_co2'] -␣
 ↪combined['cumulative_CO2_preprocessed_data']

preprocessed_data.reset_index(inplace=True)
combined_co2.reset_index(inplace=True)
threshold = 0.0000001     # The threshold
if (combined['difference'] > threshold).any():
    print('Some values exceed the threshold.')
else:
    print('All values are within the threshold.')
```

```
All values are within the threshold.
```

1.1.2 1b. Otherwise just read in the cumulative CO2 emissions data from the provided file

[84]:
```
pre_processed_input_file = input_path+'cumulative_co2.csv'

# Read the input files
df=pd.read_csv(pre_processed_input_file)
```

1.2 2. Read in the temperature data

Note, this temperature change as modelled by the NorESM2 climate model relative to the pre-industrial period. It's purely driven by the prescribed emissions, so it won't perfect represent the

actual temperatures we experienced in a given year (which are subject to chaotic fluctuations), but it's a good model.

[85]:
```
temperture_input_file = input_path+'global_temparature.nc'
```

[ ]:
```

```

Documentation for Q2 and Q3:

The temperature data is loaded from a NetCDF file (global_temparature.nc) using xarray. Then we extract the temperature data (tas), latitude (lat), and longitude (lon) from the dataset. The latitude values are used to calculate weights (using cosine transformation) for area-weighted averaging of the temperature. The temperature data is weighted by latitude using xarray's weighted() method. The weighted mean temperature is then calculated over the latitude and longitude dimensions. The global mean temperature data is merged with preprocessed CO2 data which was processed earlier, based on the year column, to create a combined dataset for analysis. A linear regression model is built using cumulative CO2 emissions (X) as the input variable and global mean temperature (y) as the output.The slope and intercept of the regression line are printed, and the model is used to predict global mean temperatures based on CO2 emissions. The regression coefficients (slope and intercept) are stored in a dictionary, converted to a pandas DataFrame, and saved to a CSV file (regression_coefficients_global_mean.csv). Two plots are generated: Cumulative CO2 vs. Global Mean Temperature: A plot of actual vs. predicted global mean temperatures against cumulative CO2 emissions. Year vs. Global Mean Temperature: A plot of actual vs. predicted global mean temperatures over the years.

And take the global mean. Don't forget to calculate and include weights for the latitude of each grid cell.

[86]:
```
temp_dataset = xr.open_dataset(temperture_input_file)

# Extract the temperature variable (tas)
temperature = temp_dataset["tas"]

# Extract latitude
latitude = temp_dataset["lat"]

# Calculate weights based on latitude
weights = np.cos(np.deg2rad(latitude))

# Now, use the `weighted()` method in xarray to apply the weights to the␣
 ↪temperature data
weighted_temp = temperature.weighted(weights)

# To compute the weighted mean, use the `.mean()` method along the latitude␣
 ↪dimension
global_mean_temp = weighted_temp.mean(dim=["lat", "lon"])
```

1.3 3. Create a simple regression model of global warming

Global warming can be surprisingly well predicted just using a linear model of cumulative CO2 emissions. This is because the CO2 emissions are the primary driver of global warming, and the CO2 stays in the atmosphere for a long time (see e.g. https://www.nature.com/articles/ngeo3031).

To get global temperature as a function of cumaltive CO2. You can use the `LinearRegression` class from `sklearn.linear_model`, with documentation provided here. You should only need to use the `fit` and `predict` methods. The `fit` method takes two arguments, the first is the input data, and the second is the output data. The `predict` method takes one argument, the input data.

Alternatively, you can also use the `statsmodels` package to get more detailed statistics on the regression. See here for documentation.

Since we're only aiming to create an interpolation model, we don't need to worry too much about keeping a test set aside. We can just use all the data to train the model. You could also use a train-test split if you want to.

[87]:
```
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm

# convert the global_mean_temp into a dataframe and rename the column time to␣
 ↪year, to merge it with the preprocessed_data dataframe
global_mean_temp_df = global_mean_temp.to_dataframe(name="global_mean_temp").
 ↪reset_index()
global_mean_temp_df.rename(columns={"time":"year"}, inplace=True)

merged_data = pd.merge(preprocessed_data, global_mean_temp_df, on="year",␣
 ↪how="inner")
merged_data = merged_data.dropna()

# Extract input (X) and output (y) variables
X = merged_data["cumulative_CO2"].values.reshape(-1, 1)  # Input: cumulative CO2
y = merged_data["global_mean_temp"].values               # Output: global␣
 ↪temperature

# Create and fit the LinearRegression model
model = LinearRegression()
model.fit(X, y)

# Print regression results
print(f"Coefficient (slope): {model.coef_[0]}")
print(f"Intercept: {model.intercept_}")

# Predict temperature based on the model
predicted_temperatures = model.predict(X)
```

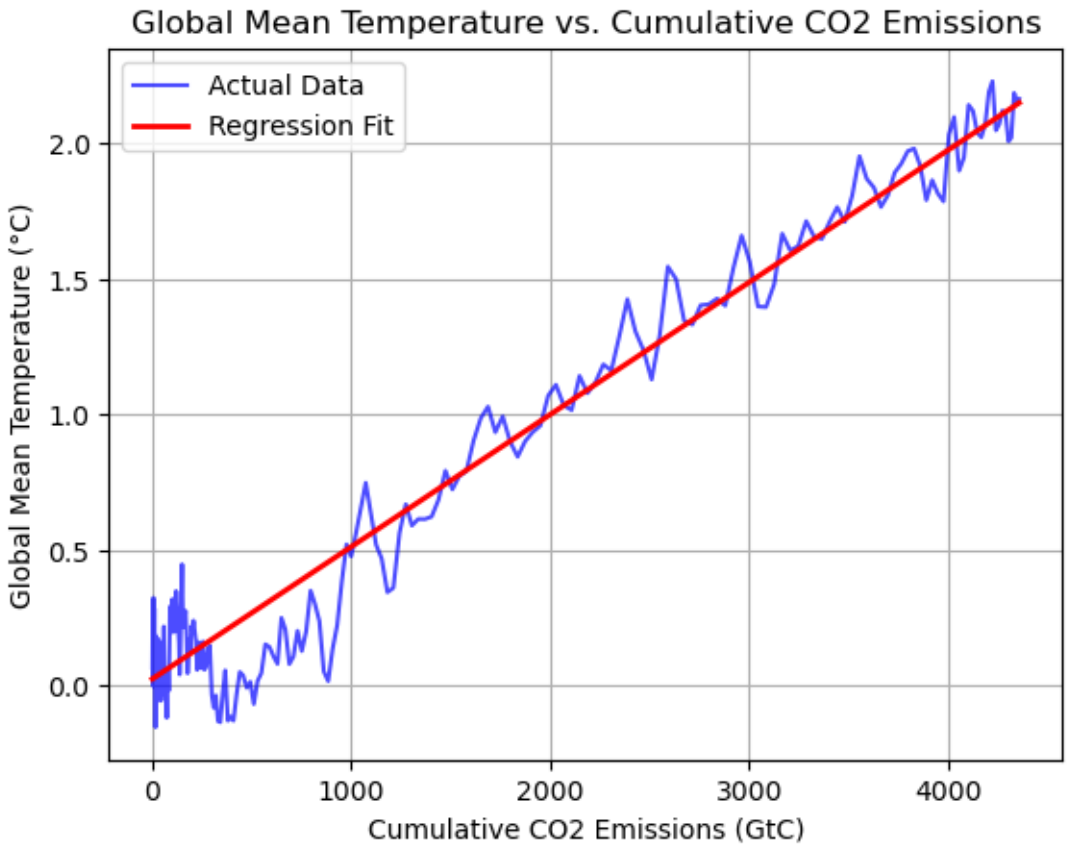
```
Coefficient (slope): 0.0004879032621412705
Intercept: 0.02554753526051723
```

[88]:
```
# Store the regression coefficients (slope and intercept) into a dictionary
regression_coefficients = {
    'slope': model.coef_[0],
    'intercept': model.intercept_
}

# Convert the dictionary to a pandas DataFrame
coefficients_df = pd.DataFrame([regression_coefficients])

# Set the file path for the CSV file
csv_file_path = '~/Project/regression_coefficients_global_mean.csv'

# Save the coefficients to a CSV file
coefficients_df.to_csv(csv_file_path, index=False)

print(f"Regression coefficients saved to {csv_file_path}")
```

```
Regression coefficients saved to
~/Project/regression_coefficients_global_mean.csv
```

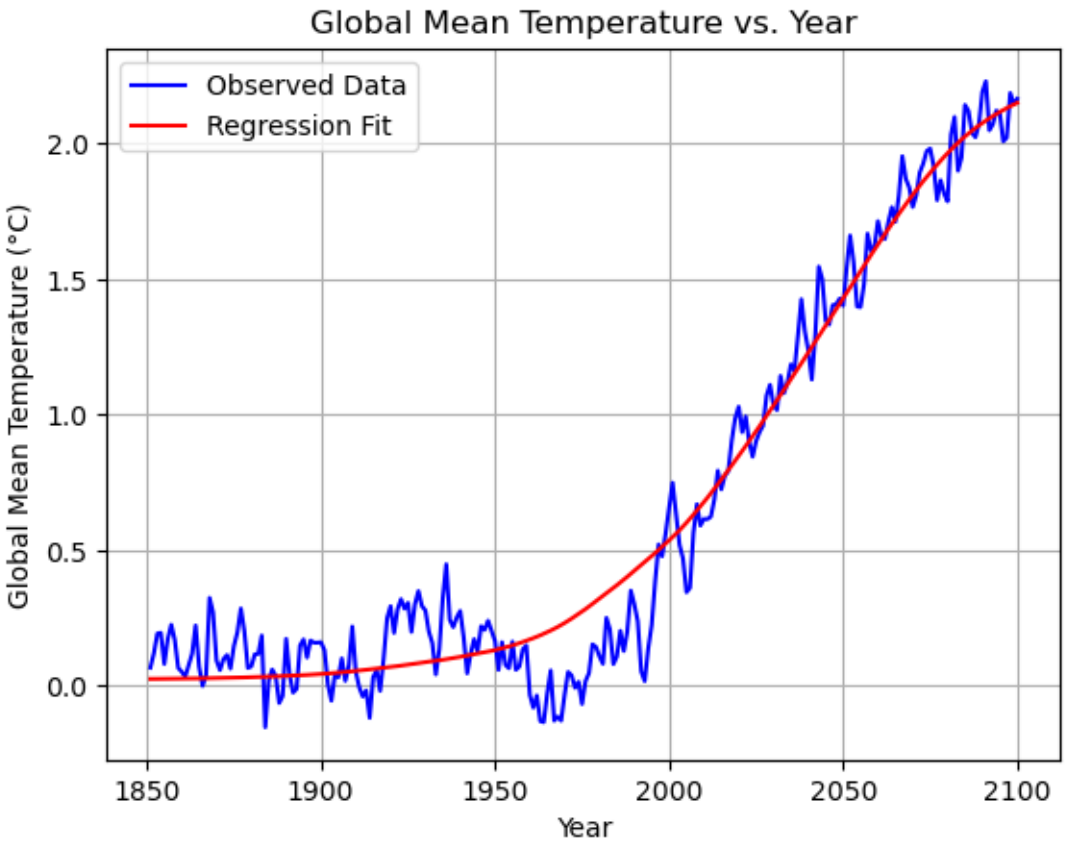
Plot gobal mean temperature as a function of cumalative CO2 emissions, along with the regression fit

[89]:
```
# Plotting the Actual data
plt.plot(merged_data["cumulative_CO2"], merged_data["global_mean_temp"],␣
 ↪color="blue", label="Actual Data", alpha=0.7)

# Plotting the Regression line
plt.plot(merged_data["cumulative_CO2"], predicted_temperatures, color="red",␣
 ↪label="Regression Fit", linewidth=2)

# Labels and title
plt.xlabel("Cumulative CO2 Emissions (GtC)")
plt.ylabel("Global Mean Temperature (°C)")
plt.title("Global Mean Temperature vs. Cumulative CO2 Emissions")
plt.legend()
plt.grid(True)

# Display the plot
plt.show()
```



Plot gobal mean temperature as a function of year, along with the regression fit

[90]:
```
#Plotting the Actualy Data
plt.plot(merged_data["year"], merged_data["global_mean_temp"], color="blue",␣
 ↪label="Observed Data")
# Plotting the Regression line
plt.plot(merged_data["year"], predicted_temperatures, color="red",␣
 ↪label=f"Regression Fit")
plt.xlabel("Year")
plt.ylabel("Global Mean Temperature (°C)")
plt.title("Global Mean Temperature vs. Year")
plt.legend()
plt.grid(True)
plt.show()
```



1.4 4. Extend this to a regional temperature model, by region, and by state

While the relationship between global temperature and cumulative CO2 emissions is very linear, the relationship between regional temperature and cumulative CO2 emissions is less so. This is because the regional temperature is affected by other factors, such as the regional distribution of land and ocean, and the regional distribution of CO2 emissions. Nevertheless, let's see how well it can do

Documentation for Q4:

The country mask data is loaded from a NetCDF file (country_mask.nc), and a 2D array representing country codes (for each latitude and longitude) is extracted. The temperature data (tas) is loaded from another NetCDF file, and latitude and longitude information is extracted. Area weights are computed using the cosine of the latitude values, and the temperature data is multiplied by these weights. The weighted temperature data is grouped by the country mask. Each group corresponds to a country, and temperature data is aggregated for each country. The mean temperature for each country is calculated by averaging the temperature data within each country group. The temperature data is renamed, and coordinates are assigned for consistency. Temperature data for the year 2023 is extracted for further analysis and visualization. A bar chart is created to visualize the warming in each country for 2023 with respect to 1850. The data is converted into a DataFrame for easier plotting. The temperature data is merged with preprocessed CO2 data on the

year column to prepare for regression analysis. A linear regression model is created to analyze the relationship between cumulative CO2 emissions and temperature for each country. The function computes the regression coefficients (slope and intercept) and R^2 (fit quality). The regression analysis is applied to each country in the merged dataset, and the results are stored in a DataFrame. Also the bar graph of R^2 values vs Country is plotted The countries are categorized based on their R^2 values into three groups: Poor Fit, Moderate Fit, and Good Fit. The regression results and categorized countries are displayed in the terminal, and the results are saved to a CSV file (regression_coefficients_by_country.csv).

Read in the country mask which is a 2D array of the same size as the temperature data, with each grid cell containing the country code of the country that grid cell is in.

[91]:
```
country_mask_file = input_path+'country_mask.nc'

# for each country and year we will have temperatures associated with it
```

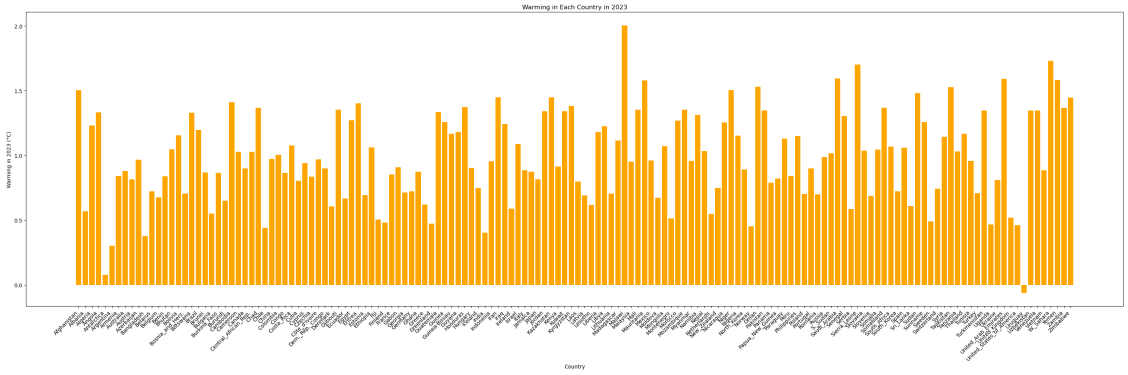
Average the spatial coordinates into countries so that you end up with a dataset that has dimensionality of the number of countries by the number of time points.

[92]:
```
country_dataset = xr.open_dataset(country_mask_file)
country_mask_array = country_dataset["__xarray_dataarray_variable__"]

# Load the temperature data and getting the weighted temperature
temp_dataset = xr.open_dataset(temperture_input_file)
temperature_data = temp_dataset["tas"]
latitude = temp_dataset["lat"]
weights = np.cos(np.deg2rad(latitude))
weighted_temp = temperature_data * weights

# Load the country mask (2D array with same lat/lon as temperature data)
country_mask = xr.DataArray(
    country_mask_array,  # 2D numpy array (country codes)
    dims=["lat", "lon"],
    coords={"lat": temp_dataset["lat"], "lon": temp_dataset["lon"]}
)

# Grouping temperature data by country and calculating the mean
country_temp = weighted_temp.groupby(country_mask)
country_mean_temp = country_temp.mean(dim="stacked_lat_lon")

# Formatting the result
country_mean_temp = country_mean_temp.rename({"__xarray_dataarray_variable__":␣
 ↪"country"})
mean_temp_by_country = country_mean_temp.
 ↪assign_coords(countries=country_mean_temp.coords["country"])
mean_temp_by_country.name = "tas"    # named so that to.dataframe() can be used
```

Plot a bar chart of the warming in each country in 2023. Note, the temperature data is baselined

to 1850.

[93]:
```
# Extract data for 2023
data_2023 = mean_temp_by_country.sel(time=2023)
# Convert to DataFrame for easy plotting and plotting the bar chart for warming␣
 ↪in each country
warming_df = data_2023.to_dataframe().reset_index()

# Plot the bar chart
plt.figure(figsize=(30, 10))
plt.bar(warming_df['country'], warming_df['tas'], color='orange')
plt.xlabel('Country')
plt.ylabel('Warming in 2023 (°C)')
plt.title('Warming in Each Country in 2023')
plt.xticks(rotation=45, ha='right')  # Rotate x labels for better readability
plt.tight_layout()
plt.show()
```

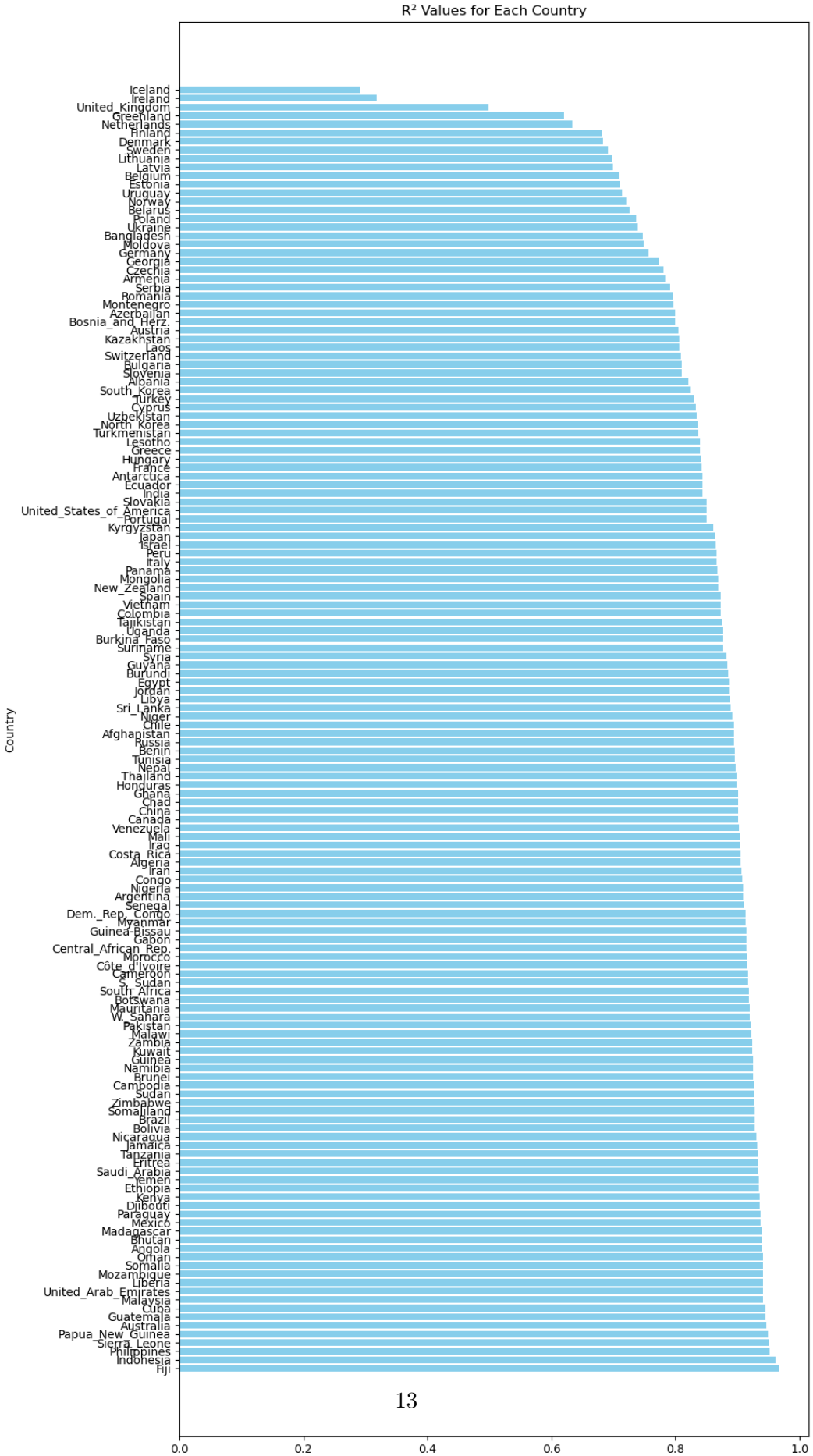


Calculate a linear regression model for each country along with the R^2 value. Plot the R^2 values for each country as a bar chart.

[94]:
```
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from sklearn.metrics import r2_score

# Converting mean_temp_by_country to a dataframe and formatting it to merge it␣
 ↪with the preprocessed_data dataframe
mean_temp_by_country_df = mean_temp_by_country.to_dataframe().reset_index()
mean_temp_by_country_df.rename(columns={"time":"year"}, inplace=True)
mean_temp_by_country_df = mean_temp_by_country_df.drop(columns='countries')
mean_temp_by_country_df.set_index('year', inplace=True)

# Merge dataframes on 'year'
merged_df = pd.merge(mean_temp_by_country_df, preprocessed_data, on='year')
```

```python
# Define a function for linear regression analysis
def regression_analysis(group):
    X = group[['cumulative_CO2']].values  # Input variable
    y = group['tas'].values  # Output variable

    # Fit linear regression model
    model = LinearRegression()
    model.fit(X, y)

    # Predict and calculate R²
    y_pred = model.predict(X)
    r2 = r2_score(y, y_pred)

    # Return coefficients and R²
    return pd.Series({
        'slope': model.coef_[0],
        'intercept': model.intercept_,
        'r_squared': r2
    })

# Apply the regression function to each group (country)
reg_coeff_by_country_df = merged_df.groupby('country').
 ↪apply(regression_analysis).reset_index()

r_squared_df = reg_coeff_by_country_df.sort_values('r_squared', ascending=False)

# Plotting
plt.figure(figsize=(10, 18))
plt.barh(r_squared_df['country'], r_squared_df['r_squared'], color='skyblue')
plt.xlabel('R² Value')
plt.ylabel('Country')
plt.title('R² Values for Each Country')
plt.tight_layout()

# Show the plot
plt.show()
```

```
/tmp/ipykernel_344/1114008950.py:35: DeprecationWarning: DataFrameGroupBy.apply
operated on the grouping columns. This behavior is deprecated, and in a future
version of pandas the grouping columns will be excluded from the operation.
Either pass `include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
  reg_coeff_by_country_df =
merged_df.groupby('country').apply(regression_analysis).reset_index()
```


R² Values for Each Country
Country
R² Value
0.0
0.2
0.4
0.6
0.8
1.0
Iceland
Ireland
United_Kingdom
Greenland
Netherlands
Finland
Denmark
Sweden
Lithuania
Latvia
Belgium
Estonia
Uruguay
Norway
Belarus
Poland
Ukraine
Bangladesh
Moldova
Germany
Georgia
Czechia
Armenia
Serbia
Romania
Montenegro
Azerbaijan
Bosnia_and_Herz.
Austria
Kazakhstan
Laos
Switzerland
Bulgaria
Slovenia
Albania
South_Korea
Turkey
Cyprus
Uzbekistan
North_Korea
Turkmenistan
Lesotho
Greece
Hungary
France
Antarctica
Ecuador
India
Slovakia
United_States_of_America
Portugal
Kyrgyzstan
Japan
Israel
Peru
Italy
Panama
Mongolia
New_Zealand
Spain
Vietnam
Colombia
Tajikistan
Uganda
Burkina_Faso
Suriname
Syria
Guyana
Burundi
Egypt
Jordan
Libya
Sri_Lanka
Niger
Chile
Afghanistan
Russia
Benin
Tunisia
Nepal
Thailand
Honduras
Ghana
Chad
China
Canada
Venezuela
Mali
Iraq
Costa_Rica
Algeria
Iran
Congo
Nigeria
Argentina
Senegal
Dem._Rep._Congo
Myanmar
Guinea-Bissau
Gabon
Central_African_Rep.
Morocco
Côte_d'Ivoire
Cameroon
S._Sudan
South_Africa
Botswana
Mauritania
W._Sahara
Pakistan
Malawi
Zambia
Kuwait
Guinea
Namibia
Brunei
Cambodia
Sudan
Zimbabwe
Somaliland
Brazil
Bolivia
Nicaragua
Jamaica
Tanzania
Eritrea
Saudi_Arabia
Yemen
Ethiopia
Kenya
Djibouti
Paraguay
Mexico
Madagascar
Bhutan
Angola
Oman
Somalia
Mozambique
Liberia
United_Arab_Emirates
Malaysia
Cuba
Guatemala
Australia
Papua_New_Guinea
Sierra_Leone
Philippines
Indohesia
Fiji

[95]:
```
# Set the file path for the CSV file
csv_file_path = '~/Project/regression_coefficients_by_country.csv'

# Save the 'reg_coeff_by_country_df' to a CSV file
reg_coeff_by_country_df.to_csv(csv_file_path, index=False)

print(f"Regression coefficients saved to {csv_file_path}")
```

```
Regression coefficients saved to
~/Project/regression_coefficients_by_country.csv
```

For which countries does the linear assumption work well, and where does it work less well?

[96]:
```
# R^2 values can be used for this

# Categorize countries based on R² values
reg_coeff_by_country_df['fit_quality'] = pd.cut(
    reg_coeff_by_country_df['r_squared'],
    bins=[-float('inf'), 0.5, 0.8, 1.0],
    labels=['Poor Fit', 'Moderate Fit', 'Good Fit']
)

# Display the categorized results
good_fit = reg_coeff_by_country_df[reg_coeff_by_country_df['fit_quality'] ==␣
 ↪'Good Fit']
moderate_fit = reg_coeff_by_country_df[reg_coeff_by_country_df['fit_quality']␣
 ↪== 'Moderate Fit']
poor_fit = reg_coeff_by_country_df[reg_coeff_by_country_df['fit_quality'] ==␣
 ↪'Poor Fit']

print("Countries where the linear assumption works well (Good Fit):")
print(good_fit)

print("\nCountries where the linear assumption works moderately well (Moderate␣
 ↪Fit):")
print(moderate_fit)

print("\nCountries where the linear assumption works poorly (Poor Fit):")
print(poor_fit)
```

```
Countries where the linear assumption works well (Good Fit):
         country     slope  intercept  r_squared fit_quality
0    Afghanistan  0.000664  -0.128632   0.894410    Good Fit
1        Albania  0.000544  -0.242526   0.821150    Good Fit
2        Algeria  0.000752  -0.129951   0.905652    Good Fit
3         Angola  0.000662   0.071806   0.940125    Good Fit
```

```
4     Antarctica  0.000068   0.018985   0.843456    Good Fit
..           …       …         …         …          …
145      Vietnam  0.000568  -0.021839   0.872669    Good Fit
146    W._Sahara  0.000769  -0.082974   0.920035    Good Fit
147        Yemen  0.000779   0.021986   0.933891    Good Fit
148       Zambia  0.000698   0.070643   0.924077    Good Fit
149     Zimbabwe  0.000708   0.025865   0.927023    Good Fit

[122 rows x 5 columns]

Countries where the linear assumption works moderately well (Moderate Fit):
              country     slope  intercept  r_squared   fit_quality
6             Armenia  0.000524  -0.263851   0.782960  Moderate Fit
9          Azerbaijan  0.000489  -0.215549   0.798815  Moderate Fit
10         Bangladesh  0.000324  -0.014025   0.747544  Moderate Fit
11            Belarus  0.000403  -0.104709   0.725605  Moderate Fit
12            Belgium  0.000307  -0.119574   0.707912  Moderate Fit
16   Bosnia_and_Herz.  0.000538  -0.199753   0.799671  Moderate Fit
35            Czechia  0.000409  -0.079320   0.780939  Moderate Fit
38            Denmark  0.000245  -0.040243   0.683638  Moderate Fit
43            Estonia  0.000355  -0.072439   0.709658  Moderate Fit
46            Finland  0.000274  -0.049191   0.682064  Moderate Fit
49            Georgia  0.000525  -0.256314   0.772189  Moderate Fit
50            Germany  0.000339  -0.092386   0.756828  Moderate Fit
53          Greenland  0.000146   0.026806   0.619755  Moderate Fit
76             Latvia  0.000329  -0.060038   0.699635  Moderate Fit
80          Lithuania  0.000349  -0.071344   0.698337  Moderate Fit
87            Moldova  0.000487  -0.207422   0.748581  Moderate Fit
89         Montenegro  0.000546  -0.247847   0.797150  Moderate Fit
95        Netherlands  0.000215  -0.064389   0.633525  Moderate Fit
101            Norway  0.000178   0.010854   0.720395  Moderate Fit
109            Poland  0.000362  -0.085483   0.735963  Moderate Fit
111           Romania  0.000536  -0.204569   0.795476  Moderate Fit
116            Serbia  0.000533  -0.238814   0.790758  Moderate Fit
128            Sweden  0.000226  -0.035351   0.691322  Moderate Fit
138           Ukraine  0.000479  -0.201937   0.739386  Moderate Fit
142           Uruguay  0.000267   0.068317   0.713985  Moderate Fit

Countries where the linear assumption works poorly (Poor Fit):
            country     slope  intercept  r_squared fit_quality
60          Iceland  0.000158   0.050206   0.291359    Poor Fit
65          Ireland  0.000088   0.006962   0.317873    Poor Fit
140  United_Kingdom  0.000127  -0.015192   0.498131    Poor Fit
```

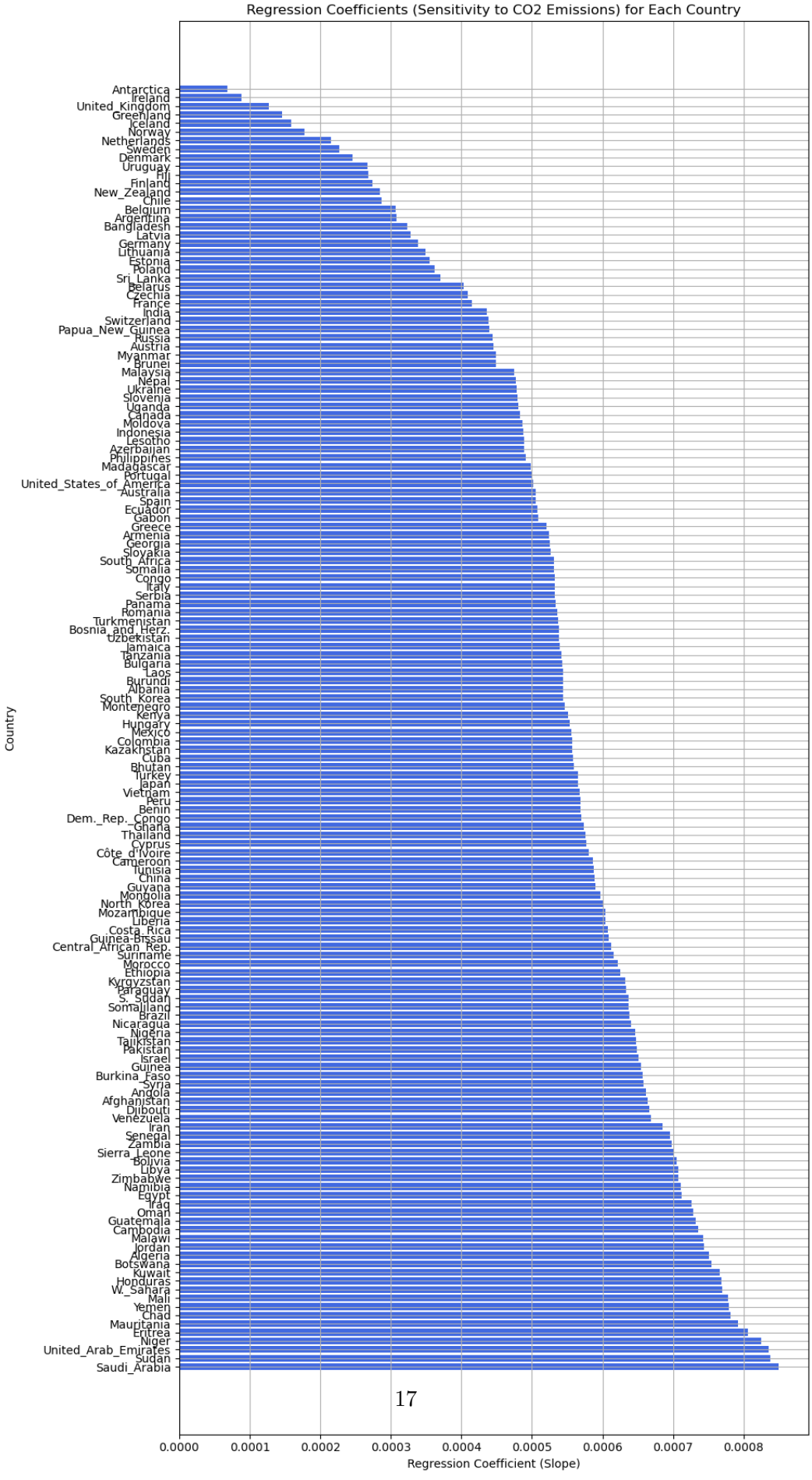
1.5 5. Plot the regression coefficients for each country

Which five countries are most sensitive to CO2 emissions and hence warming the fastest?

Documentation for Q5

The provided code is designed to analyze and visualize the regression coefficients (slopes) for each country, which represent their sensitivity to CO2 emissions. The code identifies and displays the top 5 countries most sensitive to CO2 emissions (i.e., those with the highest positive regression coefficients), and then generates a plot to visualize the regression coefficients for all countries.

[97]:
```
# Sort the countries by slope (coefficient) in descending order
sorted_results = reg_coeff_by_country_df.sort_values(by='slope',␣
 ↪ascending=False)

# Select the top 5 countries most sensitive to CO2 emissions (highest positive␣
 ↪slope)
top_5_sensitive_countries = sorted_results.head(5)

# Plot the regression coefficients for each country
plt.figure(figsize=(10, 18))
plt.barh(sorted_results['country'], sorted_results['slope'], color='royalblue')
plt.xlabel('Regression Coefficient (Slope)')
plt.ylabel('Country')
plt.title('Regression Coefficients (Sensitivity to CO2 Emissions) for Each␣
 ↪Country')
plt.grid(True)
plt.tight_layout()
# Display the plot
plt.show()

# Display the top 5 countries most sensitive to CO2 emissions
print("Top 5 countries most sensitive to CO2 emissions (fastest warming):")
print(top_5_sensitive_countries[['country', 'slope']])
```


Regression Coefficients (Sensitivity to CO2 Emissions) for Each Country
Country
Antarctica
Ireland
United_Kingdom
Greenland
Iceland
Norway
Netherlands
Sweden
Denmark
Uruguay
Fiji
Finland
New_Zealand
Chile
Belgium
Argentina
Bangladesh
Latvia
Germany
Lithuania
Estonia
Poland
Sri_Lanka
Belarus
Czechia
France
India
Switzerland
Papua_New_Guinea
Russia
Austria
Myanmar
Brunei
Malaysia
Nepal
Ukraine
Slovenia
Uganda
Canada
Moldova
Indonesia
Lesotho
Azerbaijan
Philippines
Madagascar
Portugal
United_States_of_America
Australia
Spain
Ecuador
Gabon
Greece
Armenia
Georgia
Slovakia
South_Africa
Somalia
Congo
Italy
Serbia
Panama
Romania
Turkmenistan
Bosnia_and_Herz.
Uzbekistan
Jamaica
Tanzania
Bulgaria
Laos
Burundi
Albania
South_Korea
Montenegro
Kenya
Hungary
Mexico
Colombia
Kazakhstan
Cuba
Bhutan
Turkey
Japan
Vietnam
Peru
Benin
Dem._Rep._Congo
Ghana
Thailand
Cyprus
Côte_d'Ivoire
Cameroon
Tunisia
China
Guyana
Mongolia
North_Korea
Mozambique
Liberia
Costa_Rica
Guinea-Bissau
Central_African_Rep.
Suriname
Morocco
Ethiopia
Kyrgyzstan
Paraguay
S._Sudan
Somaliland
Brazil
Nicaragua
Nigeria
Tajikistan
Pakistan
Israel
Guinea
Burkina_Faso
Syria
Angola
Afghanistan
Djibouti
Venezuela
Iran
Senegal
Zambia
Sierra_Leone
Bolivia
Libya
Zimbabwe
Namibia
Egypt
Iraq
Oman
Guatemala
Cambodia
Malawi
Jordan
Algeria
Botswana
Kuwait
Honduras
W._Sahara
Mali
Yemen
Chad
Mauritania
Eritrea
Niger
United_Arab_Emirates
Sudan
Saudi_Arabia
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008
Regression Coefficient (Slope)

```
Top 5 countries most sensitive to CO2 emissions (fastest warming):
                  country     slope
114          Saudi_Arabia  0.000850
126                 Sudan  0.000837
139  United_Arab_Emirates  0.000835
98                  Niger  0.000825
42                Eritrea  0.000807
```

1.6 6. Do an analysis of your choosing

Maybe dig into the changes in one particular country, or look at changes in the variability of temperature. Perhaps look at the chances of exceeding certian temperature limits.

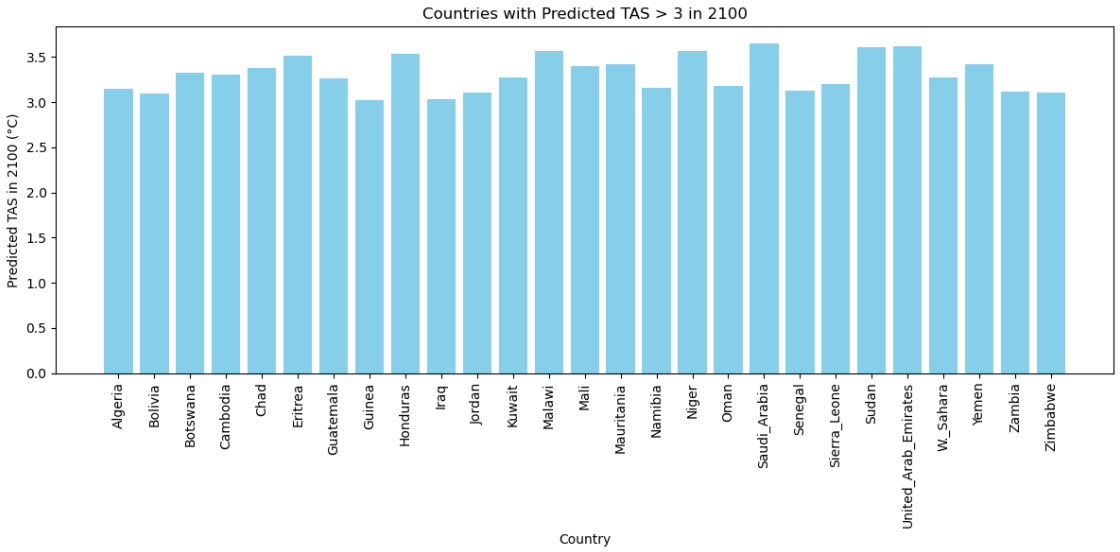
Documentation for Q6

The code is designed to predict the mean temperature anomaly (TAS) for each country in the year 2100, based on a linear regression model that relates cumulative CO2 emissions to global mean temperature for each country. The code identifies the countries that are predicted to have a mean temperature greater than 3°C in 2100 and generates a bar plot visualizing the predicted temperature for those countries.

[98]:
```
# Which countries will have a mean teamperature of more than 3°C

# Extract the cumulative CO2 for 2100 from preprocessed_data
cumulative_CO2_2100 = preprocessed_data[preprocessed_data['year'] ==␣
 ↪2100]['cumulative_CO2'].values[0]

# Define a function to calculate the predicted tas for 2100
def predict_tas_for_2100(row):
    # Use the regression equation: tas = slope * cumulative_CO2 + intercept
    predicted_tas = row['slope'] * cumulative_CO2_2100 + row['intercept']
    return predicted_tas

# Apply the function to each country in reg_coeff_by_country_df to get the␣
 ↪predicted tas for 2100
reg_coeff_by_country_df['predicted_tas_2100'] = reg_coeff_by_country_df.
 ↪apply(predict_tas_for_2100, axis=1)

# Find the countries where predicted tas is greater than 3 in 2100
countries_above_3_2100 =␣
 ↪reg_coeff_by_country_df[reg_coeff_by_country_df['predicted_tas_2100'] > 3]

# Display the countries with predicted tas greater than 3
print(countries_above_3_2100[['country', 'predicted_tas_2100']])

plt.figure(figsize=(12, 6))
```

```
# Create a bar plot where the x-axis is the countries and y-axis is the␣
 ↪predicted tas_2100
plt.bar(countries_above_3_2100['country'],␣
 ↪countries_above_3_2100['predicted_tas_2100'], color='skyblue')

# Add labels and title
plt.xlabel('Country')
plt.ylabel('Predicted TAS in 2100 (°C)')
plt.title('Countries with Predicted TAS > 3 in 2100')

# Rotate x-axis labels for better visibility if necessary
plt.xticks(rotation=90)

# Display the plot
plt.tight_layout()
plt.show()
```

	country	predicted_tas_2100
2	Algeria	3.144049
15	Bolivia	3.099122
17	Botswana	3.326899
23	Cambodia	3.305878
27	Chad	3.382170
42	Eritrea	3.515069
54	Guatemala	3.258350
55	Guinea	3.017482
58	Honduras	3.531850
64	Iraq	3.030513
70	Jordan	3.102995
73	Kuwait	3.268856
82	Malawi	3.567113
84	Mali	3.398198
85	Mauritania	3.422555
93	Namibia	3.157962
98	Niger	3.569257
102	Oman	3.182668
114	Saudi_Arabia	3.651824
115	Senegal	3.131125
117	Sierra_Leone	3.200230
126	Sudan	3.611627
139	United_Arab_Emirates	3.614342
146	W._Sahara	3.269046
147	Yemen	3.416471
148	Zambia	3.110806
149	Zimbabwe	3.109554



[ ]:

1.7 7. Make a command line interface to a prediction script

The inputs should include the name of a country (or global mean) and CO2 concentration(s). It should return the predicted temperature change relative to 1850. You can use the `argparse` package to do this. See here for documentation. Be sure to check for valid inputs.

Also provide the option to save the predictions to a CSV file.

This script should use the regression coefficients learned in the previous step so it doesn't have to use the full model output each time. You could store them in a numpy file, a pandas CSV file, or even JSON.

Documentation for Q7

A Python file is uploaded to the same location as this Jupyter Notebook. The script accepts two types of input:

"global_mean" and a float value for CO2 concentration, which returns the predicted Global Mean Temperature. "country" and a float value for CO2 concentration, which returns the predicted Mean Temperature for that specific country.

[99]:
```
! python3 Final_projectQ7.py --country "Afghanistan" 30
```

```
Predicted temperature: -0.1087118044
```

[100]:
```
! python3 Final_projectQ7.py --global_mean 24
```

```
Predicted temperature: 0.0372572136
```

[101]:
```
! python3 Final_projectQ7.py --country "Invalid_Country" 23
```

```
No data found for country: Invalid_Country
```

[102]:
```
! python3 Final_projectQ7.py --global_mean 20 --save predicted_global_mean.csv
```

```
Predicted temperature: 0.0353056005
```

[ ]:
```

```

[ ]:
```

```