CSC110 Project Proposal: Comparison of Real Temperature Values to Predicted Temperature Values for Different Representative Concentration Pathways

Can Yildiz, Alamgir Khan, Eren Findik, Giancarlo Sass Ramos

Friday, November 6, 2020

Part 2: Problem Description and Research Question

The natural atmospheric greenhouse effect is what makes life on earth possible. However, due to human activities—mainly the burning of fossil fuels—global emission of greenhouse gases (such as carbon dioxide, methane, etc.) has increased significantly over the past few years, and as more of these gases become trapped in the Earth's atmosphere, the average temperature of the world rises. This is because greenhouse gases are transparent to visible light but opaque to infrared radiation, so sunlight can travel through them but heat is blocked. Climate scientists observe this effect (among others) by collecting data (such as temperature, precipitation, etc.) and construct models to predict future temperatures for various Representative Concentration Pathways (abbreviated as RCPs). RCPs make predictions of the concentration of greenhouse gases in the atmosphere by capturing the effect of human activities and future trends on greenhouse gas emissions. The four RCPs range from very high (RCP 8.5) through to very low (RCP 2.6) future concentrations.

One such climate model is the BCCAQv2. This was developed by the Pacific Climate Impacts Consortium and aims to rectify the bias in daily precipitation series that is found in other popular climate models. This creates data that more accurately matches historical records. Essentially, this preserves the precipitation change signal while downscaling daily climate predictions relating to temperature and precipitation. This model is a mathematical equation attempting to estimate the climate trend through interactions within the atmosphere. In order to develop well-structured and long-term plans to deal with climate change, it is crucial that these estimates be as accurate as possible. For this reason, we have decided to test the accuracy of this model by comparing actual values of temperature recorded in Toronto against those predicted by the BCCAQv2 climate model, for various RCPs:**how do real temperature values compare to those predicted by the BCCAQv2 climate model?**

Part 3: Dataset Description

All datasets are in csv format

Datasets 1-4 are sourced from Environment and Climate Change Canada

Datasets 5-8 are sourced from Climate Data Canada

Description of Datasets 1, 2, 3, 4 (toronto_actual.csv, halifax_actual.csv, quebec_actual.csv, winnipeg_actual.csv) in their respective order:

A collection of the monthly summaries of the averages and extremes of temperature for the years 2003-2019 in Toronto City (Ontario), Caribou Point (Halifax), L'assomption (Quebec), and Baldur (Winnipeg) respectively.

Columns accessed: LOCAL_YEAR, LOCAL_MONTH, MEAN_TEMPERATURE

Description of Datasets 5, 6, 7, 8 (toronto_predicted.csv, halifax_predicted.csv, quebec_predicted.csv, winnipeg_predicted.csv) in their respective order:

A collection of predicted annual mean temperature (along with a range of low and high mean temperature) calculated using the BCCAQv2 climate model of each RCP (2.6, 4.5, and 8.6) for the years 2003 to 2019 (and onwards) for Ontario, Halifax, Quebec, and Winnipeg respectively

Columns accessed: DATE, RCP 2.6 MEDIAN, RCP 4.5 MEDIAN, RCP 8.5 MEDIAN

Part 4: Computational Overview

Explanation of all defined functions within our code:

Function read_actual_data():

Whilst the file (input parameter) is open:

next(reader) skips the dataset header.

The first loop appends the monthly mean temperature to a list called monthly_temps and assigns the year for the according data station. If the year is not in years list it appends the year to years list.

The second for loop calculates the mean temperature for each year and adds it into a dictionary called yearly_dict with the corresponding year.

yearly_dict is returned.

Function read_predicted_data():

Whilst the file (input parameter) is open:

years is assigned as a list of all the years that we have real temperature values for.

next(reader) skips the dataset header.

the first for loop skips each row in the dataset until it arrives at the year corresponding to the first element of the list years.

the second for loop appends the temperature of various RCPs to its corresponding list.

For each year in the list years:

a dictionary mapping rcp_dict is assigned its various RCP temperatures for that year.

rcp_dict is returned.

Function make_low_rcp_list():

For each year we read data from:

the temperature value corresponding to RCP 2.6 is is appended to a list low_rcp_list.

low_rcp_list is returned.

Functions make_median_rcp_list() and make_rcp_list():

Implemented in a very similar manner to function make_low_rcp_list.

Function calc_low_actual_pd():

actual_temps_list is assigned to a list of all the actual annual temperatures we calculated.

For each element in a list our list of predicted low RCP temperatures:

    The percentage difference between the actual and predicted temperature is calculated.

    This value is appended to a list low_rcp_pd.

low_rcp_pd is returned.

Functions calc_median_actual_pd() and calculate_high_actual_pd():

Implemented in a very similar manner to function calc_low_actual_pd.

Function plot_temp_data():

x is assigned to a list of all the years we calculated average temperatures for

Three separate lists (low_predicted_y, median_predicted_y, high_predicted_y) are assigned the values corresponding to the various RCP levels.

actual_y is assigned to a list of actual annual temperatures we calculated.

The plotting space is created.

A scatter-line graph is traced for each four of these list (containing the y-coordinates) to their corresponding year on the x-axis and is appropriately assigned a title.

The final figure is shown.

Function draw_table():

A table containing data corresponding to actual temperature, various RCP temperature values and the percentage difference between these is created.

Function draw_map():

An empty map of the provinces of Canada is imported.

A new image is created which has 2 times the width of the map, and extra height to leave space for the title.

For every city we defined, the relevant province is painted in the color that corresponds to the average temperature/RCP value of the city in the year the user typed in by using temp_to_rgb function.

The actual and predicted temperature maps are displayed side by side.

The relevant title are added to the maps *Function rcp_to_slice():*

The index value corresponding to the RCP value is returned (the temperatures corresponding to that RCP value are stored in the order of those indexes). *Function temp_to_rgb():*

The rgb value corresponding to the temperature is returned using a table to represent different color values for different temperatures (this table is included in 'screenshots.docx')

Explanation of how our program reports the result:

To draw the two coloured maps (one that represents the real temperatures and one that represents the predicted temperatures for various RCPs for all the provinces), the user is prompted to input the year and the predicted RCP value for which they would like to see the maps for. Additionally, to plot the scatter-line graph (which displays the temperature values for various RCPs and the actual temperature for a given time period) and draw the table showing the values from the graph for a city of their choice, the user is asked to input a city name. Each input has its own prompt to explain to the user exactly what is required of them.

Once the user has input all three of these, the run() method is called. The data pertaining to the various cities available for the user to select (i.e. the file names which contain real/predicted, the coordinates of the city on the map, and the name of the city) is stored as constants. The data for each city is processed using the helper functions defined in the computing_data.py file within a for loop. If the name of the city the user inputted is the same as the one being iterated in the for loop, a scatter-line graph (showing the various RCPs temperatures using unique colors plotted against their corresponding year) and a table showing the data values of the graph for that city are displayed. The user can hover over each scatter plot to see the exact temperature value for the various years and they can even hide/show any of the plotted lines in the graph.

The data necessary for the year inputted by the user is stored in the CITY_TEMPS dictionary, where the city name is mapped to a list containing actual, RCP 2.6, RCP 4.5, and RCP 8.5 temperature values respectively for that year. Next, the draw_map() function uses this dictionary to create the map: it first takes an empty map of the provinces of Canada and colors the provinces according to their capital city's actual temperatures, and then creates another map (map2) and colors the provinces according to their capital city's predicted temperatures at the RCP value the user inputted. Finally it merges these two maps side by side and adds their relevant titles.

Eventually, 1 scatter-line graph and 1 table is created for the city the user input, and a map of Canada is created for the RCP value and year typed in. The user can continue creating these visuals as much as they want - this is made possible with a while loop. In order to break out of this while loop, the user should type in 2 wrong values/names for the given prompt.

New libraries used

Our program implements the 'line and scatter plot' (used in the function plot_temp_data()) and also a 'basic table' (used in the function draw_table()) using plotly. Both of these make it possible to display a line-scatter graph and the table as we proposed in the proposal.

Aside from plotly, the new library we use is Pillow (PIL). This library allows us to manipulate images. With its subclass ImageDraw, we can add text onto the image, and with ImageFont, we can choose the font of that text. However, the most crucial method ImageDraw has is the floodfill() function, which allowed us to fill in all the areas of that given province with color.

Part 5: Instructions on How to Obtain Data Sets and Running our Program

Download all five of 'main.py', 'computing_data.py', 'reading_data.py', 'canada_map2.jpg', 'screenshots.docx' from our submitted files

Save all of these under a folder called 'Map Project'.

Inside 'Map Project' create another folder called 'datasets'.

Move all 8 datasets (toronto_actual.csv, halifax_actual.csv, quebec_actual.csv, winnipeg_actual.csv, toronto_predicted.csv, halifax_predicted.csv, quebec_predicted.csv, winnipeg_predicted.csv) mentioned in Dataset Description (also explained below) into the folder 'datasets'.

Install all the libraries listed under requirements.txt

You will need to be a library called Pillow. Open the command prompt and write 'pip install Pillow' and press enter. Wait before the library to install and close the command prompt. For reference, the website is https://pillow.readthedocs.io/en/stable/

Downloading the datasets

Datasets 1-4 all need to be downloaded from https://climate-change.canada.ca/climate-data/#/monthly-climate-summaries. For all four of these datasets, set the 'Start Date' to 2003-01, the end date to '2019-12', and the 'Data

Download Format’ to CSV.

For Dataset 1, write ‘Toronto City’ in ‘Station Table’ and select the first option that comes up. Click ‘Retrieve download links and rename the downloaded file to ‘toronto_actual.csv’

For Dataset 2, write ‘Caribou Point’ in ‘Station Table’ and select the first option that comes up. Click ‘Retrieve download links and rename the downloaded file to ‘halifax_actual.csv’

For Dataset 3, write ‘L’assomption’ in ‘Station Table’ and select the first option that comes up. Click ‘Retrieve download links and rename the downloaded file to ‘quebec_actual.csv’

For Dataset 4, write ‘Baldur’ in ‘Station Table’ and select the first option that comes up. Click ‘Retrieve download links and rename the downloaded file to ‘winnipeg_actual.csv’

Datasets 5-8 all need to be downloaded from https://climatedata.ca/download/. For all four of these datasets, select ‘Annual’ under ‘Select a frequency’, select ‘Mean Temperature’ under ‘Select a variable’, and select ‘CSV’ under ‘Select a format’.

For Dataset 5, search for ‘Toronto’ under ‘Select a location’ and select the first option that comes up. Click once on the map where the text ‘Toronto’ is written. Under ‘Save as’ write ‘toronto_predicted’. Click ‘process’ and then click ‘click here to download your data’

For Dataset 6, search for Halifax’ under ‘Select a location’ and select the third option that comes up. Click once on the map where the text ‘Halifax’ is written. Under ‘Save as’ write ‘halifax_predicted’. Click ‘process’ and then click ‘click here to download your data’

For Dataset 7, search for ‘L’assomption’ under ‘Select a location’ and select the first option that comes up. Click once on the map where the text ‘L’assomption’ is written. Under ‘Save as’ write ‘quebec_predicted’. Click ‘process’ and then click ‘click here to download your data’

For Dataset 8, search for ‘Baldur’ under ‘Select a location’ and select the first option that comes up. Click once on the map where the text ‘Baldur’ is written. Under ‘Save as’ write ‘winnipeg_predicted’. Click ‘process’ and then click ‘click here to download your data’

All of these datasets have been pre-processed. For Datasets 1-4 you will need to remove the following columns: x, y, LATITUDE, LONGITUDE, CLIMATE_IDENTIFIER, ID, LOCAL_DATE, LAST_UPDATED, ENG_PROVINCE_NAME, FRE_PROVINCE_NAME, NORMAL_MEAN_TEMPERATURE, DAYS_WITH_VALID_MIN_TEMP, DAYS_WITH_VALID_MAX_TEMP, NORMAL_PRECIPITATION TOTAL_PRECIPITATION, DAYS_WITH_VALID_PRECIP, DAYS_WITH_PRECIP_GE_1MM, NORMAL_SNOWFALL, TOTAL_SNOWFALL, DAYS_WITH_VALID_SNOWFALL, SNOW_ON_GROUND_LAST_DAY, NORMAL_SUNSHINE, BRIGHT_SUNSHINE, DAYS_WITH_VALID_SUNSHINE, COOLING_DEGREE_DAYS, HEATING_DEGREE_DAYS. For Datasets 5-8 you will need to remove the following columns: Latitude, Longitude

We have submitted these pre-processed datasets using https://send.utoronto.ca/

Please use these processed datasets as the code won’t work on the datasets downloaded from the websites i.e. download these pre-processed datasets (they are appropriately named already), save them in a folder called ‘datasets’, save this folder inside ‘Map Project’ and follow the rest of the instructions to run our program

Running our program

Open Pycharm

Click on ‘File’, then click ‘Open’, then locate where you saved ‘Map Project’ and select it and then press ‘OK’ to open it.

Once the folder is opened, Mark ‘Map Project’ as Sources Root.

Open the ‘main.py’ file in Pycharm.

Run the file in the Python Console.

You will be prompted to answer 3 questions one-by-one before the final output is displayed.

First, enter the year for which you would like to see the colored maps comparing real vs predicted temperature values (e.g. your input is 2003).

Second, enter the name of the city for which you would like to see a scatter-line graph and table representing the various RCP temperature predictions compared to the real temperatures values (e.g. your input is toronto).

Third, enter an RCP value for the comparison being made in the colored map of all the provinces of Canada (e.g. your input is RCP 4.5).

You should for 3 figures to open up: the scatter-line graph and table for the city you entered and also a colored map of the provinces of Canada.

The scatter-line graph plots years on the x-axis and on the y-axis has multiple plotted plots corresponding to the various RCPs and also the actual temperature value.

The table depicts the data from the graph but also shows the percentage differences between the RCP predicted temperature and the real temperature for all the years.

The colored map shows a comparison of real temperature values and the predicted temperature values of the RCP you input in the form of color differences.

This process can be repeated as many times as you want (for all the different years/citines) until you provide 2 incorrect inputs to a question.

You can interact with the scatter-line graph by hovering over the plotted points to see their individual precise values but you can select the option to 'Compare data on hover' to compare all plotted lines together. You can also hide/unhide lines of the graph by clicking on the key on the right.

We have provided example screenshots of how the figures might look once the code is executed in 'screenshots.docx'

Part 6: Changes Between Proposal and Final Submission

Our initial proposal was to plot one scatter-line graph depicting the various predicted RCP temperature values compared to the real temperature values recorded for Toronto and draw a table to represent this data. Our final submission does this, but it is also capable of asking the user for an input and displaying the graph and table for Quebec, Halifax, and Winnipeg as well depending on which city the user input.

Another minor change is in regards to how we read one of the dataset 'types' (Datasets 5-8). Instead of converting to the data into a Dict[datetime.date, Dict[str, List[float]]], our function instead saves the data into a Dict[int, Dict[str, float]] to simplify our code.

Lastly, as an extra challenge (depending on whether we finished the initial scatter-line graph and table), we also proposed to create two colored maps of Canada: one that represents the real temperatures for all the provinces and one that represents the predicted temperatures for various RCPs with the colors signifying the temperature value. However, in the end, we were only to implement this map for four provinces in Canada (Ontario, Quebec, Manitoba, and Nova Scotia) using some major assumptions due to limitations regarding the datasets (as explained in the discussion section).

Part 7: Discussion Section

The temperatures predicted by the BCCAQv2 climate model do not accurately predict the temperatures actually predicted in the cities tested. The actual annual temperatures oscillate wildly which causes the percentage difference between predicted RCP temperatures and actual temperatures to be very inconsistent. This is as expected because predicted RCP temperatures cannot predict annual differences in temperature. Something all graphs have in common for the various cities, is that temperature anomalies are observed for the same years. For example, the average temperatures for 2006, 2012 and 2016 are very high compared to the preceding and succeeding years(for all cities). Additionally, the average temperatures for 2004, 2009, 2014 and 2019 are very low when compared to preceding and succeeding years(except 2019, as there is no data for succeeding years). Therefore, the percent difference between RCP predicted temperatures and actual temperatures is not that important in determining the accuracy of the model. What is important is the consistency of the percent difference between RCP predicted temperatures by the BCCAv2 and actual temperatures. In Quebec and Halifax, a gradual increase in temperature is evident from the graph. In both cities the overall increase in temperature is more similar to RCP values of 2.6 and 4.5 than an RCP value of 8.5. However, in Toronto and Winnipeg, a gradual increase in temperature is not evident from the graph. This inaccuracy is demonstrated by the percentage differences between predicted and actual temperatures in the tables. The values for Quebec and Halifax are gradually increasing despite some extreme values. Likewise, the values for Toronto and Winnipeg are gradually decreasing disregarding some extreme values. Therefore the climate model is not very accurate because it only accurately predicts temperature increase for certain cities.

The major limitation we faced was in regards to the datasets we were able to find, which then affected the implementation of our algorithms. Our program consists of two 'types' of datasets: Datasets 1-4 (as mentioned in the Dataset Description) consist of real monthly temperature values for different weather stations in various cities (and thus, cities) in Canada and Datasets 5-8 (as mentioned in the Dataset Description) consist of predicted annual mean temperatures for different locations within Canada. Datasets 5-8 are simple to work with as they already contain annual values but Datasets 1-4 need their data to be computed in order to calculate the annual temperature using the monthly temperatures. However, these datasets are repeatedly missing monthly temperature values. For example, the weather station in Baldur, Winnipeg, may have monthly readings for the months of January to October for the year 2002 but may be missing November and December. This makes the calculation of the annual mean temperature impossible as we need the data for all twelve months to calculate the mean for that year, and thus, makes this year not usable for the purposes of our program. Most weather stations/locations have had this issue for many years and so finding a station that has consistent data values is very challenging, and it becomes even more challenging when we have to keep the years we are computing on consistent throughout all the cities for our map. For this reason, we have had to make oversimplifying assumptions by manipulating the datasets and reducing them to just one weather station that represents the whole province in our map/graph for a small sample number of years (only 2003-2019).

Additionally, it is a fact that temperatures are gradually rising, yet some Canadian cities don't depict this through their annual temperatures. This is because the dataset only shows annual temperatures from 2003 to 2019. This is a relatively small sample size, considering an overall rise in temperature is very slow. As a result, annual temperature anomalies such as an annual average temperature of 8.29o in Toronto, 2014 - heavily impact the results.

There are many paths for further exploration. Firstly, it would be interesting to compare the results of different climate models using the same cities, timeline and RCP values. Therefore, by researching the different methods and variables used in the climate models - it would be possible to figure out reasons for the BCCAQv2 climate model inaccurately predicting temperatures for Canadian cities. Additionally, comparing data that covers a larger timeline or cities from various countries could potentially be more accurately predicted by the BCCAQv2 climate model. Lastly, having separate lines that are able to remove annual temperature anomalies could be helpful in presenting an overall increase or decrease in annual temperatures.

References

Higher Temperatures. (2017, May 09). Retrieved November 6, 2020, from https://archive.epa.gov/climatechange/kids/impacts/signs

J, J., & M, K. (2016, January 15). How do greenhouse gases trap heat?: Socratic. Retrieved November 6, 2020, from https://socratic.org/questions/how-do-greenhouse-gases-trap-heat#: :text=Greenhouse

What are the RCPs? (n.d.). [Infographic].Retrieved November 6, 2020, from https://coastadapt.com.au/infographics/what-are-rcps

Line Charts in Python. (n.d.). Plotly. Retrieved November 6, 2020, from https://plotly.com/python/line-charts/

Scatter Plots in Python. (n.d.). Plotly. Retrieved November 6, 2020, from https://plotly.com/python/line-and-scatter/

Interactive Data Analysis with FigureWidget ipywidgets. (n.d.). Retrieved November 6, 2020, from https://plotly.com/python/figu
app/

Monthly climate summaries. (n.d.). Climate Data from Environment and Climate Change Canada. Retrieved November 6, 2020, from https://climate-change.canada.ca/climate-data/#/monthly-climate-summaries

Variable Data (Mean Temperature). (n.d.). Climate Data Canada. Retrieved November 6, 2020, from https://climatedata.ca/dow

About. (n.d.). Retrieved November 6, 2020, from https://climatedata.ca/about/4/

Python floodfill Image object. (2019, February 13). Retrieved December 11, 2020, from https://stackoverflow.com/questions/5454
floodfill-image-object

Python PIL: ImageDraw.Draw.text(). (2019, September 05). Retrieved December 13, 2020, from https://www.geeksforgeeks.org/p
pil-imagedraw-draw-text/

ImageDraw Module¶. (n.d.). Retrieved December 11, 2020, from https://pillow.readthedocs.io/en/stable/reference/ImageDraw.h

Guven, B. (2020, October 02). Adding Text on Image using Python. Retrieved December 11, 2020, from https://towardsdatascience.com/adding-text-on-image-using-python-2f5bf61bf448

Combine several images horizontally with Python. (2015, May 13). Retrieved December 13, 2020, from https://stackoverflow.com/questions/30227466/combine-several-images-horizontally-with-python

Chapter 17 – Manipulating Images. (n.d.). Retrieved December 13, 2020, from https://automatetheboringstuff.com/chapter17/

Floodfill Image using Python-Pillow. (n.d.). Retrieved December 11, 2020, from https://www.geeksforgeeks.org/floodfill-image-using-python-pillow/amp/

Pillow. (n.d.). Retrieved December 12, 2020, from https://pillow.readthedocs.io/en/stable/