Spring 6-1-2022

National Climate Data Graphical Plotting Software Review

Melissa Barnes
Portland State University

National Climate Data Graphical Plotting Software Review

by

Melissa Barnes

An undergraduate honors thesis submitted in partial fulfillment of the

requirements for the degree of

Bachelor of Science

in

University Honors

and

Computer Science

Thesis Adviser

Brenda Glascott

Portland State University

2022

This essay discusses the computer science capstone thesis work I completed. The goal for this project was to create a tool that could be used by climate scientists to extract interesting historical trends from county climate data from the United States. One example of how this data could be used is to observe when a county warmed more rapidly than others during the same dates. This information could then be studied with other contextual information to find causal links between events and the temperature warming rate. This project was completed as part of a computer science capstone in a group of seven students.

The software my group created downloads climate data from the National Weather Service for every county in the United States from 1895 to the present, stores the data in a local database, and then uses that data to make calculations and visual aids. The data includes average, minimum, and maximum temperature, precipitation, and drought values for each month. The software creates polynomial fit lines on scatter plots and heat maps. Another feature creates two fit lines, one for the full date range and one for a number of years less that can help show how accurately past data can be used to predict future data. There is also a plotting option that will split up the data by month and plot a separate line for each month as well. The data can then be compared between counties for various time intervals selected by the user. Another feature is that the data used for each plot can be exported to a csv file.

Our project was sponsored by a member of the community who had an interest in this data. He is a retired professor who has been trying to create these calculations by hand and with the help of excel, but does not have any software engineering experience. He came up with the concept and goals of this project as a way to ease the difficulty of working with the data manually. Our sponsor did not specify the technology we needed to use, so that was up to us to determine. We had a few members meet with our sponsor once a week to discuss progress and

get his feedback. Around the halfway point of our project we created an outline of the requirements and goals for the final product and had both our sponsor and our team agree to it.

Our team met semi-weekly in the beginning of the project to discuss our absolute goals and stretch goals for the project. We discussed the viability of the different goals and suggested methods to implement them. Our initial goals were to create the database, preprocess the data gathered from the online data source into a format that would work well with the database, load the data into the database, create methods to retrieve data, develop methods to apply calculations and create plot graphs with the data, and develop user interface technology selection and creation. There were some tasks we did not know how to implement initially so we created research tasks to gather more information and experiment with new methods to see what worked well. We then determined which tasks were the most important to complete and divided them amongst ourselves. Some of our design choices had to be revised as the various tasks became more integrated with each other, and at those points we would make adjustments as necessary. We kept in contact using a team Discord so that we could communicate and ask for help when issues or questions about design choices arose.

My main contribution to this project was working with our local database. The database stored weather, drought, county identifier codes, geographical information, elevations and populations. I created a file of functions that initialized the tables in the database, imported data to those tables, and retrieved the data from those tables to return to other parts of the program that needed to use it. I spent a lot of time developing PostgreSQL queries to retrieve the data we needed for plots or the user interface by using the parameters that were selected by the user. To adapt to my teammates' needs, I had to modify and create new functions throughout the entirety of the project. I ensured that the data was being returned from my functions in the format my

teammates required. For example, we chose to store some values as integers to leverage the superior performance of how the database handles integers compared to strings. However, when we worked with those values outside of the database, we needed those values to be formatted as strings of a certain length. I needed to convert the data type and pad the values with the correct amount of zeros. I also contributed to displaying some values on the user interface, but this was a much smaller portion of my work. The user interface, plotting, graphing and installation scripts were mainly created by my teammates. We worked together using the Agile development method.

Agile Software Development consists of development methodologies that are centered around an iterative style of development, where changes are made incrementally instead of aiming for one big full launch of the product all at once. We used the existing agile approach for this work which allowed us to make incremental changes and continuously deliver new features. We had semi-weekly stand-up meetings where each individual would give an update on what they had accomplished since the last meeting, what they were working on next and if there was anything blocking their progress. At our initial meetings we determined which tools we would use to create this software.

We used a Postgres database to store our data. This came as a downloadable program that we could then link to our project for use. We chose to use Python as our programming language because it is simple and effective for the tasks we needed to accomplish. We also used Pandas which is the Python Data Analysis Library to store the data we retrieved from the database as it stored it in an easily accessible format. We leveraged the psycopg2 PostgreSql Database Adaptor in order to connect to the Postgres database and query data from it. We used matplotlib to create our visual representations of the plots and graphs. We applied operations from the numpy library

to the data to create polynomial regression calculations. We used Tkinter, which is a python interface tool, to create our user interface. All of the county data we operated on came from the National Climate Data Center.

In general, all of my computer science courses have helped me to prepare for this capstone project. In all of my courses I have had to learn how to use new software tools and how to find out how they work on my own. Learning how to figure things out for myself was an absolutely necessary skill for this capstone as I was working on a portion of the program that no one else was working on. All of my courses also taught me a lot about debugging and how to figure out what was going on when my program wasn't working.

The Intro to Database Management Systems course prepared me well to work with the Postgres database. In that course I learned how to create queries to get the data that I need from the database. I also learned about primary and secondary keys and how to query for data in a relational database. The Open Source Software and Elements of Software Engineering courses taught me how to use git version control and various useful git commands. I used git version control every day with this project. I also learned about working in an agile environment and with a team. The Data Structures course helped me understand object-oriented programming and how to separate concerns for different aspects of the program. In our project we separated areas of concern into plotting, user interface, preprocessing, raw data and database operations.

We encountered obstacles throughout the entirety of the development process. One issue was miscommunication between the sponsor and our team that caused our sponsor to be unsure of whether we would complete the objectives they were looking for. We decided to meet as an entire group with our sponsor and go over everything that had been done up to that point to make sure he knew where the project stood. We also decided to send him a document that explained

what we were planning to achieve by the end of the project and what he could expect from it once completed. We had him look over the document and approve or make changes until all parties agreed on the plan. We realized that the main cause of this issue was not that we were not meeting the sponsor's expectations for the program, but that the sponsor needed more consistent communication from all of us about our progress. This resulted in the decision to include all team members, as opposed to just a few, in every weekly meeting with our sponsor.

We also had issues when comparing our computed data to our sponsor's. When we did comparisons to make sure that we were implementing the mathematical equations correctly, we initially found that our calculated data was very different from our sponsor's. To solve this problem we had a few members meet and compare data with our sponsor. We initially thought the differences between our calculations might have been caused by separate data sources, but after analyzing the sizes of the discrepancies, we determined that these issues could not be caused by separate sources as the discrepancies were too large. After analyzing our code and data in the database, we realized that some values were being imported incorrectly due to an error in our preprocessing code. We then fixed the importation functions, re-ran our calculations and brought the new results to our sponsor for more comparison.

Another issue we frequently encountered was that sometimes when two people would get to the point where they would connect their code to each other's, the implementations did not work together seamlessly. We often had to make adjustments to one side of the code to allow it to work for the other. For example, I would create functions to return data from the database to be sent to the user interface for calculations and display, but sometimes, the format of the data my code was returning did not work well for the user interface portion. We would then discuss how

the other person needed and expected the data to be formatted and I would make adjustments so that they would receive the data in that state.

An additional issue we had during the project was with formatting and displaying on different types of operating systems. On Windows Operating System the user interface would be fully visible and all buttons were accessible. However, when the same program was run on a Mac Operating System, some buttons were inaccessible and the user was unable to access all parts of the program. For this we created a new issue on our GitHub board and assigned a member to find a solution for this issue. This was a complicated issue, so while we were still researching the best final solution, we implemented a temporary fix that would allow the user to access the hidden parts of the screen. This was important so that other parts of development were not slowed down by an inability to access the full program.

I have gained a lot from this experience. I have learned how important it is to have a solid and clear set of written expectations that my team and the sponsor agree to. It is important that these expectations are clearly outlined in the beginning to avoid scope creep, which is the tendency for a project's requirements to increase over the project lifecycle. It is important that both parties know what is possible, expected and promised during the duration of the project.

I have also gained an understanding of how helpful it is to define tasks clearly from the beginning. When the tasks are created it is important to spend the time to thoroughly define what that part of the code should accomplish, what parameters it should expect to receive and how the return values should be formatted. This is important because if these details are not clearly defined, it can result in a lot of revising that could otherwise be avoided. One person who creates the task may have a clear idea of how the feature should be implemented, while the person implementing that feature may have a completely different idea of how it should work.

Additionally, others on the team who are not working on that feature may have code that relies on it and need to know how their code will connect to the feature. This could have helped avoid some of our issues when we connected two or more pieces of code together after they had been developed separately.

I now understand that it is important to regularly push updates from a local branch to the main development branch, as well as to regularly merge development branch changes into local branches. This is important to avoid complicated merge conflicts that can take time and resources to resolve and to make sure that we are working with the most recent version of the program. It is also important for continuous integration that we have a separate main branch that only merges the development branch changes in when those changes are fully functional. This ensures that there is always a clean working version of the program, even if a bug accidentally gets introduced to the development branch.

I have acquired a lot of technical knowledge about working with databases. I learned how to set up and connect to the Postgres database. I experienced working with PostgreSQL syntax and understanding how it differs from other languages I have experience with. I also learned how to implement exception handling for Postgres transactions and connections. I studied how to retrieve data from the database and store it in a Pandas dataframe, as this was a convenient method to store our data and to use it throughout the program.

If I could go back to the beginning of the project I would have the sponsor and our team work on a requirements document from the very beginning that outlines the expectations and commitments of what we will create. I think that a clearer understanding from the beginning would have solved a lot of communication issues we ran into in the middle of the project. I

would also have had all of the team members meet every week with the sponsor from the beginning of the project.

I would possibly store the ids for the county data as strings instead of integers. One of the reasons we did not do this was that it can impact performance. I would first check to see how much of an impact this change would have and if it wasn't significant, I would keep them stored as strings. While we stored the values as integers in the database, whenever I would retrieve the data I had to convert them to strings and append missing zeros to the front. The database truncates leading zeros for integer data types. It would have been a lot simpler to handle these values if they were stored as strings because the leading zeros would not have been truncated by the database, so I would have had less reformatting to do.

I would also make the formatting of the user interface a top priority earlier on in the process. There were times when it was difficult for me to test certain aspects of the program because the user interface was not fully functional. Not all team members faced this problem as it depended on which operating system we were working on. Luckily, most of my work did not depend on the user interface being fully functional, but there were a few times when it did block my work.

Our program is on track to meet the requirements agreed upon by our sponsor and the team. The data we have calculated is now consistent with the sample data our sponsor has calculated. We have implemented all types of graphing requested by our sponsor. Our database is functional and able to be reproduced on any user's local machine. We have connected all pieces of the code that were created independently. Complications with how the user interface is displayed on different operating systems has been resolved. We have created a readme file that gives detailed instructions on how to install and use the software. We implemented an installation

script to help ease difficulties we anticipated might arise when a user who is unfamiliar with software is trying to install our program. Overall, our team worked hard to address issues as they arose and ended the capstone with a finished product that met our sponsor's expectations.

Annotated Bibliography

Group, P. S. Q. L. G. D. (2022, April 8). PostgreSQL. Retrieved April 8, 2022, from https://www.postgresql.org/

PostgreSQL is an open source object-relational database system. We used this to create a local database for each user to store the county weather and drought data, county id codes and county geographical data. The strengths of using postgres are that it is open-source, largely compliant with SQL-standard, and works well with geographical data and matplotlib. One weakness to consider when using PostgreSQL is that it is comparatively slower than other popular database systems, so would not be the best choice for a program that heavily relies on performance.

National Climatic Data Center. (n.d.). *National Climatic Data Center Climate Division Dataset*. Index of /pub/data/cirs/climdiv. Retrieved April 8, 2022, from https://www1.ncdc.noaa.gov/pub/data/cirs/climdiv/

ŋClimDiv is a monthly divisional dataset which also contains statewide, regional and CONUS data from 1895 to present for the following elements: Temperature, Precipitation, HDD/CDD, Drought. The strengths of using this dataset were that it had the climate and drought data we were looking to use for almost every year and county in the United States. One drawback to using this data was that a few counties were missing data for years before 1925 and Hawaii. Another limitation was that the location of the weather stations for this data is not included and some of the weather stations likely moved at different times between the beginning of the dataset and the end. Differences in weather station locations and elevations could affect the weather data provided.

NumPy. (n.d.). Retrieved April 8, 2022, from https://numpy.org/

Numpy is a package for scientific computing with Python that provides powerful operations on arrays. We used this to contain and operate on arrays of data for our plotting methods. The advantage to using this was that Numpy comes with built in functions to operate on arrays that we did not have to create from scratch.

Pandas. pandas. (n.d.). Retrieved April 8, 2022, from https://pandas.pydata.org/

Pandas is an open source data analysis and manipulation tool, built on top of the Python programming language. We used Pandas dataframes to store all of the data we gathered from the database. One advantage to using Pandas was that it is built on top of NumPy and works well with many other python libraries. Pandas also makes it very easy to perform certain operations on the data that would otherwise be meticulous and time-consuming such as adding or removing an entire column of data. It is not a tool that could be used with other programming languages as it is python specific.

PSYCOPG2. PyPI. (n.d.). Retrieved April 8, 2022, from https://pypi.org/project/psycopg2/

Psycopg is a popular PostgreSQL database adapter for the Python programming language. Its main features are the complete implementation of the Python DB API 2.0 specification and the thread safety. We used Psycopg to establish connections to the PostgreSQL database and in handling exceptions from the database. It is specifically built to work with PostgreSQL and would not be useful for other databases.

Tkinter - Python interface to TCL/TK. tkinter - Python interface to Tcl/Tk - Python 3.10.4 documentation. (n.d.). Retrieved April 8, 2022, from https://docs.python.org/3/library/tkinter.html

The Tkinter package is the standard Python interface to the Tcl/Tk GUI toolkit. We used Tkinter to create our user interface because it was easy to implement a basic interface and was fairly customizable due to its layering nature. One disadvantage was that some formatting issues were difficult to debug and it might not be the best choice for an application that requires more advanced widgets.

Visualization with python. Matplotlib. (n.d.). Retrieved April 8, 2022, from https://matplotlib.org/

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. We used Matplotlib to create our graphs. Matplotlib is an excellent library that can create many different types of high quality visualizations. It was very easy to use and already included many built-in functions to adjust the graphs that we would have otherwise had to construct from scratch. This is a python specific library and would not be suitable for other languages.