Analyzing Historical Winter Weather

Kole Frazier

Final Project

Physics 2300 Scientific Computing

Weber State University

Fall 2017

1. Introduction

During this semester, I was given the tools needed to perform analysis on large sets of data. I decided to put these tools to work and analyze a much larger set of data than I had analyzed before. I decided to write a program that could handle a large set of weather data from the National Oceanic and Atmospheric Administration (NOAA). The focus would be on cold weather and whether the Winter season in the Salt Lake City area has been changing over the past decade.

Due to the process of reading in, parsing and plotting data being a process that does not initially require user input, I built my program to be self-contained and automatically generate the desired information. When ran, it will pull in all information from the data sets and ready the information to be plotted for analysis.

1.1 Purpose

The NOAA publishes the data collected from its many weather stations and allows third parties and general citizens to request sets of raw data. When the NOAA publishes this weather data, it does not analyze the information for you. Nor does the NOAA ensure that every piece of data is complete.

This problem of having so much data – which may not be perfect - suddenly in your hands is a common obstacle that writing a program can assist in overcoming. While there are many different languages that may have helped in some cases for this program (such as SQL queries or F#), the programming language used during this course and to write this program is Python.

1.2 Scope

The NOAA's website allows anyone to request data sets for many different weather parameters. I chose to limit the scope of my project to a single decade, starting with 2007 and going to 2017. To help reign in the scope of the project further, I requested weather data such as precipitation records and temperature records for the Salt Lake City area. I received two sets of data to work with: a data set with entries for every single day from all stations in the area and a data set with yearly averages for each station.

The data sets I received were far from perfect. In some cases, random parts of data would be missing. This is due to the data simply not being available that day, or the NOAA station may have received too many erroneous values, resulting in the value for that day being discarded.

The data sets I received were huge. Particularly the daily readings data set with just over 112,000 lines of data. As such, I had to work with the data without reaching out to the hard drive for another line while keeping the numbers for each set of data accurate.

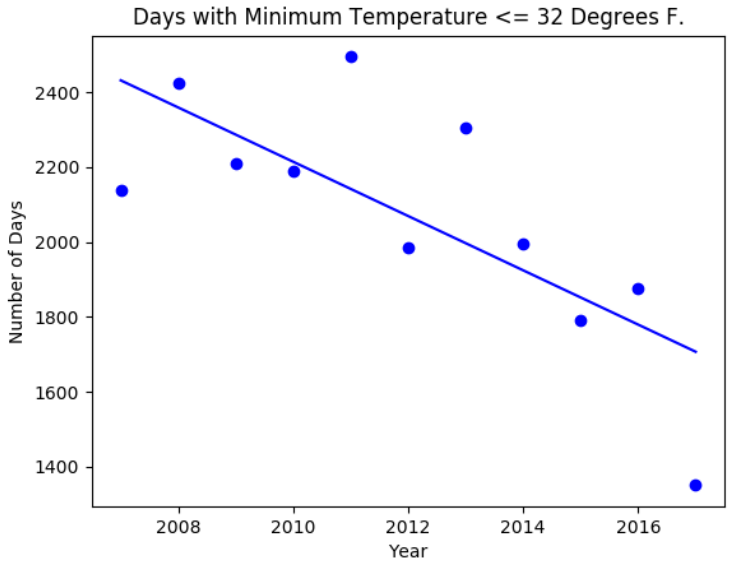

Figure 1

I created a chart that I thought would display the average number of freezing days for each year. Instead, I got a count of how many different entries (from all stations) reported freezing temperatures.

1.3 Using the Program

The program I wrote is very simple to use and could easily be modified to assist in analyzing other sets of data. Using the program requires the Python 2.7 framework to be installed. The program will look for two files in the same directory as the program's source file:

1. WeatherData-SLC-Daily.csv
2. WeatherData-SLC-Yearly.csv

Running the program requires no user input and no flags. It can be ran just like running another Python program:

```
C:\Program Directory\> python WinterWeatherAnalysis.py
```

After reading in the data files, the program will generate the plots listed in the main scope of its internal scoping. Instead of displaying any plots, plots will be saved in the same directory as the program file.

2. Top Down Design

1. Brainstorm uses for the data – what kind of output would I want to create.
2. Request the needed data from the NOAA.
3. Plan my approach to the data and the general flow of my program.
4. Begin prototyping program and general methods to compile data.

5. Test and refine.

2.1 Pseudo Code

This section covers the program's pseudo code. It is broken down into two sections:

General Flow – Outlines the program's general work flow, without detailed pseudo code about used methods.

Data Methods – Outlines in more detail the methods created to filter and compile data via different formats.

2.1.1 General Flow

```
Import libraries: matplotlib, matplotlib.pyplot, csv
Create dictionary to hold CSV file names under keys: 'Daily', 'Yearly'.

Read in all lines for each file, assign data to its own variable.
      Use a generic method that will simply read in the entire file as raw data.
      Pass raw data to a method that will organize the in-memory data into a
dictionary. This method will use the first row of the CSV files has keys for the
dictionary of data then organize the rest of the raw data under these functions.
Begin prepping data for plots. Data will need to be processed for each individual
plot.
Filter date using data methods as needed for each plot.
Then generate the plot.
Export plot to file.
```

2.1.2 Data Methods

```
Method: CleanDataFloat
Summary: Takes in two lists of data and casts the data to a float. Used to generate
safe-to-plot pairs of data, as any empty/incomplete entries will be discarded.
Parameters: x (list), y (list)
Workflow:
      Instantiate two new lists to export cleaned data into.
      Iterate over the lists. If an entry is blank, skip it. Otherwise, cast to
float and append to the appropriate list.
      Return both lists.

Method: GetYearsFromDates
Summary: Take in a list of formatted dates and an optional split character. Return a
list of years from the given list. NOAA data format is "YYYY-MM-DD", so dates must be
in this format.
Parameters: Formatted Dates (list), (optional) Split Character (default '-')
Workflow:
      Instantiate a new list to contain output.
      Iterate over the given list, splitting the value on the split character.
      Append year part to list.
      Return list.

Method: AverageDataByKey
Parameters: Keys (List), Values (List)
```

Summary: Takes in a list of keys and a list of values. Calculates an average for each key-value associated pair, returns lists with unique keys and averages for the unique keys.
Workflow:
Instantiate two lists to contain keys that have been handled and the calculated averages for each key.
Iterate over the keys list, tracking both value and index.
Check if each key is in the handled keys list.
If it is, continue to next entry.
Otherwise, add it to the handled keys list, use a list comprehension to find all indexes of values that match the key positions. Get the associated values, calculate the average and append it to the averages list.
Return handled keys and calculated averages lists

Method: FilterUnderEqualToValue
Parameters: keys (list), data (list), value (int or float)
Summary: Finds all key-data pairs in the given lists where the data is less-than or equal to the value parameter.
Workflow:
Instantiate two lists to contain keys and data that meet the criteria.
Iterate over the data list, tracking index and data entry.
If the data entry is less-than or equal to the value parameter, save it to the pre-instantiated lists.
Return both lists of filtered data.

Method: CountForKey
Parameters: keys (list), data (list)
Summary: Finds how many times each unique entry (key) is used in the keys list.
Workflow:
Instantiate two lists to contain unique keys and counts.
Iterate over the keys list.
If it is in the pre-instantiated unique keys list, skip it.
Otherwise, use a list comprehension to find how many times the key exists. Append that number to the counts list.
Return the unique keys and counts lists.

Method: FilterDataForKeyValue
Parameters: Keys (list), Data1 (list), Data2 (list), Target (dynamic typed)
Summary: Returns two lists of data associated with the Target Key. Example: Get all temperature and weather data where the key is equal to some location.
Workflow:
Instantiate two lists to contain filtered data.
Iterate over the keys.
If the key matches the value, append associated data to the already instantiated lists.
Otherwise, continue to next entry.
Return sets of filtered data.

3. Results

Through the data processed and plotted, I noticed that there has been a general trend of decline in many aspects of the Winter season for the Salt Lake area.

First, analyzing the data shows that there has been a slow increase in average temperature, with the number of days remaining in freezing temperatures decreasing.

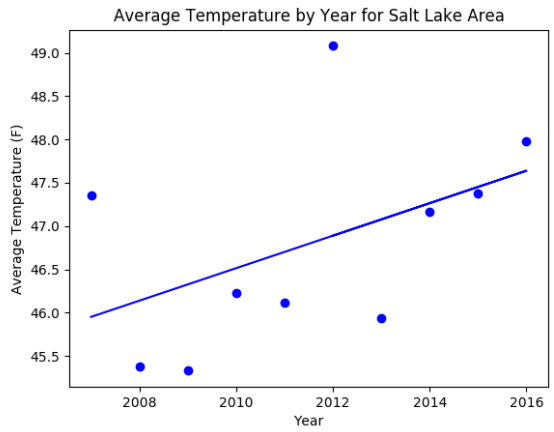

Figure 2 - Average Temperature by Year for the Salt Lake Area

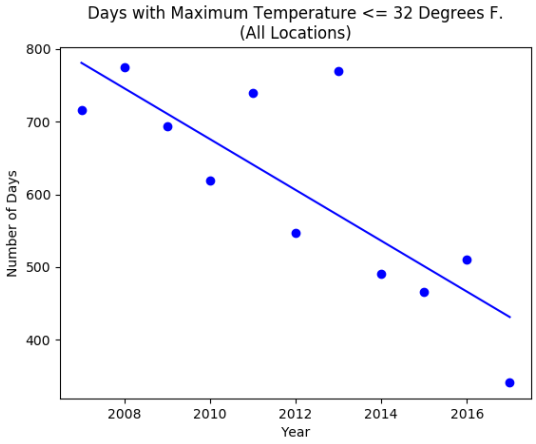

Figure 3 – Days where the maximum temperature did not rise above 32 degrees Fahrenheit.

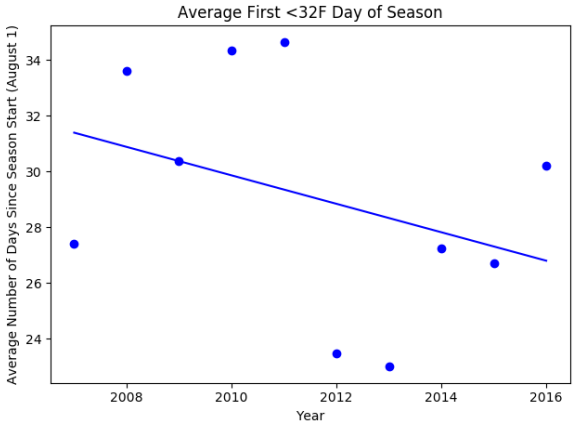

Figure 4 - Average first day with freezing temperatures.
(Note: The NOAA begins tracking this date at the end of Summer.)

Furthermore, I found that in recent years, the amount of total precipitation received throughout the year is steadily lower than 5 to 10 years ago. Combined with fewer days spent in freezing temperatures, there inherently has lowered the amount of snow received.

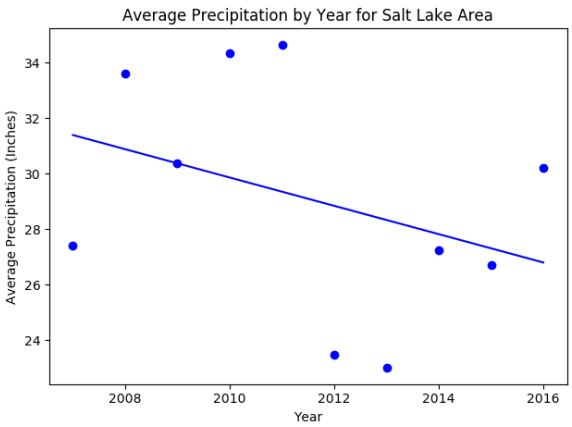

Figure 5 – Average Precipitation by Year

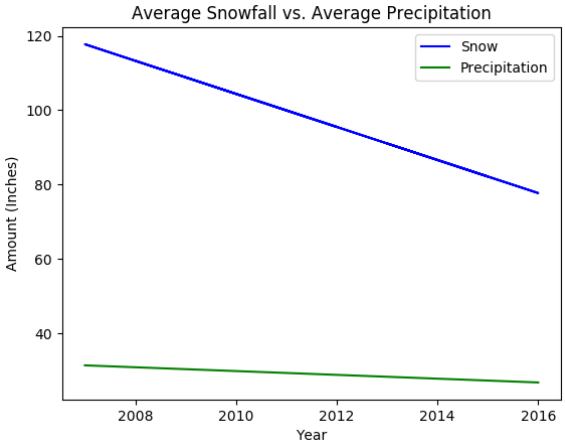

Figure 6 – Snowfall and Precipitation Trends

I found that having plenty of great data for a decade was enough to show a trend in many cases. However, when the data was narrowed down, it did not always provide enough information to show a trend. This was noticed when the data was narrowed down to specific areas. The figure below is a prime example of this. The trend line generated for the Salt Lake Triad Center daily weather information does not seem to fit the shown data. Having a couple extremes on the graph's scale did not help the trendline.

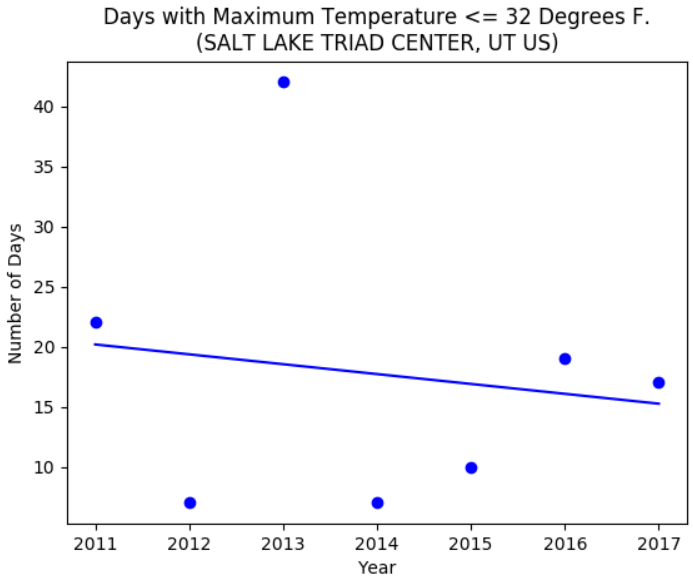

Figure 7

4. Final Thoughts

I am pleased with the program I wrote and the information it generates. If I were to continue working on this project or restart it, I would consider expanding the amount of data analyzed to multiple decades worth of data from a much larger region. Refactoring the program to follow better programming standards (such as classes to group methods) would be a very easy way to increase this project's maintainability.

This kind of project could also fit into topics I have learned from other classes – such as web page design and API design. It would not be too difficult to take what I have created here and adjust it to generate plottable data and trends based on a few user parameters. The information could then be delivered via an API, or to a JavaScript function that would create an interactive plot for a user to work with.

This has given me a look into raw data analysis. This is an area of analysis I have considered before working on before as a side project, but have never pursued. I was honestly surprised with how fun (pardon the nerdy expression) transforming raw data into something readable could be.