November 2018

MNXB01: Course Project.

Sebastian Grabowski, Malik Mohsin Abbas and Frank Johansson

Department of Astronomy and Theoretical Physics, Lund University

1 Introduction

The Swedish Meteorological and Hydrological Institute (SMHI) have a large collection of temperature records dating back to the 1950s at various locations in Sweden. Used for this report were also the set of data with average daily temperatures of Uppsala that started on January 12:th 1722 and ended in 2013. The purpose of the project was to gather interesting information regarding the Swedish climate, with a heavy emphasis on developing a basic C++ program (with the use of ROOT) that will be used in achieving that goal.

The report is set up as follows. Chapter 2 contains a overview of the typical data science workflow used as a guideline for simplifying the collaboration. There will also discuss the idea of the three different observables that were chosen and a little more about the acquired data. Later in the same chapter, the focus lies on the program. Here we will go through the different functional parts of the program, followed by the result section **??** presenting the produced histograms and plots. A discussion of the mentioned results and analysis methods concerning the program are provided in chapter 4, followed by a review and conclusions in chapter 5.

2 Method/Analysis Procedure

The job was divided into three main parts (as can be seen in figure 1)

- Read data and output structured data (according to some template)
- Analyze the output data (for example: calculating the mean value, etc.). Then output results into file.
- Create histograms and plots

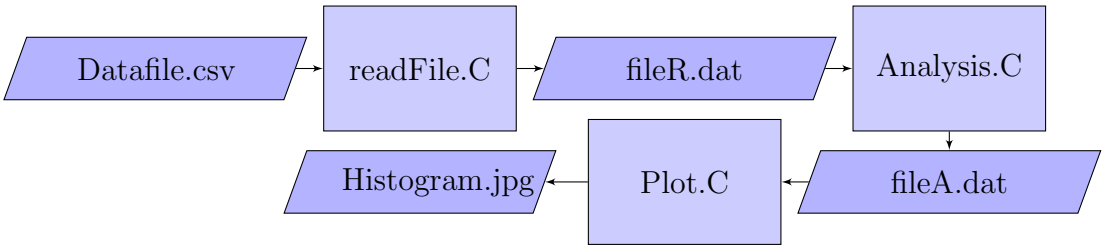

Figure 1: An overview of the different parts of the code and the output in between each step. The readFile script reformates the raw data file according to a pre-establish format. The output file is then used by Analysis.C script. This script gathers the necessary information and produces the final observable files, which then can be plotted with Plot.C.

The code was structured so that each method read a file and output another file. The reason for this structure was to simplify the collaboration effort and to help finding any errors. Since each piece of code should work individually and we are able to check the output, we could focus on creating the correct output at each step.

2.1 readFile.C

First we wanted to read the available data and store it in another output file which can be easily red for analysis purposes. Also, since the raw data has some bad quality data so we wanted to filter that out. For this purpose we created a file named "readFile.C" which contains a function called "readfile".

The readFile function reads the collected temperature data from the csv file and rewrites it in an easier to read manner. It also lets the user specify the time span for which to make calculations.

To start with, the function opens the input file and skips to the first useful line. Then it reads every line while checking if the date on that line lies within the specified interval. After that it replaces the delimiting semicolons with spaces to fit the analysis part of the code. There are some lines that contains text which is cut away at this point. The function also checks for bad data point, denoted by a "Y" and removes them. Finally it prints the good quality data in the required output format to an output-file named "XXR[cityname].dat".

2.2 Analysis.C

The Analysis.C script is to analyze and obtain insights from the reformed data output file, gained from the before mentioned readFile.C script. The way it works is by taking 3 arguments containing information about which of the three histogram-type is wanted, for which city and where to find these data-files (of type "XXR[cityname].dat"). The Analysis.C script can be run from the terminal (./Analysis ¡type¿ ¡cityname¿ ¡directory¿) or from inside ROOT, by calling to the "analyze(string type, string cityname, string directory)" method.

There are three types of available data analysis (as can be seen in figure 2): mT which calculates the daily averages temperature for a given city or all available "XXR[cityname].dat" files inside a given directory; lH which gathers how often a given day of the year was the warmest or coldest (on average) for every year in a given city; fT finds the highest and lowest temperature for each day for each year. When the values have been calculated, the program proceeds with deleting the used data files and prints out the results to an output-file named "XXA[cityname].dat".

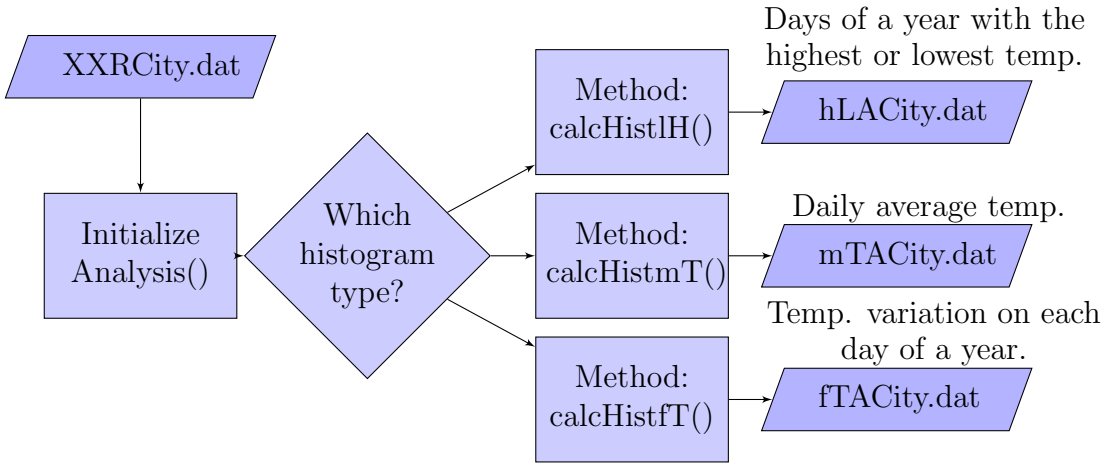

Figure 2: Overview of the Analysis method.

2.3 Plot.C

The final thing to do is to present our analyzed results in a visual manner by using root to construct some graphs and histograms. To do this we implemented a function "plot" in "Plot.C".

First, the plot function identifies the type of plot to create from the file name and then reads the file and creates a canvas. Depending on the first two letters of the file name, the function will then plot the data in a graph, a one dimensional histogram with a fitting function or a three dimensional histogram.

If the file name begins with "mT" it will creates a graph showing the mean temperature of each day over all years as shown in figure 4.

If the file name begins with "hL" it creates a one dimensional histogram showing the counts of warmest and coldest day each year as shown in figure 5.

Finally if the file name begins with "fT" it creates a three dimensional histogram showing the lowest and highest temperatures on each day for each year as shown in figure 6.

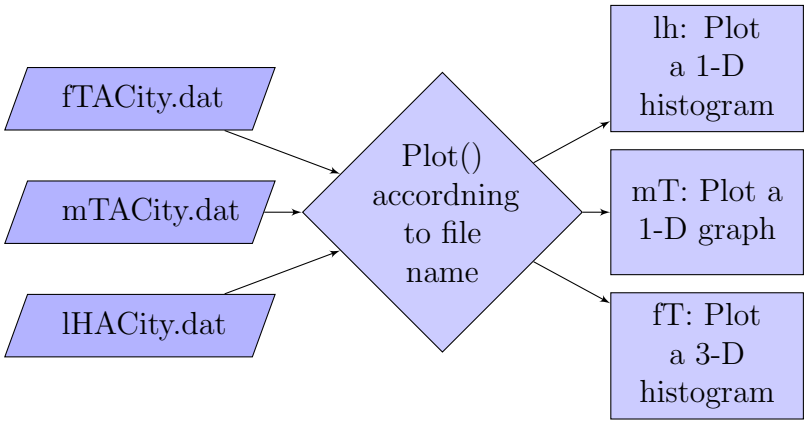

Figure 3: Overview of the plot method.

3 Results

readFile.C successfully reads the raw and stores int the output file according to desired format for the analysis purpose. Analysis.C file does all the three required analyses and creates the new files according to respective analysis types. Finally, Plot.C plots the data according to our analyses as shown in figures 4, 5, 6.

Fig 4 shows the mean temperature of all cities for each day between 1961 and 2015. Fig 5 shows the histogram of how often each day of the year was coldest or warmest for "Borås". Fig 6 shows a 3-D histogram of the highest and lowest temperatures for each day overall years for "Borås".

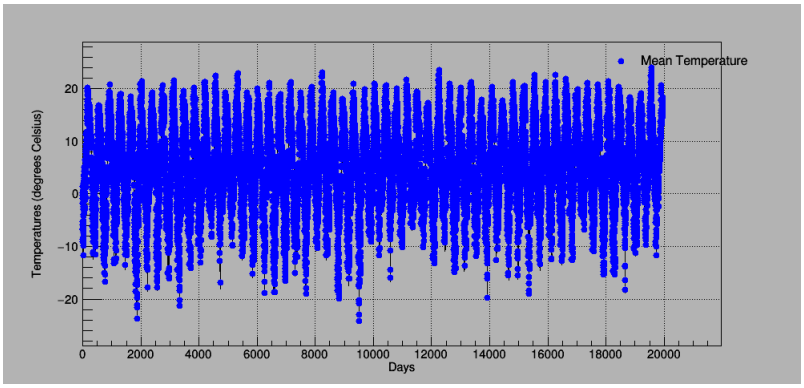

Figure 4: The mean temperature of all cities for each day between 1961 and 2015.

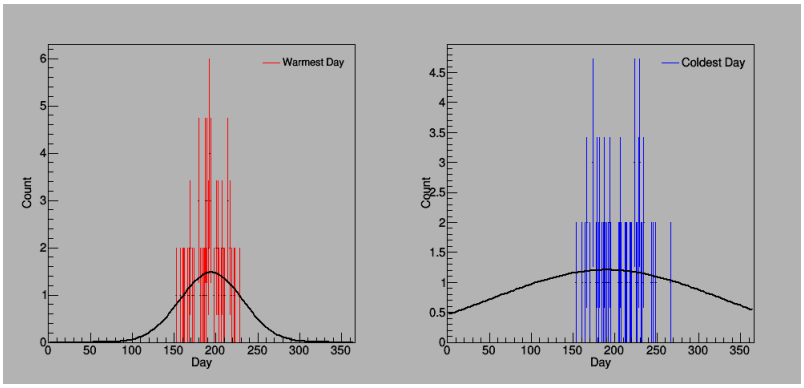

Figure 5: Histograms of how often each day of the year was coldest or warmest for "Borås". The left histogram shows the warmest day and starts counting at the first of January. The right histogram shows the coldest day and starts counting at the first of July.

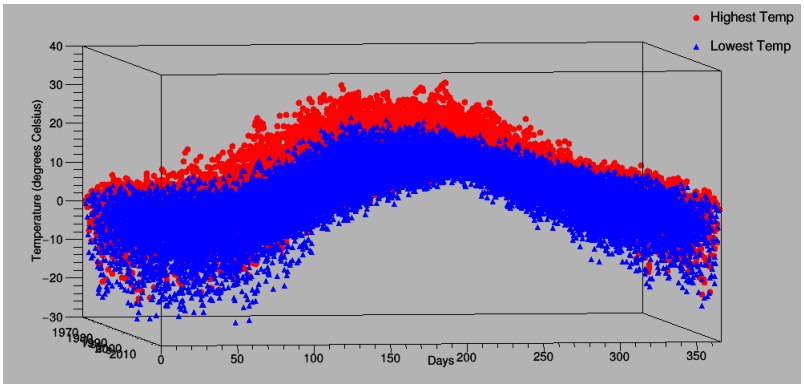

Figure 6: 3-D histogram of the highest and lowest temperatures for each day overall years for "Borås".

4 Discussion

In figure 4 we can see that the mean temperature fluctuates around $\sim$5°C. We expected to see some effect of global warming but we are probably looking at a too small sample from a too short time period.

In figure 5 we can see that a Gaussian fits well to the distribution of both the warmest and the coldest day of the year. We can also see that the variance is higher for the coldest day of the year compared to the warmest day of the year.

Figure 6 displays the temperature but it is hard to draw some conclusions from it, as there are a large amount of data point with a large spread. One could replace the points with Gaussian fits for each year in order to make it more readable.

5 Conclusion

Since the focus of the project was to create the code and not to create informative plots, the plots we made might not be as informative as you would like from a scientific report. We prioritized creating different types of graphical representation of our results.

The project was also limited by time and the fact that all participants have other courses and obligations.

The project did however give hands on experience with basic programming in C++ and how to work in collaboration using GitHub. It also gave us a better understanding of using root to show our results graphically. Overall it was a good learning experience.