Weather Report
Insights drawn from weather measurements in Sweden
FYTN03

Leo Zethraeus, Piotr Yartsev, Xi-Zhen Liu

November 2019

Contents

1 Introduction

The Swedish Meteorological and Hydrological Institute (SMHI) routinely records the temperature at various locations in Sweden. In some places, this has been going on for hundreds of years. The goal of this project is to find at least three interesting results from the SMHI data. The Data set consists of essentially these parameters: Time (in format year-month-day-hour:minute), Temperature, longitude, latitude, altitude. We are using C++ and bash script to clean and process the data, then use ROOT to visualize those data and do further analysis. The report is divided up in three parts, by three main questions to investigate:

- **Part A**: The rising temperature in Uppsala and surroundings. This part aims to detect changes or trends in the annual average temperature in and around Uppsala from the period 1722-2013. Of particular interest is to see if an appropriate function fitted on the annual averages can predict future temperatures.

- **Part B**: How many days a year is the average temperature in Lulea below 0 C°?
- **Part C**: When is the hottest and coldest day most likely to occur in one of the locations around Uppsala?

The report is set up as follows: Sec. 1 introduces the motivation behind each specific question that we wish to investigate. In Sec. 2 we describe how the code works, from reading files to generating plots. Sec. 3 contains the answer of our questions, with some plots. In Sec. 4 we discuss our result and explain what we have done.

1.1 Part A: The rising temperature in Uppsala and surroundings

With the help of the collected data of over 200 years of daily measurements of temperature in Uppsala and surroundings, the yearly average temperatures can be computed and compared from 1722 until 2013. To more clearly see trends over a few years a time, two moving means of these yearly averages with an interval of ten years and twenty years, respectively, are computed and shown in Fig.2. From the yearly averages, the parameter of a chosen function can be fit to try to see if it detects trends such as global warming since the industrial revolution (here approximated to 1880) even though the data set only contains measurements from Uppsala and the surroundings. Once a function has been fit, it might be interesting to see what it predicts about the future by extrapolating.

1.2 Part B: Number of days with average temperature bellow 0 C °in Lulea each year.

To see the trends of the number of days with average temperature bellow 0 C °in Lulea each year we have to clean and process the date in the Lulea cvs file (done with **C++** and **BASH**) then plot the results (using **ROOT**). This is an interesting thing to look at as the number of cold days would probably be affected by global warming, so to study this effect may give a insight into how the climate can be affected over the course of many years.

1.3 Part C: Which is the hottest and coldest day most likely to occur in one of the locations in Uppsala?

We want to visualize the stats of the date the hottest and coldest temperature appear in each year. It may present a normal distribution, and by using ROOT, we can plot the histogram showing this distribution, including the mean and standard deviation. We can use a Gaussian function to fit our to check how the distribution nears normal distribution.

2 Method

2.1 Part A: The rising temperature in Uppsala and surroundings

2.1.1 Cleaning up the Data with C++

Before doing any data analysis, the data was cleaned up with the C++ script annualtemp.cpp. The annualtemp.cpp script takes the unprocessed SMHI dataset from Uppsala (found in CleanData/datasets) and produces a txt file with space separated values in the format (year averagetemp)[1], one row per year and annual average, in the folder ProcessedData/UppsalaData called annualtemp.txt.

2.1.2 Plotting and extrapolating using ROOT

To plot and fit a function, a root/C++ script written called tempYearplotandpred.cpp. In this script, the annual means from the previously produced annualmean.txt was filled in an array. This array was used to compute a total mean over the entire period. Three 1D histograms were filled with temperatures. One called upTemp, only filled with temperatures above the total mean (for temperatures below the total mean they were filled with the total mean instead), and one with downTemp was filled with temperatures below the total mean. This was in order to be able to plot deviation from the total mean in different colors in the same figure.

A third histogram totalTemp was filled, which simply contained all the annual average temperatures, and was used for fitting a fourth degree polynomial function. The fitted function, and the deviation-from the mean plot are shown in Fig. 2. Once the fourth degree polynomial had been fitted to the histogram totalTemp, it could be used to extrapolate temperatures to a given year in the future (or past).

2.1.3 Computing moving means

In order to compute and plot a moving mean, a graph was incremented with points from an array containing the annualtemps like the histogram totalTemp. With a fixed box interval representing the number of years to average over, for

[1]Example: (1756 5.6)

a fixed bin (or year) in the array with all the annual average temperatures, the average was taken over the nearest neighbour years by incrementing in another nested for-loop. In this way, points were added to a graph in order to produce a smooth curve with the moving mean. Two moving means were computed, one with a box-interval of 10-years and one with 20-years.

2.1.4 How to run the code:

All C++ scripts for cleaning up **Part A** were included and implemented as class methods in a class called TempTrender, where the methods for the other parts of this report where included as well. To get the extrapolated year, simply call tempYearplotandpred(int yeartoExtrapolate) once logged in to root in the "code" directory. It should work automatically if you just log in to root in the "code" directory where the rootlogon.C file is located. You can also just call Project() in root in the "code" directory, if the compilation was successful, the default yeartoExtrapolate when called in project.cpp being 2050.

2.2 Part B: Number of days with average temperature bellow 0 C °in Lulea each year.

For the program we predominately used BASH as it is very good at manipulating strings, which we thought was very good for this task as, after we converted the cvs to a txt file, all the files we would deal with in this task would have been txt. files with strings.

2.2.1 Cleaning the data with BASH

Cleaning the data involved many steps. First we extract all lines from the cvs that contain **;G** to an file called **data_temp_smhi-opendata_Lulea.txt** as the **G** denotes lines that have good data. We remove any characters following the **G** as it includes text describing the cvs file which we do not need. Using

```
while IFS=';' read -r one two three four
```

we separated the data text file by semicolons. We move the first part (the dates) into a new txt file called **data_date_smhi-opendata_Lulea.txt**. We ignore part two and part four as they are the time when the measurement was taken and the quality of the measurement because for our purpose the data is useless. We also ignore part three that contains the temperature measurement as we will extract them later.

We then want to put all the temperature measurement for each day on one line. We achieve this by first removing all the duplicate dates from **data_date_smhi-opendata_Lulea.txt**, then reading said txt file line by line we grab all the temperature measurement from the **data_temp_smhi-opendata_Lulea.txt** that contains the date from the date file using the grep command and we put the temperatures in a new file called **data_temp_day_smhi-opendata_Lulea.txt** seperating the values by a semicolon.

2.2.2 Calculating daily average temperature with C++

By putting the names of the file **data_temp_smhi-opendata_Lulea.txt** into a temperary file called **Input.txt** we can parse the temporary file as the argument for an **C++** file. We create a **C++** file that opens the **data_temp_smhi-opendata_Lulea.txt** file using

```
std::ifstream
```

and then iterate over each line in the txt file. For each line a two variables

```
double number=0;
int i=0;
```

are created. Then the line is separated by a semicolon and each part is converted to an double variable. That variable is added to the **number** variable and for each part add one to **i**. Then we add the average number of each line, $\frac{\textbf{number}}{\textbf{i}}$, to a new document called **day_temp_med_day.txt**. Using the same method as when we put all the daily temperatures on the same line we put all the temperatures for each year on separate lines. We then count the number of occurrences of the character - using

```
grep -o "-" | wc -l
```

and put it into a file called **Final.txt**. We remove the fist ten lines in both **Final.txt** and **dat_years_smhi-opendata_Lulea.txt** as the values for the number of days on those lines in **Final.txt** is significantly smaller then all the other and do not seam like good data.

2.2.3 Plotting using ROOT

Plotting with **ROOT** is done in a similar fashion as in the case with the **C++** where the files **Final.txt** and **dat_years_smhi-opendata_Lulea.txt** are parsed as arguments into the **ROOT** file called **graph.C**. It reads the input files line by line and extracts the Numbers/dates and put them into two arrays, **x** and **y**. The two arrays are then plotted against each-other and a linear fit is added with

```
gr->Fit("pol1")
```

where "pol1" references a polynomial of degree one.

2.2.4 How to run the code:

In the folder **code** if you run clean **cleaning_data.sh** it will execute all the data cleaning, the **C++** calculations and will plot a graph with **ROOT** automatically and save the plot in the **images** folder. You can the find all the cleaned data in the **CleanData/datasets/smhi-opendata_Lulea** folder.

2.3 Part C: Which is the hottest and coldest day most likely to occur in one of the locations in Uppsala?

2.3.1 Construct a class Date with C++

To find out the hottest and coldest day of a year, we have to extract date information from the data. This means we have to do some conversion among string type data, number type data, and calculated day of the year. Moreover, this conversion might be useful for other tasks since almost all the analysis take date into consideration. Other attributes like temperature can also be stored in the class. Therefore, we build a class Date to solve the conversion and reusing problems. This class can let us easily convert datatype by calling its methods, which considerably reduces our tasks later.

2.3.2 Read the data with C++

We use C++ filestream library to read data word by word, which is split by space and newline character. Then we parse those readed string to the type it should be. For example, year should be integer, temperature should be double. After, we construct a object of class Date and set those variables as the object's members. For each year, we find the hottest and coldest date by comparing the current processing date with the previous peak temperature date. Once we finish scanning a year, the peak date will be push to the back of a vector, storing all the peak data.

2.3.3 Plotting with ROOT

Each time the year is scanned through, we can fill the data to the histogram created by ROOT. By doing so, we can fill out all the data after finishing reading data without another round for filling. And the class Date also makes things easier by the function **day_of_year()**.

2.3.4 How to run the code

In **project.cpp**, provide your path of the data file and create an object **tempTrender t(pathToFile)**, then call the member function **hotCold_Upp()**. The histogram would be saved in **/images** directory.

3 Results

3.1 Part A: The rising temperature in Uppsala and surroundings

The results from Part A are shown in Fig 2. The moving mean with a 10-year box interval is plotted as a solid black line, the one with a 20-year box interval is plotted as a solid gray line and the fitted polynomial as a green dashed line. The increase in temperature is clearly visible from the upwards curved polynomial from around 1980 − 2013, and also from the moving means, which fluctuates less than the annual averages and mostly stays above the total mean in the last two decades.

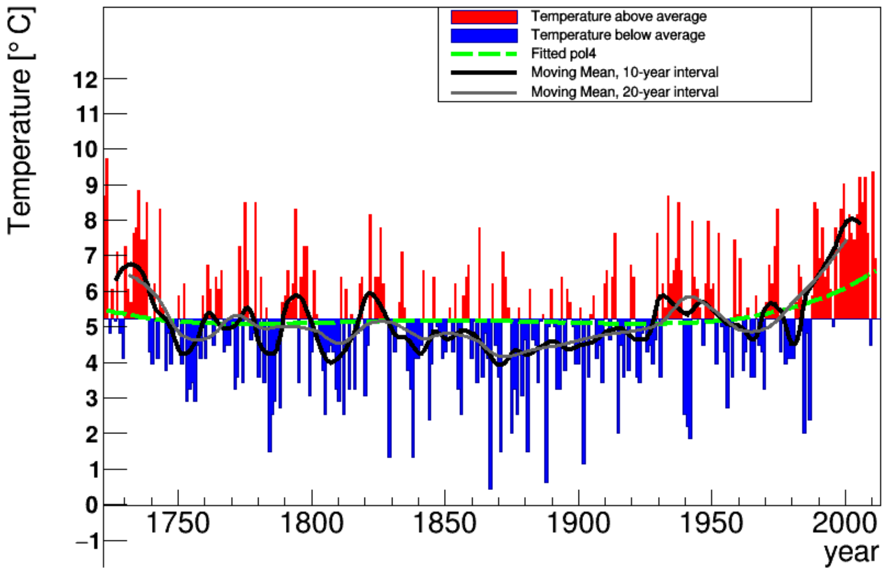

Figure 1: Histogram showing the annual average temperatures in Uppsala and surroundings from 1722 to 2013. The average temperature over the whole period is 5.2°. Temperatures above this total average are shown in red, those below are shown in blue. Two moving averages (”Moving Mean”) are shown as solid black and gray lines, and a fit of a polynomial of fourth degree is shown as a green dashed line.

If you use the function tempPerplotandpred(2050) it gives you an extrapolated value of 7.5 degrees Celsius. The reduced $\chi^2 \approx 0.8$ for the polynomial fit, suggesting a slight underfitting. That is hardly surprising as the annual temperature average has chaotic fluctuations and thus is hard to fit with a linear continuous function.

3.2 Part B: Number of days with average temperature bellow 0 C °in Lulea each year.

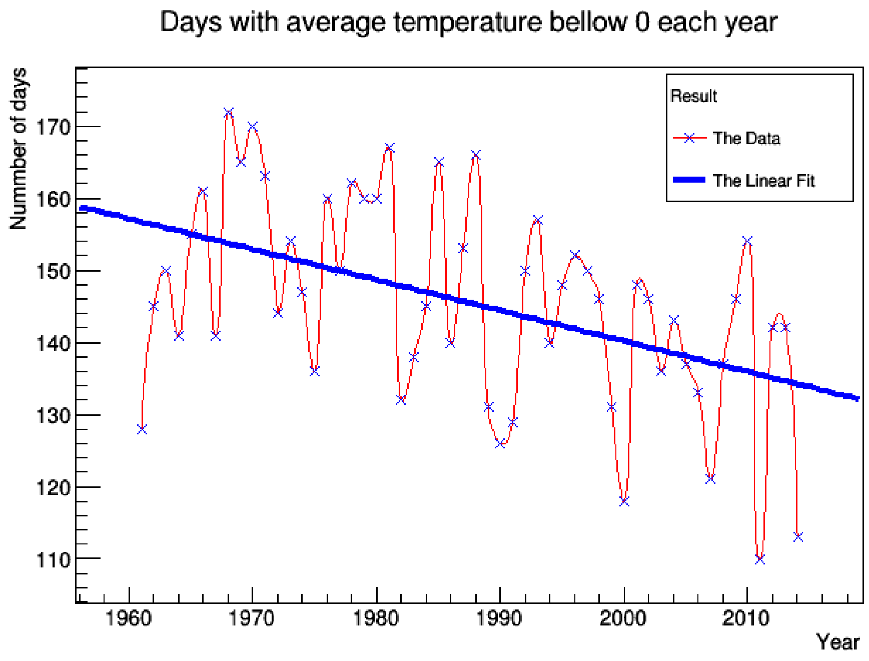

Figure 2: Graph showing the data where the crosses are the data, number of days bellow 0 C °against the dates which spans from 1961 to 2014, with the blue line showing a linear fit.

3.3 Part C: Which is the hottest and coldest day most likely to occur in one of the locations in Uppsala?

The warmest and coldest day of each year

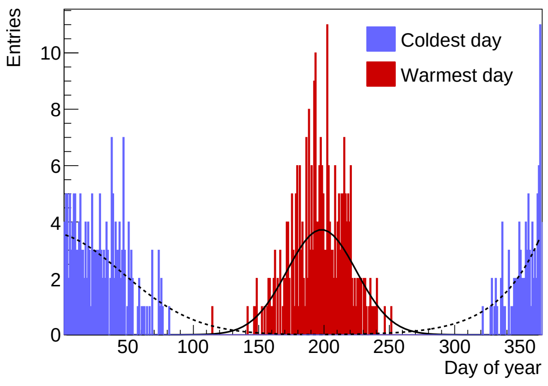

Figure 3: final histogram

As the histogram shows, the most likely coldest day is on the end of the year (31/12), and the most likely hottest day is on 13/7.

4 Discussion

4.1 Part A: The rising temperature in Uppsala and surroundings

From Fig. 2 it is clear that the average temperature is rising not only globally, but also on a local level in mid-Sweden. In this report we have confirmed concerns of global temperature rising using measurements from Uppsala and surroundings, but it would be interesting to compare with data from other places around the world, to see how the changes correlate. It would also be interesting to see which locations are more affected by the global warming than others. This is left for future research.

4.2 Part B: Number of days with average temperature bellow 0 C °in Lulea each year.

The choice to program predominantly in **BASH**, while convenient, turned out to be somewhat bad idea as the code runs very slow, upwards 30 minutes per city. Originally the idea was to plot the data from all the cities but it turned out to be impossible as running the code for all of them would have taken $30\ min \times 9 = 4.5\ hours$.

We had to remove the first points from the final data as they were all in the 0-2 range while the rest of the data was in the 110-170 range. After inspecting the txt files in **CleanData/datasets/smhi-opendata_Lulea** folder showed that it was do to the first few years only having good data starting from march-may with rare measurement in December, January and February, the coldest months of the year. That would naturally drastically decrease the number of measured days bellow 0 C°those years. It is possible to remove those years automatically in the code, for example creating a function that only years that have at least X% of the days in the cold months, but we did not have time to make such a function so it had to be done manually.

From Figure 2 we can see that the number of days bellow 0 C°each year varies quite a bit, but there seams to be a trend down, which would make sense as when the earth gets hotter do to global warming the number of days bellow 0 C °would naturally decrees. There seams to be a periodicity in the fluctuations, but the data is to noisy for a sin fit so we did not do one.

4.3 Part C: The warmest and coldest day of each year

The question is to try to find out the possibilities of dates to become the hottest and coldest date. We read the data of Uppsala and then plot the histogram of the occurrence for each date being hottest or coldest.
We noticed some problem:

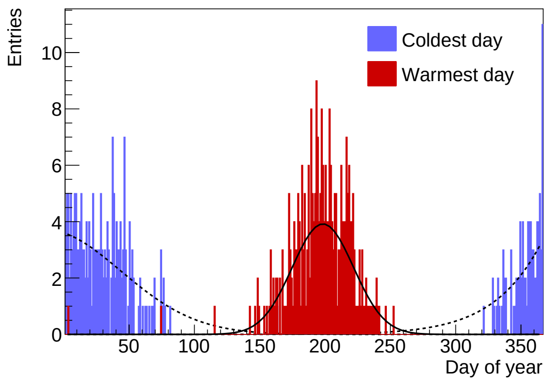

Figure 4: first histogram

1. There are some coldest date in summer, and also some hottest date in winter, which isn't make sense.
2. Winter is distributed at the first and the end of the year, so we use two Gaussian function to fit on those two parts. However, the plot should be circular in practice. We should take both parts into consideration when we draw fit line.

Then we figured out the cause of the first problem. Because the Uppsala data set is a combination of some places around Uppsala. For this question, we ignored those places other than Uppsala. This makes the data became incomplete. For example, data of 1766 are all from Stockholm after April 5, thus the hottest date in 1766 would be April 4, which does not make sense. To solve this, we only accept coldest days which is less than 100 or more than 300 in day of the year, and hottest days which is between 100 and 300 in day of the year.
To solve the second problem, we extend the length of the x-axis of histogram to 732, twice of a year. Then we just copy the tail to the front, and also the front to the tail. We plot two Gaussian fit lines for coldest date, one at the front and the other one at the end, and these line must be the same. By doing so, we can take all the coldest date into consideration. Finally, we just need to combine all the fit lines in same plot, and only take from day 1 to day 366 and ignore others.

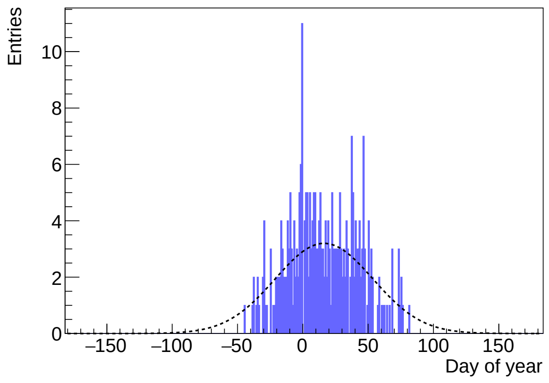

Figure 5: extended histogram at the begin of the year

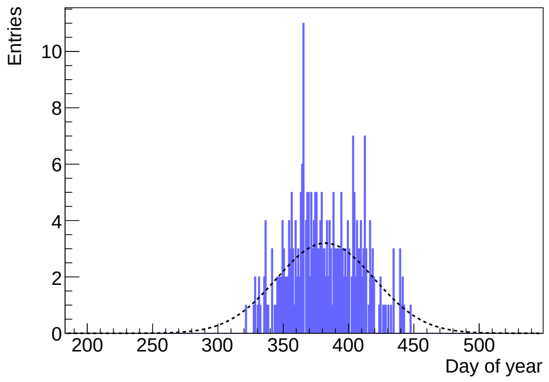

Figure 6: extended histogram at the end of the year

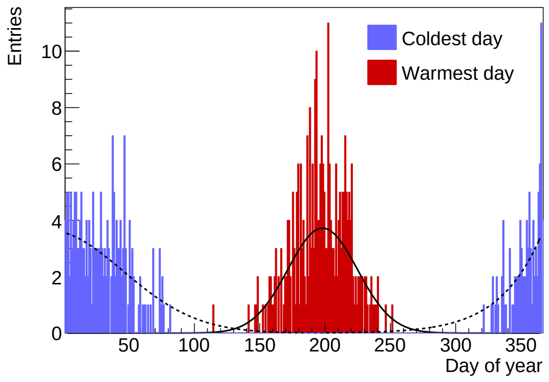

Figure 7: final histogram