MNXB01-project

Alexander Huusko
David Madsen
Lisa Vergara
Lucas Hellström

Contents

1 Data read in system

The first challenge part of the project was to create a data read in system in order to allow data analysis. The data files come in the form on .csv files which has the data structure seen in Figure 1.

```
1   Stationsnamn;Klimatnummer;Mäthöjd (meter över marken)
2   Falsterbo;52230;2.0
3
4   Parameternamn;Beskrivning;Enhet
5   Lufttemperatur;momentanvärde, 1 gång/tim;degree celsius
6
7   Tidsperiod (fr.o.m);Tidsperiod (t.o.m);Höjd (meter över
8   1951-01-01 00:00:00;2015-11-30 23:59:59;5.0;55.3836;12.8
9
10  Datum;Tid (UTC);Lufttemperatur;Kvalitet;;Tidsutsnitt:
11  1951-01-01;06:00:00;-2.4;Y;;Kvalitetskontrollerade histo
12  1951-01-01;12:00:00;-0.8;Y;;Tidsperiod (fr.o.m.) = 1951-
13  1951-01-01;18:00:00;0.0;Y;;Tidsperiod (t.o.m.) = 2015-09
14  1951-01-02;06:00:00;1.0;Y;;Samplingstid = Ej angivet
15  1951-01-02;12:00:00;1.4;Y;;
16  1951-01-02;18:00:00;1.4;Y;;Kvalitetskoderna:
17  1951-01-03;06:00:00;2.0;Y;;Grön (G) = Kontrollerade och
18  1951-01-03;12:00:00;1.4;Y;;Gul (Y) = Misstänkta eller ag
19  1951-01-03;18:00:00;1.2;Y;;
20  1951-01-04;06:00:00;2.0;Y;;Orsaker till saknade data:
21  1951-01-04;12:00:00;2.0;Y;;  stationen eller givaren har
22  1951-01-04;18:00:00;1.8;Y;;  kvalitetskontrollerna har f
23  1951-01-05;06:00:00;2.0;Y
```

Figure 1: Data structure of the files used for analysis. Note that for this file line 1-10 contains no data points.

The first couple of lines consists of information which is irrelevant for reading in, which is why a for loop was used in order to skip the first lines until it reaches the first data point, which can be identified by its "year-month-day;hour:minute:second;temperature" format.

In order to obtain the information in each data point it was first split into three separate strings at each ";". In the next step the "year-month-day" string was split further at each "-" and stored into separate vectors. The same was done to the "hour:minute:second" string for each ":" and the hour was stored in a vector (the data points contains no minutes or seconds and are thus not stored in vectors). Last step was to convert the temperature to a float and then store it in a vector.

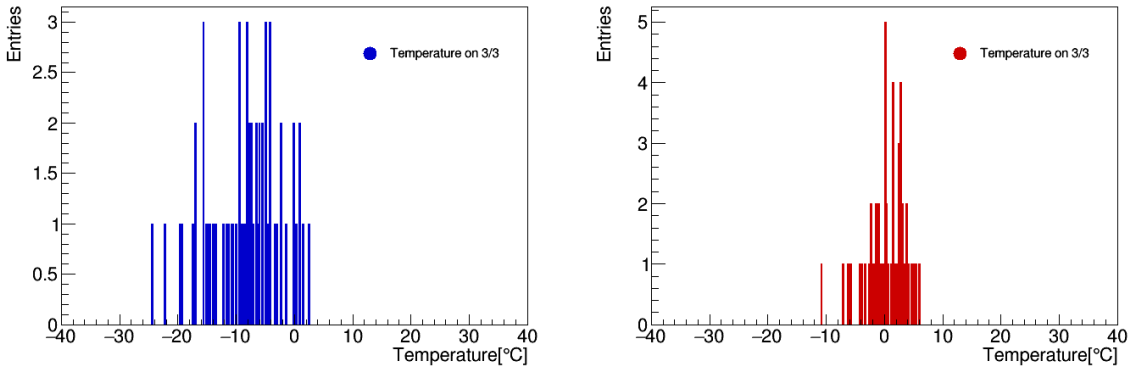
The next step was to create a date vector which calculates the time in terms of year (in decimal) so that one can display all the data points in the same plot without having for example data point "1951-06-11 6:00:00" and data point "1951-06-11 12:00:00" having the same x-value. The code itself converts hours, days and months into equivalent amount of one year and adds it all together with the year and then stores the result in a vector. The code also checks if the current year is a leap year and then takes into consideration the extra day when converting things to years.

2 The Temperature of a Given Day

The aim of this part of the project is to find the temperature of a given day of the year. The day chosen for this analysis is March 3rd. For the sake of comparison two datasets are chosen: Luleå (located in the north of Sweden) and Lund (located in the south).

In order to analyze the data, a function called "tempOnday" was defined. Inside the function, which accepted two arguments: the month and the day to be analyzed, a for loop which ran through all the data points - except the last one - was created. Inside the for loop the code first controlled if the

data point corresponded to the month and the day given as arguments to the function. If it did, the temperature corresponding to this data point was added to a variable called "tempYear". The amount of times this was done was tracked with a counter. The code continued with controlling if the year of the current data point was the same as the consecutive year. If this was not the case, and the current year differed from the consecutive one, the variable "tempYear" was divided by the counter (that is, by the amount of times a new temperature had been added to the variable). This was done because in the dataset several temperatures were given for each day, and thus this way the average temperature for March third on each year was calculated. When this average was calculated, it was saved inside a vector "tempVec" for later use and the counter and variable "tempYear" were reseted. In a final if-statement it was checked if the data point was the last one. This because the original for loop, only looped through until the second last data point. Thus if the current data point was the last, the variable "tempYear" was also divided by the counter and an average for the last year in the dataset was obtained and saved in the vector "tempVec". When the for loop was finished the vectors obtained (one for each city) were used to fill the histograms shown in figure 2. After the histograms were made, the mean temperature of the given day together with the standard deviation of the temperatures were found through the histograms, but also by calculation.



(a) Luleå dataset.

(b) Lund dataset.

Figure 2: The temperature of March 3rd.

In figure 2a and 2b the histograms created with the Luleå and Lund datasets respectively, can be seen. The mean temperature in Luleå on March third according to the histogram (and the calculations) is $-8.37 \pm 6.02°$ C. While in Lund the mean temperature on the same day is $0.44 \pm 3.33°$ C. As can be seen the mean temperature in Lund is much higher than the one in Luleå, which is not strange considering the locations of these two cities. Another thing to notice is the difference in the standard deviations, which is smaller for the Lund dataset than for the Luleå dataset. This can also be directly seen from the histograms in figure 2.

3 The warmest and coldest days of the year

The task is to find the warmest and coldest day for a number of different years. The data is analysed by first using a for loop which will go through all values in the dataset. For every data point the temperature is checked and if it is higher than the previously highest temperature for the year, the day, temperature and month of that day is recorded. This is also done for the coldest days and an example of the code can be seen below.

```
for (k = 0; (unsigned)k < (year.size()-1); k++) {
        if (temperature.at(k) > hottest){
                hottest = temperature.at(k);
                hottestDay = day.at(k);
                hottestMonth = month.at(k);
        }
}
```

At the end of each year the highest and coldest days are saved inside vectors which will be used later. These vectors are used to fill three separate histograms. One for the warmest days, one for the coldest days in the beginning of the year and one for the coldest days at the end of the year. The coldest days are separated to make the guassian fit work.

With the values in histograms the histograms are plotted in the same canvas and guassian fits are applied. By using guassian fits we can obtain a mean for the warmest and coldest days and also the uncertainties of these values. By using the Lund dataset, shown in figure 3, we can see that the mean of the warmest day is: 196 ± 3 and for the coldest day: 24 ± 8. These results shows that the warmest day is most likely to occur in the middle of June and the coldest day is most likely to occur in the end of January.

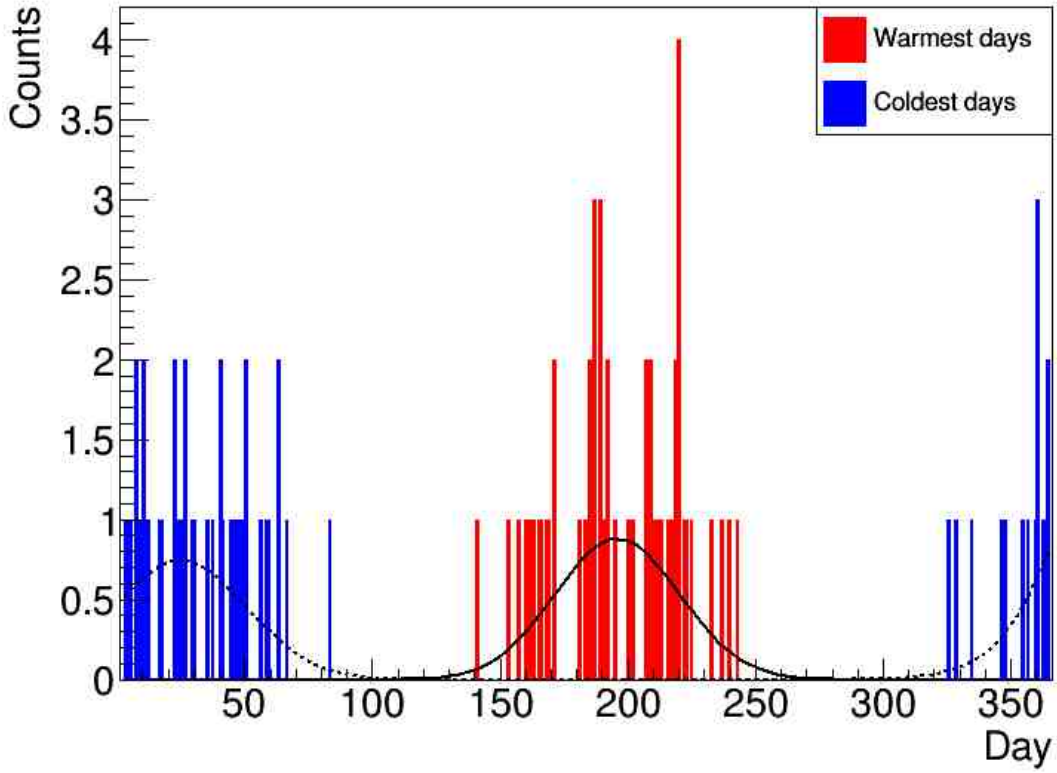


Figure 3: Plot of the warmest and coldest days for each year. Also a guassian fit for both.

4 Highest temperature of the year

The aim of this subproject was to obtain the highest temperature of the year for all available cities and compare them. In order to do this we created a function in the main (project.cpp) file which reads in all of the city objects which is needed for the plotting in the end.

The code uses one function which is part of the tempTrender class. The function uses a variable called "highestTemperature" which is used in order to determine the highest temperature of the year. The variable is first put to a low temperature (0 celsius) as we know the highest temperature will guaranteed be higher than this. The function then starts a for loop which goes over all the data points. If the temperature of the data point is higher than the highest temperature variable, it replaces its value. At the end of each year the highest temperature and the current year is stored in two vectors before resetting the variable. This way one will obtain the highest temperature of each year in an easily accessed vector.

Next step of the subproject was to plot all of the results together in one plot in order to make comparisons easier. The plotting used a for loop and a pointer to reduce the code length by avoiding nine sets of (x,y) lists declarations and value assignments. Figure 4 shows the results of the subproject.

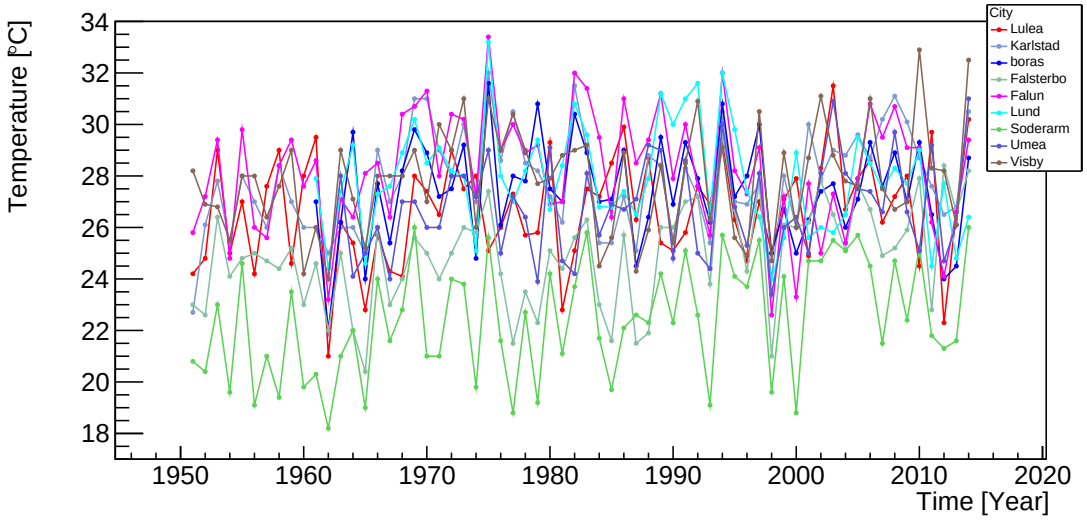


Figure 4: Highest temperature per year for Borås, Falsterbo, Falun, Karlstad, Luleå, Lund, Söderarm, Umeå and Visby.

The results are very messy, however one can see that Söderarm is constantly at the bottom and for some years all of the cities have a raised temperature. It could have been smart to split it up into two separate figures in order to increase readability.