# MNXB01-project

Alexander Huusko
David Madsen
Lisa Vergara
Lucas Hellström

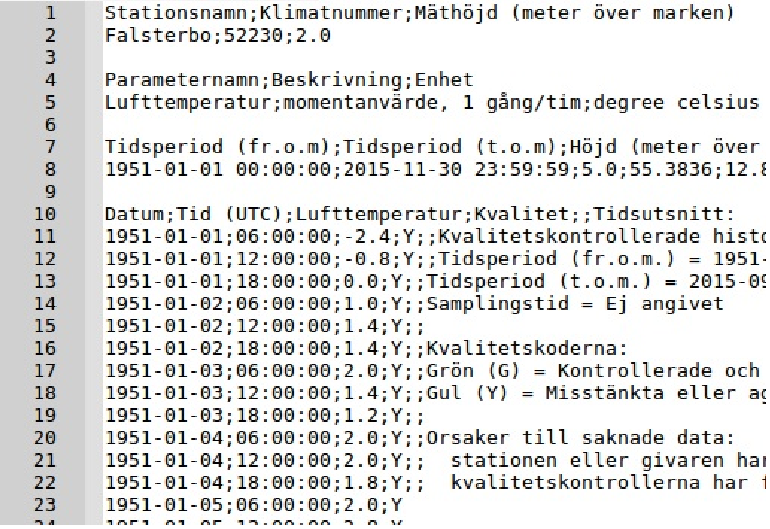
# Contents

# 1 Introduction

By analysing weather data we can obtain a lot of information about our climate, changes to it and fluctuations from year to year. This project will use ROOT from Cern to analyse data from a number of weather stations positioned in different parts of Sweden. Using this data we pick a date and measure the temperature for that specific date over a few years in order to see the fluctuations over the years. Next we looked at the temperature of all days over a year. We also search for the warmest and coldest days of every year. Finally we looked at the highest temperature of a year for all recorded datasets.

# 2 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
24  1951-01-05;12:00:00;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.

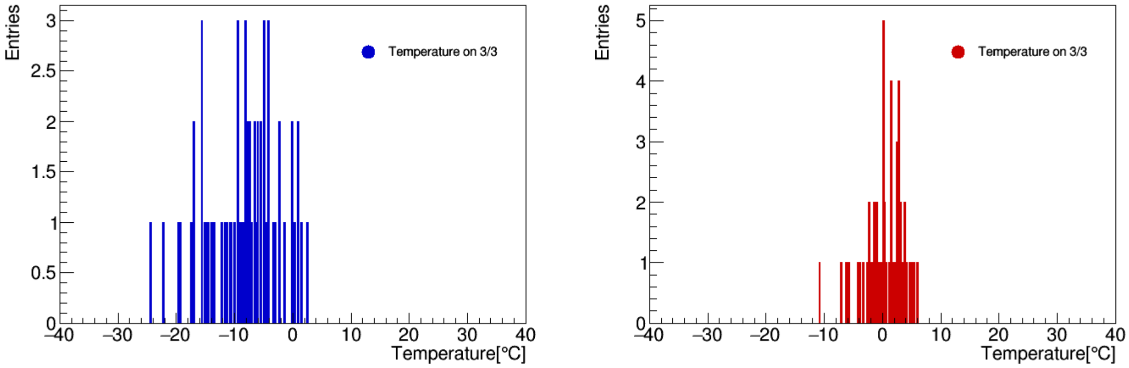
In order to obtain the information in each data point it was first split into three separate strings at each ";". In the next step the "year-month-day" string was split further at each "-" and stored into separate vectors. The same was done to the "hour:minute:second" string for each ":" and the hour 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.

## 3 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.

# 4 Average temperature of each day of the year

## 4.1 Method

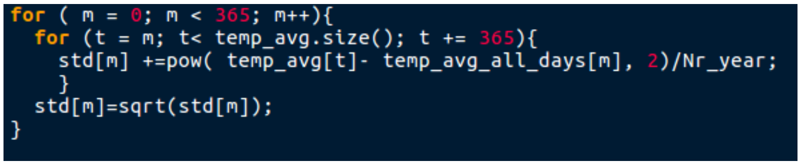
The assignment is to take the average temperature of each day of each year. The analysis is started with a for loop over all the data points in the code. An if statement was then added in order to ignore all leap years, this was due to the short timescale of the project, and to simplify the task. For the years that is not a leap year the code goes into another if statement which checks if there have been several measurements on one day. The temperatures of one day is then summed up and divided with the number of temperatures for that day to receive the average. These average temperatures are then stored in one vector.

In order to sum up the temperature of one day for the different years two for loops were used. The first for loop goes through all the years and the second loop that is inside the first one loops over 365 days and adds each day into an array. This way the array will become the sum of the temperatures of one day for all the years. To receive the average another for loop were used that divided all elements of the array with the number of years.

To calculate the standard deviation, the formula for the standard deviation was used which can be seen in equation 1 below, where $\sigma$ denotes the standard deviation and $\mu$ is the mean.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)} \tag{1}$$

The code for the standard deviation can be seen in figure 3. Two for loops were used one that loops over a full year (365 days). Inside that for loop another for loop was used to sum the standard deviation for the same day of each year.



```
for ( m = 0; m < 365; m++){
  for (t = m; t< temp_avg.size(); t += 365){
    std[m] +=pow( temp_avg[t]- temp_avg_all_days[m], 2)/Nr_year;
    }
  std[m]=sqrt(std[m]);
}
```

*Figure 3: The code for the standard deviation calculation, where temp_avg is the average for all the days for all the years and temp_all_days is the average temperature of each day of the years.*

## 4.2 Result and discussion

The result which can be seen in figure 4 shows the average temperature of each of the year for Lund.

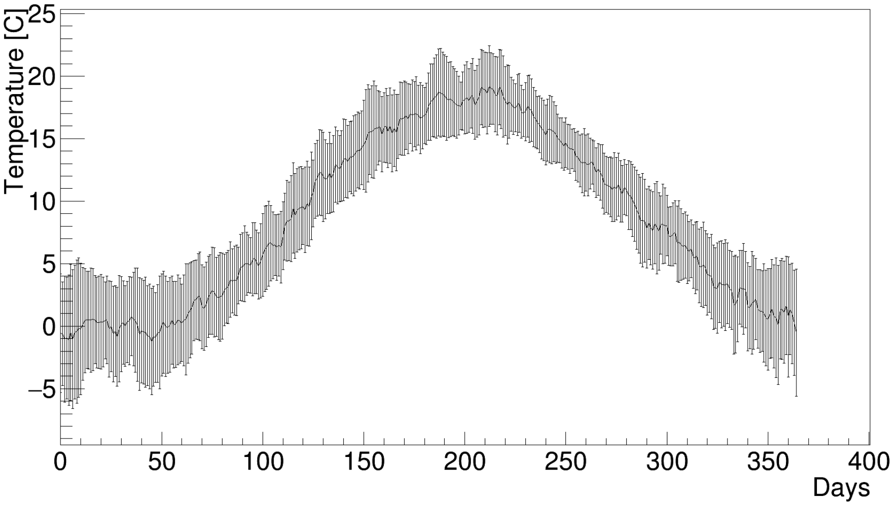


*Figure 4: The average temperature of each day of the year in Lund.*

As can bee seen in figure 4 the result is as one would expect, that the temperature is higher and more stable in the summer. In the winter it's colder and the temperature is less stable thus the larger standard deviation.

Even though the plot of the result gives an result that agrees with reality, the code has some major flaws. In order to simplify the task some assumptions were made which cost the accuracy of the result. One assumption that effected the result the most was the assumption that the dataset had measurements for all days which wasn't the case. The effect this have is that the days in the plot might not be at the right position in the plot thus not being accurate.

If we were to redo this task another approach would have been taken were the inconsistencies of the datasets would have been taken into account.

## 5 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 5, we can see that the mean of the warmest day is: $196 \pm 3$ and for the coldest day: $24 \pm 8$. 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.

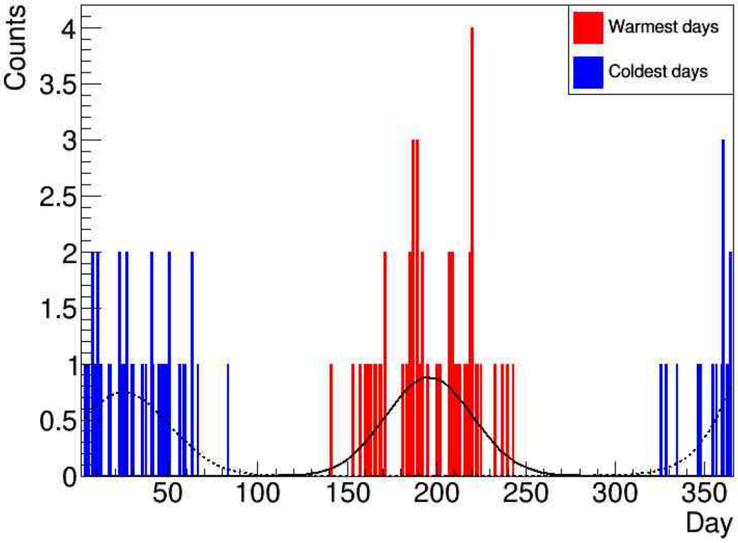


*Figure 5: Plot of the warmest and coldest days for each year. Also a guassian fit for both.*

## 6 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 6 shows the results of the subproject.

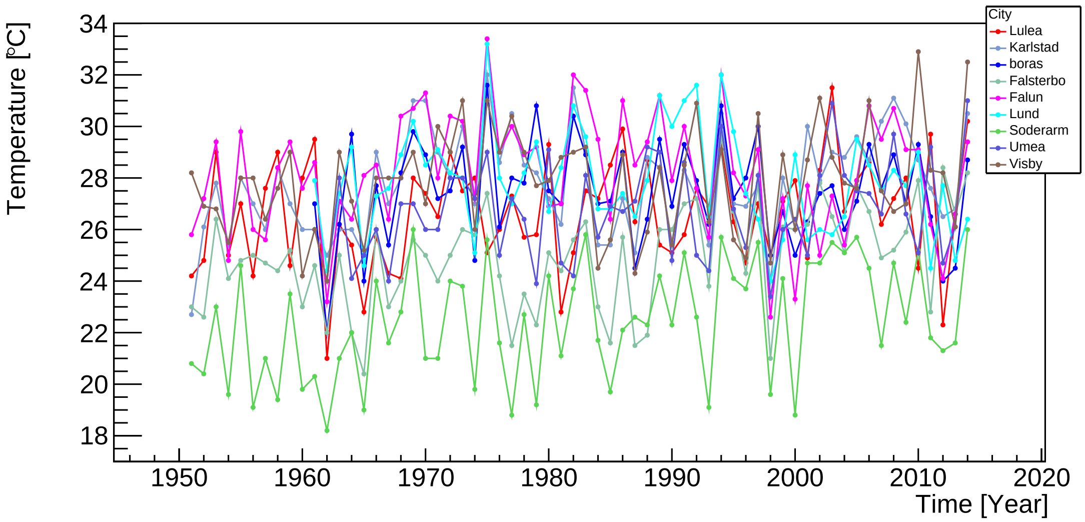


*Figure 6: Highest temperature per year for Borås, Falsterbo, Falun, Karlstad, Luleå, Lund, Söderarm, Umeå and Visby.*

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.

# 7 Conclusion

By analyzing data, a great deal of hidden information can be obtained. This could be seen in this project, where a huge amount of temperatures were analyzed. Once the first challenge of the project - and perhaps the most important one - was passed and the data could be read, the actual analysis could begin.

First two histograms were made, with the datasets from Luleå and Lund, of March 3rd. From these histograms, the mean temperatures of that day in the two cities were found to be $-8.37°$C for Luleå and $0.44°$C for Lund. The average temperature of each day, also in Lund, was then calculated. The plotted results showed that the temperature starts low at the beginning of the year, increases towards the middle of the year and then decreases again towards the end of it. Thereafter, the warmest and coldest days in Lund were searched for. A histogram was made and afterwards fitted with a gaussian. The mean for the warmest day was found to be 196 and for the coldest it was 24, meaning that the warmest day in Lund is most likely to occur in June, while the coldest is most likely to occur in January. In the final part of this project, the highest temperature of the year for all cities, was obtained. In the results all cities were plotted in the same (messy) graph. From this graph it can be seen that the highest temperatures climb and fall together.

Although the results obtained in this project are all quite expected and not very surprising, it is always good to be able to confirm ideas and perceptions. Moreover, finding out what kind of information can be extracted with the right analysis tools and what the many possibilities for this are, is fascinating.