# Final Project:
# MNXB11 - Introduction to Programming for Scientists

Shilpa Rani Sahu, Yuxin Lin, Sebastian Wiederkehr

October 31, 2023

## 1 Introduction

For this project, there were a variety of datasets provided. All of them contain recordings of air temperatures from cities in Sweden, the earliest dating back to the 18th century. The data was gathered by the Swedish Meteorological and Hydrological Institute (SMHI). It is publicly available and can be accessed through their website.
Out of this great pool of available data, we chose to examine the dataset for the weather station in Lund, as it felt the most relevant as we are experiencing the temperatures here firsthand. Furthermore, it has a large range, even when compared to the other datasets available, with its first entry dating back to the first of January 1780.

## 2 Research and Goals

As the supposed extent of the project was set by the instructions, we first decided on the three research questions we wanted to answer with our dataset on air temperatures in Lund. We came up with the following:

1. How much did the mean air temperatures on April the 1st vary in Lund since 1780?
2. What were the ,on average, warmest and coldest days in Lund since 1780?
3. Can we observe long-term trends in air temperature in Lund?

Especially the third question, arguably the most interesting one, gives further reason why we chose to use the data from Lunds weather station. Due to its proximity to the sea, it is an area with oceanic climate. This ensures minimal temperature fluctuations between the seasons and thus making it easier to observe long term trends as well as making the results more general.

These were our initial goals. However, as the project went along, we had to reconsider our aims due to some problems we faced, which are going to be further explained in section 5 Complications. Our renewed goals were aimed at comparing the different datasets we produced during our analysis (see subsection 3.1 and 3.2). Thus, the now goal of our project was to compare said datasets over the course of the oldest available data for Lund, the year 1780.

# 3 Method

## 3.1 Data preparation

The datasets we were provided with still include Metadata, such as the exact location of the weather station, the different time periods in which the data was gathered as well as information about the quality of the data. Additionally, the data is portrayed in a rather unusual format, using semicolons instead of commas to separate each entry. Since we need clean comma-separated values, as a first step we have to clean the data and replace the separators. Therefore we wrote a shell (bash) script, as this is usually the most straightforward tool for this kind of job. It is called *cleandata.sh*. As a first step, this bash script skips over the first 13 lines of the original datafile, as those are filled with the metadata described above. Afterwards, it iterates through each line; first making sure that the contents are valid data, and then extracting the information for *date*, *time* and *temperature*. Whilst doing this, it gets rid of additionally metadata stored next to the actual data in some lines. The output is then saved into a new file called *processed-data.csv*. In this step, we also leave out the data describing the quality of each observation.

## 3.2 Data extraction

Now that data is prepared in a suitable format, we then read and extract data that we need from the csv file. The goal of this step is to obtain the day and the average temperature of the day,which is critically important for data analysis. First, we need to read csv file line by line.This is achieved by using while loop and getline function, which is the most common and simple way to read file line by line. A much more advanced way to read file is using csv library but it requires knowledge and research of specific csv library, which may take much effort. Next, we are supposed to parse the line and extract data into three variables day,time and temperature.In a typical csv file, data in the same line is separated by comma.We can make use of this feature and define comma as delimiters. By using getline function then we can separate a line into three strings. However, we should convert the type of temperature into double so we can do some calculations later on. We use the simple "¿¿" operator to extract data from stream and put it into variable.By defining the type of variable temperature as double,we achieve type conversion of variable.Then we want to calculate the daily mean temperature.It seems pretty easy but it becomes confusing as the times of measurement differs from day to day.Sometimes the temperature is measured once per day, sometimes twice a day and even three times a day.It means we have to identify whether the measurements take place in the same day and put the measurements of the same day into the same group. Besides, we have to calculate the times of measurement for every specific day so we know the number to be divided for the sum of the all temperature.To create a unique day identifier, we define a map, whose key is day and value is vector of temperature.The key of the map can therefore be served as the identifier and only temperature of the same day of the key will be put into the specific vector.The reason why we use vector is that we can adding elements to it easily and obtain the number of elements of it by simply calling size function, which definitely fits our need to calculate the times of measurement per day. By using map and vector, the confusing part of this problem is solved. Then we just make use of for loop to sum up the value of elements for each vector and divide it by the times of measurement of

that day and we can finally get daily mean temperatures. The output of this C++ code is Extracted-data.csv, which contains the day and its mean temperature.

### 3.3 Analysis

The goal of this step was to create histograms based on our datasets using ROOT-libraries. As a good way to plot data using ROOT is to use vectors, we first need to extract the temperature data from the csv-files. To facilitate this step, we make use of the *commata*-library to parse csv-files. Documentation for this library can be found here. First, we create the three needed vectors to store our data, two strings for day and time and a double for temperature. Then, we make use of the class commata::table_scanner to set up an object *scanner* to parse through our csv-file, starting in the first line. The corresponding function .set_field_scanner() is then used to configurate the scanner-object, assigning each column in our data to a vector. Finally, commata::parse_csv() is used to parse the datafile, using our set up scanner. Std::ifstream() enables us to input our csv-file. Now, as the data is ready, we first set up our Canvas to draw our histogram on. In the next step, we create a histogram named 'temperature data' with a title, axis labels and a range from -20 to 40. Then, we create a simple for-loop: We loop through the temperature vector 767 times, filling each value into our histogram. This number equals exactly the amount of data entries for the year 1780 in Lund. Finally, we set the histogram colour to red and draw it on the earlier created Canvas, saving the canvas as 'temp.pdf'. Thus, we have created a histogram showing the occurrence of temperatures in Lund throughout the year 1780, using our *processed-data.csv* dataset.
Since we want to compare both datasets we produced, we iterate this process for the second one, *Extracted_data.csv*. Thus, we redo the just described steps, but with one vector less as the time-data was left out in the extracting step as described above.

## 4 Results

The results we produced can be seen in Figure 1 and Figure 2. They portray the different temperatures which were measured in the year 1780 in the Lund weather station and their prevalence. As explained above, the two histograms are a result of using both datasets we produced. Therefore, Figure 1 shows the occurrence of temperatures with one to three observations per day, taken at different times. Therefore, Figure 1 has 767 entries, the number of observations in the year 1780. Figure 2 however displays only one observation per day, with average values for each day if multiple temperatures were recorded, thus only having 365 entries. Notably, the mean temperature using multiple entries a day is .32 °C warmer than when using daily averages. Another difference between both histograms can be found in the standard deviations: the standard deviation is .24 smaller for Figure 2. This was to be expected, as taking daily averages automatically deletes the more extreme values (which would have been observed in the morning or the night), therefore making the data less dispersed around the mean. Regarding the validity of our results, there are no obviously wrong measurements, as the range of temperatures is very likely to have happened. However the almost bimodal distribution of values raises some suspicion. Most likely, this is due to the different times when each temperature was observed. With more time, we could have produced a third histogram using only observations gathered at the same time, thus eliminating this bias.

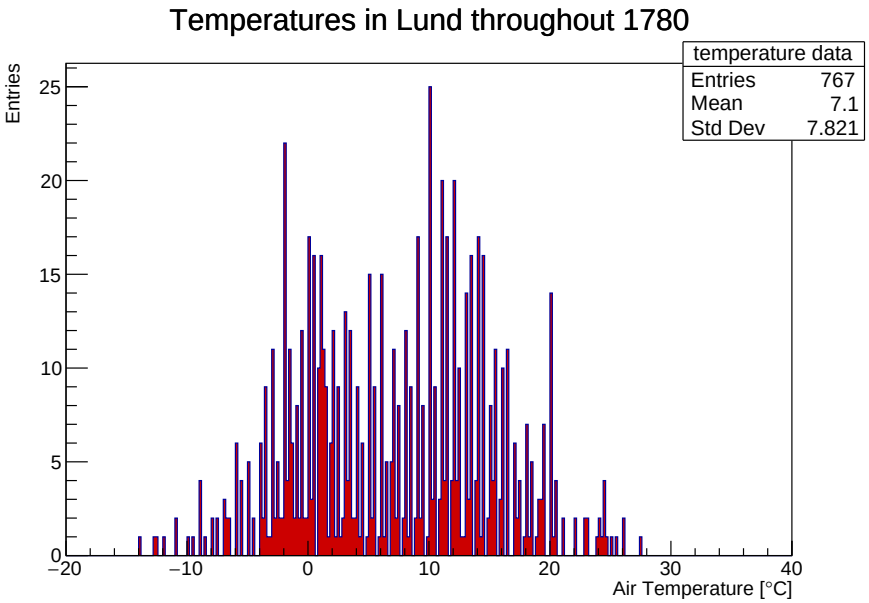


Figure 1: Multiple daily entries, *processed-data.csv*

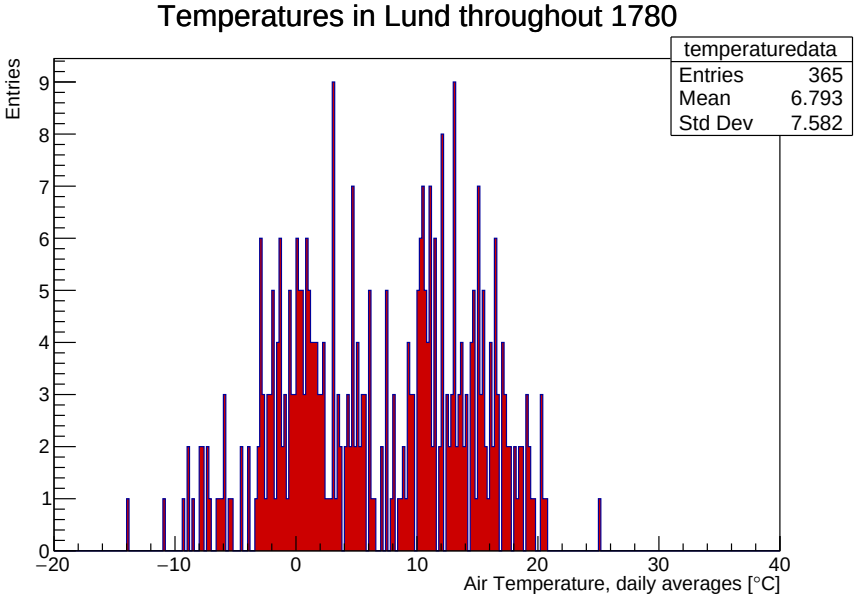


Figure 2: one daily entry, *Extracted-data.csv*

# 5 Complications

As you might have noticed, we were not able to reach our goals and carry out our planned analyses. This came as a result of multiple complications and problems we faced during our process, which we will now describe. We faced difficulties with implementing libraries where we spent a lot of time doing Homework Tut 8-11. Also the bash script was very slow to produce pre processed data. Overall somewhat overwhelmed with scope of the task, difficult to keep overview.We were not able to produce planned results, project is not as structured as would be optimal.BUT: still produced interesting plots.

# 6 Conclusion

We have successfully pre-processed and extracted the Swedish climate data by using bash script and C++. Data Analysis is done by plotting histograms using Commata and ROOT libraries.Overall,We gained some experience how to deal with C++ language,ROOT,Bash script and pushing and pulling by using Github.