MNXB11 - Introduction to Programming and Computing for Scientists
Project

Team C
Lucas Åstrand, Georgios Floros, Adib Shaker, Keyu Yang

November 5, 2024

1 Summary

The aim of this project is to analyze weather data collected from various cities in Sweden using the tools we learned in this course, including Bash, C++, and ROOT. We began by preparing the provided data, which was in CSV format, using a Bash script to remove comments and headers. Next, we converted the cleaned data into a ROOT file for analysis. With this ROOT file, we were able to produce the following results.

2 Data Structure & Preparation

The data, provided by SMHI (Sveriges Meteorologiska och Hydrologiska Institut), is in *.csv* format and contains historical temperature records from various climate observation sites, spanning from the 16th century to 2023. The dataset is organized into four columns: date, time, temperature values, and a measurement quality metric, which indicates higher quality data with 'green' and lower quality data with 'yellow'. The data was cleaned and prepared using a Bash script called smhicleaner.sh, that got rid of everything but the dataset.

In order to effectively make use of the dataset in C++ and ROOT, a class is defined and used to fill a root TTree, which is subsequently saved in a TFile. Within the TFile, data is stored on a measurement branch, where the individual values (corresponding to the columns of the csv file) are stored as leaves. The date and csv parsing libraries are used to parse the dates and times, originally given in yyyy-mm-dd and hh:mm:ss formats respectively, converting them into separate components. This structure allows for an efficient representation of the data, providing access to any statistic by simply giving a parameter. For instance, one can easily extract the temperature measurement for a given date and time.

3 Structure of the Project

The execution of the entire project follows this structure:

1. The initial csv data file is cleaned and prepared by the smhicleaner.sh for further analysis.
2. The measurement.cxx file defines the Measurement class, which encapsulates environmental measurement data, including date and time components, temperature, and measurement quality, along with corresponding constructors, destructors, and setter/getter methods.

3. The data is converted to root format, using the writeTree.cxx file, which parses the date and time, populates a Measurement object with this information, and fills a ROOT TTree with the populated measurements for later analysis.
4. The execution of the ROOT functionalities in the project is automated using the root_macro.C file.
5. The run_project.sh script serves as the main automation tool for executing the project.

4 given_year_temp()

The purpose of this function is to generate a graph displaying the average daily temperature for each day of a specified year, provided as input when the function is called. To achieve this, the program first calculates the average temperature for each day and then maps it to a corresponding day on the graph. This mapping system defines a key for each day of the year (specifically, the day's sequential number within the year) that is the same for all the entries of the same day. The function accounts for the varying number of days in each month, as well as leap years, to ensure accuracy.

For a given year, for instance 1998, the program generates a graph with points representing the average temperature for each day and a line connecting these points. Additionally, it fits the following cosine function:

$$y = P_0 + P_1 \cos(\frac{2\pi}{365}x + P_2) \tag{1}$$

to the graph to model seasonal temperature fluctuations, displaying the fit parameters in the ROOT environment. Finally, the graph is saved as a PDF file in the program's base directory. An output example of the mentioned function can be seen in Figure1.

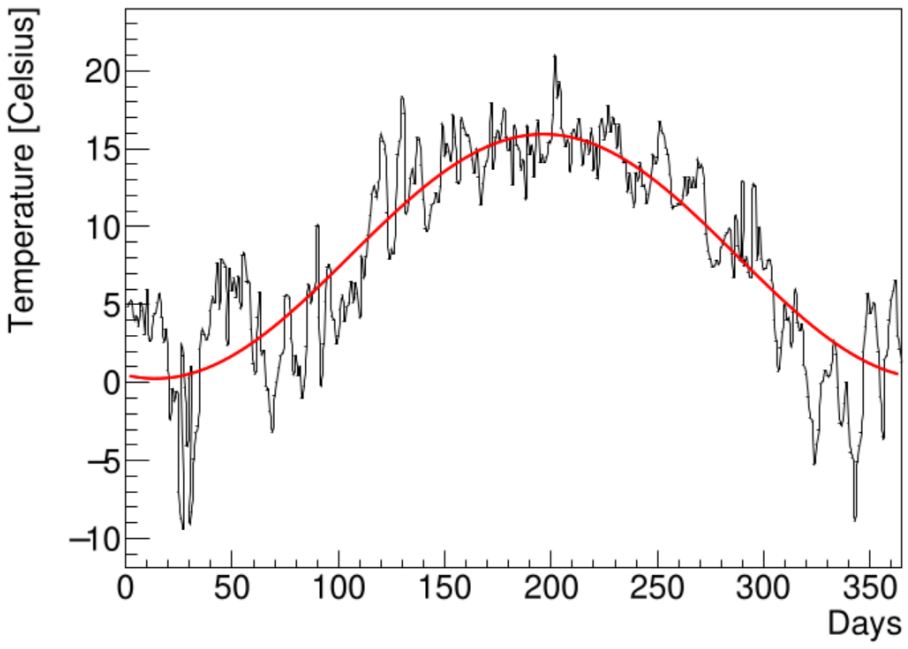

Figure 1: Graph showing the mean temperature of each day for the year 1998, along with a fitted curve based on equation 1.

5 yr_low_high_temp()

This function analyzes of annual extreme temperatures in Lund, Sweden, from 1900 to 2020, excluding the year 1957 due to data irregularities. It visualizes the highest and lowest recorded temperatures for each year from the provided temperature dataset stored in a ROOT file.

Data Processing and Methodology

The function starts by loading the temperature data from the ROOT file into the program, where it calculates the annual maximum and minimum temperatures. However, the data from 1957 was excluded, as incomplete records from that year could compromise the accuracy of the analysis. This focused approach ensures that only reliable data is used for visualizing temperature extremes over the years.

Visualization Approach

The processed data is plotted on a scatter plot with the following details:

- **X-axis**: Represents the years, ranging from 1900 to 2020, excluding 1957.
- **Y-axis**: Depicts the temperature extremes in degrees Celsius.
- **Data Points**: Two distinct color schemes are used:
  - **Blue Points**: Represent the annual lowest temperatures.
  - **Red Points**: Represent the annual highest temperatures.

The scatter plot effectively illustrates trends and variations in temperature extremes over the observed period. The output is shown in Figure 2.

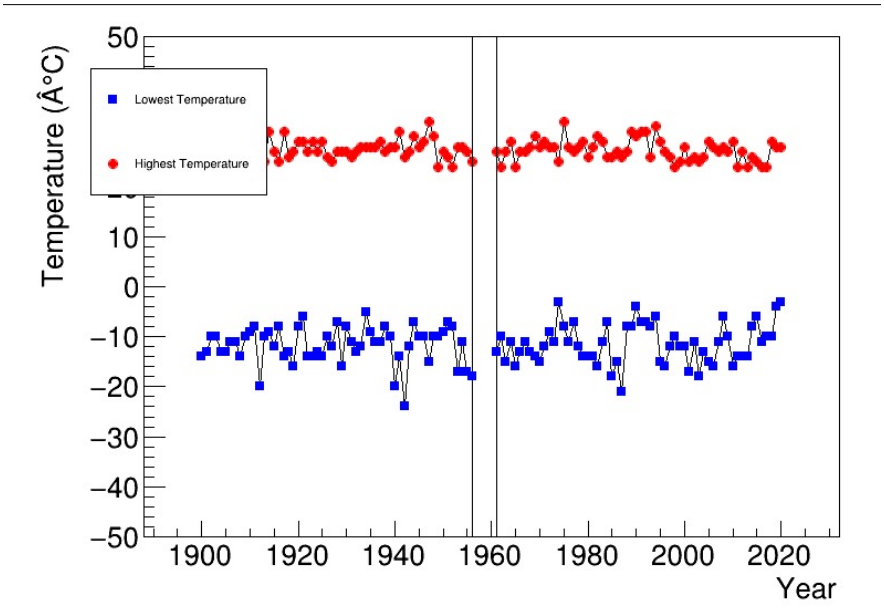

Figure 2: Yearly lowest and highest temperatures in Lund, Sweden, from 1900 to 2020 (excluding year 1957).

6 performFourierAnalysis()

This function performs a Fourier analysis on a given dataset of temperature data, in order to identify periodic patterns and possible temperature cycles over time. The program's operation can be split up into different parts, each performing a different task. The individual parts are described in the following paragraphs.

Data Pre-processing: The program reads the full dataset of temperature measurements and implements a two-pass algorithm in order to firstly calculate daily and then monthly averages of the temperature data. This is done by utilizing a map to correlate temperature measurements to a specific day or month. Following this, the program de-trends the data by subtracting the overall mean temperature. A graph is then filled with the de-trended data, constructing the first portion of the resulting output, a graph displaying the monthly temperature deviation against time.

Signal Processing: In order to reduce possible noise in the data, the program applies a 4-point Gaussian smoothing to the dataset. Later, the program performs a Fast Fourier Transform (FFT) to the de-trended data, extracting the magnitude of the periodic patterns in the data. The result of the Fourier Transform is then used to fill a frequency spectrum graph, constructing the second portion of the output, a graph of the magnitude of the periodic patters against the period.

Visualization: The program finally creates a two-panel visualization, showing the de-trended data in the upper portion, while displaying the frequency spectrum in the lower portion, alongside a short help-panel to aid the understanding of the graph.

Figure 3, shows the results of the *performFourierAnalysis()* function, when applied to a dataset of temperature measurements in Lund, Sweden.

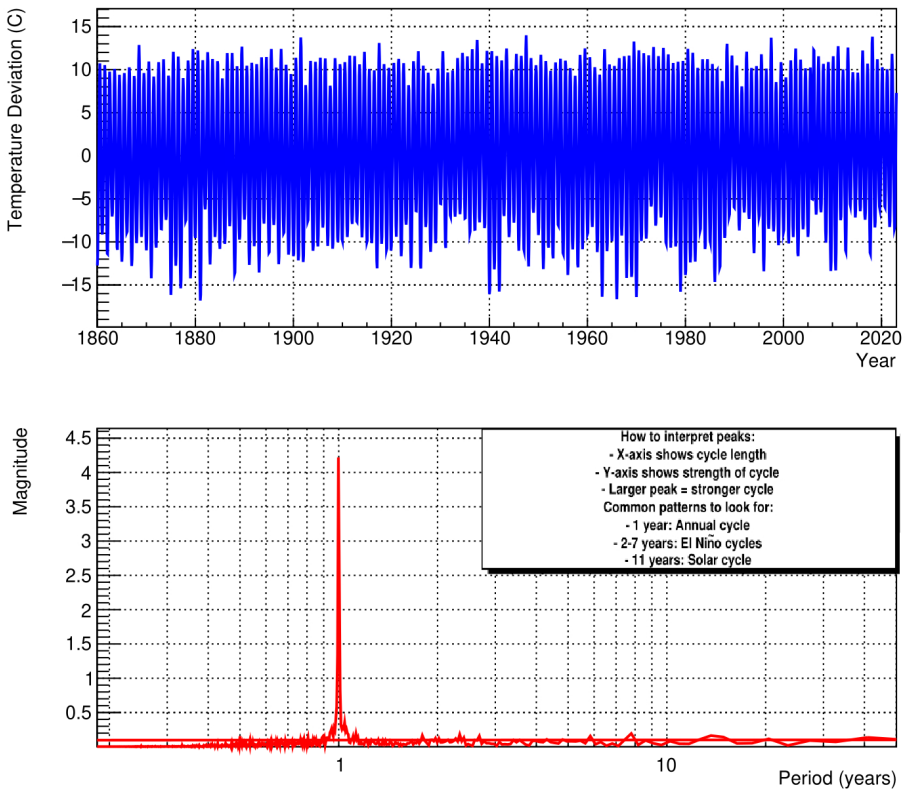

Figure 3: The output of the *performFourierAnalysis()* function.

7 yr_avg_temp()

This function is intended to provide visualized information about the trend of the temperature between two given intervals. The yr_avg_temp() function calculates average temperatures using a map to correlate temperature records by year and month. Later using a linear fit function, a linear correlated line is shown in the plots. Using the slope of the fitted line, the function returns the rate of temperature change within the input time interval. The results can contribute to understanding climate change and its effects in the analyzed region, providing valuable data for further research.

A sample result made by the data from Lund and between 1900 and 2022 is shown in Figure.4

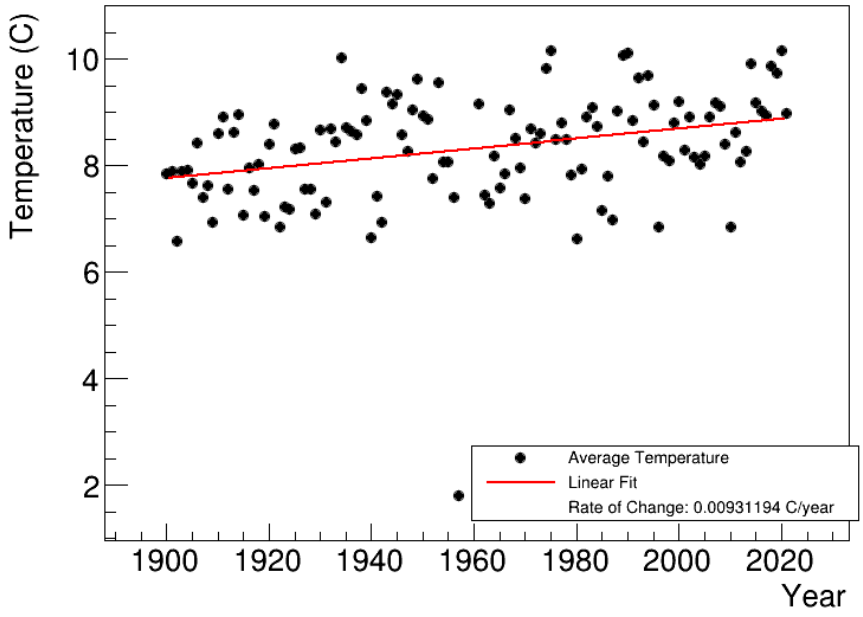

Figure 4: Plot of the average temperature of Lund between 1900 and 2022.

8 outliers()

The outliers() function is designed to identify temperature outliers within a specified date range. It reads data from the ROOT file, calculates outliers based on the Interquartile Range (IQR) method, and writes the outlier information to a text file.

The IQR represents the range within which the central 50% of the data points lie. It is defined as the difference between the third quartile (Q3) and the first quartile (Q1):

$$IQR = Q3 - Q1 \tag{2}$$

where Q1 is the value below which 25% of the data points fall, and Q3 is the value above which 25% of the data points fall. An outlier is defined as a data point that falls below:

$$Q1 - 1.5 \times IQR \tag{3}$$

or above:

$$Q3 + 1.5 \times IQR \tag{4}$$

A sample of outliers detected by this function from Lund's data and between 1800 and 2022 is shown below

```
Outlier Temperatures from 1800 to 2022:
1800-01-02: -19 C
1800-01-03: -19 C
1802-01-11: -19.5 C
1802-01-12: -24 C
1802-01-12: -21.8 C
1802-01-12: -22 C
1802-01-13: -20 C
1802-01-13: -21.2 C
1802-01-14: -24 C
.
.
```