**Manual**

**Content**

***1. Input Interface***

1. **Adding files**
2. **XML-file**

***2. Script***

1. **Examples operations**
2. **Answering questions**

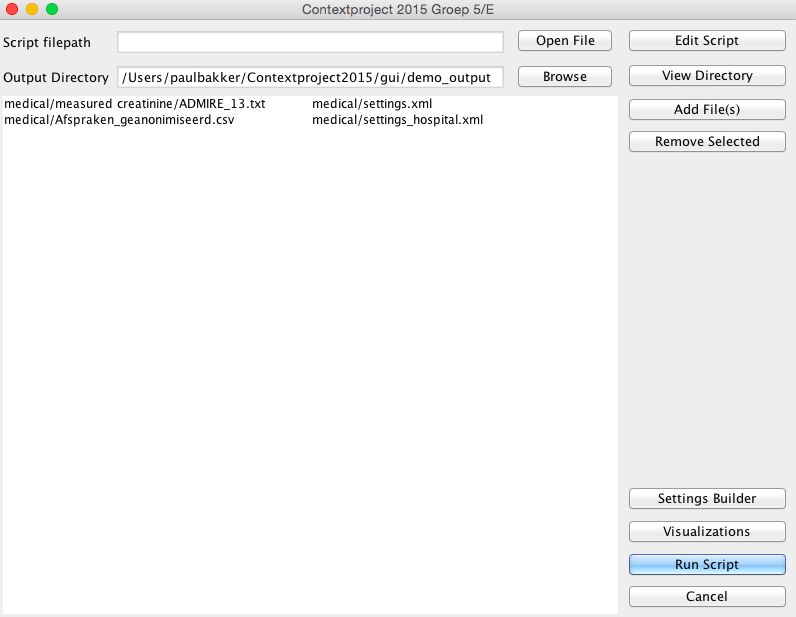
***3. Progress Interface***

***4.***﷒﷒ ***Visualization Interface***

1. **Frequency chart**
2. **Box plot**
3. **State transition diagram**
4. **Stem leaf plot & histogram**

**1. Input Interface**

When you start this program this is the first thing you will see. Here you can import all the necessary files to start the analysis of the program.

****

As you can see we can select the script file at the top of the program. After it’s selected we can edit the script by pressing the edit script button. This will open the default editor of the file specified by user’s operating system.

Beneath that we can select the output directory and view it if necessary. If the directory doesn’t exist it will be created.

In the bottom right you can jump straight to the visualizations, as well as the builder of the XML-files or start running your script.

**1.1 Adding files**

In the middle part of this interface you can see the files you have selected and start adding more by pressing the “Add File(s)” button. You can import csv files, as well as excel files into this program. For each file you will need to select a separate XML file so the program can detect all the columns and formats in the data file.

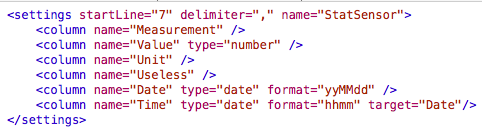
**1.2 XML-file**

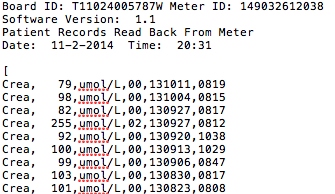
For each data file there has to be a XML-file select so that the program can detect the column names and the format of the file.

This is what is specified in the XML-file:

* + *Name of the table*
  + *Delimiter of the values*
  + *Start line of the data*
  + *Each column with its name and format*

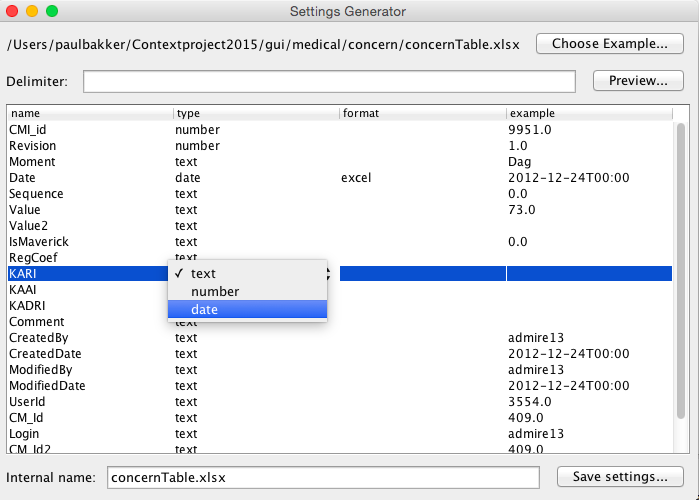
Here is an example of a XML-file for the given data file:

****

****

The term target is only used in the case of time. This is used to couple the date value with the time value so we can sort the created table by time.

Creating a XML-file for each file you want to import can be cumbersome so we also created a Settings builder. This will create most of the XML-file automatically for the user. The user only needs to specify the delimiter and format of each column.

****

To make importing files that were the output of a previous analysis much easier we automatically create a XML-file of the output, making using the program in rapid succession a lot faster.

**2. Script**

To perform analysis on the data you need to create a script. In the script you specify what kind of operation you want to perform in a given order, so performing sequential data analysis. Here we will give some examples of each operation and the combining of operations. The detailed syntax will be a separate document.

**2.1 Examples operations**

Here we will give an example for each operation to make clear what is does:

**CONSTRAINT**

*Table with the name website*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Login | Date | Value | Comment | MeasVar |
| Admire13 | 23-06-14 | 145 | Wandelen | Crea |
| Admire13 | 24-06-14 | 140 | Fietsen | Crea |
| Admire14 | 23-06-14 | 164 | Lopen | Crea |
| Admire13 | 25-06-14 | 154 | Rennen | Crea |
| Admire15 | 23-06-14 | 163 | Lopen | Crea |
| Admire16 | 23-06-14 | 143 | Eten | Crea |
| Admire13 | 26-06-14 | 137 | Drinken | Crea |

We can filter on each column given a specific value. So if we use the following script:

**CONSTAINT [website].[Login] == “Admire13”**

This has as output the following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Login | Date | Value | Comment | MeasVar |
| Admire13 | 23-06-14 | 145 | Wandelen | Crea |
| Admire13 | 24-06-14 | 140 | Fietsen | Crea |
| Admire13 | 25-06-14 | 154 | Rennen | Crea |
| Admire13 | 26-06-14 | 137 | Drinken | Crea |

**CODE**

*Table with the name website*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Login | Date | Value | Comment | MeasVar |
| Admire13 | 23-06-14 | 145 | Wandelen | Crea |
| Admire13 | 24-06-14 | 140 | Fietsen | Crea |
| Admire14 | 23-06-14 | 164 | Lopen | Crea |
| Admire13 | 25-06-14 | 154 | Rennen | Crea |
| Admire15 | 23-06-14 | 163 | Lopen | Crea |
| Admire16 | 23-06-14 | 143 | Eten | Crea |
| Admire13 | 26-06-14 | 137 | Drinken | Crea |
| Admire13 | 27-06-14 | 133 | Drinken | Crea |

We can create a code for different kind of behaviors in the data. If we want to add a code to the records where the comment “Eten” is followed by “Drinken” only once.

**CODE [website] ON {1 [website].[Comment] == “Eten” } {1 [website].[Comment] == “Drinken” } AS “code\_eten\_drinken”**

This has as output:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Login | Date | Value | Comment | MeasVar | Code |
| Admire13 | 23-06-14 | 145 | Wandelen | Crea |  |
| Admire13 | 24-06-14 | 140 | Fietsen | Crea |  |
| Admire14 | 23-06-14 | 164 | Lopen | Crea |  |
| Admire13 | 25-06-14 | 154 | Rennen | Crea |  |
| Admire15 | 23-06-14 | 163 | Lopen | Crea |  |
| Admire16 | 23-06-14 | 143 | Eten | Crea |  |
| Admire13 | 26-06-14 | 137 | Drinken | Crea |  |
| Admire13 | 27-06-14 | 133 | Drinken | Crea |  |

As you notice, no code is added as “Eten” in column Comment isn’t followed by “Drinken” once, but twice. We can change the 1 into a 2, but if we don’t care how many times “Eten” is followed by “Drinken” but still want to add these records to a code we can simply use this code:

**CODE [website] ON {1 [website].[Comment] == “Eten” } {\* [website].[Comment] == “Drinken” } AS “code\_eten\_drinken”**

Which now gives as output:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Login | Date | Value | Comment | MeasVar | Code |
| Admire13 | 23-06-14 | 145 | Wandelen | Crea |  |
| Admire13 | 24-06-14 | 140 | Fietsen | Crea |  |
| Admire14 | 23-06-14 | 164 | Lopen | Crea |  |
| Admire13 | 25-06-14 | 154 | Rennen | Crea |  |
| Admire15 | 23-06-14 | 163 | Lopen | Crea |  |
| Admire16 | 23-06-14 | 143 | Eten | Crea | Code\_eten\_drinken |
| Admire13 | 26-06-14 | 137 | Drinken | Crea | Code\_eten\_drinken |
| Admire13 | 27-06-14 | 133 | Drinken | Crea | Code\_eten\_drinken |

**CONNECT**

*Table with the name website*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Login | Date | Value | Comment | MeasVar |
| Admire13 | 23-06-14 | 145 | Wandelen | Crea |
| Admire13 | 24-06-14 | 140 | Fietsen | Crea |
| Admire13 | 25-06-14 | 164 | Lopen | Crea |
| Admire13 | 26-06-14 | 154 | Rennen | Crea |
| Admire13 | 27-06-14 | 163 | Lopen | Crea |
| Admire13 | 28-06-14 | 143 | Eten | Crea |
| Admire13 | 29-06-14 | 137 | Drinken | Crea |
| Admire13 | 30-06-14 | 133 | Drinken | Crea |

*Table with the name sensor*

|  |  |  |
| --- | --- | --- |
| Value | CreateDate | Time |
| 145 | 23-06-14 | 09:15 |
| 140 | 24-06-14 | 09:43 |
| 163 | 25-06-14 | 09:56 |
| 255 | 26-06-14 | 05:34 |
| 163 | 27-06-14 | 23:12 |
| 143 | 27-06-14 | 13:21 |
| 137 | 28-06-14 | 11:11 |

We can connect these two tables by selecting on which column we want to merge for each table:

**CONNECT [website].[Date] TO [sensor].[CreateDate]**

This will merge the two Date columns no matter the name and sort the table based on the created column. Columns with the same name will also be merged into one, while columns that only exist in one table will left empty in the records of the other table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Login | Date | Value | Comment | MeasVar | Time |
| Admire13 | 23-06-14 | 145 | Wandelen | Crea |  |
|  | 23-06-14 | 145 |  |  | 09:15 |
| Admire13 | 24-06-14 | 140 | Fietsen | Crea |  |
|  | 24-06-14 | 140 |  |  | 09:43 |
| Admire13 | 25-06-14 | 164 | Lopen | Crea |  |
|  | 25-06-14 | 163 |  |  | 09:56 |
| Admire13 | 26-06-14 | 154 | Rennen | Crea |  |
|  | 26-06-14 | 255 |  |  | 05:34 |
| Admire13 | 27-06-14 | 163 | Lopen | Crea |  |
|  | 27-06-14 | 143 |  |  | 13:21 |
|  | 27-06-14 | 163 |  |  | 23:12 |
| Admire13 | 28-06-14 | 143 | Eten | Crea |  |
|  | 28-06-14 | 143 |  |  | 11:11 |
| Admire13 | 29-06-14 | 137 | Drinken | Crea |  |
| Admire13 | 30-06-14 | 133 | Drinken | Crea |  |

**CHUNK**

*Table with the name sensor*

|  |  |  |
| --- | --- | --- |
| Value | CreateDate | Time |
| 145 | 23-06-14 | 09:15 |
| 140 | 24-06-14 | 09:43 |
| 163 | 25-07-14 | 09:56 |
| 255 | 26-07-14 | 05:34 |
| 163 | 27-07-14 | 23:12 |
| 143 | 27-07-14 | 13:21 |
| 137 | 28-08-14 | 11:11 |

We can split the data into different chunks based on time. So we can chunk for example on year, month and day. We can also chunk on a number of years, months or days. In this example we will chunk on one month:

**CHUNK [website] USING MONTH 1**

This gives the following output:

|  |  |  |  |
| --- | --- | --- | --- |
| Value | CreateDate | Time | Chunk |
| 145 | 23-06-14 | 09:15 | Chunk 0 |
| 140 | 24-06-14 | 09:43 | Chunk 0 |
| 163 | 25-07-14 | 09:56 | Chunk 1 |
| 255 | 26-07-14 | 05:34 | Chunk 1 |
| 163 | 27-07-14 | 23:12 | Chunk 1 |
| 143 | 27-07-14 | 13:21 | Chunk 1 |
| 137 | 28-08-14 | 11:11 | Chunk 2 |

**BETWEEN**

*Table with the name website*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Login | Date | Value | Comment | MeasVar |
| Admire13 | 23-06-14 | 145 | Wandelen | Crea |
| Admire13 | 24-06-14 | 140 | Fietsen | Crea |
| Admire13 | 25-06-14 | 164 | Lopen | Crea |
| Admire13 | 26-06-14 | 154 | Rennen | Crea |
| Admire13 | 27-06-14 | 163 | Lopen | Crea |
| Admire13 | 30-06-14 | 143 | Rennen | Crea |
| Admire13 | 29-06-14 | 137 | Drinken | Crea |
| Admire13 | 30-06-14 | 133 | Drinken | Crea |

We can determine the time between the comment “Lopen” and “Rennen” by using the following operation:

**BETWEEN [website].[Comment] [Website].[Date] [Website].[Date] "Lopen" "Rennen”**

This will give the following output in which the values in time\_before are hours:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Login | Date | Value | Comment | MeasVar |  |
| Admire13 | 23-06-14 | 145 | Wandelen | Crea | Time\_before |
| Admire13 | 24-06-14 | 140 | Fietsen | Crea |  |
| Admire13 | 25-06-14 | 164 | Lopen | Crea | 24 |
| Admire13 | 26-06-14 | 154 | Rennen | Crea |  |
| Admire13 | 27-06-14 | 163 | Lopen | Crea | 72 |
| Admire13 | 30-06-14 | 143 | Rennen | Crea |  |
| Admire13 | 29-06-14 | 137 | Drinken | Crea |  |
| Admire13 | 30-06-14 | 133 | Drinken | Crea |  |

**COMBINE**

*Table with the name website*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Login | Date | Value | Comment | MeasVar |
| Admire13 | 23-06-14 | 145 | Wandelen | Crea |
| Admire13 | 24-06-14 | 140 | Fietsen | Crea |
| Admire13 | 25-06-14 | 164 | Lopen | Crea |
| Admire13 | 26-06-14 | 154 | Rennen | Crea |
| Admire13 | 27-06-14 | 163 | Lopen | Crea |
| Admire13 | 28-06-14 | 143 | Eten | Crea |
| Admire13 | 29-06-14 | 137 | Drinken | Crea |
| Admire13 | 30-06-14 | 133 | Drinken | Crea |

*Table with the name sensor*

|  |  |  |
| --- | --- | --- |
| Value | CreateDate | Time |
| 145 | 23-06-14 | 09:15 |
| 140 | 24-06-14 | 09:43 |
| 163 | 25-06-14 | 09:56 |
| 255 | 26-06-14 | 05:34 |
| 163 | 27-06-14 | 23:12 |
| 143 | 27-06-14 | 13:21 |
| 137 | 28-06-14 | 11:11 |

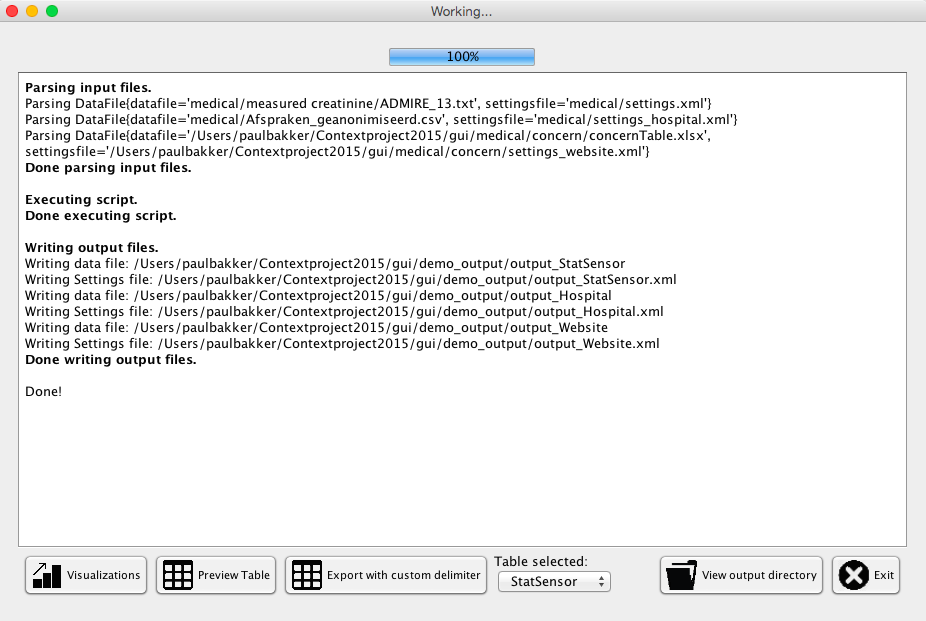
We can combine these two tables by selecting on which column we want to merge for each table, the difference with connect is that records are combined here instead of kept in their original state:

**COMBINE [website].[Date] TO [sensor].[CreateDate]**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Login | Date | Value | Comment | MeasVar | Value | CreateDate | Time |
| Admire13 | 23-06-14 | 145 | Wandelen | Crea | 145 | 23-06-14 | 09:15 |
| Admire13 | 24-06-14 | 140 | Fietsen | Crea | 140 | 24-06-14 | 09:43 |
| Admire13 | 25-06-14 | 164 | Lopen | Crea | 163 | 25-06-14 | 09:56 |
| Admire13 | 26-06-14 | 154 | Rennen | Crea | 255 | 26-06-14 | 05:34 |
| Admire13 | 27-06-14 | 163 | Lopen | Crea | 163 | 27-06-14 | 23:12 |
| Admire13 | 27-06-14 | 163 | Lopen | Crea | 143 | 27-06-14 | 13:21 |
| Admire13 | 28-06-14 | 143 | Eten | Crea | 137 | 28-06-14 | 11:11 |

**3. Progress Interface**

When the script is running we will show what is happening in the program and give errors if something went wrong. After the script is finished this interface will also give you some option on what to do with the output.



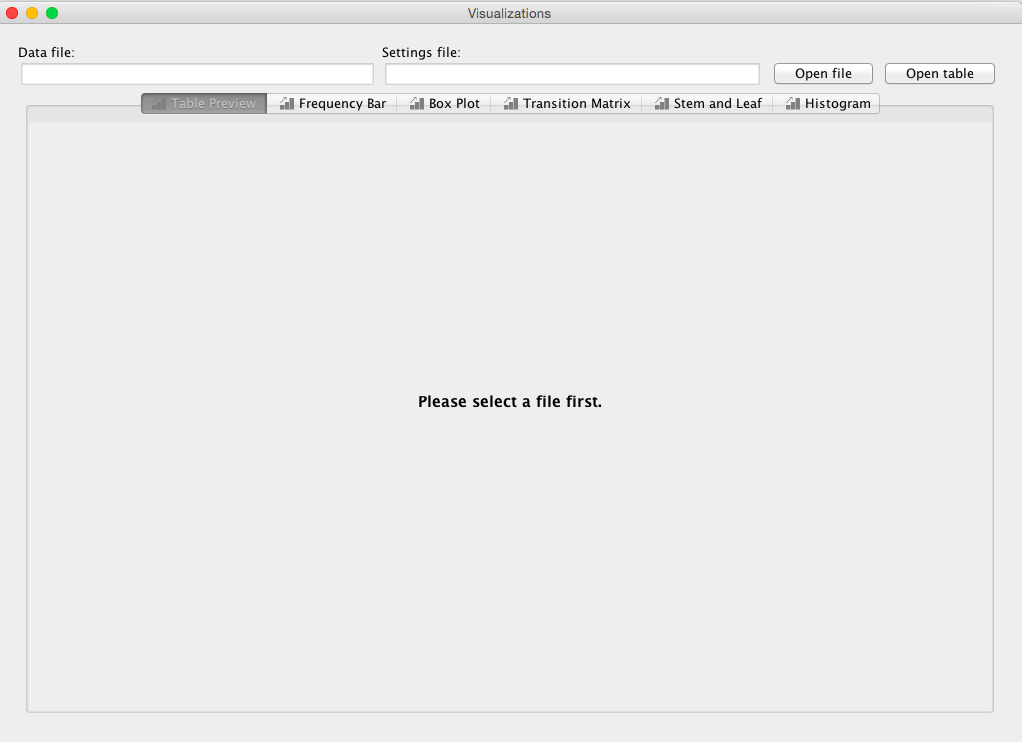
You can start immediately with visualizing the selected table from the output, which we will discuss in the next section of this manual. You can also preview the created table so you can see if the script worked as intended, this also shows the selected table from the output.

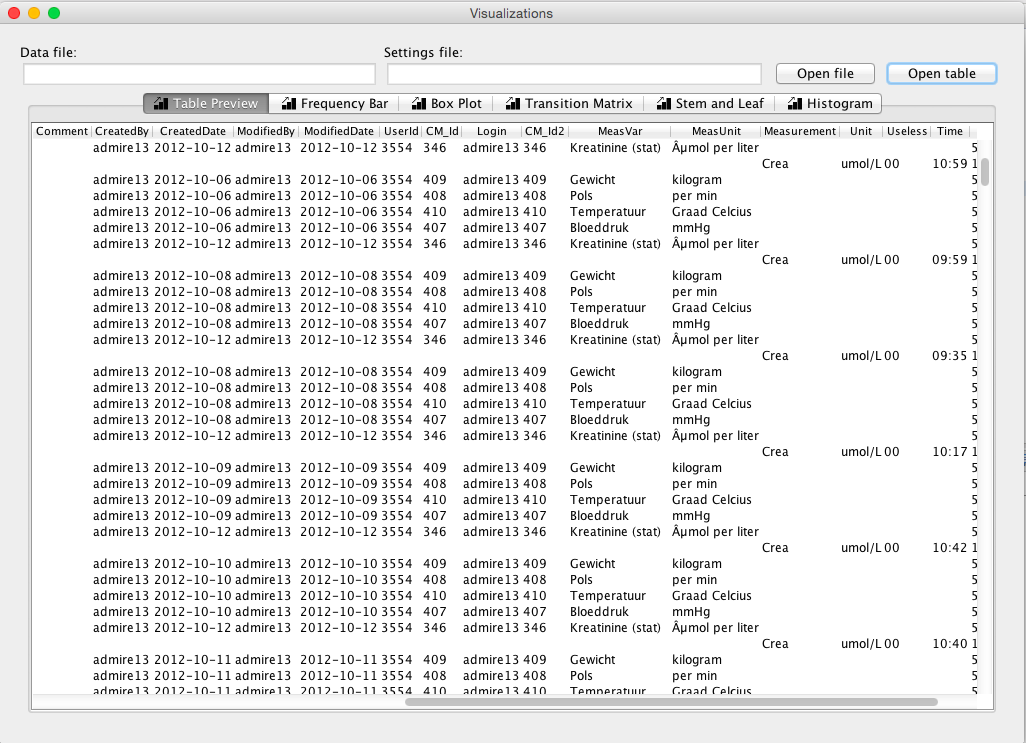
It’s also possible to export a file again with a different delimiter. The default delimiter of this program is ‘;’ as this is used by Excel.

On the right side there is a button to open the output directory where you will find the exported data files, the automatically created XML-files of those data files and serializable files of each data file. The serializable files are used for visualization which we will discuss next.

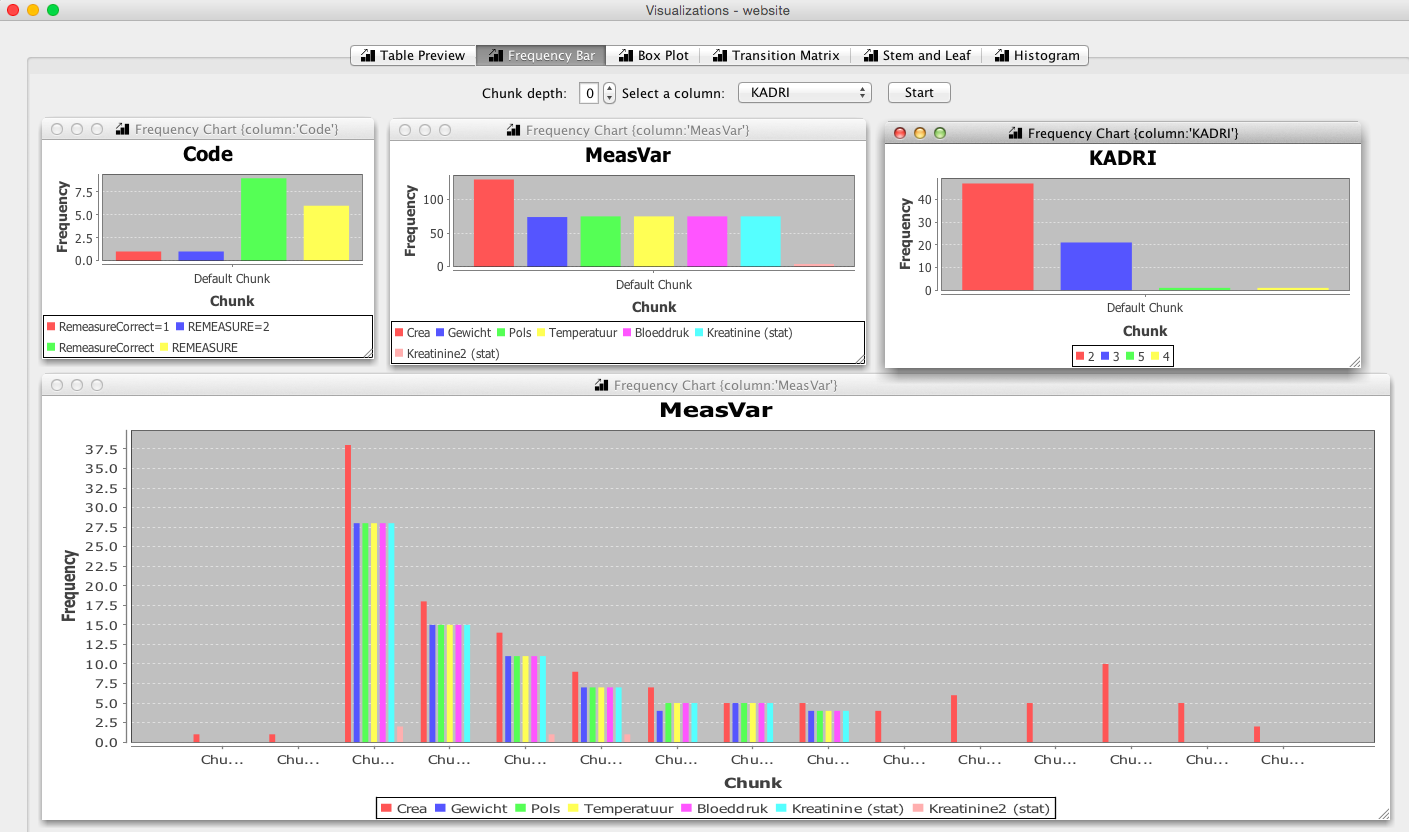
**4. Visualization**

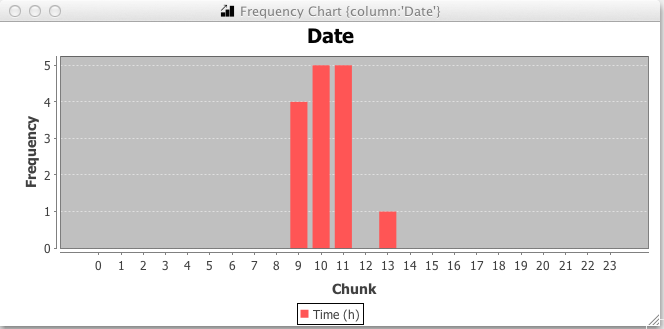
This program can also visualize the data before you load the newly created into an analysis program. We create a ser file separate from the other output files to feed into the visualizations. After you ran the script you can directly jump into the visualization GUI but you can also start the visualization GUI right away and import a ser file. You can also import csv and excel files with their XML-files into the visualization GUI, but not every feature will work if you import those files instead of the ser file. We will discus this in the separate visualizations. If you open a file, you will see the table first and can then determine what to visualize.



****

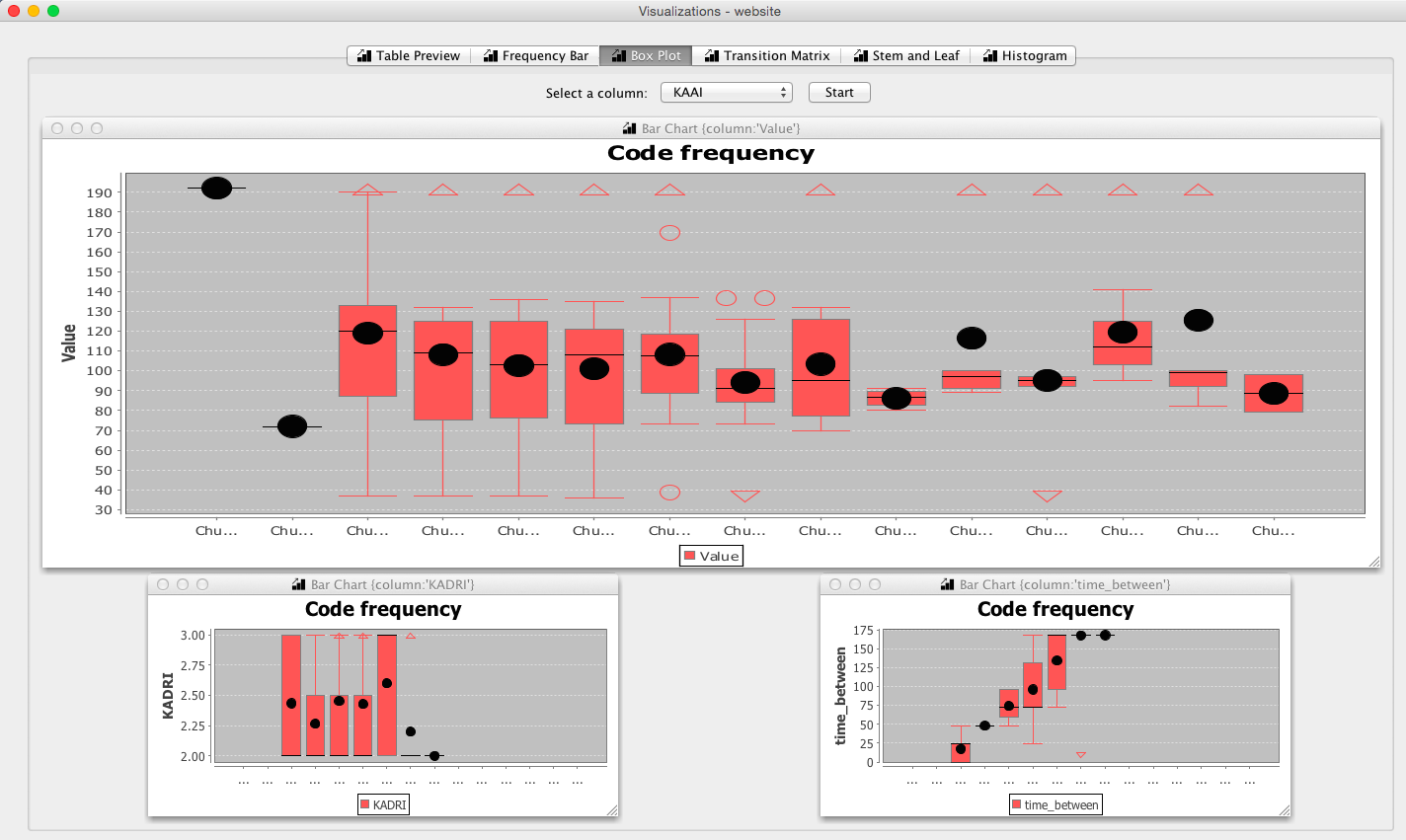
**4.1 Frequency chart**

First you can make a frequency chart based on each column. This can show you the frequency of the unique values in that column. If you have imported a ser file our jumped straight from the progress UI to the visualization you can also determine the frequency of values or codes per chunk. This can also be done if you have chunked more then one time with the chunk depth. Codes can be visualized with the option “\*CODE\*”.

If you have a time column in your file, the frequency has also the possibility to show this without labels but with the hours as indexes as shown below:

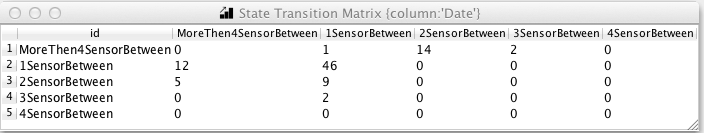
**4.2 Box Plot**

A boxplot can be made of every column in which values are only integers. The columns with strings or dates are automatically filtered out in the dropdown to avoid errors. Importing a ser file doesn’t provide extra features in a box plot. If there are chunks in the data in the boxplot will visualize the data per chunk.



**4.3 State Transition Diagram**

If you want to determine the transitions between codes in the data, you only have to select the column on which you want to sort the data. Only columns containing dates will be visible in the dropdown to avoid errors.



**4.4 Stem Leaf Plot and Histogram**

The final visualization is stem leaf plot and histogram. This will give an indication on what kind of range of values are in a selected column. This is only possible on column with integers and other column are filter out of the dropdown.

