Processing Picogreen standard curve results measured by a plate reader

By Thomas Heigl

2021 January 24th

https://github.com/chuddy-ibk/picogreen-DNA-concentration

**Background information:**

In molecular biology, handling DNA (nucleic acid) is a daily activity. For some experiments, it is crucial to know the precise concentration of the DNA molecules in a carrier substance e.g. in water. One way to measure the DNA concentration is to measure the absorbance of light (260 nm) when shone through the carrier substance as DNA absorbs light with this wavelength well. However, a major disadvantage of this technique is that some proteins and RNAs also absorb at 260 nm, which leads to overestimation of the true DNA concentration if your sample is contaminated. .   
A more precise method exists, which uses a fluorescent dye (e.g. Picogreen) to stain double-stranded nucleic acid. To measure your sample’s DNA concentration you need to know how much fluorescence a specific concentration of DNA emits. This is why the KIT bought from companies deliver their DNA Standard with a known concentration. Diluting the DNA Standard will tell you the fluorescent value for different DNA concentrations. When the concentration is plotted with the fluorescent value, you can appreciate that the increase of DNA leads to an increase of measured fluorescence. With the help of the Standard curve, you can measure your sample’s fluorescence for different dilutions and calculate the concentration.

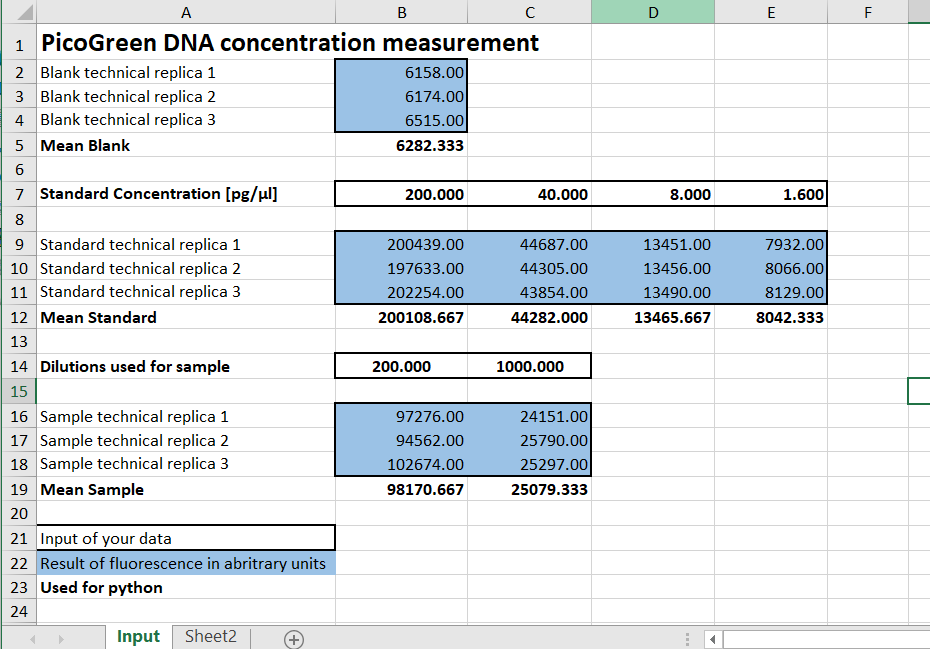
**Motivation:**

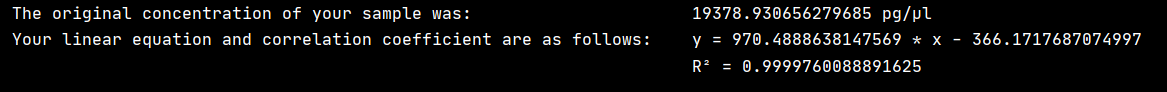
Calculation of the Standard curve as well as of your sample concentration can be cumbersome, which is why Excel is already a useful tool to increase speed of calculations. However, doing the analysis with a program, written in Python, will allow adding more and more features with time so that in the end you put in the data and the program tells you how to continue the experiment.   
Automating some work in the lab will make life much easier and the focus can be put on more important parts of your studies.

**Tutorial:**

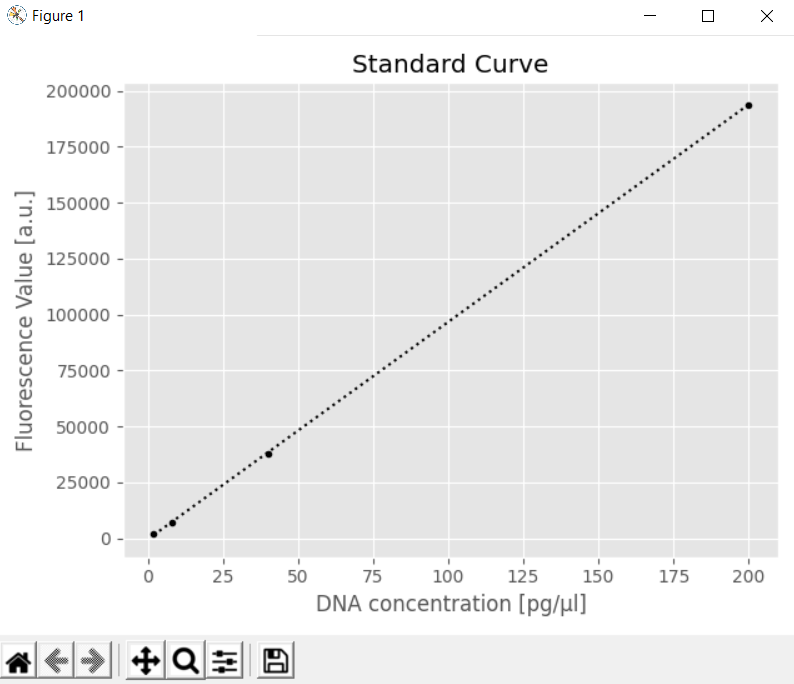
*This program needs following modules:* ***xlrd*** *(2.0.1),* ***numpy*** *(1.19.5) and* ***matplotlib*** *(3.3.3)*

This program uses input from a standard .xls file. The file “test.xls” contains an input sheet which we will access by running the program. In this file, the user input into the .xls file is marked by the cells with thicker borders. The background color indicates the input of fluorescent values you receive from a plate reader. The python program extracts the bold cells. Below your data, I added a legend.



Running “picogreen\_analysis\_no-test-cases.py” will result in following output when the test.xls file has not been changed yet:   


and a new window with a plotted graph will be opened:



The .xls file can be changed to provoke programmed error calls, e.g. deleting a value from a cell, which is used for extraction from the program. As an example, when B5 is deleted, following error message is raised: “ValueError: No numbers were found in named row: 'Mean Blank'”

The “picogreen\_analysis.py” includes also the 32 test cases and is the program in total. Putting everything into a folder and copy-pasting the folder somewhere else did not change the functionality of my program. Only the print-out of the tests looked differently, however, all the tests were still correct.

**Reflection on implementation:**

*I had the test.xlsx file as a template, which I have programmed for myself but also for others’ use and so I oriented the python program to the excel one in regards of looking for the correctness of the results.*

This was, as expected, many hours of work. The preliminary product for this take-home examination is still a well working program. For practicing reasons I also tried to implement as much from this course as possible. This is why I tried to handle lists, tuples and dictionaries. Maybe it would be simpler written with fewer data types. I also had one early version with direct user input to tell the file name to be opened, however this would only increase the lines of code and I already have much.

**Programming and Testing:**

Finding necessary functions in modules I have not heard of, lead to much trial and error approaches. Furthermore, when working with external files, the user can manipulate so many different parameters that in the end I decided to give the user some reminders when something in the template file was changed, however, at some point I needed to assume that the user does only handle the input area. I don’t know if doc-tests can be used here efficiently.

Confusing for me was how to test for something that is not used in the argument of a function. For example, my program uses one function to look at the first column to find out where the important rows are by finding distinct names in that column. If a name is found two times in that row, an error is raised but I could not think of a test for that error raise. Only if you change the –xls file directly you will reach that error call. There are some other error calls that can be reached while changing the .xls file. One messy and crowdy solution is to create a .xls file for every error call but this would lead to several test files and maybe confusion.

**Plans:**

I want to continue on this program, as I am halfway through I think.

The sample concentration should be used to calculate the DNA molecules per µl and then the program should tell me how much I need to dilute my sample to reach a wanted DNA concentration. It should also tell me if I have enough sample or if I need to start my following experiment with a lower concentration (after I put in the amount of sample I have of course). In the end my sample will be used as a Standard curve for quantifying virus DNA and therefore the number of viral particles I was able to produce. The test.xlsx file shows some of the future steps to be programmed.   
I also want to find out how to make my program an executable so that people won’t need to download python but only execute the program and put in the results in the .xls file.