ISSN 1846-6168 (Print), ISSN 1848-5588 (Online)
https://doi.org/10.31803/tg-20210326070107

ATLaS: Assistant Software for Life Scientists to Use in Calculations of Buffer Solutions

Ugur Comlekcioglu*, Nazan Comlekcioglu

Abstract: Many solutions such as percentage, molar and buffer solutions are used in all experiments conducted in life science laboratories. Although the preparation of the solutions is not difficult, miscalculations that can be made during intensive laboratory work negatively affect the experimental results. In order for the experiments to work correctly, the solutions must be prepared completely correctly. In this project, a software, ATLaS (Assistant Toolkit for Laboratory Solutions), has been developed to eliminate solution errors arising from calculations. Python programming language was used in the development of ATLaS. Tkinter and Pandas libraries were used in the program. ATLaS contains five main modules (1) Percent Solutions, (2) Molar Solutions, (3) Acid-Base Solutions, (4) Buffer Solutions and (5) Unit Converter. Main modules have sub-functions within themselves. With PyInstaller, the software was converted into a stand-alone executable file. The source code of ATLaS is available at https://github.com/cugur1978/ATLaS.

Keywords: ATLaS; buffer; Pandas; Python; solution; Tkinter

1 INTRODUCTION

There is a high demand for computer-based support for diverse scientific research, but the software that suits the need cannot always be found [1, 2]. While researchers lack the necessary programming skills to develop the required software, software developers are generally unfamiliar with scientific research [3]. Although it is necessary to conduct interdisciplinary studies to solve this problem, it is important to train software developers who are experts in certain scientific fields [4, 5]. Because the cooperation to be achieved by the design, development and distribution of free software by scientists will enable researchers to spend more time on their research [6].

Using programming languages such as C++, Python, Fortran or Pearl, a variety of software, pipelines and packages have been developed to meet all analysis demands of scientific or industrial researchers. Examples of these software include BioCoder[7], Tellurium [8], Biotite [9]. Various web-based tools allow researchers to handle their studies outside of the command-line environment; however, these programs need a stable internet connection and thus hinder offline analysis. Internet-free software with a graphical user interface (GUI) has greatly helped researchers in their work [10].

Python is an interpreted, object-oriented, open-source language, has possibly become the actual standard for investigative, interactive, and computation-driven scientific research [11]. Its ease of learning and use combined with the open-source nature of the language make it an ideal platform for scientific computing [8]. Python help develop computational research tools by providing a balance of clarity and flexibility without decreasing performance [12]. Python stands out with its scientific libraries as well as features such as object oriented and functional programming [13]. Scientific softwares use of a very rich available collection of scientific libraries such as SciPy, NumPy, and Matplotlib. SciPy, for example, has become the de facto standard for leveraging scientific algorithms in Python, with over 600 unique code contributors, thousands of dependent packages, more than 100,000 dependent repositories [14].

In scientific applications, requirements can vary greatly, from basic scripts for usual data analysis with pre-existing toolkits and software to complex general-purpose structures that search to automate and facilitate a wide variety of computational jobs [15]. The place of experimental study in laboratories has always supposed a high profile at all levels of science. However, errors in laboratory experiments can occur in any step of the process. Casadevall et al. [16] reported that more than half of all error-related retractions were attributed to laboratory error. Experimental errors have negative effects in terms of cost, time and effort. In addition, insufficient description of materials and methods limits the reproducibility of the scientific researches [17]. Methods reproducibility mentions to the providing of adequate detail about study protocols and data so the same protocols could, in theory or in actuality, be completely repeated [18]. Protocols need to be highly detailed in order to reduce experimental errors and increase the reproducibility of experiments. However, methods in the literature are usually given without a detailed and complete account of procedures [7].

In this study, we are introducing a stand-alone software ATLaS (Assistant Toolkit for Laboratory Solutions), which is an assistant toolkit that includes basic and widespread calculations of solutions with a user-friendly GUI. This toolkit developed to save time and energy of wet-lab researchers from various calculations of preparing solutions. Additionally, the purpose of this software is to minimize calculation errors in solution preparation in experiments. The code of ATLaS is open, where the software's code could be altered, enhanced or combined in various laboratory softwares. ATLaS is a basic package software that carries out a range of calculations to prepare various solutions for a variety of scientific experiments. Considering that not every laboratory has access to the internet, ATLaS is intended to be software independent from the internet.

2 IMPLEMENTATION

ATLaS is platform-independent software that can be run under all operating systems. The code of the software has been licensed under MIT and freely available at https://github.com/cugur1978/ATLaS. ATLaS was developed using Python 3.8.5, an interpreted object-oriented high-level programming language that has become broadly adopted in the scientific community due to its clean and open syntax with uncomplicated semantics that make it instinctive to learn, active developer community and free availability [19, 20]. Additionally, Python is very beneficial while building a prototype more quickly with less code. PyCharm 2020.3.2 was used as the coding editor. PyCharm offers a smart code editor and a powerful graphical debugger [21]. Tkinter package is very useful to construct the GUI that contain components such as windows, list boxes, entries etc. Tkinter supplies usability to link codes and functions by pressing buttons and show the output in a variety of options determined by the developer [22].

Modularization is the process of organizing the code into reusable and parametrizable components. Modularization prevents errors, makes code more readable and testable, and simplifies code reuse over redundant duplication [15]. Therefore, ATLaS is designed to consist of 5 main modules. Functions are placed in these modules. A function is a block of code that expresses the solution to a small, independent problem or task. Structuring a program by functions also helps with the modularity and interpretability of the program code and ultimately simplifies the debugging process - an important consideration in all programming projects [19]. The modules used in the calculation of percent and molar solutions contain 3 functions each. These functions perform (1) calculation of percent/molar solutions, (2) dilution and (3) percent/molar conversions. Chemicals in Acid & Base and Buffer Modules are presented to the user as a list. Information such as empirical formula, formula weight of these chemicals are saved in Microsoft Excel files. ATLaS makes calculations by calling these values from the Excel files. Unit Converter module performs concentration conversions under volume, mass and density options. PyInstaller was used to convert ATLaS to a standalone executable software. PyInstaller gathers packages accessed in python software and assemblies locally installed packages in the directory where the application can find the packages in this directory and take the required function from these packages in place of the function in the system. The flow chart used in the design of the software is given in Fig. 1.

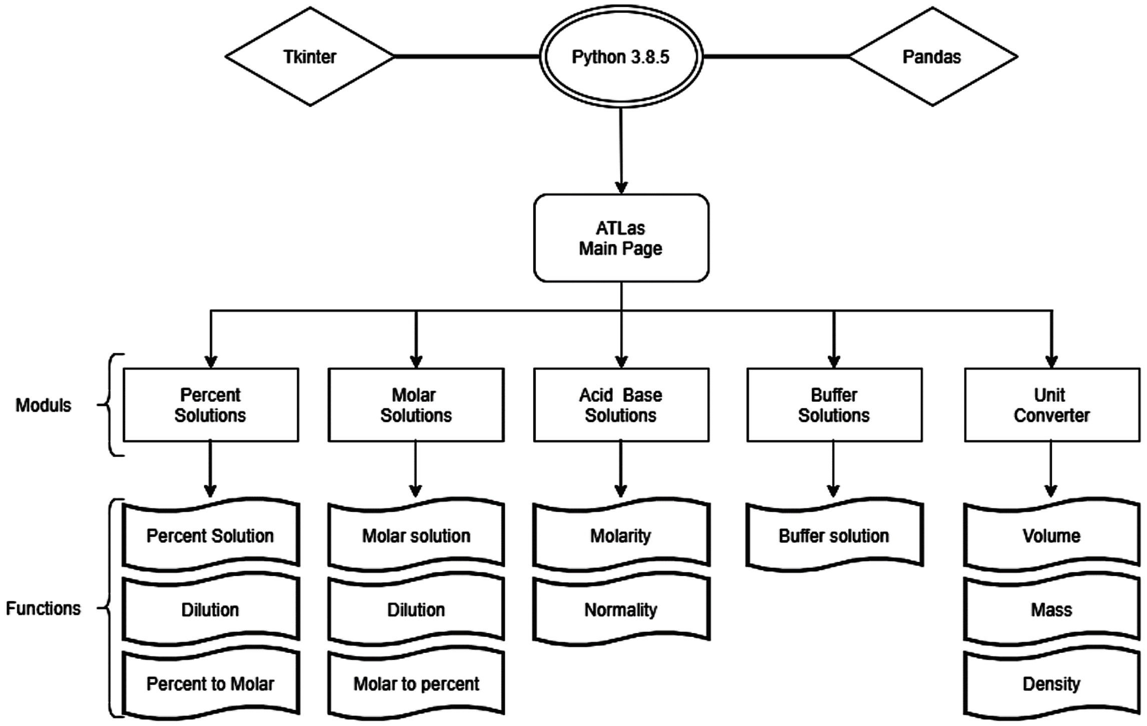


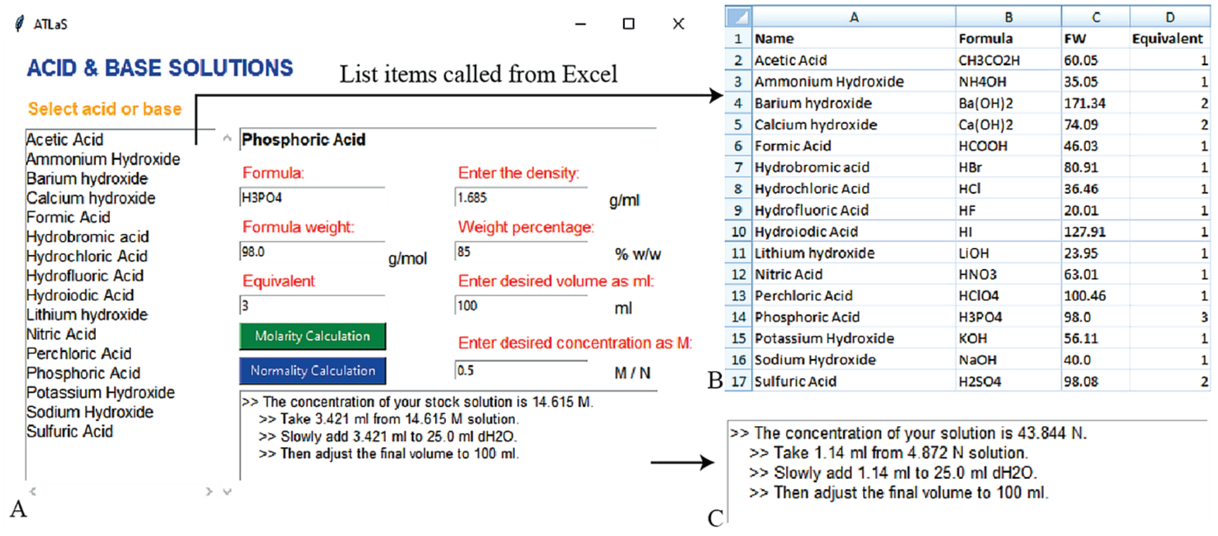
Figure 1 The flowchart used in the development of ATLaS

3 RESULTS AND DISCUSSION

The research in life science laboratories requires a diverse collection of software tools. These tools range from laboratory equipment software to data analysis software, and custom software developed to handle the relatively numerous calculations. Researchers increasingly depend on these tools to handle complicated laboratory protocols. Ultimately, research needs in laboratory motivated us to develop an assistant toolkit called ATLaS. The goal of ATLaS is to assist wet-lab scientist with various domains in life science laboratory researches. The benefits it provides at a functional level are unified functions, interface simplicity and efficient usage. Solution calculator is the first domain in ATLaS. We developed a calculator for preparing solutions in experimental studies. In this software, we focused our attention on reducing calculation errors in solution preparation by researchers. Thus, this domain allows researchers to prevent time, labor and financial losses by hindering experimental errors raised from faulty calculations in solution preparation. Although solution calculations are made on various websites on the Internet, not every site can meet all the calculations needed. For this reason, the researcher may have to navigate various sites to make the calculation he/she wants. In addition, the lab may not have stable internet access or internet connection speeds may be low. ATLaS collects the calculations that may be needed while preparing solutions by life scientists in a single software. It does not require internet connection with its stands-alone structure. The versatility of a program comes from the variety of functions it has. ATLaS includes 10 functions under 5 modules. Since the documentation will become more important in software development, modules and the overall functions of ATLaS were illustrated in the user manual and available at https://github.com/cugur1978/ATLaS/blob/main/ATLaS_User_Manuel.pdf

The modular development of ATLaS provided an advantage in debugging. The modules written separately were brought together after the errors were corrected. In this way, the codes can be read more easily for those who want to develop ATLaS by cloning. A good GUI makes an application intuitive and easy to use. Tkinter, which is default graphical user interface widget set for Python, was used for GUI. Tkinter was selected because of its availability on all operating systems [23]. Graphical user interfaces allow the user to send information into the program without accessing the code [24]. One of our main purposes was that the GUI of ATLaS had a simple interface. It can be confusing to present all parameters in a single interface, so ATLaS is designed modularly. Thus, even the first person to use the program can make calculations without difficulty.

Scientific software developed by researchers typically appeals to a very narrow audience [25]. Effective scientific software should be coded to meet the needs of the researcher over a long period. Scientific software may produce the intended results in developers hands, however, the potential new user may not benefit in a similar manner. In order to enable ATLaS be flexible enough to adapt to a variety of laboratories, chemical information did not inserted into the codes. Taking into account the difference of the solutions or chemicals needed in different laboratories, ATLaS allows the researcher to create a chemical database in Acid-Base and Buffer solution modules. The information in these modules is called from the Microsoft Excel file located in the same folder as ATLaS. Python language uses the Pandas library for processing spreadsheet-format data such as Excel and therefore to read the data in Excel files, Pandas library was imported to ATLaS. If the researcher enters the information of the chemicals required, depending on the format in the Excel file, ATLaS will use this information in calculations. Finally, calculation results are reported to the user as a recipe. Thus, the results were made more understandable especially for students and young researchers who start working in research laboratories.



Acetic Acid
Ammonium Hydroxide
Barium hydroxide
Calcium hydroxide
Formic Acid
Hydrobromic acid
Hydrochloric Acid
Hydrofluoric Acid
Hydroiodic Acid
Lithium hydroxide
Nitric Acid
Perchloric Acid
Phosphoric Acid
Potassium Hydroxide
Sodium Hydroxide
Sulfuric Acid

Phosphoric Acid

Formula: H3PO4
Formula weight: 98.0 g/mol
Equivalent: 3
Enter the density: 1.685 g/ml
Weight percentage: 85 % w/w
Enter desired volume as ml: 100 ml
Enter desired concentration as M: 0.5 M / N

```
>> The concentration of your stock solution is 14.615 M.
   >> Take 3.421 ml from 14.615 M solution.
   >> Slowly add 3.421 ml to 25.0 ml dH2O.
   >> Then adjust the final volume to 100 ml.
```

	A	B	C	D
1	Name	Formula	FW	Equivalent
2	Acetic Acid	CH3CO2H	60.05	1
3	Ammonium Hydroxide	NH4OH	35.05	1
4	Barium hydroxide	Ba(OH)2	171.34	2
5	Calcium hydroxide	Ca(OH)2	74.09	2
6	Formic Acid	HCOOH	46.03	1
7	Hydrobromic acid	HBr	80.91	1
8	Hydrochloric Acid	HCl	36.46	1
9	Hydrofluoric Acid	HF	20.01	1
10	Hydroiodic Acid	HI	127.91	1
11	Lithium hydroxide	LiOH	23.95	1
12	Nitric Acid	HNO3	63.01	1
13	Perchloric Acid	HClO4	100.46	1
14	Phosphoric Acid	H3PO4	98.0	3
15	Potassium Hydroxide	KOH	56.11	1
16	Sodium Hydroxide	NaOH	40.0	1
17	Sulfuric Acid	H2SO4	98.08	2

```
>> The concentration of your solution is 43.844 N.
   >> Take 1.14 ml from 4.872 N solution.
   >> Slowly add 1.14 ml to 25.0 ml dH2O.
   >> Then adjust the final volume to 100 ml.
```

Figure 2 Acid and base solution module (A), Excel file called from ATLaS (B) and Normal Calculation output (C)

As an example, the calculations required for preparing 0.5 M and 0.5 N phosphoric acid solutions are shown in the application in Fig. 2. Acid and Base Solution module in ATLaS contains 16 acids and bases. The formulas, formula weights and equivalent values of these chemicals are called from the Excel file. The researcher could add more acid and bases to this Excel file, thus the chemical database could be expanded by the user. However, density and weight percentage values are requested from the user. This is because these values may differ according to the brands of chemicals used in laboratories. As seen in Fig. 2, when phosphoric acid is selected from the list and the relevant fields are filled, the outputs can be seen when the molarity or normality buttons are pressed. Instead of giving the calculated values directly to the user, the preparation of the solution comes as a recipe as more understandable laboratory instructions. For this, the calculated values are assigned to the defined variables and placed in the codes containing instructions.

The open-source code of ATLaS enables the developers to be able to better understand the methodology and reproduce the results. Open-source software development has had remarkable effect on scientific research. Open sharing not only enables the scientific approach through replication, validation, and error checking, sharing is the key to a sustainable forthcoming for computational research, and publishers require open-source code for reviewing the software used to generate results [26]. However, the retainability of softwares after publication is presumably the most important problem faced by scientists who develop it [27]. Additionally, version control is necessary for sustainable software development [25]. In order to overcome these issues, ATLaS uploaded to GitHub, which is a popular web-based hosting service for Git version control. GitHub provides a useful software development platform, wherein developers can upload their open-source projects. The most important issue in a repository is having a license that clearly defines the permissions and restrictions attached to the code and other files in the repository [28]. Since ATLaS was deposited in GitHub under the license of MIT, which gives permissions without limitation to use copy and modify, it is possible for other developers to develop this software.

4 CONCLUSIONS

We took into account "Release early, release often" as an open-source mantra, and reported the first version of ATLaS in this study. We aim to present new versions to researchers by adding different modules to ATLaS. Scientific software development is a significant need that has to be fulfilled. It is essential that experiments be analyzed in a reproducible manner. Computational research software systems allow for the standardization of analysis pipelines, thus enables the scientific studies repeatable in different laboratories.As scientists, we need assistant softwares that will standardize laboratory protocols and calculations. Development of open-source software is a collective process and therefore our expectation is that different scientists and software developers will be involved in the development of ATLaS.

5 REFERENCES

[1] Kohl, M., Wiese, S., & Warscheid, B. (2011). Cytoscape: software for visualization and analysis of biological networks. In: Hamacher M., Eisenacher M., Stephan C. (eds) *Data Mining in Proteomics. Methods in Molecular Biology (Methods and Protocols), vol 696*. Humana Press. https://doi.org/10.1007/978-1-60761-987-1_18

[2] List, M., Ebert, P., & Albrecht, F. (2017). Ten simple rules for developing usable software in computational biology. *PLoS Computational Biolology, 13*(1), e1005265. https://doi.org/10.1371/journal.pcbi.1005265

[3] Wang, A., Yan, X., & Wei Z. (2018). ImagePy: an open-source, Python-based and platform-independent software package for bioimage analysis. *Bioinformatics, 34*(18), 3238-3240. https://doi.org/10.1093/bioinformatics/bty313

[4] Deardorff, A. (2020). Why do biomedical researchers learn to program? An exploratory investigation. *Journal of the Medical Library Association, 108*(1), 29. https://doi.org/10.5195%2Fjmla.2020.819

[5] Dagdia, Z. C., Avdeyev, P., & Bayzid, M. S. (2021). Biological computation and computational biology: survey, challenges, and discussion. *Artificial Intelligence Review*, 1-67. https://doi.org/10.1007/s10462-020-09951-1

[6] Morin, A., Urban, J., & Sliz, P. (2012). A quick guide to software licensing for the scientist-programmer. *PLoS Computational Biology, 8*(7), e1002598. https://doi.org/10.1371/journal.pcbi.1002598

[7] Ananthanarayanan, V. & Thies, W. (2010). Biocoder: A programming language for standardizing and automating biology protocols. *Journal of Biological Engineering, 4*(1), 1-13. https://doi.org/10.1186/1754-1611-4-13

[8] Choi, K., Medley, J. K., König, M., Stocking, K., Smith, L., Gu, S., & Sauro, H. M. (2018). Tellurium: an extensible python-based modeling environment for systems and synthetic biology. *Biosystems*, 171, 74-79. https://doi.org/10.1016/j.biosystems.2018.07.006

[9] Kunzmann, P. & Hamacher, K. (2018). Biotite: a unifying open source computational biology framework in Python. *BMC Bioinformatics, 19*(1), 1-8. https://doi.org/10.1186/s12859-018-2367-z

[10] Chen, C., Chen, H., Zhang, Y., Thomas, H. R., Frank, M. H., He, Y., & Xia, R. (2020). TBtools: an integrative toolkit developed for interactive analyses of big biological data. *Molecular Plant, 13*(8), 1194-1202. https://doi.org/10.1016/j.molp.2020.06.009

[11] Millman, K. J. & Aivazis, M. (2011). Python for scientists and engineers. *Computing in Science & Engineering, 13*(2), 9-12. https://doi.org/10.1109/MCSE.2011.36

[12] Perez, F., Granger, B. E., & Hunter, J.D. (2010). Python: an ecosystem for scientific computing. *Computing in Science & Engineering, 13*(2), 13-21. https://doi.org/10.1109/MCSE.2010.119

[13] Rashed, M. G. & Ahsan, R. (2012). Python in computational science: applications and possibilities. *International Journal of Computer Applications, 46*(20), 26-30. https://doi.org/10.5120/7058-9799

[14] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., & Bright, J. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods, 17*(3), 261-272. https://doi.org/10.1038/s41592-019-0686-2

[15] Balaban, G., Grytten, I., Rand, K. D., Scheffer, L., & Sandve, G. K. (2021). Ten simple rules for quick and dirty scientific programming. *PLOS Computational Biology, 17*(3), e1008549. https://doi.org/10.1371/journal.pcbi.1008549

[16] Casadevall, A., Steen, R. G., & Fang, F. C. (2014). Sources of error in the retracted scientific literature. *The FASEB Journal, 28*(9), 3847-3855. https://doi.org/10.1096/fj.14-256735

[17] Prinz, F., Schlange, T., & Asadullah, K. (2011). Believe it or not: how much can we rely on published data on potential drug targets? *Nature reviews Drug Discovery, 10*(9), 712-712. https://doi.org/10.1038/nrd3439-c1

[18] Goodman, S. N., Fanelli, D. & Ioannidis, J. P. (2016). What does research reproducibility mean? *Science Translational Medicine, 8*(341), 341ps12-341ps12. https://doi.org/10.1126/scitranslmed.aaf5027

[19] Ekmekci, B., McAnany, C. E., & Mura, C. (2016). An introduction to programming for bioscientists: a Python-based primer. *PLoS Computational Biology, 12*(6), e1004867. https://doi.org/10.1371/journal.pcbi.1004867

[20] Fourment, M. & Gillings, M. R. (2008). A comparison of common programming languages used in bioinformatics. *BMC Bioinformatics, 9*(1), 1-9. https://doi.org/10.1186/1471-2105-9-82

[21] Jeon, J. & Kim, H. U. (2020). Setup of a scientific computing environment for computational biology: Simulation of a

genome-scale metabolic model of *Escherichia coli* as an example. *Journal of Microbiology, 58*(3), 227-234. https://doi.org/10.1007/s12275-020-9516-6

[22] Habib, P. T., Alsamman, A. M., & Hamwieh, A. (2019). BioAnalyzer: Bioinformatic software of routinely used tools for analysis of genomic data. *Advances in Bioscience and Biotechnology, 10*(03), 33. https://doi.org/10.4236/abb.2019.103003

[23] Vlachoudis, V. (2009). FLAIR: a powerful but user friendly graphical interface for FLUKA. *Proceedings of International Conference on Mathematics, Computational Methods & Reactor Physics,* Saratoga Springs, New York.

[24] Moruzzi, G. (2020). Essential Python for the Physicist, Springer International Publishing. https://doi.org/10.1007/978-3-030-45027-4

[25] Taschuk, M. & Wilson, G. (2017). Ten simple rules for making research software more robust. *PLOS Computational Biology, 13*(4), e1005412. https://doi.org/10.1371/journal.pcbi.1005412

[26] Osborne, J. M., Bernabeu, M. O., Bruna, M., Calderhead, B., Cooper, J., Dalchau, N., Dunn, S.-J., Fletcher, A. G., Freeman, R., Groen, D., Knapp, B., McInerny, G. J., Mirams, G. R., Pitt-Francis, J., Sengupta, B., Wright, D. W., Yates, C. A., Gavaghan, D. J., Emmott, S., & Deane, C. (2014). Ten simple rules for effective computational research. *PLOS Computational Biology, 10*(3), e1003506. https://doi.org/10.1371/journal.pcbi.1003506

[27] Prlić, A. & Procter, J. B. (2012). Ten simple rules for the open development of scientific software. *PLOS Computational Biology, 8*(12), e1002802. https://doi.org/10.1371/journal.pcbi.1002802

[28] Perez-Riverol, Y., Gatto, L., Wang, R., Sachsenberg, T., Uszkoreit, J., Leprevost, F. D. V., Fufezan, C., Ternent, T., Eglen, S. J., & Katz, D. S. (2016).Ten simple rules for taking advantage of Git and GitHub. *PLOS Computational Biology, 12*(7), e1004947. https://doi.org/10.1371/journal.pcbi.1004947

Authors' contacts:

Ugur Comlekcioglu
(Corresponding author)
Kahramanmaras Sutcu Imam University,
Biotechnology Laboratory,
46050, Kahramanmaras, Turkey
E-mail: cugur1978@gmail.com

Nazan Comlekcioglu
Kahramanmaras Sutcu Imam University,
Biology Department,
46050, Kahramanmaras, Turkey