Makara Journal of Technology

Volume 23 | Issue 2 Article 4

8-2-2019

Use of the “DNAChecker” Algorithm for Improving Bioinformatics Research

Nausheen Bhat
Department of Bioinformatics, School of Life Sciences, Indonesia International Institute for Life Sciences, Jakarta Timur 13210, Indonesia

Ezra Bernadus Wijaya
Department of Bioinformatics, School of Life Sciences, Indonesia International Institute for Life Sciences, Jakarta Timur 13210, Indonesia

Arli Aditya Parikesit
Department of Bioinformatics, School of Life Sciences, Indonesia International Institute for Life Sciences, Jakarta Timur 13210, Indonesia, arli.parikesit@i3l.ac.id

Recommended Citation
Bhat, Nausheen; Wijaya, Ezra Bernadus; and Parikesit, Arli Aditya (2019) "Use of the “DNAChecker” Algorithm for Improving Bioinformatics Research," *Makara Journal of Technology*: Vol. 23: Iss. 2, Article 4.
DOI: 10.7454/mst.v23i2.3488
Available at: https://scholarhub.ui.ac.id/mjt/vol23/iss2/4

Makara J. Technol. 23/2 (2019), 72-77
doi: 10.7454/mst.v23i2.3488

Use of the “DNAChecker” Algorithm for Improving Bioinformatics Research

Nausheen Bhat[1], Ezra Bernadus Wijaya[1,2], and Arli Aditya Parikesit[1*]

1. Department of Bioinformatics, School of Life Sciences, Indonesia International Institute for Life Sciences, Jakarta Timur 13210, Indonesia
2. Department of Bioinformatics and Medical Engineering, Asia University, Taichung 41354, Taiwan

[*]*e-mail: arli.parikesit@i3l.ac.id*

Abstract

Basic Local Alignment Sequencing Tool (BLAST) is a bioinformatics tool used for analyzing nucleotide sequences with regards to their similarity. BLAST can be found online on biological databases such as the National Center for Biotechnology Information (NCBI) and other such repositories. The mechanism of BLAST allows the target sequence to be compared with other sequences to find regions of local similarity, and thus, a comparability quotient that determines the resemblance between the sequences is created. Due to the open-platform nature of the online databanks, several sequences can be accepted with little to no interjections regarding the quality of sequence submitted. An example of unclean nucleotide sequences can be based on the number of non-template nucleotides, denoted as “N,” present within the sequence. Here we develop a self-established nucleotide sequence reading program known as “DNAChecker,” which helps identify the quality of the target sequence and therefore proposes the effectiveness of the BLAST result. DNAChecker is an inbuilt, program that runs on Python 3.4 and was implemented in the United States Agency for International Development (USAID) project conducted in Indonesia International Institute for Life Sciences. Although DNAChecker has proven to be useful, it has a lot of room for improvements, such as having a more objectively accurate means of differentiating between good and bad sequences.

Abstrak

Penggunaan Algoritma “DNA Checker” untuk Pengembangan Riset Bioinformatika. Basic Sequence Alignment Tool (BLAST) adalah aplikasi bioinformatika yang digunakan untuk menganalisis sekuens nukleotida sehubungan dengan pensejajarannya. BLAST dapat ditemukan secara daring di database biologis seperti Pusat Nasional untuk Informasi Bioteknologi (NCBI) dan repositori lainnya. Mekanisme BLAST memungkinkan sekuens target untuk dibandingkan dengan sekuens lain untuk menemukan daerah kesamaan lokal, dan dengan demikian, dapat menghasilkan perbandingan yang menentukan kemiripan antara sekuens. Karena sifat platform terbuka dari bank data daring, beberapa urutan dapat diterima dengan sedikit atau tanpa interupsi terkait kualitas urutan yang disampaikan. Contoh urutan nukleotida tidak baik dapat didasarkan pada jumlah nukleotida non-template, dilambangkan sebagai "N," yang hadir dalam urutan. Di sini kami mengembangkan program pembacaan urutan nukleotida yang dikenal sebagai "DNAChecker," yang membantu mengidentifikasi kualitas urutan target dan karenanya meningkatkan keefektifan hasil pencarian BLAST. DNAChecker adalah program *inbuilt*, yang berjalan pada Python 3.4 dan diimplementasikan di proyek Badan Pembangunan Internasional Amerika Serikat (USAID) yang dilaksanakan di Institut Bioscientia Internasional Indonesia. Meskipun DNAChecker terbukti bermanfaat, ia tetap seyogyanya ditingkatkan fiturnya, seperti memiliki cara yang lebih akurat secara obyektif untuk membedakan urutan yang baik dan buruk.

Keywords: DNAChecker, Python, NCBI, BLAST, USAID

1. Introduction

DNA amplification is an essential process that manipulates the DNA fragments of a sample such that a sequence is produced. Sequence identification requires the use of the Basic Local Alignment Sequencing Tool (BLAST), which is a bioinformatics tool that allows nucleotide sequence comparison via the alignment of a target sequence against a nucleotide databank, where the most similar sequences can be identified [1]. Although the results of BLAST have a defined accuracy, it is not certain how “clean” or precise the target sequence may have been. During amplification, a common error occurs where the lack of concentration of any nucleotide is undetermined [2],[3], and consequently, the corresponding region is either left empty or identified as a non-template nucleotide, denoted simply as “N” [4]. These non-template nucleotides may be the reason for variations between the expected and actual results from BLAST. In many cases, where the BLAST result is unknown, the presence of non-template nucleotides may allow changes within the sequence that may cause a disruption in the read sequence and consequently in the overall research information [4],[5].

As experienced with the United States Agency for International Development (USAID) project’s DNA amplification, several non-template nucleotides were present, which had to be manually dealt with, so that not the whole sequence is read. Due to the advancements in computational analysis for biological data, the Python programing language can be implemented to script a code that allows the sequence to be read prior to BLAST analysis and determine the quality of the sequence based on a uniform criterion [6], [7]. Several computational algorithms for large-scale DNA analysis have been implemented, but not for a specialized task as explained in this research [8]–[10]. In various nucleotide-reading computer-based programs, uncertain nucleotides within a sequence are defined by the letter “N.” The more the occurrences of “N” within a given nucleotide sequence, the “uglier” the sequence. The ideal sequence would be a chain consisting of the peak sequence of one of the four nucleotides: guanine (G), thymine (T), cytosine (C), or adenine (A).

The purpose of this study is to create a calculated nucleotide sequence analyzer that can dictate accurate approval measures in order to reduce the chances of BLAST unclean nucleotide sequences and prevent possible unsatisfactory results. The function of the program, DNAChecker, designed by members of the USAID project, is to analyze whole sequences and eventually determine whether the sequence is clean enough to proceed for the next step of sequence identification, which is performed by BLAST.

2. Experimental

Material. DNAChecker was created using a Hewlett Packard Pavilion laptop with the following specifications: Intel® Core™ i7-4510U CPU @ 2.00 GHz 2.60 GHz, RAM 12.0 GB, Windows 10 64-bit Operating System, × 64-based processor.

The program used for creating DNAChecker was Python 3.4, with additional plugins, such as Biopython, to further upgrade the features of DNAChecker [6]. In addition to the Python software, a DNA sequence visualizing software, FinchTV, was used to make an initial identification of the ABI (Applied Biosystem) format chromatogram file (.ab1) into a more basic text format (.txt), which only represents the nucleotides and not the other measurable factors that FinchTV can express, such as the nucleotide concentration and such [11]. This was done to easily input the sequence into DNAChecker, since accessing and reading text files is easier, compared to ABI format files.

Data Management and Coding. Figure 1 shows the interface of the DNA sequence visualizing software FinchTV. As can be seen in the figure, there are peaks that represent nucleotides with the highest concentrations. With this, we can conclude that a certain sequence will have the given nucleotide in order. However, when there is a disruption in peak, such as shown in the beginning, the software expresses it as an “N.” This “N” nucleotide is the indicator of a good and properly expressed and analyzed sequence. The fewer “N” present within a sequence, the better the sequence.

Ideally, the sequence would have no non-template nucleotide (N), just as it is present in the DNA. Finally, after accessing the DNA sequence, we can copy the information and paste it onto a text document-based application, such as Notepad. Figure 2 is the representation of the previously shown raw data files that have been converted into text files and thus are readable from a notepad application. This is an essential step that allows the sequence to be read by the DNAChecker program; if the ABI format were employed, the reading, as well as the coding within the Python program, would not be efficient from the perspective of time and memory. It is also important to notice the directory of this folder. Since the Python program will read the files from this folder, it must be kept in the specified directory that Python could load into the PC’s RAM.

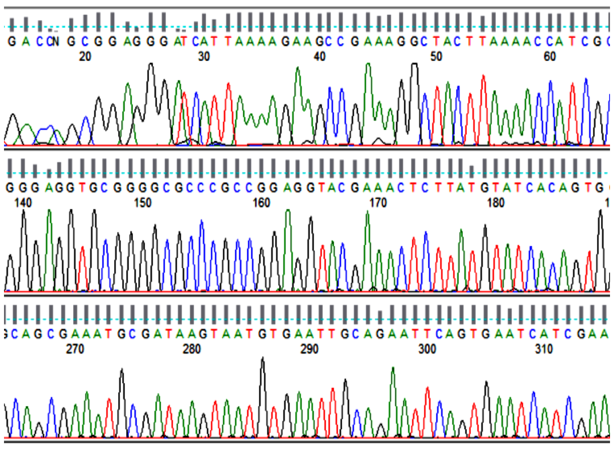
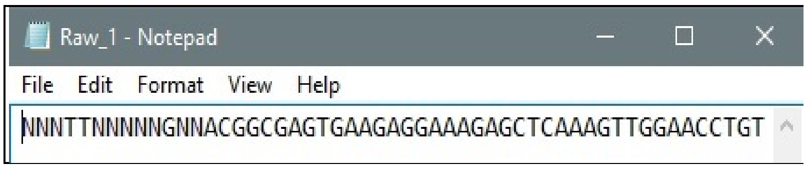


Figure 1. The Interface of the DNA Sequence Visualizing Software, FinchTV

Figure 2 shows the target DNA sequence that is read as a text file. The only information that is displayed in this format is the nucleotide in sequence.

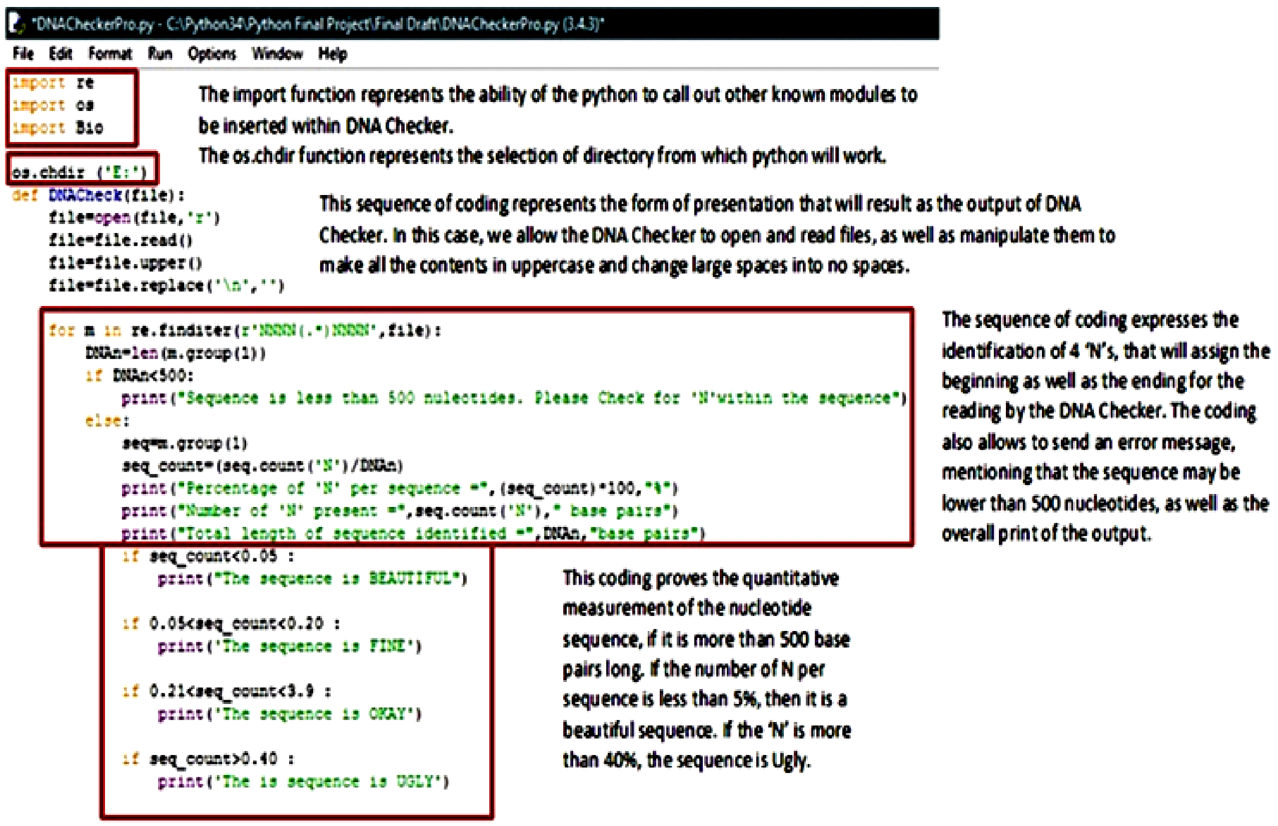
The import function allows other previously built functions to be inserted into the DNAChecker program so that its functions can be utilized. The "re" module or the Regular Expression module is inputted within the frame of the string operation and can be used to allow the recognition of various possible strings patterns [12]. While the import os is the main function to direct Python into a specific directory of choice, import Bio is the function to import the Biopython module into the main Python 3.4 software (Figure 3). Although, at this point, there is no use of the Biopython module since there are no functions that require any coding that the Biopython module provides. However, for the further development of this project, the Biopython module can be applied to make use of the coding that Biopython can provide.

As a part of managing biological data, it is necessary to ensure that all files are readable and designed to be uniformly adjusted. Therefore, the "file" call requires the "open" coding as a well-read function, represented by the "r" code. To uniformly present all accessible data, the nucleotide pairs must be capitalized and the spacing arrangements must be taken care of to avoid the esthetically uneven data. For that, the coding "file.upper" creates an uppercase default input, and as a part of the Python built-in rule, "/n" represents the largely gapped spaces, which are then replaced with" ," to express that these spaces are to be replaced with nothing in between and are thus equally spaced.



```
NNNTTNNNNNGNNACGGCGAGTGAAGAGGAAAGAGCTCAAAGTTGGAACCTGT
```

Figure 2. DNA Sequence Shown in a Text Document


DNACheckerPro.py - C:\Python34\Python Final Project\Final Draft\DNACheckerPro.py (3.4.3)
File Edit Format Run Options Window Help

```
import re
import os
import Bio
os.chdir ('E:')
def DNACheck(file):
    file=open(file,'r')
    file=file.read()
    file=file.upper()
    file=file.replace('\n','')
    for m in re.finditer(r'NNNN(.*)NNNN',file):
        DNAn=len(m.group(1))
        if DNAn<500:
            print("Sequence is less than 500 nulectides. Please Check for 'N'within the sequence")
        else:
            seq=m.group(1)
            seq_count=(seq.count('N')/DNAn)
            print("Percentage of 'N' per sequence =",(seq_count)*100,"%")
            print("Number of 'N' present =",seq.count('N')," base pairs")
            print("Total length of sequence identified =",DNAn,"base pairs")
            if seq_count<0.05 :
                print("The sequence is BEAUTIFUL")

            if 0.05<seq_count<0.20 :
                print('The sequence is FINE')

            if 0.21<seq_count<3.9 :
                print('The sequence is OKAY')

            if seq_count>0.40 :
                print('The is sequence is UGLY')
```

The import function represents the ability of the python to call out other known modules to be inserted within DNA Checker.

The os.chdir function represents the selection of directory from which python will work.

This sequence of coding represents the form of presentation that will result as the output of DNA Checker. In this case, we allow the DNA Checker to open and read files, as well as manipulate them to make all the contents in uppercase and change large spaces into no spaces.

The sequence of coding expresses the identification of 4 'N's, that will assign the beginning as well as the ending for the reading by the DNA Checker. The coding also allows to send an error message, mentioning that the sequence may be lower than 500 nucleotides, as well as the overall print of the output.

This coding proves the quantitative measurement of the nucleotide sequence, if it is more than 500 base pairs long. If the number of N per sequence is less than 5%, then it is a beautiful sequence. If the 'N' is more than 40%, the sequence is Ugly.

Figure 3. The Inputted Coding for the whole DNAChecker Program

The “re.finditer” is a convenient tool to identify and match string patterns that are read from the left to right, and the “m” represents the group that holds the information of the sequences, shown previously as m.group(1) [13, 14]. The “re.finditer” function in this case helps to identify four sequential “N”s in a sequence, from where the reading for the sequence will be tracked. To end the search of the sequence, the function will have to find another group of four “N”s, as shown in the coding above. The representation of four “N”s is a helpful way to start the sequence from an ugly point in the beginning and to read the rest of the neat sequence until the next four “N”s. However, if the occurrence of the four “N”s happens earlier than before the necessary amount of sequence is recorded, an error message will be shown, saying that the sequence needs to be at least 500-base-pairs long, which will be set as a minimum base pair count. If the sequence count is more than 500, then the program can move onto the next step.

Finally, the next set of coding acts as a rule for the quantitative measurement for the sequence, which will decide the beauty of the extracted sequence. The code summarizes the requirements to identify the number of “N”s in the sequence and the total number of base pairs in the sequence itself.

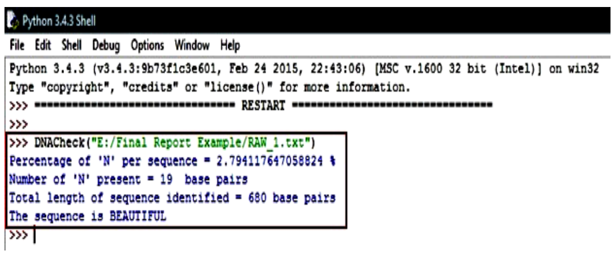
If the amount of “N”s per sequence reaches a certain level, the sequence will be judged for its beauty. As instructed to the program, if the number of “N”s per sequence is less than 5%, then the sequence is deemed BEAUTIFUL. If the sequence is between 5% and 20%, the sequence is deemed FINE, whereas if the sequence is between 21% and 39%, the sequence is considered OKAY and reconsideration is needed on cleaning up the nucleotides sequence by replacing the N base with the best peaks shown on Finch TV. Finally, if the percentage of “N”s is above 40% of the whole seThe result of the input is the determined verdict of the DNA sequence quality, ranging from beautiful to unreadable. Figure 5 shows the result of a sample analyzed with the DNAChecke; the total sequence length, total amount of “N”s, and percentage of “N”s present within the sequence are all displayed. Finally, the DNAChecker determines the quality of the sequence. Figure 6 displays different raw data sequences with different measurements based on the number of non-template nucleotides present in the individual sequence. As observed, the percentage of non-template nucleotides in the first raw data is less than 5%, and therefore, the sequence receives a BEAUTIFUL rating. However, when the percentage of non-template nucleotides within the sequence is above 5% and below 20%, the sequence is deemed FINE. Finally, the last sequence shows that the sequence of the Raw_12.txt file has multiple “N”s before its minimum limit, i.e., 500 base pairs. The result prints out as shown.

There are no computational means of measuring DNA sequence quality, as DNA amplicons can only be treated via wet laboratory methods. Hence, there is no developed standard for determining how clean a sequence should be before being processed for any experiment.

Similarly, there is no standard for the percentage measurements used in DNAChecker, but the brackets are based on fine estimates of the number of non-template nucleotides that occur within the middle section of the sequence. The beginning and the ending of the sequence tend to be very “noisy,” and therefore, the sequence may not qualify as a good sample. However, we utilize multiple N’s in the beginning and the end of the sequence to provide the starting and ending points for the program to read the sequence. This creates a suitable measure of the significantly more important and neater sequence, which is improved than the earlier efforts [15-17].

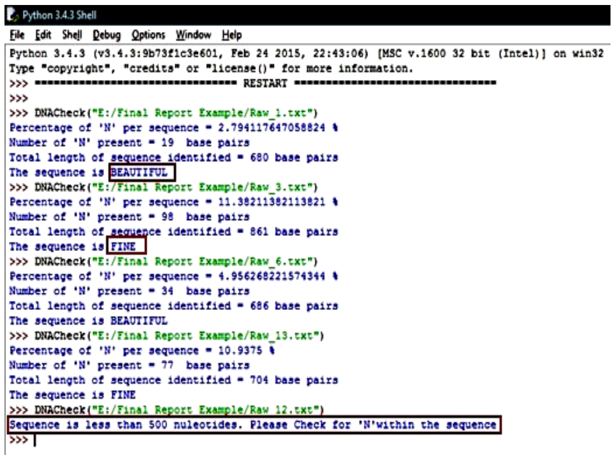
DNAChecker has proven to be useful for the general identification of clean sequences for BLAST. This program can be applied in various projects that deal with various sequences from organisms, since it acts as an efficient tool to filter out the good sequences from the bad ones while providing information based on the sequence that can be used in the database for further comparison. In a research grant, known as the USAID grant, Indonesia International Institute for Life Sciences (i3L) has been working on the development of the microbial diversity of lands, as well as the identification of biofuel-potent microbes [6-8]. DNAChecker can be employed in the USAID project, as it can analyze the identified sequences from the microbes and after determining the cleanliness of the sequence, it can provide a decent secondary proof of the accurate results from BLAST, as well as improve the databases with more information. In the research world, where scientific measures for various processes, such as next-generation sequencing and metagenomics, are carried out on computers, DNAChecker holds the important role of pre-analyzing such digital results that will help in assuring the purity of the sequence.

DNAChecker is intended to be one of three different programs, along with multiBLAST and GenBank Checker. MultiBLAST, is the process of analyzing several files together using BLAST to efficiently work on multiple sequences and obtain data faster. Although several multiBLAST codings are provided on the internet [18], this project intends to integrate several personal touches into creating a newer multiBLAST coding, with the previous coding acting as a backbone. GenBank Checker introduces another level of advanced data collection, which improves on the next step of multiBLAST, to identify the microorganism, along with its details and especially its trusted publications.



```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
>>> DNACheck("E:/Final Report Example/RAW_1.txt")
Percentage of 'N' per sequence = 2.794117647058824 %
Number of 'N' present = 19  base pairs
Total length of sequence identified = 680 base pairs
The sequence is BEAUTIFUL
>>>
```

Figure 5. The Result of the First Sequence Shown



```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
>>> DNACheck("E:/Final Report Example/Raw_1.txt")
Percentage of 'N' per sequence = 2.794117647058824 %
Number of 'N' present = 19  base pairs
Total length of sequence identified = 680 base pairs
The sequence is BEAUTIFUL
>>> DNACheck("E:/Final Report Example/Raw_3.txt")
Percentage of 'N' per sequence = 11.38211382113821 %
Number of 'N' present = 98  base pairs
Total length of sequence identified = 861 base pairs
The sequence is FINE
>>> DNACheck("E:/Final Report Example/Raw_6.txt")
Percentage of 'N' per sequence = 4.956268221574344 %
Number of 'N' present = 34  base pairs
Total length of sequence identified = 686 base pairs
The sequence is BEAUTIFUL
>>> DNACheck("E:/Final Report Example/Raw_13.txt")
Percentage of 'N' per sequence = 10.9375 %
Number of 'N' present = 77  base pairs
Total length of sequence identified = 704 base pairs
The sequence is FINE
>>> DNACheck("E:/Final Report Example/Raw 12.txt")
Sequence is less than 500 nuleotides. Please Check for 'N'within the sequence
>>>
```

Figure 6. The Results of Various Sequences from the Initial Raw Data

GenBank Checker is intended to access the GenBank accession number of the given sequence, which also currently exists, using the Biopython module [6, 19]. However, GenBank Checker has the property to save most of the necessary selected information into the database, allowing the database to be significantly more detailed.

Other than improvements through the development of other programs, DNAChecker itself can be improved in having a more detailed quantitative measurement, with more levels of cleanliness. This program has the potential to be scaled up for genome and proteome annotation filtering methods that currently still use simple scripting applications [20–22]. One shortcoming that must be fixed is the identification of the N base based on the highest peak that can be visualized from Finch TV, and a coding that can efficiently allow such facilities needs to be added.

3. Conclusion

Advancements in computational science for biological data has allowed the creation of a simple program that can be helpful in simply providing an additional assurance for the accuracy of biological data. This report also shines some light on the fact that there are no significant criteria for uploading any quality of sequences onto DNA databases. With the help of DNAChecker, a basis can be adapted to ensure that only sequences of high quality are uploaded onto the online databanks. The use of DNAChecker is not only limited to the USAID project since it can be used to analyze any given sequence as long as the sequence can read the four N's chain as a beginning and ending, so that the sequence can print out the intended results. Considering the above, DNAChecker can offer a lot in the world of research.

Acknowledgement

The author would like to thanks Institute of Research and Community Empowerment, Indonesia International Institute for Life Sciences, and I3L-USAID Project for supporting this research. Thanks also go to Faried Irmansyah with his team from I3L IT Department for providing support and infrastructure. Lastly, thanks also goes to Direktorat Riset dan Pengabdian Masyarakat, Direktorat Jenderal Penguatan Riset dan Pengembangan Kementerian Riset, Teknologi dan Pendidikan Tinggi Republik Indonesia for providing Hibah Penelitian Dasar DIKTI/LLDIKTI III 2019 No. 1/AKM/PNT/2019. The author declares that there is no conflicting interest.

References

[1] M. Johnson, I. Zaretskaya, Y. Raytselis, Y. Merezhuk, S. McGinnis, T.L. Madden, Nucleic Acids Res. 36 (2008) W5-9.
[2] P. Rice, I. Longden, A. Bleasby, Trends Genet. 16 (2000) 276.
[3] D.A. Benson, I. Karsch-Mizrachi, D.J. Lipman, J. Ostell, E.W. Sayers, Nucleic Acids Res. 37 (2009) D26.
[4] B. Steipe, B. Schiller, A. Plückthun, S. Steinbacher, J. Mol. Biol. 240 (1994) 188.
[5] E.H. Akand, K.M. Downard, Mol. Phylogenet. Evol. 112 (2017) 209.
[6] P.J.A. Cock, T. Antao, J.T. Chang, B.A. Chapman, C.J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, M.J.L. de Hoon, Bioinformatics 25 (2009) 1422.
[7] B. Chapman, J. Chang, ACM SIGBIO Newsl. 20 (2000) 15.
[8] M.I. Khan, C. Sheel, Am. J. Bioinforma. 2 (2013) 15.
[9] D.P. M., R. Prabha, A. Rai, D.K. Arora, Am. J. Bioinforma. 1 (2012) 10.
[10] R.M. Al-Khatib, R. Abdullah, N.A. Rashid, J. Comput. Sci. 5 (2009) 680.
[11] K.N. Mishra, D.A. Aaggarwal, D.E. Abdelhadi, D.P.C. Srivastava, Int. J. Comput. Appl. 3 (2010) 39.
[12] N. Tabuchi, E. Sumii, A. Yonezawa, Electron. Notes Theor. Comput. Sci. 75 (2003) 95.

[13] J.C. Brown, J. Virol. Antivir. Res. 5 (2016) 1.

[14] B. Steele, J. Chandler, S. Reddy, in: Algorithms Data Sci., Springer International Publishing, Cham, 2016, pp. 313–342.

[15] T. Cajka, L.A. Garay, I.R. Sitepu, K.L. Boundy-Mills, O. Fiehn, J. Nat. Prod. 79 (2016) 2580.

[16] L.A. Garay, I.R. Sitepu, T. Cajka, O. Fiehn, E. Cathcart, R.W. Fry, A. Kanti, A. Joko Nugroho, S.A. Faulina, S. Stephanandra, J.B. German, K.L. Boundy-Mills, J. Ind. Microbiol. Biotechnol. 44 (2017) 1.

[17] L.A. Garay, I.R. Sitepu, T. Cajka, I. Chandra, S. Shi, T. Lin, J.B. German, O. Fiehn, K.L. Boundy-Mills, J. Ind. Microbiol. Biotechnol. 43 (2016) 887.

[18] P.J.A. Cock, T. Antao, J.T. Chang, B.A. Chapman, C.J.Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B.Wilczynski, M.J.L. de Hoon. Bioinf. 25 (2009) 1422.

[19] B. Chapman, Genome Informatics 299 (2003) 298.

[20] A.A. Parikesit, P.F. Stadler, S.J. Prohaska, Open Ser. Informatics 4 (2012) 1.

[21] A.A. Parikesit, S. Prohaska, P. Stadler, N. Biotechnol. 27 (2010) S44.

[22] A.A. Parikesit, P.F. Stadler, S.J. Prohaska, in: Ext. Abstr. Ger. Conf. Bioinforma., 2011, pp. 9–11.