ACTA SCIENTIFIC MICROBIOLOGY (ISSN: 2581-3226)

Volume 3 Issue 6 June 2020

Short Communication

SeqProperties: A Python Command-Line Tool for Basic Sequence Analysis

Maurice HT Ling*

Colossus Technologies LLP, Singapore, HOHY PTE LTD, Singapore

***Corresponding Author:** Maurice HT Ling, Colossus Technologies LLP, Singapore, HOHY PTE LTD, Singapore.

Received: April 27, 2020

Published: May 20, 2020

Command-line tools are the fundamental building blocks of main bioinformatics applications [1,2]. Despite a burden to non-computer affine biologists [3], command-line tools can serve as underlying operations for chaining into analysis pipelines [4-10] or for integration into applications with graphical user interfaces [11].

Biological sequence analysis is recognized as a fundamental skill in bioinformatics [12-15] with many application [16]. Hence, this short communication presents SeqProperties, a Python command-line tool for basic sequence analysis and had been used for a number of studies [17,18] within my research group. It is part of the Bactome project (https://github.com/mauriceling/bactome) and is licensed under GNU General Public Licence version 3 for academic and non-commercial purposes only.

SeqProperties is implemented as a command-line tool using Python 3 and Python-Fire module (https://github.com/google/python-fire), which aims to simplify the implementation of command-line interface in Python 3. This has been exemplified in previous tools [19-22]. A total of 31 functions spanning 8 categories had been implemented. SeqProperties is a constant work-in-progress and new analytical methods will be added as required by the needs of or at the request of fellow researchers. The fundamental input into SeqProperties is a FASTA file containing one or more FASTA record(s) while the command-line format is python seqproperties.py [Function] [one or more options]. Brief descriptions of the functions are as follow.

Description

1. Function showIDs prints out the sequence IDs of all the FASTA records in the FASTA file to the output format of <count> : <sequence ID> where count is the numeric running order and sequence ID is the sequence ID of the FASTA record.

2. Function showDesc prints out the sequence IDs and descriptions of all the FASTA records in the FASTA file to the output format of <count> : <sequence ID> : <description> where description is the description of the FASTA record.

Complement and translation

1. Function complement generates the complement sequence of each FASTA record. This is done using reverse_complement function in Biopython library [23] where each FASTA sequence is assumed to be in 5'→3', this function will generate the complementary sequence in 5'→3' orientation rather than 3'→5' orientation. The output will be in FASTA format.

2. Function translate translates all the FASTA records from nucleotide sequence(s) to amino acid sequence(s), using Biopython library [23], to the output format of <sequence ID> : <amino acid sequence> where amino acid sequence is the translated amino acid sequence of the FASTA record.

Count statistics

1. Function nlength counts the number of nucleotides (number of bases) in each FASTA record. This assumes that each FASTA record contains a nucleotide sequence. It generates an output format of <sequence ID> : <nucleotide length> where nucleotide length is the number of nucleotides in the sequence.

2. Function plength counts the number of amino acids (peptide length) by peptide FASTA record. This assumes that each FASTA record contains a peptide sequence. It generates an output format of <sequence ID> : <peptide length> where peptide length is the number of amino acids in the peptide.

3. Function count counts the frequency of each character in each nucleotide sequence (by FASTA record) and generate a frequency table with order dependent on the input sequence.

4. Function codoncount generates the codon usage frequency table by each FASTA record. This assumes that each FASTA record contains a nucleotide sequence. It will generate a frequency table of codons by alphabetical order.

5. Function aacount translates each nucleotide sequence (by FASTA record) and generate a frequency table of the amino acids, by alphabetical order of IUPAC 1-character code.

6. Function ngram generates the n-grams by each FASTA record and generates the output in the format of <sequence ID> : <list of n-gram counts> : <list of n-gram identities>.

7. Function reverse processes each FASTA record for the presence of a sub-sequence and its reverse. For example, this is to see if a DNA sequence has both GATCTA and ATCTAG in its sequence. It generates the output in the format of <sequence ID> : <length of reverse> : <sequence> : <reversed sequence>.

8. Function asymfreq processes the dipeptide frequency data file to asymmetric frequency or C190 described by Carugo [24]. The dipeptide frequency data file is a comma-delimited file in the format of <dipeptide>, <count> and can be readily generated from n-gram output. The output of this function is in the format of <dipeptide> : <antidipeptide> : <C190 score>.

9. Function propensity processes the dipeptide frequency data file to propensity described by Carugo [24] and generates the output in the format of <dipeptide> : <propensity score>.

Nucleotide composition

1. Function gc generates the %GC by each FASTA record and generates the output in the format of <sequence ID> : <%GC>.

2. Function g generates the %G by each FASTA record and generates the output in the format of <sequence ID> : <%G>.

3. Function gci generates the %GC of the i-th base in each codon (of j length) by each FASTA record and generates the output in the format of <sequence ID> : <%GC of i-th base>.

4. Function gi generates the %G of the i-th base in each codon (of j length) by each FASTA record and generates the output in the format of <sequence ID> : <%G of i-th base>.

5. Function a generates the %A by each FASTA record and generates the output in the format of <sequence ID> : <%A>.

6. Function ai generates the %A of the i-th base in each codon (of j length) by each FASTA record and generates the output in the format of <sequence ID> : <%A of i-th base>.

Peptide properties

1. Function mw calculates the molecular weight, using Biopython library [23], by each FASTA record and generates the output in the format of <sequence ID> : <molecular weight>.

2. Function aromaticity calculates the aromaticity index by each FASTA record by Biopython library [23] using method described in Lobry and Gautier [25] and generates the output in the format of <sequence ID> : <aromaticity index>.

3. Function instability calculates the instability index by each FASTA record by Biopython library [23] using method described in Guruprasad [26] and generates the output in the format of <sequence ID> : <instability index>.

4. Function isoelectric calculates the isoelectric point (pI) by each FASTA record using Biopython library [23] and generates the output in the format of <sequence ID> : <isoelectric point>.

5. Function secstruct calculates the secondary structure fractions by each FASTA record using Biopython library [23] and generates the output in the format of <sequence ID> : <helix fraction> : <turn fraction> : <sheet fraction>.

6. Function gravy calculates the hydropathy, also known as GRAVY (Grand Average of Hydropathy), by each FASTA record, which is calculated by Biopython library [23] using method described in Kyte and Doolittle [27] and generates the output in the format of <sequence ID> : <GRAVY value>.

7. Function flexibility calculates the flexibility by each FASTA record. This is calculated by Biopython library [23] using method described in Vihinen., *et al.* [28] and generates the output in the format of <sequence ID> : <flexibility value>.

8. Function extinction calculates the molar extinction coefficient by each FASTA record using Biopython library [23] and generates the output in the format of <sequence ID> : <extinction coefficient assuming reduced cysteine> : <extinction coefficient assuming non-reduced cysteine>.

Sequence alignment

1. Function p align takes a FASTA file and calculate pairwise alignments between all the sequences in the file, using either Smith-Watermann algorithm [29] or Needleman-Wunsch algorithm [30] in Biopython library [23]. The output is in the format of <count> : <alignment score> : <sequence ID 1> : <sequence ID 2> where alignment score is the calculated pairwise alignment score, sequence ID 1 and 2 are the sequence IDs of the 2 FASTA records used for pairwise alignment.
2. Function palign2 takes 2 FASTA files (a query FASTA file and a database FASTA file) and calculate pairwise alignments in FASTA file to database FASTA file, using either Smith-Watermann algorithm [29] or Needleman-Wunsch algorithm [30] in Biopython library [23]. The full output is in the format of <count> : <alignment score> : <sequence ID from query> : <sequence ID from database> while the summarized output format is <count> : <minimum alignment score> : <average

alignment score> : <standard deviation of alignment score> : <maximum alignment score> : <sequence ID from query> where minimum, average, standard deviation, and maximum alignment scores are calculated from pairwise alignment scores.

Open reading frame finding

1. Function ORF finds open reading frames (ORF) for each FASTA record in a given FASTA file. An ORF is basically computed as a stretch of sequence flanked by a start and stop codon. The default output is in the format of <count> : <sequence ID> : <start position> : <stop position> : <strand> : <length of ORF> : [<sequence of ORF>] with <sequence of ORF> as optional, depending on input options. If FASTA file output is required, the description line for each sequence will be in the format of <count>|<sequence ID>|<start position>|<stop position>|<strand>|<length of ORF>.

Sampling

1. Function rselect selects a random set of sequences from a given FASTA file and will generate the output in the format of <count> : <sequence ID> : <sequence>, or a FASTA output format.

Conflict of Interest

The author declares no conflict of interest.

Bibliography

1. S Liu and Z Liu. "Introduction to Linux and command line tools for bioinformatics". In Bioinformatics in Aquaculture, Z. J. Liu, Ed. Chichester, UK: John Wiley and Sons, Ltd (2017): 1-29.
2. T Seemann. "Ten recommendations for creating usable bioinformatics command line software". *Gigascience* 2.1 (2013): 15.
3. DK Morais., *et al*. "BTW - Bioinformatics through Windows: an easy-to-install package to analyze marker gene data". *Peer Journal* 6 (2018): e5299.
4. SP Sadedin., *et al*. "Bpipe: a tool for running and managing bioinformatics pipelines". *Bioinformatics* 28.11 (2012): 1525-1526.
5. A Zhou., *et al*. "PipelineDog: a simple and flexible graphic pipeline construction and maintenance tool". *Bioinformatics* 34.9 (2018): 1603-1605.
6. J Leipzig. "A review of bioinformatic pipeline frameworks". *Briefings in Bioinformatics* 18.3 (2017): 530-536.
7. J Dahlberg., *et al*. "Arteria: An automation system for a sequencing core facility". *Gigascience* 8.12 (2019): giz135.
8. M Kotliar., *et al*. "CWL-Airflow: a lightweight pipeline manager supporting Common Workflow Language". *Gigascience* 8.7 (2019): giz084.
9. J Bedő. "Bio Shake: a Haskell EDSL for bioinformatics workflows". *Peer Journal* 7 (2019): e7223.
10. G Wang and B Peng. "Script of Scripts: a pragmatic workflow system for daily computational research". *PLoS Computational Biology* 15.2 (2019): e1006843.
11. M Joppich and R Zimmer. "From command-line bioinformatics to bioGUI". *Peer Journal* 7 (2019): e8111.
12. U Saeed and Z Usma. "Biological sequence analysis". In Computational Biology, Codon Publications (2019).
13. TW Tan., *et al*. "A proposed minimum skill set for university graduates to meet the informatics needs and challenges of the '-omics' era". *BMC Genomics* 10.3 (2009): S36.
14. MA Wilson Sayres., *et al*. "Bioinformatics core competencies for undergraduate life sciences education". *PLoS ONE* 13.6 (2018): e0196878.
15. RE Tractenberg., *et al*. "The mastery rubric for bioinformatics: a tool to support design and evaluation of career-spanning education and training". *PLoS ONE* 14.11 (2019): e0225256.
16. JX Lim., *et al*. "Sequence composition". In Encyclopedia of Bioinformatics and Computational Biology, Elsevier (2019): 323-326.
17. JH Kim and MH Ling. "Proteome diversities among 19 archaebacterial species". *Acta Scientific Microbiology* 2.5 (2019): 20-27.
18. A Maitra and MH Ling. "Codon usage bias and peptide properties of Pseudomonas balearica DSM 6083T". *MOJ Proteomics and Bioinformatics* 8.2 (2019): 27-39.
19. M H Ling. "Island: a simple forward simulation tool for population genetics". *Acta Scientific Computer Sciences* 1.2 (2019): 20-22.
20. MH Ling. "RANDOMSEQ: Python command–line random sequence generator". *MOJ Proteomics and Bioinformatics* 7.4 (2018): 206-208.
21. MH Ling. "SEcured REcorder BOx (SEREBO) based on blockchain technology for immutable data management and notarization". *MOJ Proteomics and Bioinformatics* 7.6 (2018): 169-174.

22. MH Ling. "Draft implementation of a method to secure data by file fragmentation". *Acta Scientific Computer Sciences* 1.2 (2019): 10-13.

23. PJA Cock., *et al*. "Biopython: freely available Python tools for computational molecular biology and bioinformatics". *Bioinformatics* 25.11 (2009): 1422-1423.

24. O Carugo. "Frequency of dipeptides and antidipeptides". *Computational and Structural Biotechnology Journal* 8.11 (2013): e201308001.

25. JR Lobry and C Gautier. "Hydrophobicity, expressivity and aromaticity are the major trends of amino-acid usage in 999 Escherichia coli chromosome-encoded genes". *Nucleic Acids Research* 22.15 (1994): 3174-3180.

26. K Guruprasad., *et al*. "Correlation between stability of a protein and its dipeptide composition: a novel approach for predicting in vivo stability of a protein from its primary sequence". *Protein Engineering, Design and Selection* 4.2 (1990): 155-161.

27. J Kyte and RF Doolittle. "A simple method for displaying the hydropathic character of a protein". *Journal of Molecular Biology* 157.1 (1982): 105-132.

28. V Mauno., *et al*. "Accuracy of protein flexibility predictions". *Proteins: Structure, Function, and Bioinformatics* 19.2 (1994): 141-149.

29. TF Smith and MS Waterman. "Identification of common molecular subsequences". *Journal of Molecular Biology* 147.1 (1981): 195-197.

30. SB Needleman and CD Wunsch. "A general method applicable to the search for similarities in the amino acid sequence of two proteins". *Journal of Molecular Biology* 48. 3 (1970): 443-453.