MethodsX 9 (2022) 101803

Contents lists available at ScienceDirect

MethodsX

journal homepage: www.elsevier.com/locate/mex

Method Article

CoCoView - A codon conservation viewer via sequence logos

Beatriz Rodrigues Estevam, Diego Mauricio Riaño-Pachón*

Computational, Evolutionary and Systems Biology Laboratory (LabBCES), Center for Nuclear Energy in Agriculture, University of São Paulo, Piracicaba/SP, Brazil

ABSTRACT

Sequence logos are a simple way to display a set of aligned sequences, and they are useful to identify conserved patterns. Since their introduction, several tools have been developed for generating these representations at the single residue level (amino acids or nucleotides). We have developed a tool to build sequence logos of protein-coding sequences at the codon level, allowing more accurate analysis of coding-sequences as they represent synonymous and non-synonymous changes instead of showing only changes that imply on amino acid substitutions. We built CoCoView on top of the Logomaker Python API. It creates codon sequence logos from a multiple sequence alignment of protein-coding sequences. Some properties of the data and the generated logos can be controlled by the end-users, such as data redundancy, plot type and alphabet color.

- Split aligned sequences into codon positions;
- For each position compute codon frequency and information content;
- Use the computed information to plot the graphic.

ARTICLE INFO
Method name: CoCoView: A Codon Conservation Viewer via Sequence Logos
Keywords: Consensus sequence, Information theory, Codons representation, Conserved patterns
Article history: Available online 29 July 2022

* Corresponding author.
E-mail address: diego.riano@cena.usp.br (D.M. Riaño-Pachón).
Social media: (D.M. Riaño-Pachón)

https://doi.org/10.1016/j.mex.2022.101803

Specifications table

Subject area	Bioinformatics
More specific subject area	Sequence analysis
Name of your method	CoCoView: A Codon Conservation Viewer via Sequence Logos
Name and reference of original method	Consensus sequence display via Sequence logos [1].
Resource availability	CoCoView.py and additional information are available on project's GitHub: https://github.com/labbces/CoCoView

Method details

Background information

Introduced by Schneider and Stephens (1990) sequence logos are composed of stacks of letters for each position of the multiple sequence alignment, following the conceptual bases of information theory [1–3]. The height of the stack (Rseq [Eq. 1]) is proportional to the conservation of the position; it is defined as the difference between the maximum possible entropy (Smax), defined as log2 of the number of symbols, and the observed entropy (H(l) [Eq. 2]). The height of a given base/amino acid/codon within the stack (Height [Eq. 3]), is measured by the product of its frequency and Rseq [Eq. 3] [1,4].

$$R_{seq}(l) = S_{max} - H(l) = log_2 N - H(l) \tag{1}$$

$$H(l) = -\sum_{n=1}^{N} f(n,l) log_2 f(n,l) \tag{2}$$

$$Height(n,l) = f(n,l) R_{seq}(l) \tag{3}$$

Where f(n,l) represents the frequency of the symbol n (nucleotide, codon, or amino acid) at position l. N is the number of distinct symbols for a given alphabet (nucleotides, codons, or amino acids). Following this, Smax, for DNA and RNA that both have 4 nitrogenated bases, is log2(4) = 2 bits; for proteins with 20 different amino acids it is log2(20) ≈ 4.32 bits and for 64 codons it is log2(64) = 6 bits. Notice that when allowing for ambiguous nucleotides, the number of possible 'codons' would be higher, and so the Smax.

Codons have an important role in biology, they are the information unit in protein-coding sequences, during the process of translation. Changes in codon usage can have important functional consequences, for instance, even changes between synonymous codons can impact protein folding [5] or can affect the rate of protein elongation [6]. Analyzing codon usage on a positional basis allows the identification of consensus/conserved sequences and their variants in DNA regions that represent active, cleavage, and allosteric sites in proteins, and also to analyze regulatory regions, as in mRNA sites that enhance or repress protein translation [7] and mRNA splicing regions [8].

There is a lack of current and easy-to-use tools to visualize codon variation on a positional basis, as previous implementations are no longer available [9]. We developed CoCoView, exploiting Logomaker [10] to create codon sequence logos.

Materials and methods

We developed CoCoView as a single python v3 script, tested on v3.7 and v3.9, to generate the codon sequence logos. It is available at https://github.com/labbces/CoCoView and runs on the command-line interface. CoCoView relies on some external libraries that should be installed in advance: argparse [11], pandas [12], matplotlib [13], logomaker [10], json [14], and biopython [15]. We are using Logomaker as a base due to its flexibility, and also because among other features, it offers the possibility to transform probability matrices into bit matrices and to define where each symbol or glyph will be located on the plot [10].

Input

CoCoView only requires a file with aligned nucleotide sequences in FASTA format that must contain aligned sequences whose length is multiple of three, it assumes that the sequence starts with a complete codon. It also has some command-line switches that can alter the behavior of the program, we will describe these later. As output two files are produced, the matrix computed, either with bits or probabilities, which was used to build the logo and the sequence logo in either png or pdf format.

Command-line arguments for CoCoView

Required, input FASTA file: “fastaFile”: CoCoView only requires a single input file. The script can only deal with single nucleotide symbols following the modern IUPAC nucleotide code nomenclature for incompletely specified bases [16]. Ambiguous nucleotides can pose problems to define the codons, so CoCoView allows the user to filter out sequences based on the fraction of ambiguous nucleotides present, using the argument “degreeOfUncertainty”, see below. We recommend using at least 40 sequences to avoid underestimation of entropy [4].

Optional, –prefixFileName: CoCoView produces two output files. One of them is a matrix that can have bits or probabilities (see –matrixLogoType) and that is used to build the codon logo. The other output file is the codon logo in figure format (see –logoFormat). The value of this argument is used as a prefix to create these two output files.

Optional, –imageTitle: This argument is a string that will appear as the title at the top of the sequence logo. If not provided by the user a title will be automatically generated from the input file name

Optional, –matrixLogoType: CoCoView builds the codon logo based on a matrix, which can be:

- a probability matrix: A matrix of N (rows) x M (columns), in which N are the codon positions in the multiple sequence alignment, and M are the different codons. Each cell has the proportion (probability) of a given codon in a given position. The sum of all codon proportions for a given position must add to 1.
- a bit matrix, default option: This is a transformation of the probability matrix, maintaining the same geometry, using the conceptual framework in equations 1 to 3. Each cell in the matrix represents the Height [Eq. 3] of a given codon in a given position, in bit units.

Optional, –alphaColor: CoCoView can use four different palettes of colors for the codon logos. Codons can be colored following the properties of their corresponding amino acids.The options are: “weblogo_protein (default)”, “charge”, “chemistry” and “hydrophobicity”.

Optional, –degreeOfUncertainty: Ambiguous nucleotides are allowed in the input sequence, however when they are present there is uncertainty about the amino acids they code for. With this argument the user can filter out sequences that have a proportion of ambiguous nucleotides greater than degreeOfUncertainty, using a floating-point number between 0 and 100. For example, a degreeOfUncertainty set to 30% will exclude all sequences of length equal to 12 that have at least 4 ambiguous nucleotides.

Optional, –datasetType: If duplicated sequences are present in the input dataset, setting this argument to ‘nonreduntant’ will remove duplicates from the analyses. This option is useful for small datasets. When very large datasets are used (thousands of sequences with hundreds/thousands of residues), users are advised to use third-party tools to generate non-redundant sequence sets, eg., cd-hit [17] or UCLUST [18]. Setting ‘nonreduntant’ may be of interest when the user wants to visualize less frequent codons. Default value ‘redundant’.

Method validation - brief example

Transcription factors are proteins that bind DNA and regulate the expression of target genes. AP2 is a transcription factor involved in the regulation of growth and development, fruit ripening, defense response, and metabolism in plants [19]. In order to illustrate the benefits of a per-codon variation representation, we generated sequence logos using WebLogo [4] (per nucleotide analysis, Fig. 1A)

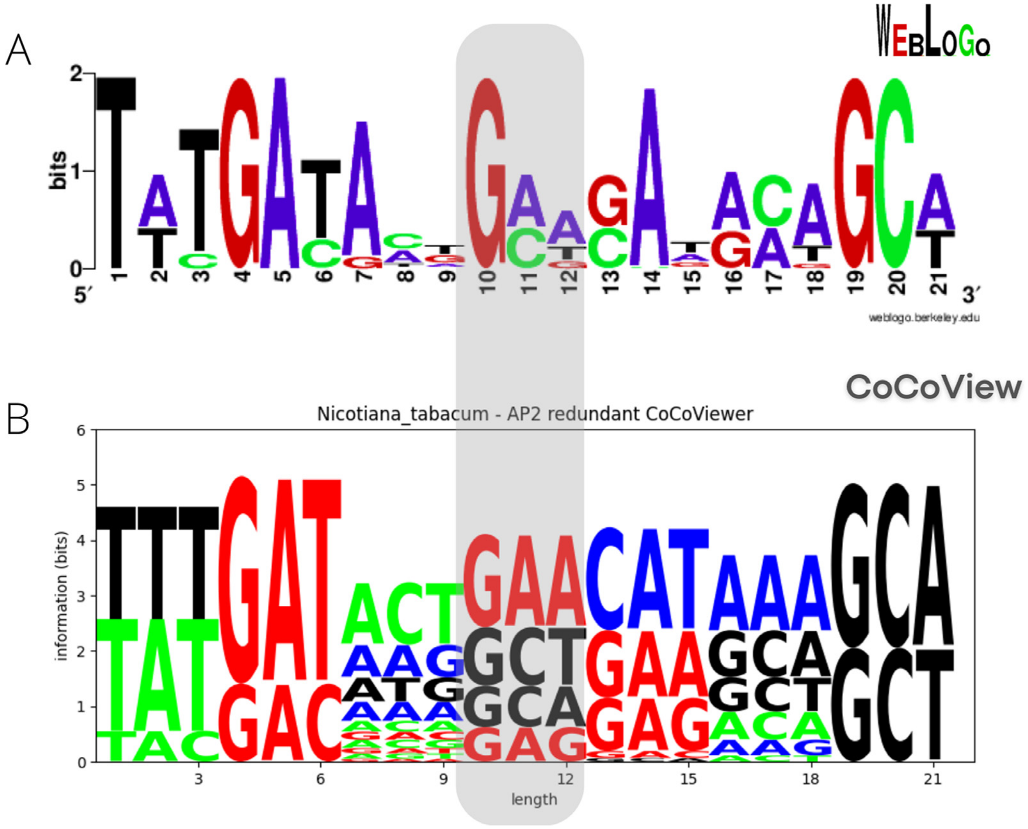

Fig. 1. CoCoView logo based on a multiple sequence alignment of a region of AP2 transcription factor coding sequences from Nicotiana tabacum. (A) Sequence logo generated using WebLogo [4], representing a per-nucleotide analysis. (B) Sequence logo generated using CoCoView (per-codon analysis). A per-nucleotide analysis could erroneously suggest that some codons are common, which can be ruled out on a per-codon visualization. Exemplified by the codon “GAT”, at the position highlighted in gray on both sequence logos, which can be interpreted as a common codon in the per-nucleotide analysis. However, in the per-codon analysis, this codon does not occur at this position.

and CoCoView (per codon analysis) for a region of the multiple sequence alignment of the coding sequences of AP2 from Nicotiana tabacum (Fig. 1B). In Fig. 1A, please note positions 10th to 12th, which represent the 4th codon of that region of the CDS, one could incorrectly draw the conclusion that the triplet “GAT ” is common at that position, based on the conservation of the individual nucleotides. However, when looking at the sequence logo based on condons on Fig. 1B, it is clear that “GAT” is not common at all at this position.

Conclusion

Here we presented CoCoView, a method to construct sequence logos using codons, which allows for a more detailed analysis of sequence conservation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Beatriz Rodrigues Estevam: Software, Writing – original draft, Writing – review & editing. **Diego Mauricio Riaño-Pachón:** Conceptualization, Software, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Acknowledgments

The authors would like to thank Danilo B. Rocha, Scientific initiation student from the Computational, Evolutionary and Systems Biology Laboratory, for fruitful discussions.

Founding: Beatriz R. Estevam acknowledges funding by the Programa Unificado de Bolsas (PUB) - University of São Paulo [project number: 193-2020/21; Diego M. Riaño-Pachón is grateful for fellowship from the National Council for Scientific and Technological Development CNPq (310080/2018-5).

References

[1] T.D. Schneider, R.M. Stephens, Sequence logos: a new way to display consensus sequences, Nucleic Acids Res. 18 (1990) 6097–6100 10.1093%2Fnar%2F18.20.6097.
[2] T.D. Schneider, New approaches in mathematical biology: information theory and molecular machines, in: F. Raulin (Ed.), J. Chela-Flores, Springer, Netherlands, Dordrecht, 1996, pp. 313–321, doi:10.1007/978-94-009-1712-5_28. Chem. Evol. Phys. Orig. Evol. Life Proc. Fourth Trieste Conf. Chem. Evol. Trieste Italy 4–8 Sept. 1995.
[3] T.D. Schneider, Sequence logos, machine/channel capacity, Maxwell's demon, and molecular computers: a review of the theory of molecular machines, Nanotechnology 5 (1994) 1–18, doi:10.1088/0957-4484/5/1/001.
[4] G.E. Crooks, G. Hon, J.-M. Chandonia, S.E. Brenner, WebLogo: a sequence logo generator, Genome Res. 14 (2004) 1188–1190, doi:10.1101/gr.849004.
[5] I.M. Walsh, M.A. Bowman, I.F. Soto Santarriaga, A. Rodriguez, P.L. Clark, Synonymous codon substitutions perturb cotranslational protein folding in vivo and impair cell fitness, Proc. Natl. Acad. Sci. 117 (2020) 3528–3534, doi:10.1073/pnas.1907126117.
[6] P.S. Spencer, E. Siller, J.F. Anderson, J.M. Barral, Silent substitutions predictably alter translation elongation rates and protein folding efficiencies, J. Mol. Biol. 422 (2012) 328–335, doi:10.1016/j.jmb.2012.06.010.
[7] T. Tuller, A. Carmi, K. Vestsigian, S. Navon, Y. Dorfan, J. Zaborske, T. Pan, O. Dahan, I. Furman, Y. Pilpel, An evolutionarily conserved mechanism for controlling the efficiency of protein translation, Cell 141 (2010) 344–354, doi:10.1016/j.cell.2010.03.031.
[8] W. Qu, P. Cingolani, B.R. Zeeberg, D.M. Ruden, A bioinformatics-based alternative mrna splicing code that may explain some disease mutations is conserved in animals, Front. Genet. 8 (2017) https://www.frontiersin.org/article/10.3389/fgene.2017.00038. (accessed April 27, 2022).
[9] V. Sharma, D.P. Murphy, G. Provan, P.V. Baranov, CodonLogo: a sequence logo-based viewer for codon patterns, Bioinformatics 28 (2012) 1935–1936, doi:10.1093/bioinformatics/bts295.
[10] A. Tareen, J.B. Kinney, Logomaker: beautiful sequence logos in Python, Bioinformatics 36 (2020) 2272–2274.
[11] T. Waldmann, argparse: Python command-line parsing library, n.d. https://github.com/ThomasWaldmann/argparse/ (accessed April 17, 2022).
[12] J. Reback jbrockmendel, W. McKinney, J.V. den Bossche, T. Augspurger, M. Roeschke, S. Hawkins, P. Cloud gfyoung Sinhrks, P. Hoefler, A. Klein, T. Petersen, J. Tratner, C. She, W. Ayd, S. Naveh, J.H.M. Darbyshire, M. Garcia, R. Shadrach, J. Schendel, A. Hayden, D. Saxton, M.E. Gorelli, F. Li, M. Zeitlin, V. Jancauskas, A. McMaster, T. Wörtwein, P. Battiston, pandas-dev/pandas: Pandas 1.4.2, Zenodo, 2022. 10.5281/zenodo.6408044.
[13] T.A. Caswell, M. Droettboom, A. Lee, E.S. de Andrade, T. Hoffmann, J. Hunter, J. Klymak, E. Firing, D. Stansby, N. Varoquaux, J.H. Nielsen, B. Root, R. May, P. Elson, J.K. Seppänen, D. Dale, J.-J. Lee, D. McDougall, A. Straw, P. Hobson, hannah, C. Gohlke, A.F. Vincent, T.S. Yu, E. Ma, S. Silvester, C. Moad, N. Kniazev, E. Ernest, P. Ivanov, matplotlib/matplotlib: REL: v3.5.1, Zenodo, 2021. 10.5281/zenodo.5773480.
[14] T. Bray. The JavaScript Object Notation (JSON), data interchange format. 2014.
[15] 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, Biopython: freely available Python tools for computational molecular biology and bioinformatics, Bioinformatics 25 (2009) 1422–1423, doi:10.1093/bioinformatics/btp163.
[16] A. Cornish-Bowden, Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984, Nucleic Acids Res. 13 (1985) 3021–3030.
[17] L. Fu, B. Niu, Z. Zhu, S. Wu, W. Li, CD-HIT: accelerated for clustering the next-generation sequencing data, Bioinformatics 28 (2012) 3150–3152, doi:10.1093/bioinformatics/bts565.
[18] R.C. Edgar, Search and clustering orders of magnitude faster than BLAST, Bioinformatics 26 (2010) 2460–2461, doi:10.1093/bioinformatics/btq461.
[19] C. Gu, Z.-H. Guo, P.-P. Hao, G.-M. Wang, Z.-M. Jin, S.-L. Zhang, Multiple regulatory roles of AP2/ERF transcription factor in angiosperm, Bot. Stud. 58 (2017) 6, doi:10.1186/s40529-016-0159-1.