Brief Report

EvolCat-Python: A User-Friendly Toolkit for Exploring the Language of Life

Robert Friedman *

Posted Date: 27 May 2025

doi: 10.20944/preprints202505.2059.v1

Keywords: bioinformatics; Python, genetic sequence data; sequence alignment; sequence distance; sequence format conversion

 doi:10.20944/preprints202505.2059.v1

Brief Report

EvolCat-Python: A User-Friendly Toolkit for Exploring the Language of Life

Robert Friedman †

Department of Biological Sciences, University of South Carolina, Columbia, SC 29208, USA; bob.friedman.2@gmail.com

† Retired.

Abstract: The blueprint of life, DNA, holds vast amounts of information. Scientists studying how life evolves and how different species are related often need to compare these genetic blueprints. This involves sifting through long strings of genetic code, finding similarities and differences, and understanding how these changes have occurred over time. Many powerful computer programs exist to help with this, but they can sometimes be complex to use, especially when dealing with very large amounts of data or when needing to perform many analytical steps in a row. This report introduces EvolCat-Python, a collection of computer tools designed to make some of these common tasks in genetic analysis easier and more accessible. It builds on ideas from an initial design document for a software package called "EvolCat," which aimed to provide straightforward, command-line tools for scientists, particularly biologists, who need to analyze genetic sequences without necessarily being expert programmers. EvolCat-Python brings this vision to life using the popular and versatile Python programming language. Hosted publicly on GitHub (https://github.com/bob-friedman/EvolCat-Python), this toolkit offers a range of functions for handling and analyzing DNA and protein sequences.

Keywords: bioinformatics; Python; genetic sequence data; sequence alignment; sequence distance; sequence format conversion

1. Introduction: Making Sense of Genetic Information

The blueprint of life, DNA, holds vast amounts of information. Scientists studying how life evolves and how different species are related often need to compare these genetic blueprints. This involves sifting through long strings of genetic code, finding similarities and differences, and understanding how these changes have occurred over time. Many powerful computer programs exist to help with this, such as the phylogeny-focused PAUP* (Swofford 2003) and PAML (Yang 1997), but they can sometimes be complex to use, especially when dealing with very large amounts of data or when needing to perform many analytical steps in a row.

This report introduces EvolCat-Python, a collection of computer tools designed to make some of these common tasks in genetic analysis easier and more accessible. It builds on ideas from an initial design document for a software package called "EvolCat," which aimed to provide straightforward, command-line tools for scientists, particularly biologists, who need to analyze genetic sequences without necessarily being expert programmers. EvolCat-Python brings this vision to life using the popular and versatile Python programming language. Hosted publicly on GitHub (https://github.com/bob-friedman/EvolCat-Python), this toolkit offers a range of functions for handling and analyzing DNA and protein sequences, drawing inspiration from the utility of command-line suites like Phylip (Felsenstein 1989) and EMBOSS (Rice et al. 2000).

2. What Is EvolCat-Python?

Imagine EvolCat-Python as a digital toolbox for biologists. Instead of physical tools, it contains a set of specialized computer scripts. Each script is designed to perform a specific job related to

 doi:10.20944/preprints202505.2059.v1

understanding genetic information. For example, one tool might convert genetic data from one file format to another, making it easier to use with different analysis programs. Another tool might help find specific patterns or "words" within a long stretch of DNA, while others can translate the genetic code of DNA into the protein sequences that build and run our bodies, a foundational step in many molecular evolutionary studies (Stajich et al. 2002).

The original design for EvolCat emphasized the need for tools that could be run from a computer's command line – a text-based way of interacting with the computer. This is particularly useful for scientists because it allows them to automate repetitive tasks by writing simple scripts that tell the computer to run several tools in sequence, perhaps on hundreds or thousands of files. EvolCat-Python follows this principle, providing a set of Python scripts that can be easily run and combined in this way, similar in spirit to how researchers might use components of the BioPerl toolkit (Stajich et al. 2002). It makes use of well-established Python libraries for biology, like Biopython, to handle many of the underlying complex tasks.

3. Key Tools in the EvolCat-Python Toolkit

EvolCat-Python is organized into a library of core functions and a set of ready-to-use scripts, each contributing to a comprehensive analytical environment. A significant aspect of the toolkit involves managing and reformatting genetic data files. Scientists frequently encounter genetic information in diverse formats, and EvolCat-Python facilitates interoperability through several conversion utilities. For instance, gb2fasta.py converts files from the detailed GenBank format, often sourced from major repositories like NCBI (Wheeler et al. 2002), to the simpler FASTA format, which is widely used for sequence data. Other converters include fas2csv.py for changing FASTA to a spreadsheet-friendly CSV format, and fas2phy.py for preparing data for the PHYLIP suite (Felsenstein 1989). The toolkit also supports reverse conversions, such as phy2fas.py (PHYLIP to FASTA) and phy2meg.py. To ensure data consistency, clean_fasta_name.py helps standardize the labels or headers within FASTA files.

Beyond file management, EvolCat-Python provides tools for directly interpreting genetic information. The gbCDS.py script, for example, can delve into information-rich GenBank files to extract the DNA sequence of a gene's coding region—the segment that dictates the structure of a protein—and also provide its translated protein sequence. Similarly, translate_seq.py takes DNA sequences in FASTA format and translates them into their corresponding protein sequences, offering flexibility by allowing users to specify different reading frames or genetic codes.

For comparing and analyzing the sequences themselves, the suite offers several powerful scripts. The dot_plot.py tool generates a visual "dot plot" from two sequences, which is an intuitive way to identify regions of similarity or repeated patterns between them, a fundamental aspect of biological sequence comparison (Pearson and Lipman 1988). When searching for specific, short genetic patterns within a longer DNA sequence, such as signals that might regulate gene activity, approximate_string_match.py can find occurrences even if the match isn't perfect. Within a single DNA sequence, find_tandem_repeats.py is designed to locate sections where a short pattern is repeated consecutively, which can be important for understanding aspects of gene regulation or genetic instability. Furthermore, the count_kmers.py script is useful for counting the frequency of short DNA "words" (k-mers) of a defined length, an analysis that has applications in areas like genome assembly or the identification of unique sequence signatures.

Preparing sequences for detailed evolutionary studies often involves several preprocessing steps, which EvolCat-Python aims to simplify. Evolutionary comparisons typically require sequences to be aligned so that corresponding positions can be compared, often using programs like ClustalW (Thompson et al. 1994). After alignment, the nogaps.py script can be used to remove columns that contain gaps (representing insertions or deletions in some sequences), as these can sometimes complicate subsequent evolutionary calculations. If a researcher needs to focus on a specific part of a longer sequence, extract_region.py allows for the precise cutting out of that segment. For studies involving sequences from multiple sources or experiments, merge_fastas.py can combine several

 doi:10.20944/preprints202505.2059.v1

FASTA files into a single file; notably, if the same sequence label appears in different input files, this tool can concatenate (join end-to-end) their respective sequences. Given the double-stranded nature of DNA, the rev_comp.py script is also provided to generate the "reverse complement" of DNA sequences, a frequently required transformation in sequence analysis.

A core strength of EvolCat-Python, reflecting the emphasis of the original EvolCat design, lies in its tools for calculating evolutionary distances between genetic sequences. The calculate_k2p.py script computes the Kimura 2-Parameter (K2P) distance, a widely used measure that estimates the genetic divergence between DNA sequences while accounting for the fact that certain types of mutations occur more frequently than others (Graur and Li 1997). This script also provides a statistical measure of error for the distance estimate and calculates the ratio of two key mutation types (transitions and transversions). For a broader perspective, calculate_dna_distances.py offers a more comprehensive analysis, calculating several different DNA distance measures—including Jukes-Cantor and K2P distances—between all pairs of sequences provided in a file. Each of these distance metrics is based on different underlying assumptions about the process of DNA evolution (Graur and Li 1997). These distance-based approaches are foundational for many types of phylogenetic reconstruction, including neighbor-joining methods (Saitou and Nei 1987). The accurate estimation of synonymous and nonsynonymous substitution rates (Nei and Gojobori 1986), crucial for inferring natural selection (Hughes et al. 1990), is another area the original design aimed to support, and tools are planned to contribute to such analyses, potentially by preparing data for or parsing output from specialized software like PAML (Yang 1997).

Biological investigations often result in tabular data, and EvolCat-Python includes utilities for handling such datasets. The join_tables.py script can merge two tab-delimited text files (akin to simple spreadsheets) based on values in a common column, an operation similar to a "join" in database systems. For rearranging tabular data, transpose_tsv.py can flip a tab-delimited table, converting its rows into columns and vice-versa, while transpose_text_matrix.py performs a similar function for matrices composed of single characters. The table_to_binary.py script offers a way to simplify numerical tables by converting their entries into binary values (0s and 1s) based on whether the original numbers meet a specified threshold. To help identify recurring entries in datasets, print_duplicate_column_values.py scans the first two columns of a table and reports any values that appear more than once. Finally, sort_numbers.py provides a straightforward utility for taking a list of numbers, each on a new line, and arranging them in numerical order.

Recognizing the central role of the BLAST (Basic Local Alignment Search Tool) program (Altschul et al. 1997) in bioinformatics, the initial EvolCat design placed importance on tools for its use. EvolCat-Python addresses this with several scripts. The parse_blast_text.py script takes the standard text output generated by a BLAST search and reorganizes the information about each sequence match into a more structured and readable format. For researchers who prefer to work with data in tables, blast_to_table.py converts the BLAST text output into a tab-delimited format, which is convenient for importing into spreadsheet programs or for further scripted analysis; this tool also usefully excludes "self-hits" where a sequence simply matches itself. When dealing with many BLAST results, find_blast_top_pairs.py helps to sift through a table of BLAST-like matches to identify the highest-scoring unique pairings between different query and subject sequences, which can be particularly helpful for identifying distinct sets of related genes or proteins. This functionality is essential for tasks like identifying homologous sequences for phylogenetic analysis or exploring gene family evolution across diverse datasets, such as those available through resources like Ensembl (Hubbard et al. 2007). These tools aim to simplify the often complex task of managing and interpreting large volumes of BLAST output, a common step in studies of gene duplication (Friedman et al. 2004) or comparative prokaryotic genomics (Hughes et al. 2005).

Beyond these specific categories, EvolCat-Python also includes other useful utilities. For instance, iupac_to_regexp.py translates standard genetic ambiguity codes (e.g., 'R' representing either A or G) into patterns known as regular expressions, which allow for more flexible and powerful searching of DNA or protein sequences. The overall architecture is supported by libraries, which

 doi:10.20944/preprints202505.2059.v1

consolidate common functions used across the various scripts. The ability to export data for use with other sophisticated phylogenetic programs like Tree-Puzzle (Schmidt et al. 2002) or Molphy (Adachi and Hasegawa 1996) remains an important design consideration for interoperability.

4. How EvolCat-Python Helps Scientists

The main goal of EvolCat-Python is to make common tasks in studying DNA and evolution more straightforward. By providing these tools as simple command-line scripts, scientists can save time by automating repetitive steps instead of performing them manually across numerous files. The command-line approach is also particularly well-suited for processing the large amounts of genetic data generated by modern sequencing technologies. Crucially, when an analysis is conducted using a script, that script serves as an exact and executable record of all the steps taken. This greatly enhances the ability of the scientist to repeat the analysis later, or for other scientists to understand and replicate the work, which is a cornerstone of good scientific practice (Monniaux 2008). Furthermore, scientists can easily link different EvolCat-Python scripts together, or combine them with other existing software, to create custom analysis pipelines precisely tailored to their specific research questions and datasets.

The development of EvolCat-Python involved converting original Perl code with the assistance of AI tools, followed by essential human review and refinement to ensure the accuracy and functionality of the Python versions. This approach facilitates the modernization of useful, established tools, making them available within the widely adopted and versatile Python programming environment.

5. Important Note for Users

The EvolCat-Python tools are provided for research and informational purposes. As with any scientific software, it is important for users to understand the underlying methods of the tools they employ, to critically evaluate the results obtained, and to ensure that these methods and results are appropriate for their specific scientific investigation. Comparing outcomes with those from other established software is often a good practice for validation, but is instead a requirement at this time. Many of the scripts and their underlying implementation of methods require further testing and revision.

6. Conclusion and Availability

EvolCat-Python offers a practical and growing set of Python-based tools designed to support scientists working with genetic sequences and studying the processes of evolution. It aims to fulfill the vision of the original EvolCat design by providing accessible, command-line driven utilities that can simplify complex tasks, enable the analysis of large datasets, and contribute to more efficient and reproducible research. The toolkit is open-source, encouraging community use and feedback.

EvolCat-Python is freely available on GitHub: https://github.com/bob-friedman/EvolCat-Python Instructions for using individual scripts can be found in the documentation within the repository.

Funding: This research received no external funding.

Acknowledgement: Original content by the author, which was further adapted and enhanced by an AI Assistant, Jules/Gemini 2.5 Pro, (Google, ver. 05/06/2025).

Conflicts of Interest: The author declares no conflict of interest.

 doi:10.20944/preprints202505.2059.v1

References

1. Adachi J, Hasegawa M (1996) MOLPHY version 2.3: programs for molecular phylogenetics based on maximum likelihood. Computer Science Monographs of Institute of Statistical Mathematics 28: 1-150.
2. Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 25: 3389-3402.
3. Felsenstein, J. (1989). PHYLIP - Phylogeny Inference Package (Version 3.2). Cladistics, 5, 164-166.
4. Friedman R, Ekollu V, Rose JR, Hughes AL (2004) Dblox: a genome-wide test for ancient segmental duplication. Bioinformatics 20: 2834-2835.
5. Graur D, Li, W-H (1997). Molecular evolution. Sunderland, MA: Sinauer Associates.
6. Hubbard TJ, Aken BL, Beal K, Ballester B, Caccamo M, Chen Y, et al. (2007) Ensembl 2007. Nucleic Acids Res. Database issue.
7. Hughes AL, Ekollu V, Friedman R, Rose JR (2005) Gene family content-based phylogeny of prokaryotes: the effect of search criteria. Syst Biol 54: 268-276.
8. Hughes AL, Ota T, Nei M (1990) Positive Darwinian selection promotes charge profile diversity in the antigen binding cleft of class I major-histocompatibility-complex molecules. Mol Biol Evol 7: 515-524.
9. Nei M, Gojobori T (1986) Simple methods for estimating the numbers of synonymous and nonsynonymous nucleotide substitutions. Mol Biol Evol 3: 418-426.
10. Pearson WR, Lipman DJ (1988) Improved Tools for Biological Sequence Comparison. PNAS 85: 2444-2448.
11. Rice P, Longden I, Bleasby A (2000) EMBOSS: The European Molecular Biology Open Software Suite. Trends in Genetics 16: 276-277.
12. Saitou N, Nei M (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol Biol Evol 4: 406-425.
13. Schmidt HA, Strimmer K, Vingron M, von Haeseler A (2002) TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing. Bioinformatics. 18:502-504.
14. Stajich JE, Block D, Boulez K, Brenner SE, Chervitz SA, Dagdigian C, et al. (2002) The Bioperl toolkit: Perl modules for the life sciences. Genome Res 12: 1611-1618.
15. Swofford DL (2003) PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods). Version 4. Sinauer Associates, Sunderland, Massachusetts.
16. Thompson JD, Higgins DG, Gibson TJ (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. Nucleic Acids Research 22: 4673-4680.
17. Wheeler DL, Church DM, Federhen S, Lash AE, Madden TL, Pontius JU, et al. (2002) Database resources of the National Center for Biotechnology Information. Nucleic Acids Res 30: 13-16.
18. Yang Z (1997) PAML: a program package for phylogenetic analysis by maximum likelihood. Computer Applications in BioSciences 13: 555-556.