PxBLAT: An Ergonomic and Efficient Python Binding Library for BLAT

Yangyang Li[1] **and Rendong Yang**[1,2] ✉

[1]Department of Urology, Northwestern University Feinberg School of Medicine, Chicago, IL 60611; [2]Robert H. Lurie Comprehensive Cancer Center, Northwestern University Feinberg School of Medicine, Chicago, IL 60611

Summary: We introduce PxBLAT, a Python library designed to enhance usability and efficiency in interacting with the BLAST-like alignment tool (BLAT). PxBLAT provides an intuitive application programming interface (API) design, allowing the incorporation of its functionality directly into Python-based bioinformatics workflows. Besides, it integrates seamlessly with Biopython and comes equipped with user-centric features like server readiness checks and port retry mechanisms. PxBLAT removes the necessity for system calls and intermediate files, as well as reducing latency and data conversion overhead. Benchmark tests reveal PxBLAT gains a ~20% performance boost compared to BLAT in the Python environment.

Availability and Implementation: PxBLAT supports Python (version 3.8+), and pre-compiled packages are released via PyPI (https://pypi.org/project/pxblat/) and Bioconda (https://anaconda.org/bioconda/pxblat). The source code of PxBLAT is available under the terms of an open-source MIT license and hosted on GitHub (https://github.com/ylab-hi/pxblat). Its documentation is available on ReadTheDocs (https://pxblat.readthedocs.io/en/latest/).

Software Libraries | Sequence Analysis | BLAT

Correspondence: ***rendong.yang@northwestern.edu***

Introduction

The rise of Python as a preferred programming language within bioinformatics is widely acknowledged as a result of its user-friendly nature, extensive libraries, and unparalleled versatility (Perkel). A variety of libraries have been crafted to augment Python's interface, thereby amplifying the adaptability and compatibility of bioinformatics tools (Putri et al.; Cock et al.). For instance, Biopython (Cock et al.), a preeminent bioinformatics library, furnishes interfaces to tools like Basic Local Alignment Search Tool (BLAST) (Altschul et al.) and Clustal (Higgins and Sharp).

Besides, the relentless progression of genome sequencing technologies has precipitated a dramatic surge in the availability of genomic data. The necessity for tools to decipher and manipulate these data has become paramount within both the research and clinical domains. BLAT (Kent) is a standout within the bioinformatics landscape and is recognized for its capability to swiftly conduct genome sequence alignments. It offers a faster alternative to BLAST (Altschul et al.) for aligning DNA sequences to the human genome (Kent). Despite its widespread use and endorsement within the bioinformatics community, interfacing with BLAT can present challenges. BLAT, implemented in the C language, solely extends utilities via command-line interfaces (CLIs), rendering its integration into Python-based projects inconvenient. Moreover, BLAT triggers system calls and data conversions that cause significant overhead. Until now, a comprehensive approach to enhancing the usability of BLAT has been absent when incorporated into Python-based projects.

Here, we propose PxBLAT, which empowers users to utilize BLAT programmatically, facilitating its seamless integration into novel algorithms or analytical pipelines. PxBLAT serves as a bridge, bringing the high-performance capabilities of BLAT into the Python environment while maintaining reproducibility. The overarching aim of this work is to fill the observed gap by providing a Python binding library specifically tailored for BLAT, addressing both its efficiency and ergonomic concerns.

Implementation

The design of PxBLAT adheres to the principles of readability and simplicity, aiming for an intuitive user experience that diminishes the learning curve. Striving to curtail complexity while enhancing usability and performance, we extracted the implementation of BLAT from the codebase of the UCSC Genome Browser (UCSC), thus reducing dependencies. We retain the original C codebase and reimplement the utilities of BLAT (V37.1) including *faTwoBit*, *gfServer*, and *gfClient* CLI, using the C++ programming language (Kent). Subsequently, we integrate the current C++ code and develop PxBLAT via Pybind11 (Jakob et al., 2016). The approach facilitated direct interaction with the functions of the BLAT without modifying its original source code, thereby preserving the integrity and performance of the BLAT while broadening its capabilities.

The query result of PxBLAT conforms to the *QueryResult* class of Biopython (Cock et al.), enabling manipulation of the query result using Biopython

features (Listing 1). Furthermore, PxBLAT eliminates the need for intermediate files, allowing all operations to be executed in memory. This eradicates the common obstacle of data conversion into specific formats, enabling users to focus more on the core sequence alignment task. Input and output files have been made optional, offering more flexible choices. The use of system calls, though functional, can induce latency and create performance bottlenecks. PxBLAT reduces the need for system calls, thereby improving efficiency (Table 1). Moreover, PxBLAT allows for the retrieval of server status without manipulating log files, a process that can be troublesome in concurrent environments. The library incorporates ergonomic features such as checking and waiting for server readiness for alignment, retrying different ports if the current one is occupied, and using an existing server if one has already been started.

We provide a variety of examples and documentation to help the user get started (Listing 1). Specifically, PxBLAT provides APIs, including Class *Server*, *Client*, and other free functions, to reproduce the BLAT suit. Class *Server* and *Client* have the same utilities as CLI *gfServer* and *gfClient*, respectively, but with greater flexibility. Free functions, for example, *start_server*, *query_server*, *status_server*, *fa2twobit*, *twobit2fa*, etc., serve as potential usage in different contexts. The library has been tested and developed with continuous integration (CI) and continuous development (CD) to ensure code quality. It uses type annotations for public classes and functions to guarantee quality and correctness via a type checker and static analyzer. The type annotations also make PxBLAT more user-friendly in developing environments, where the function signatures can be suggested and corrected automatically. Besides, PxBLAT contains CLIs implemented by its API. The CLI includes completion for different shells, improving versatility.

```python
from pxblat import Server
from pxblat import Client

client = Client(
    host="localhost",
    port=65000,
    seq_dir="ref/",
    min_score=20,
    min_identity=90,
)

server_option =
↪  Server.create_option().build()
with Server(
    host="localhost",
    port=65000,
    two_bit="ref/reference.2bit",
    option=server_option
) as server:
    work()  # work that consumes time
    server.wait_for_ready()
    result1 = client.query("ATCG")
    result2 = client.query("AtcG")
    result3 = client.query(["ATCG", "ATCG"])
    result4 = client.query(["cgTA",
    ↪  "fasta.fa"])

    for hsp in result1.hsps:
        print(f"{hsp}")
```

Listing 1. | **API Example.** The code snippet shows how to use the API of PxBLAT, and the query result can be iterated. More code examples can be found at `https://pxblat.readthedocs.io/en`

Benchmarking

The result and performance of PxBLAT are benchmarked against BLAT (V37.1) using a dataset of 1000 FASTA Files. The testing dataset, which consists of five groups, is randomly sampled from chromosome 20 of the Human Genome (hg38), and each sample includes one sequence. The dataset includes varying lengths of sequence to cover a range of typical usage scenarios. The length of each sequence ranges from 1000 bp to 3000 bp (Figure S1). We summarized the high-scoring pairs (HSPs) of all hits in each sample derived from BLAT and PxBLAT and compared their predictions. We observed that PxBLAT reproduces the HSPs of BLAT across all tested samples, which ensures the correctness of PxBLAT (Table S1).

The benchmarking was conducted on an Apple M1 Pro with macOS 13.4.1 22F82 arm64. We use system calls to launch BLAT and the *time* library to measure the execution time. Generally, PxBLAT gains ~20 % speedup compared to BLAT (Table 1). In summary, PxBLAT offers tangible benefits according to reduced execution time and improved user experience, proving its value as an enhancement to BLAT's functionality.

Acknowledgements

Special thanks to the team who maintain the UCSC codebase and users from the bioinformatics community whose valuable feedback and suggestions were pivotal in refining PxBLAT's design and functionality.

Conflict of interest

None declared.

Funding

This work was supported by the National Institute of General Medical Sciences [R35GM142441].

Table 1. Performance Benchmarking

Data Group	Samples (No.)	BLAT (Sec.)	PxBLAT (Sec.)	Speedup
benchmark/fas/1	200.00	61.40	50.41	1.22
benchmark/fas/2	200.00	61.70	51.10	1.21
benchmark/fas/3	200.00	55.34	46.44	1.19
benchmark/fas/4	200.00	56.98	47.72	1.19
benchmark/fas/5	200.00	54.55	46.44	1.17

Data availability

The dataset for benchmarking and testing is available at the GitHub repository https://github.com/ylab-hi/pxblat. The path of the dataset is *benchmark/fas*.

Code availability

The PxBLAT, along with the source code, is publicly available in the GitHub repository at https://github.com/ylab-hi/pxblat. The documentation is available at ReadtheDocs https://pxblat.readthedocs.io/en/latest/. The script for benchmarking is available at *tests/test_result.py* in the repository.

Reference

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. Basic local alignment search tool. 215(3):403–410. doi: 10.1016/S0022-2836(05)80360-2.

Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., and family=Hoon, p. u., given=Michiel J. L. Biopython: Freely available Python tools for computational molecular biology and bioinformatics. 25(11):1422–1423. doi: 10.1093/bioinformatics/btp163.

Higgins, D. G. and Sharp, P. M. CLUSTAL: A package for performing multiple sequence alignment on a microcomputer. 73(1):237–244. doi: 10.1016/0378-1119(88)90330-7.

Jakob, W., Rhinelander, J., and Moldovan, D. pybind11 — seamless operability between c++11 and python, 2016. https://github.com/pybind/pybind11.

Kent, W. J. BLAT—The BLAST-Like Alignment Tool. 12(4):656–664. doi: 10.1101/gr.229202.

Perkel, J. M. Programming: Pick up Python. 518(7537):125–126. doi: 10.1038/518125a.

Putri, G. H., Anders, S., Pyl, P. T., Pimanda, J. E., and Zanini, F. Analysing high-throughput sequencing data in Python with HTSeq 2.0. 38(10):2943–2945. doi: 10.1093/bioinformatics/btac166.