Bioinformatics, 36(3), 2020, 918–919
doi: 10.1093/bioinformatics/btz615
Advance Access Publication Date: 2 August 2019
Applications Note

Genome analysis

PyRanges: efficient comparison of genomic intervals in Python

Endre Bakken Stovner[1,2,3,4,*] **and Pål Sætrom**[1,2,3,4]

[1]Department of Computer Science, [2]Department of Clinical and Molecular Medicine, [3]Bioinformatics Core Facility and [4]Department of Public Health and Nursing, K.G. Jebsen Center for Genetic Epidemiology, Norwegian University of Science and Technology, Trondheim 7013, Norway

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on May 31, 2019; revised on July 26, 2019; editorial decision on July 30, 2019; accepted on August 1, 2019

Abstract

Summary: Complex genomic analyses often use sequences of simple set operations like intersection, overlap and nearest on genomic intervals. These operations, coupled with some custom programming, allow a wide range of analyses to be performed. To this end, we have written PyRanges, a data structure for representing and manipulating genomic intervals and their associated data in Python. Run single threaded on binary set operations, PyRanges is in median 2.3–9.6 times faster than the popular R GenomicRanges library and is equally memory efficient; run multi-threaded on 8 cores, our library is up to 123 times faster. PyRanges is therefore ideally suited both for individual analyses and as a foundation for future genomic libraries in Python.

Availability and implementation: PyRanges is available as open source under the MIT license at https://github.com/biocore-NTNU/pyranges and the documentation exists at https://biocore-NTNU.github.io/pyranges/

Contact: endrebak85@gmail.com

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Comparing sets of intervals is a fundamental task in genomics, and a few basic operations allow for answering many scientific questions. For example, to find genes potentially targeted by a transcription factor, one can intersect the sets of intervals representing gene positions and representing transcription factor binding sites to identify those that overlap.

Several toolboxes of genomic operations exist, such as bedtools (Quinlan and Hall, 2010) and bedops (Neph *et al.*, 2012) for the command line and GenomicRanges (Lawrence *et al.*, 2013) for the R programming environment.

GenomicRanges is a data structure for representing and operating on genomic intervals and their metadata, which are stored as a 2D table in memory. By providing methods for access and for set operations on genomic intervals, programmers can use the R programming language to manipulate and analyse the contents of GenomicRanges. Consequently, GenomicRanges is a powerful tool for writing complex and custom genome analyses. Indeed, in R, GenomicRanges is a foundational library, and a cornerstone of genomics packages in the R Bioconductor project (Gentleman *et al.*, 2004).

Python is currently ranked as the most popular programming language in the world (according to *IEEE Spectrum*'s compound metric; Cass and Bulusu, 2018, https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2018) and is much used in data science and bioinformatics, yet it lacks a GenomicRanges implementation. Instead, the current way to do genomics in Python is to use pybedtools, a Python wrapper for bedtools (Dale *et al.*, 2011). This solution has several drawbacks including relying on external software, heavy I/O and disk use for each operation, lack of position-based querying, the inability to use the Python data science stack on pybedtools objects and limiting users to work with a few bioinformatics file formats instead of arbitrary genomic data (see Supplementary Discussion for details). The PyRanges library remedies this situation by providing a Python GenomicRanges implementation which is multi-threaded, fast and memory efficient.

2 Library

2.1 Implementation

The PyRanges data structure is logically represented as a 2D table. Each row represents an interval, and the columns each describe either a part of the location (chromosome, start position, end position and optionally, strand) or metadata (name, score, exon number or any arbitrary value desired by the user). The underlying implementation uses a dictionary that maps chromosome and strand pairs to their respective 2D tables; however, this division is largely invisible to the user. The data in the 2D tables are stored in Pandas DataFrames, thus allowing the vast Python science stack to be used seamlessly with PyRanges. Furthermore, DataFrames allow for storing the data contiguously in native data types, such as integers, floats or categoricals, to ensure memory efficiency.

To make PyRanges fast, its operations are written in Cython or C. Moreover, by keeping the data belonging to each chromosome in separate DataFrames, these logically distinct data can easily be independently processed. In this way we avoid the substantial time costs of splitting and merging the data for each operation. PyRanges provides parallel processing through the Ray framework (Moritz *et al.*, 2018), resulting in a speed-up provided the data are sufficiently big (see Section 2.3).

2.2 Functionality

PyRanges' main functionality includes functions for reading genomic intervals from files, and unary and binary functions for manipulating one and two sets of genomic intervals. File reading functions support common formats such as bed, GTF/GFF and bam. Unary functions manipulate single PyRanges by subsetting, clustering or computing coverage; that is, the number of intervals overlapping each genomic position. Binary functions include operations, such as intersection, nearest and subtract that create a new set of genomic intervals by comparing two sets of intervals. See the Supplementary Material for a full list of PyRanges' operations.

PyRanges also uses and provides two stand-alone libraries useful beyond bioinformatics. One library (pyrle) implements run-length encoding arithmetic, which is useful to compactly represent and efficiently do arithmetic on the coverage (or any other nucleotide-associated score) of sets of regions. The other library (NCLS) implements the Nested Containment List, which is an immutable interval tree with better memory efficiency and speed than a regular interval tree both for tree construction and interval queries (see Supplementary Timings).

2.3 Performance

The PyRanges library has been extensively benchmarked for both speed and memory use (Fig 1; Supplementary Timings). We used two types of data for testing: (i) libraries of reads only, i.e. they included no metadata and were hence more lightweight and (ii) GTF annotations. We used unsorted test files generated by bedtools random for hg38 to simulate the read files. To create a large GTF, we used sampling with replacement on the Gencode hg38 GTF.

For binary operations, PyRanges in single-threaded mode was 6.5–31 (median 14) and 9.8–36 (median 24) times faster than pybedtools on 1e6 and 1e7 intervals, respectively. Compared to GenomcRanges, PyRanges was 1.3–16 (median 2.3) and 1.9–84 (median 9.6) times faster. Run multi-threaded on 8 cores, the speed-ups for the same operations on 1e7 intervals were 13–63 and 1.8–123 times compared to pybedtools and GenomicRanges, respectively. For all operations, PyRanges run single-threaded on 1e7 intervals had a median speed-up of 26 and 4.0 times and used a median 5.7 and 5.1 times less memory compared to pybedtools and GenomicRanges, respectively.

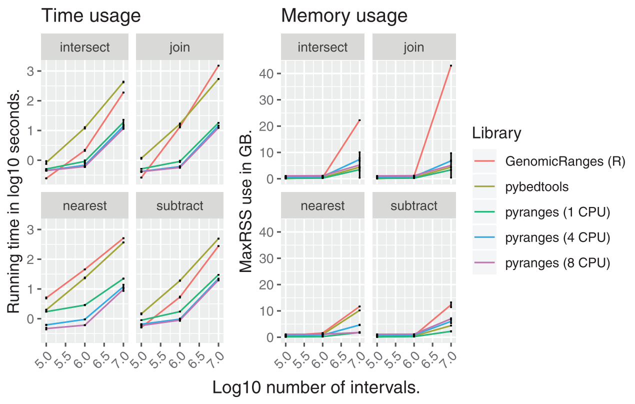

Fig. 1. Running time (left) and memory usage (right) as a function of the number of intervals for four common binary functions on genomic intervals; see Supplementary Timings for complete benchmark results

3 Conclusion

PyRanges is an efficient and feature-rich library for genomics in the extremely popular Python programming language, and the only one of its kind. We therefore expect it to be a boon to current and future bioinformaticians and researchers working in Python.

Funding

This work was supported by the Research Council of Norway [230338]; and Stiftelsen K.G. Jebsen.

Conflict of Interest: none declared.

References

Dale,R.K. *et al.* (2011) Pybedtools: a flexible Python library for manipulating genomic datasets and annotations. *Bioinformatics*, **27**, 3423–3424.

Gentleman,R.C. *et al.* (2004) Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol.*, **5**, R80.

Lawrence,M. *et al.* (2013) Software for computing and annotating genomic ranges. *PLoS Comput. Biol.*, **9**, 1–10.

Moritz,P. *et al.* (2018) Ray: A distributed framework for emerging AI applications. In: *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pp. 561–577.

Neph,S. *et al.* (2012) Bedops: high-performance genomic feature operations. *Bioinformatics*, **28**, 1919–1920.

Quinlan,A.R. and Hall,I.M. (2010) Bedtools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, **26**, 841–842.