ruby-htslib と hts.cr：Ruby および Crystal 向け HTSlib インターフェース

kojix2

2026 年 8 月 23 日

概要

HTSlib は、ゲノムデータの読み書きに用いられる C ライブラリであり、samtools および bcftools の基盤をなす (Bonfield et al., 2021; Danecek et al., 2021)。hts.cr と ruby-htslib は、HTSlib を Crystal および Ruby から利用できるようにする。両ライブラリは、SAM/BAM/CRAM、VCF/BCF、インデックス付き参照配列（Faidx）、Tabix ファイル、pileup をサポートする。

ruby-htslib は、システムの HTSlib に 直接リンクする Ruby のネイティブ C 拡張を使用する。一方、hts.cr は Crystal の FFI を介して HTSlib を呼び出す。これらのライブラリにより、Ruby または Crystal を用いる研究者や開発者は、外部コマンドを介さずにゲノムデータをプログラム内で処理できる。

必要性

ゲノムデータに対する多くの定型処理には、専用のコマンドラインツールが提供されている。しかし、研究上の問いに応じて複数のフィールドを組み合わせて選別・集計したり、レコードを探索的に調べたり、結果を独自のデータ構造に格納して別のライブラリと連携させたりする場合には、既存ツールのオプションだけでは処理を表現できず、独自のスクリプトやプログラムが必要となることがある。HTSlib は、ファイル形式の解釈、圧縮、インデックスを用いた検索などを C API として提供する。これらを高水準言語から扱うには、関数を呼び出すだけでなく、HTSlib のファイル、ヘッダー、レコードとその生存期間を、各言語のデータモデルに対応づける必要がある。

Python、Nim、Rust、C++ には、pysam、cyvcf2、hts-nim、rust-htslib、vcfpp など、成熟した HTSlib インターフェースが存在する (Köster et al., 2025; Li, 2024; Pedersen & Quinlan, 2017, 2018; Pysam Developers, 2025)。ただし、これらは各言語向けのインターフェースであり、Ruby または Crystal の プログラムから直接利用することはできない。Ruby では、BioRuby のプラグインである bio-samtools が、SAMtools を利用したアラインメント処理、pileup、変異解析、可視化の高水準インターフェースを提供してきた (Etherington et al., 2015; Goto et al., 2010; Ramirez-Gonzalez et al., 2012)。これに対し、ruby-htslib は独立した HTSlib に直接接続し、アラインメントとバリアントのファイル、ヘッダー、レコードを、Ruby で解析プログラムを構築するための構成要素として提供する。hts.cr は、同じ HTSlib の機能を Crystal の静的型とネイティブコンパイルから利用し、処理の中心部分を別の言語で実装することなく、ストリーミング型のバイオインフォマティクスツールを記述できるようにする。両ライブラリは、HTSlib の機能をそれぞれの言語に適した形で 組み合わせられるようにする。

ソフトウェア設計

両ライブラリは、HTSlib のファイル、ヘッダー、レコードを共通のレコード指向モデルで表す。このモデルは、ネイティブリソースの生存期間を管理し、走査中だけ借用する値と走査後も保持する値を区別する。Bam は SAM/BAM/CRAM、Bcf は VCF/BCF のファイルを表し、それぞれが対応する Header を保持する。ファイルオブジェクトの each は Record を一件ずつ返し、走査中は同じオブジェクトとネイティブバッファを使い回す。この方法はファイル全体をメモリに展開せず、割り当てを抑えて処理できる一方、走査後もレコードを保持する場合には複製する必要がある。レコードのフィールドは Ruby または Crystal のメソッドから参照・更新でき、アラインメントの 補助タグとバリアントの INFO・FORMAT フィールドは型付きの値として扱われる。領域検索と書き込みにも同じファイル、ヘッダー、レコードのオブジェクトを用いる。

この共通モデルにすべての操作を当てはめるのではなく、アクセスの単位が異なる機能には専用のオブジェクトを用いる。インデックス付き FASTA は参照配列の一部を取得する Faidx、Tabix で索引付けされたテ

キストは指定領域に重なる行を返す `Tabix` として表す。pileup は独立した ファイル形式ではなく、`Bam` から得られる位置単位のビューとして、参照配列上の位置とそこに重なるアラインメントの集合を列挙する。各ファイルオブジェクトはブロック付きで開くことができ、ブロックの処理後にファイルを閉じ、ネイティブリソースを解放する。

このオブジェクトモデルを共有する一方、借用した値の公開方法と、値を保持するための経路には各言語の利用形態を反映する。`ruby-htslib` の C 拡張は、HTSlib のポインターを型付きの Ruby オブジェクトに格納し、ガベージコレクションの際に対応する解放関数を呼び出す。走査ではネイティブのレコードバッファを使い回し、走査後も保持するレコードやフィールドは独立した Ruby オブジェクトに コピーする。BCF の FORMAT フィールドには、コピーせずに再利用バッファを参照する借用 view も用意する。したがって、値を Ruby の配列や文字列として保持する経路と、借用した値を走査中に処理して割り当てを抑える経路を、用途に応じて選択できる。

`hts.cr` は、HTSlib のポインターを Crystal の静的型付きオブジェクトで包み、借用と所有の区別を型とブロックの有効範囲に反映する。ネイティブバッファは、ブロック内だけ有効な `Slice` や借用 view として参照でき、フィールドをプリミティブ値の iterator で処理すれば、中間的な配列や文字列を生成しない。走査後も必要な値だけをコピーすることで、ヒープへの割り当てとガベージコレクタの負荷を抑える。この構成は、Crystal のネイティブコンパイルを利用した ストリーミングツールにおいて、HTSlib のデータを効率よく逐次処理できるようにする。

性能評価

各ライブラリの実行時オーバーヘッドを調べるため、同じ処理を C、`hts.cr`、`ruby-htslib` で実装し、スループットを比較した。三つの実装はいずれも同じ HTSlib（1.22.1-51-gcd2a6f61）を使用し、Ubuntu 26.04 LTS 上でシングルスレッドで実行した。C の基準実装は gcc 15.2.0（`-O2`）、`hts.cr` 0.4.0 は Crystal 1.21.0（LLVM 20.1.8、`--release`）、`ruby-htslib` 0.6.0 は Ruby 4.0.6 でビルドした。したがって、この比較は、HTSlib 自体の違いではなく、言語境界と各 API のデータ表現に伴うコストを反映する。

評価には、2 Mbp の参照配列に対する 100 bp の single-end リード 300,000 件を含む BAM（平均深度約 15×）と、GT、DP、AD、GL の FORMAT フィールドを持つ 20 サンプル、50,000 サイトの BCF を合成して用いた。BAM は座標順にソートし、両ファイルにインデックスを 作成した。領域検索と pileup の対象は、14,902 件のリードが重なる 100 kb の区間（`chr1:500,000-600,000`）とした。pileup では base quality の下限を 13 とし、mapping quality によるフィルタリングは行わなかった。各ワークロードを 5 回実行し、表にはスループットの中央値を示す。初回後は入力がページキャッシュに収まった。スクリプトは `benchmark` ディレクトリに収録している。

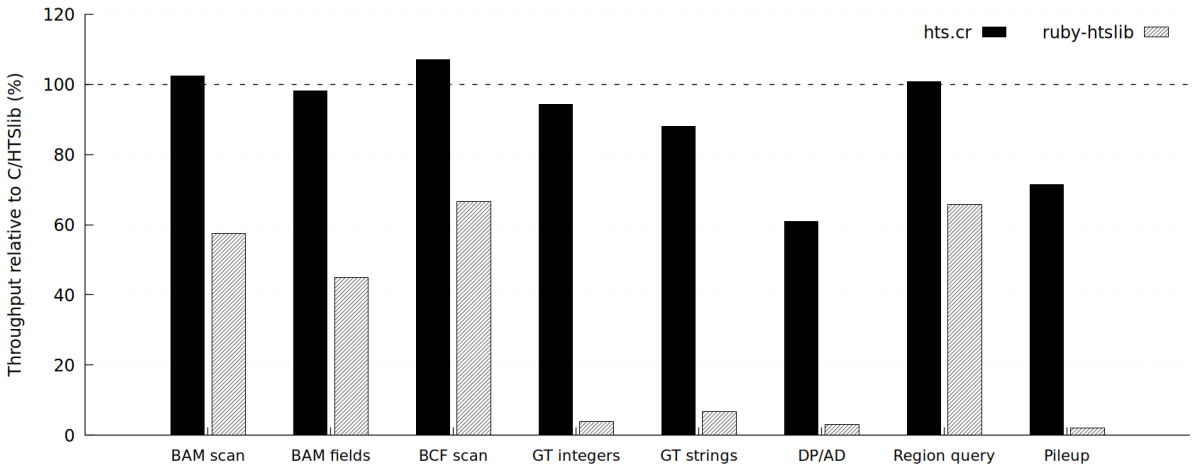

Figure 1: C/HTSlib 実装を基準としたスループットの中央値。領域検索には 20 回反復時の 1 回あたりの平均値を用いた。

ワークロード	C/HTSlib	`hts.cr`	`ruby-htslib`
BAM レコードの逐次走査	2,938,000 records/s	3,009,000 records/s	1,689,000 records/s
flag と座標を参照する BAM 走査	2,949,000 records/s	2,895,000 records/s	1,326,000 records/s
BCF レコードの逐次走査	1,624,000 records/s	1,738,000 records/s	1,080,000 records/s
FORMAT/GT の整数走査	1,425,000 records/s	1,345,000 records/s	54,000 records/s
FORMAT/GT の文字列変換	438,000 records/s	386,000 records/s	29,000 records/s
FORMAT/DP・AD の走査	1,357,000 records/s	827,000 records/s	43,000 records/s
領域検索（初回／20 回反復時の平均）	2,622,000 / 2,674,000 records/s	2,634,000 / 2,694,000 records/s	1,725,000 / 1,761,000 records/s
pileup の塩基カウント	6,792,000 columns/s	4,846,000 columns/s	140,000 columns/s

レコードの逐次走査、整数値の FORMAT フィールドへのアクセス、領域検索では、`hts.cr` の スループットは C の 94〜107% であった。サンプルごとの GT を文字列に変換する処理では C の 88%、FORMAT/DP・AD の走査では 61%、pileup の塩基カウントでは 71% であった。ネイティブバッファを直接処理できる経路では C に近い性能が得られ、文字列の生成や 高水準の view を介した値の走査には追加のコストが伴うことを示している。

`ruby-htslib` は、逐次走査と領域検索で C の 45〜67% のスループットを示した。FORMAT フィールドや pileup の処理では差が大きく、C に対するスループットは 2〜7% であった。FORMAT/GT の整数走査と FORMAT/DP・AD の走査には、割り当てを抑える iterator または 借用 view を用いたが、Ruby の値への変換とブロック呼び出しはレコード内の値ごとに発生する。GT の文字列変換と pileup では、所有型の値を返す `genotype_strings` と `each_base_counts` を用いており、コピーとオブジェクト生成のコストも含まれる。なお、三つの実装はすべて同じレコード数を処理し、pileup で数えた塩基の総数も 1,017,916 で 一致した。

この評価は、各言語の優劣を一般化するものではなく、本稿で提供する API の代表的な実行経路を比較したものである。結果は単一の実行環境と合成データセットに基づくため、実データを用いたアプリケーション全体の実行時間や、大規模ファイルに対する性能を予測するものではない。

研究への影響

BioCrystal では、`hts.cr` は、式に基づいて BAM/CRAM をフィルタリングする コマンドラインツール `bam-filter` と、libui-ng を用いた BAM viewer `bamboo` に 利用されている (bio-cr, 2024b, 2024a)。

AI 利用に関する開示

本稿の作成に Codex を使用した。

謝辞

`ruby-htslib` の開発は、Ruby Association Grant 2020 から一部支援を受けた (Ruby Association, 2020)。著者は、HTSlib、SAMtools、BCFtools、Ruby、Crystal の開発者と、関連するバイオインフォマティクスコミュニティに感謝する。資金提供者はソフトウェアの設計と投稿の決定に関与していない。著者は利益相反がないことを宣言する。

参考文献

bio-cr. (2024a). *Bamboo: A lightweight BAM file viewer written in Crystal.* GitHub repository. https://github.com/bio-cr/bamboo

bio-cr. (2024b). *Bam-filter: Use simple expressions to filter a BAM/CRAM file*. GitHub repository. https://github.com/bio-cr/bam-filter

Bonfield, J. K., Marshall, J., Danecek, P., Li, H., Ohan, V., Whitwham, A., Keane, T., & Davies, R. M. (2021). HTSlib: C library for reading/writing high-throughput sequencing data. *GigaScience*, *10*(2), giab007. https://doi.org/10.1093/gigascience/giab007

Danecek, P., Bonfield, J. K., Liddle, J., Marshall, J., Ohan, V., Pollard, M. O., Whitwham, A., Keane, T., McCarthy, S. A., Davies, R. M., & Li, H. (2021). Twelve years of SAMtools and BCFtools. *GigaScience*, *10*(2), giab008. https://doi.org/10.1093/gigascience/giab008

Etherington, G. J., Ramirez-Gonzalez, R. H., & MacLean, D. (2015). Bio-samtools 2: A package for analysis and visualization of sequence and alignment data with SAMtools in Ruby. *Bioinformatics*, *31*(15), 2565–2567. https://doi.org/10.1093/bioinformatics/btv178

Goto, N., Prins, P., Nakao, M., Bonnal, R., Aerts, J., & Katayama, T. (2010). BioRuby: Bioinformatics software for the Ruby programming language. *Bioinformatics*, *26*(20), 2617–2619. https://doi.org/10.1093/bioinformatics/btq475

Köster, J., Schröder, C., Marks, P., Lähnemann, D., Holtgrewe, M., Gehring, J., & contributors. (2025). *Rust-htslib: HTSlib bindings and a high-level Rust API for reading and writing BAM, VCF, and BCF files*. GitHub repository / crates.io. https://github.com/rust-bio/rust-htslib

Li, Z. (2024). Vcfpp: A C++ API for rapid processing of the variant call format. *Bioinformatics*, *40*(2), btae049. https://doi.org/10.1093/bioinformatics/btae049

Pedersen, B. S., & Quinlan, A. R. (2017). cyvcf2: Fast, flexible variant analysis with Python. *Bioinformatics*, *33*(12), 1867–1869. https://doi.org/10.1093/bioinformatics/btx057

Pedersen, B. S., & Quinlan, A. R. (2018). Hts-nim: Scripting high-performance genomic analyses. *Bioinformatics*, *34*(19), 3387–3389. https://doi.org/10.1093/bioinformatics/bty358

Pysam Developers. (2025). *Pysam: A Python module for reading, manipulating and writing genomic data sets*. GitHub repository. https://github.com/pysam-developers/pysam

Ramirez-Gonzalez, R. H., Bonnal, R., Caccamo, M., & Maclean, D. (2012). Bio-samtools: Ruby bindings for SAMtools, a library for accessing BAM files containing high-throughput sequence alignments. *Source Code for Biology and Medicine*, *7*(1), 6. https://doi.org/10.1186/1751-0473-7-6

Ruby Association. (2020). *Ruby association grant*. Program page. https://www.ruby.or.jp/en/about/activities/grants_and_fundings/