# Genome Sequencing with ROOT

Aditya Pandey
Birla Institute of Technology, Mesra

Mentors:-
Vassil Vassilev, Martin Vasilev

## Abstract:-

Genomic sequencing workflows produce two essential outputs: alignment data (BAM files containing billions of sequencing reads mapped to a reference genome) and variant data (VCF files containing millions of detected mutations and polymorphisms). These formats serve different purposes but are fundamentally linked, variants are called from alignments, and alignments provide the evidence supporting each variant call.
The sequence alignment output is usually saved in BAM format. Since first published in 2009, BAM has quickly become the most popular file format to store short-read alignments and is generally considered as the starting point for most NGS analysis tasks. Short-variant genotypes are typically stored in Variant Call Format (VCF) files. These two formats remain separate, requiring researchers to maintain parallel file systems, use different software tools, and manually coordinate queries across both data types.
Manual review of aligned reads for confirmation and interpretation of variant calls is an important step in many variant calling pipelines for next-generation sequencing data. Visual inspection can greatly increase the confidence in calls, reduce the risk of false positives, and help characterize complex events. This clinical need examining variants alongside their supporting read evidence highlights the fundamental connection between alignment and variant data that current formats fail to capture.

## Problem Statement:-

### Fragmented Storage and Tooling

While VCFs offer a condensed summary of variants, BAM files hold a wealth of contextual information crucial for distinguishing true variants from artifacts. Understanding variant calls in the context of corresponding BAM alignments is an important process. Currently, clinicians must load both files into visualization tools like IGV, manually navigate to each variant position, and inspect the underlying reads, repeating this process for every candidate variant.

### Compression Limitations

CRAM is becoming increasingly popular for data storage, as it has all the advantages of BAM but is more compact in size (50–80% file size reduction). However, some CRAM files use lossy compression and thus cannot be completely faithfully restored back to the original BAM source data.Quality scores dominate file size, consuming 60-70% of compressed BAM/CRAM data, yet current formats do not exploit the full statistical structure of this data.

VCF Scaling Crisis

VCF file sizes grow superlinearly due to increasing numbers of variants discovered with increasing sample size. Significant increases in the size of VCF files causes workflows to be slower and makes sharing and analysing files more time-consuming and resource-expensive. Export of data from gnomAD to VCF is in the petabyte range. Doubling the size of gnomAD would make export and interchange in VCF format completely infeasible.

The usual row-wise encoding of the VCF data model emphasises efficient retrieval of all data for a given variant, but accessing data on a field or sample basis is inefficient. Row-wise data storage is fundamentally unsuitable and a more scalable approach is needed.

## Proposed Solution:-

RAMTools plans to create an unified genomic data format that accepts BAM files as input, stores variant information from VCF files, and achieves high compression through columnar storage and specialized codecs. The format is built on ROOT's RNTuple, a columnar I/O system developed at CERN for high-energy physics data that handles petabyte-scale datasets.

BAM Input Support

The system reads BAM files via htslib, preserving all mandatory SAM fields: read name (QNAME), bitwise flags (FLAG), reference name (RNAME), position (POS), mapping quality (MAPQ), CIGAR string, mate information (RNEXT, PNEXT, TLEN), sequence (SEQ), and quality scores (QUAL). All auxiliary tags (MD, NM, RG, barcodes) are also preserved.

VCF Variant Information

The system imports variant data from VCF files, storing all eight mandatory columns (CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO) plus FORMAT fields and per-sample genotypes. GT (genotype) encodes alleles as numbers: 0 for the reference allele, 1 for the first allele listed in ALT, 2 for the second allele. The separator indicates whether the alleles are phased ('|') or unphased ('/') with respect to other data lines. INFO fields are parsed into typed columns rather than stored as key-value text, enabling efficient queries on specific annotations.

High Compression Through Columnar Storage

Unlike row-wise BAM and VCF formats, RNTuple stores each field in a separate column. This enables type-specific compression: quality scores are compressed using context-aware entropy coding that exploits correlation with read position and base identity; sequences use reference-based compression storing only differences from the reference genome; positions use delta encoding exploiting sorted order; genotypes use 2-bit packing for biallelic variants with run-length encoding for homozygous stretches.

Integrated Queries

Because alignments and variants share a common positional index, querying a genomic region returns both data types together. Querying a specific variant returns the variant annotation along with all reads overlapping that position. This directly supports clinical validation workflows where manual review of aligned reads is often performed to reduce the risk of false positives and incorrect results.

## Deliverables:-

1. C++ library supporting BAM input and unified RNTuple output with variant information.
2. Integrated query API returning alignments and variants for a genomic region
3. Compression benchmarks against CRAM 3.1 and bgzipped VCF on 1000 Genomes and GIAB datasets
4. Documentation covering format specification, API reference, and usage examples

## Project Timeline (Jan 20 - Apr 20):-

| **Week** | **Tasks** |
|---|---|
| 1 | htslib integration for BAM input, extend existing SAM parser |
| 2 | Complete field mapping (all 11 SAM fields + aux tags) to RNTuple |
| 3 | BAM → RNTuple pipeline, round-trip validation with BAM export |
| 4 | Design variant schema in RNTuple (CHROM, POS, REF, ALT, evidence fields) |
| 5 | MD tag parsing for mismatch detection, CIGAR-based indel extraction |
| 6 | Variant evidence aggregation (read depth, allele counts, quality scores) |
| 7 | Buffer week (catch-up, bug fixes, unexpected issues) |
| 8 | Positional indexing linking variants to supporting reads |
| 9 | Multi-sample variant merging, genotype encoding |
| 10 | Unified query API: region → alignments + extracted variants |
| 11 | Compression benchmarks vs CRAM 3.1 on 1000 Genomes & GIAB |
| 12 | Performance optimization, variant extraction accuracy validation |
| 13 | Format specification, API reference, usage examples, final submission |