# C++ library and tools for next generation sequence data

*M. Trost, H.M. Kang, P. Anderson, B. Li, W. Chen, C. Fuchsberger, X. Zhan , A. Tan, G.R. Abecasis*
Dept of Biostatistics and Center for Statistical Genetics, Univ Michigan, Ann Arbor, MI, 48109, USA;

## INTRODUCTION

In order to handle the increasing volume of next generation sequencing and genotyping data created and developed:

- **C++ Library** – open source, freely available (GPL license), easy to use APIs
  - File/Stream I/O – uncompressed, BGZF, GZIP, stdin, stdout
  - Common file formats – SAM/BAM, FASTQ, GLF
    - Indexed access to BAM files
    - Accessors to get/set values
  - Utility classes, including:
    - Cigar – interpretation and mapping between query and reference
    - Pileup – structured access to data by individual reference position
- **Standalone Tools** – efficiently process and analyze data using this library

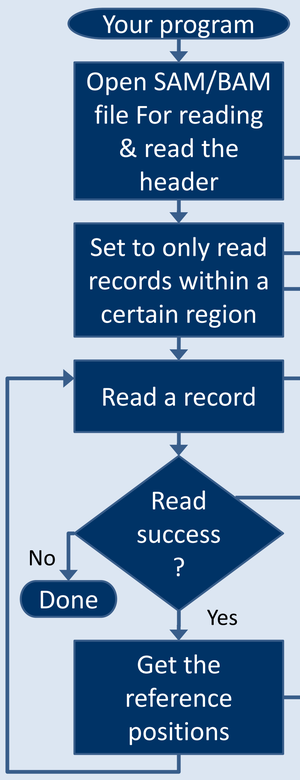
## URL FOR LIBRARY & TOOLS

**http://genome.sph.umich.edu/wiki/Software**

- Download of tools and library
- Description and usage information
- Class descriptions, examples, & FAQs

## SAMPLE C++ CODE USING LIBRARY

- Print reference positions for BAM file reads that overlap the specified positions.




```
void printRefPositions(std::string inFile, std::string indexFile,
                       std::string rname, int startPosition,
                       int endPosition)
{
    SamFileHeader header;
    // Open the bam file for reading and read the header.
    SamFile samIn(inFile.c_str(), SamFile::READ, &header);

    // Open the bam index file for reading.
    samIn.ReadBamIndex(indexFile.c_str());

    // Set the section to be read.
    samIn.SetReadSection(rname.c_str(), startPosition, endPosition);

    SamRecord record;
    // Keep reading BAM records until they aren't anymore.
    while(samIn.ReadRecord(header, record))
    {
        // Print the reference positions associated with this read.
        std::cout << "Read " << samIn.GetCurrentRecordCount() << ":";
        Cigar* cigar = record.getCigarInfo();
        for(int i = 0; i < record.getReadLength(); i++)
        {
            int refPos =
                cigar->getRefPosition(i, record.get1BasedPosition());
            if(refPos != Cigar::INDEX_NA)
            {
                std::cout << "  " << refPos;
            }
        }
        std::cout << "\n";
    }
}
```

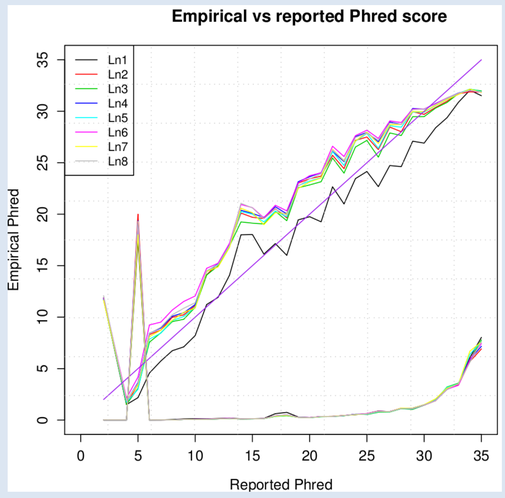
```
bam$ ./printRefPositions sortedBam.bam
sortedBam.bam.bai 1 1013 1751
Read 1:  1011  1012  1013  1014  1015
Read 2:  1751  1752  1753  1754  1755
bam$
```

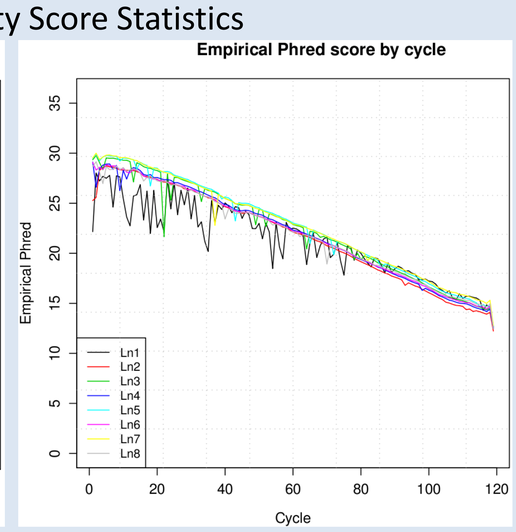
## SAM/BAM TOOLS

- **VerifyBamID** – Check sample identities for contamination/sample swap
  - Genotype concordance based detection
  - Estimate based on population allele frequencies without genotype data
- **Validate** – Check file format & print statistics
- **Convert** – Convert between SAM & BAM
- **SplitChromosome** – Split into 1 file per Chromosome
- **WriteRegion** – Write only reads in the specified region
- **Filter** – Soft clip ends with too high mismatch % and mark unmapped if quality of mismatches is too high
- **PolishBam** – Add/Update header lines & add RG tag to each record
- **QPLOT** - Calculate & plot summary statistics
- **RGMergeBam** – Merge sorted BAM files adding Read Groups
- **SplitBam** – Split into 1 file per Read Group
- **TrimBam** – Trim end of reads, changing read ends to 'N' & quality to '!'
- **Pileup** – Pileup every base or just bases in specified region and write VCF
- **Deduper** – Mark or remove duplicates
- **Recalibrator** – Resource-efficient tool, which recalibrates base qualities based on an adaptive logistic regression model

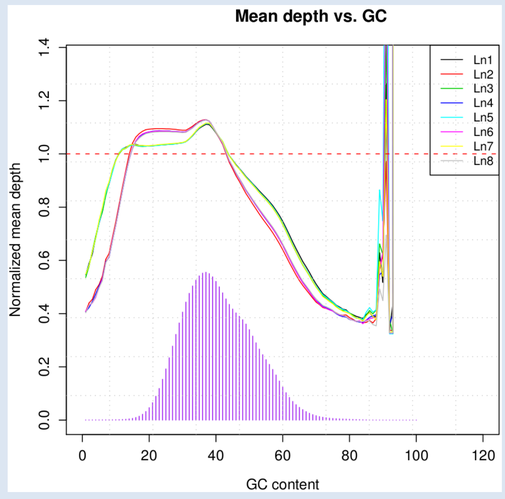
## QPLOT SAMPLE OUTPUT

Base Call Quality Score Statistics

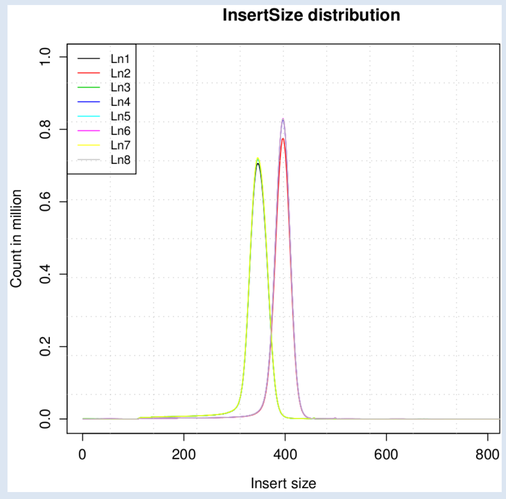



GC Bias Statistics




Insert Size Statistics




- Additional plots include: Depth Distribution, Depth Coverage, and Empirical Q20 Count

## BAM VALIDATE SAMPLE OUTPUT

```
bamValidator
bamValidator$ ./bam validate --verbose --in testInvalid.sam --printableErrors 4
Record 1
INVALID_QNAME (ERROR) : Invalid Query Name - the string length (256) does not match the specified query name length
(0).
INVALID_QNAME (WARNING) : Invalid Query Name (QNAME) length: 256.  Length with the terminating null must be between
2 & 255.

Record 2
INVALID_POS (WARNING) : POS out of range (536870912) must be between 0 and (2^29)-1.

Record 3
FAIL_PARSE: Too few columns (1) in the Record, expected at least 11.

Record 5
FAIL_PARSE: flag, 9K7, is not an integer.


Number of records read = 18
Number of valid records = 2

Error Counts:
        FAIL_PARSE: 12
        INVALID_QNAME: 2
        INVALID_POS: 3

TotalReads      6.00
MappedReads     6.00
PairedReads     2.00
ProperPair      0.00
DuplicateReads  0.00
QCFailureReads  0.00

MappingRate(%)  100.00
PairedReads(%)  33.33
ProperPair(%)   0.00
DupRate(%)      0.00
QCFailRate(%)   0.00

TotalBases      26.00
BasesInMappedReads      26.00
Returning: 5 (FAIL_PARSE)
bamValidator$
```

Report errors for badly formatted files

Basic file statistics

## ADDITIONAL TOOLS

- **FastQValidator** – Check format of FASTQ file
  - Reports errors for badly formatted files
  - Reports Base Composition Statistics (%reads at each read index)
- **VcfGenomeStat** – Print flanking sequences and how often they appear for input VCF file

*Additional support including VCF and association analysis will be provided in the near future.*

Support from National Institutes of Health (U01HG00521401 & HG005552)
References: Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R; 1000 Genome Project Data Processing Subgroup. The Sequence Alignment/Map (SAM) format and SAMtools. *Bioinformatics*, 2009;25, 2078-2079. [PMID: 19505943]