# SeqArray – A storage-efficient high-performance data format for WGS variant calls

**Dr. Xiuwen Zheng**

**Department of Biostatistics**

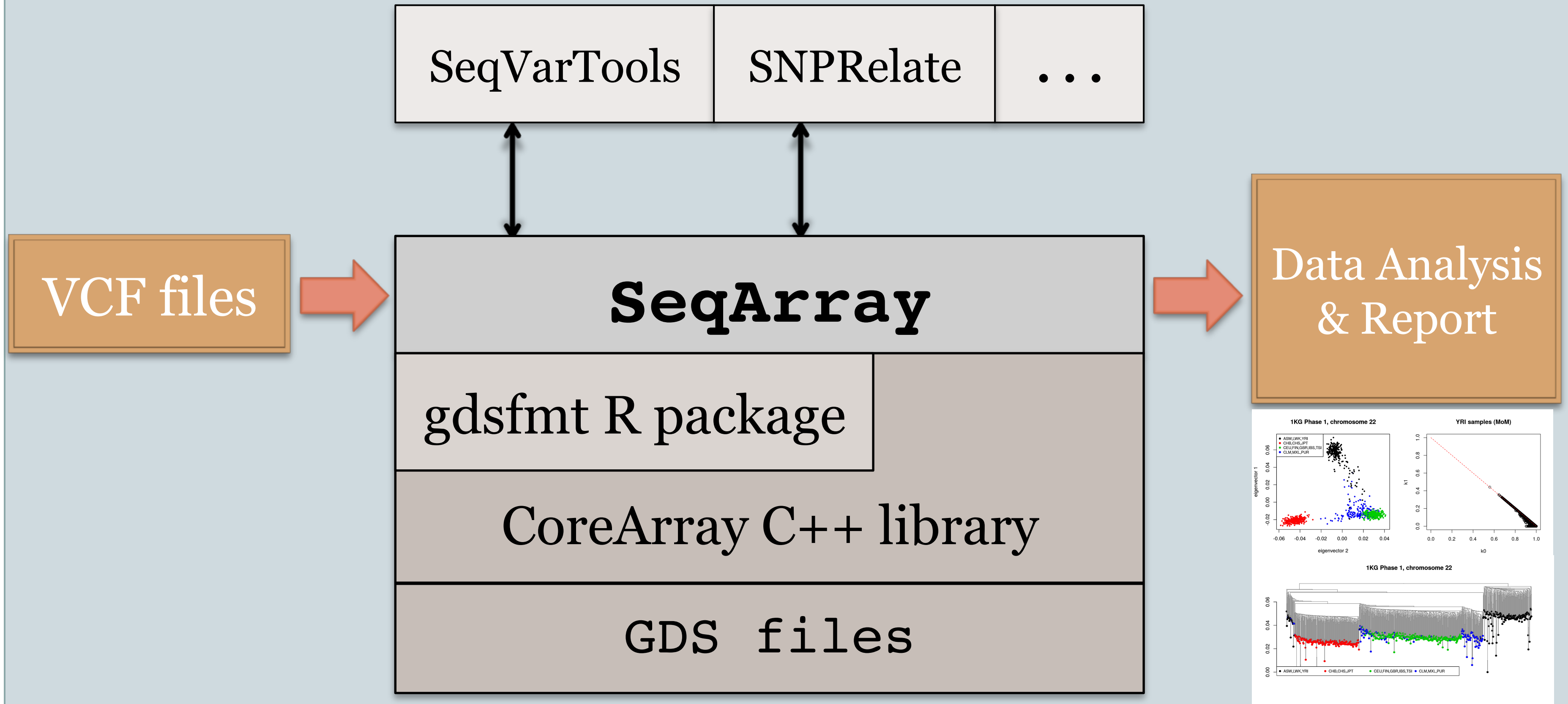
**University of Washington – Seattle**

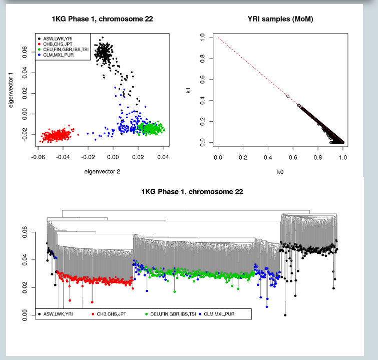
**WA, USA**

## Introduction

- Whole-genome sequencing (WGS) data is being generated at an unprecedented rate
  - 1000 Genomes Project Phase 3 (1KG)
    - 81 million variants and 2,504 individuals
    - http://www.1000genomes.org
  - Trans-Omics for Precision Medicine (TOPMed) program
    - NIH-funded
    - WGS: 140 million variants and 9,109 individuals
  - Variant Call Format (VCF)
    - a generic and flexible text-based format
    - VCF files are large and data retrieval is relatively slow

## Methods – Workflow





GDS – Genomic Data Structure
SeqArray – BioC package (http://www.bioconductor.org/packages/SeqArray)

## Methods – Advantages

- SeqArray provides the same capabilities as VCF
- Stores data in a binary and array-oriented manner
  - efficient access using the R language
- Genotypes are stored in a compressed manner
  - 2-bit array to store alleles (95% sites are bi-allelic)
  - rare variants: highly compressed
  - 1KG, 203.5 billion genotypes -> 4.3G (2.26% if a byte stores a genotype)
- Parallel access
  - multiple cluster nodes and/or cores

## Methods – Key Functions

**Table 1**: The key functions in the SeqArray package.

| Function | Description |
|---|---|
| seqVCF2GDS | Reformats VCF files |
| seqSetFilter | Defines a data subset of samples or variants |
| seqGetData | Gets data from a SeqArray file with a defined filter |
| seqApply | Applies a user-defined function over array margins |
| seqParallel | Applies functions in a computing cluster |

## Benchmark

- Dataset:
  - the 1000 Genomes Project Phase 3, chromosome 1
  - 6,468,094 variants, 2,504 individuals
  - original VCF.gz file: 1.2G
  - reformat to a SeqArray file: 458M (zlib compression)

- Calculate the frequencies of reference alleles
  - 1. R code (sequential version)
  - 2. R code (parallel version)
  - 3. R and C++ integration via the Rcpp package

## Benchmark – Test 1 (sequentially)

```
# load the R package
library(SeqArray)

# open the file
genofile <- seqOpen("1KG_chr1.gds")

# apply a user-defined function over variants
system.time(afreq <- seqApply(genofile, "genotype",
  FUN = function(x) { mean(x==0L, na.rm=TRUE) },
  as.is="double", margin="by.variant")
)
```

"x" looks like:

```
                 sample
  allele [,1] [,2] [,3] [,4] [,5]
    [1,]    0    1    0    1    1
    [2,]    0    0   NA    1    0
```

0 – reference allele
1 – the first alternative allele

the user-defined function

**10.8 minutes** on Linux with Intel® Xeon® CPU @2GHz and 128GB RAM

# Benchmark – Test 2 (in parallel)

```
# load the R package
library(parallel)

# create a cluster with 4 cores
cl <- makeCluster(4)

# run in parallel
system.time(afreq <- seqParallel(cl, genofile,
  FUN = function(file) {
        seqApply(file, "genotype", as.is="double",
            FUN = function(x) mean(x==0L, na.rm=TRUE))
  }, split = "by.variant")
)
```

the user-defined function distributed to 4 different computing nodes

divide genotypes into 4 non-overlapping parts according to different variants

**3.1 minutes** (vs. 10.8m in Test 1)

## Benchmark – Test 3 (C++ Integration)

```
library(Rcpp)
cppFunction('double CalcAlleleFreq(IntegerMatrix x) {
  int nrow = x.nrow(), ncol = x.ncol();
  int g, cnt=0, zero_cnt=0;
  for (int i = 0; i < nrow; i++) {
    for (int j = 0; j < ncol; j++) {
      if ((g = x(i, j)) != NA_INTEGER) {
        cnt ++;
        if (g == 0) zero_cnt ++;
  }}}
  return double(zero_cnt) / cnt;
}')
system.time(
  afreq <- seqApply(genofile, "genotype", CalcAlleleFreq,
      as.is="double", margin="by.variant")
)
```

dynamically define an inline C/C++ function in R

**1.5 minutes** (significantly faster! vs. 10.8m in Test 1)

## Comparison

**Table 2**: Format conversion and genotype compression with 1KG data (minutes).

| Software | Format | File Size | 1 core | 4 cores |
|---|---|---|---|---|
| bcftools | VCF.gz to VCF.gz [1] | 14.4 G | 2.6 h | -- |
|  | VCF.gz to BCF | 12.3 G | 4.5 h | -- |
| BGT | VCF.gz to BGT [2] | 3.5 G | 2.4 h | -- |
| SeqArray | VCF.gz to GDS (zlib) | 5.7 G | 2.3 h | 0.9 h |
|  | VCF.gz to GDS (lzma) | 2.6 G | 6.3 h | 2.2 h |

[1]: merging VCF files of 22 chromosomes without format conversion
[2]: BGT does not store the phasing states and INFO annotations

## Comparison

**Table 3**: Genotype decompression on all autosomes of 1KG data (minutes).

| Format (algorithm) | File Size | Speed [1] | Time |
|---|---:|---:|---:|
| VCF.gz | 14.4 G | -- | 21.9 m [2] |
| BCF | 12.3 G | -- | 13.6 m [2] |
| BGT (pbwt[3]) | 3.5 G | 455.3 | 7.45 m |
| GDS (zlib) | 5.7 G | 759.7 | 4.46 m |
| GDS (lzma) | 2.6 G | 629.1 | 5.39 m |

[1]: million genotypes per second
[2]: the lower bound of running time: gzip decompression
[3]: positional Burrows-Wheeler transform

## Discussion

- SeqArray is of great interest to
  - R users involved in data analyses of WGS variants
  - particularly those with limited experience of high-performance computing (HPC)

- Major updates in BioC release 3.3 (April 2016)
  - 4x speedup in VCF import
  - decoding genotypes is 2x faster
  - supports GRanges / GRangesList efficiently
  - . . .

## Acknowledgements

- Department of Biostatistics, University of Washington – Seattle, WA
  - Stephanie M. Gogarten
  - Adrienne Stilp
  - David Levine
  - Cathy Laurie