# SeqArray: an R/Bioconductor Package for Big Data Management of Genome-Wide Sequence Variants

**Dr. Xiuwen Zheng**

**Department of Biostatistics**

**University of Washington – Seattle**

# Introduction

- Thousands of gigabyte genetic data sets provide significant challenges in data management, even on well-equipped hardware
  - The 1000 Genomes Project Phase 1 (1KG): ~39 million variants (differences from the reference genome) of 1092 individuals
  - http://www.1000genomes.org
  - Variant Call Format (VCF) files: genotypes + annotation, totaling ~184G in a compressed manner

# Methods

- CoreArray (C++ library)
  - designed for large-scale data management of genome-wide variants
  - data format (GDS) to store multiple array-oriented datasets in a single file

- Two R packages
  - gdsfmt – R interface to CoreArray Genomic Data Structure (GDS) files
  - SeqArray – specifically designed for data management of genome-wide sequence variants from Variant Call Format (VCF) files

# Methods – Advantages

- 1. Direct access of data without parsing VCF text files
- 2. Stored in a binary and array-oriented manner
  - 2 bits are employed as a primitive type to store alleles (e.g., A, G, C, T)
  - efficient access of variants using R language
- 3. Genotypic data stored in a compressed manner
  - rare variants -> highly compressed without sacrificing access efficiency
  - e.g., 1KG, 26G genotypes -> 1.5G by the zlib algorithm (5.8%!)
- 4. Run in parallel!

# Methods – Key Functions

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

| Function | Description |
|---|---|
| seqVCF2GDS | Reformats VCF files to GDS format |
| seqSummary | Gets the summary (# of samples, # of variants, INFO/ FORMAT variables, etc) |
| seqSetFilter | Sets a filter to sample or variant (define a subset of data) |
| seqGetData | Gets data from a sequencing file (from a subset of data) |
| seqApply | Applies a user-defined function over array margins |
| seqParallel | Applies functions in parallel |

# Benchmark

- Dataset:
  - the 1000 Genomes Project Phase 1, chromosome 1
  - 3,007,196 variants, 1092 individuals
  - the original VCF file: 10.7G (compressed!)
    - genotypes + annotations
  - reformat to a single SeqArray file: 10.6G
- Calculate the frequencies of reference alleles
  - 1. R code (sequential version)
  - 2. R code (parallel version)
  - 3. Seamless R and C++ integration via the Rcpp package (sequential version)

# Benchmark – Test 1 (sequentially)

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

# open the file
genofile <- seqOpen("1KG.chr1.gds")

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

the typical "x" looks like:

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

0 – reference allele
1 – the first alternative allele

**the user-defined function**

**~6.8 minutes** on a Linux system with two quad-core Intel processors (2.27GHz) and 32 GB RAM

# Benchmark – Test 2 (in parallel)

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

# create a computing cluster with 4 cores
seqParallelSetup(4)

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

**the user-defined kernel function distributed to 4 different computing nodes**

**divide genotypes into 4 non-overlapping parts**

**~2.1 minutes** (vs 6.8m in Test 1)

# Benchmark – Test 3 (C++ Integration)

```
library(Rcpp)
cppFunction('double RefAlleleFreq(IntegerMatrix x) {
    int nrow = x.nrow(), ncol = x.ncol();
    int cnt=0, zero_cnt=0, g;
    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", RefAlleleFreq,
        as.is="double", margin="by.variant")
)
```

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

**~0.7 minutes** (significantly faster!!! vs 6.8m in Test 1)

# Conclusion

- SeqArray will be of great interest to
  - R users involved in data analyses of large-scale sequencing variants
  - particularly those with limited experience of parallel / high-performance computing
- SeqVarTools (Bioconductor)
  - variant analysis, such like allele frequency, HWE, Mendelian errors, etc
  - functions to display genotypes / annotations in a readable format

# Acknowledgements

- Department of Biostatistics at University of Washington – Seattle,
  - Genetic Analysis Center:
    - Dr. Stephanie M. Gogarten
    - Dr. Cathy Laurie