A Tutorial for the R Package SNPRelate

Xiuwen Zheng
GENEVA Coordinating Center
Department of Biostatistics
University of Washington

Jul 18, 2012

Contents

1 Overview

Genome-wide association studies (GWAS) are widely used to help determine the genetic basis of diseases and traits, but they pose many computational challenges. We developed gdsfmt and SNPRelate (high-performance computing R packages for multi-core symmetric

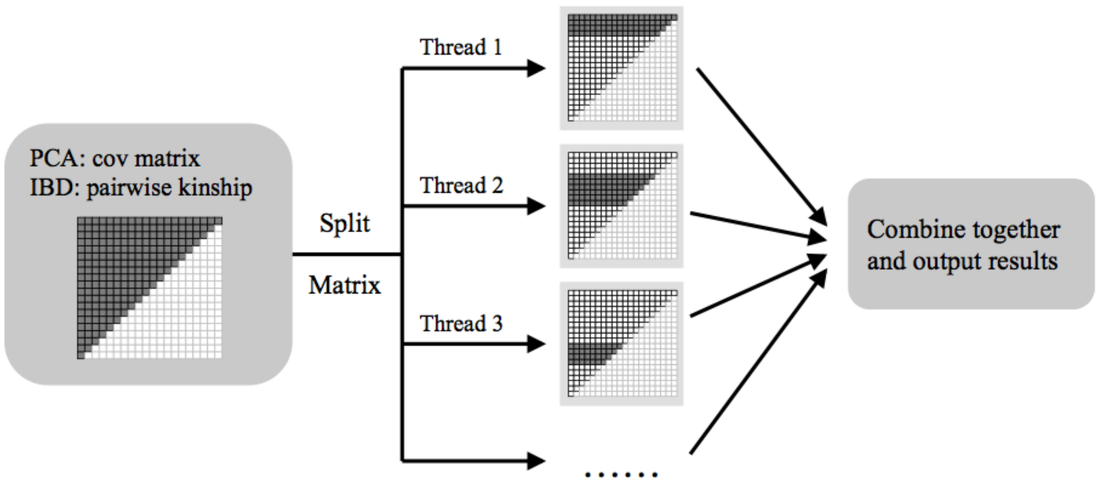


Figure 1: Flowchart of parallel computing for principal component analysis and identity-by-descent analysis.

multiprocessing computer architectures) to accelerate two key computations in GWAS: principal component analysis (PCA) and relatedness analysis using identity-by-descent (IBD) measures. The kernels of our algorithms are written in C/C++ and have been highly optimized. The calculations of the genetic covariance matrix in PCA and pairwise IBD coefficients are split into non-overlapping parts and assigned to multiple cores for performance acceleration, as shown in Figure 1. Benchmarks show the uniprocessor implementations of PCA and IBD are ∼10 to 45 times faster than the implementations provided in the popular EIGENSTRAT (v3.0) and PLINK (v1.07) programs respectively, and can be sped up to 70∼250 folds by utilizing multiple cores.

R is the most popular statistical programming environment, but one not typically optimized for high performance or parallel computing which would ease the burden of large-scale GWAS calculations. To overcome these limitations we have developed a project named CoreArray (`http://corearray.sourceforge.net/`) that includes two R packages: gdsfmt to provide efficient, platform independent memory and file management for genome-wide numerical data, and SNPRelate to solve large-scale, numerically intensive GWAS calculations (i.e., PCA and IBD) on multi-core symmetric multiprocessing (SMP) computer architectures.

This vignette takes the user through the relatedness and principal component analysis used for genome wide association data. The methods in these vignettes have been introduced in the paper of Zheng *et al.* (2012).[1] For replication purposes the data used here are taken from the HapMap Phase II project. These data were kindly provided by the Center for Inherited Disease Research (CIDR) at Johns Hopkins University and the Broad Institute of MIT and Harvard University (Broad). The data supplied here should not be used for any purpose other than this tutorial.

[1]Zheng, Xiuwen., *et al.* A High-performance Computing Toolset for Relatedness and Principal Component Analysis in GWAS. *Submitted.*

2 Preparing Data

2.1 Data formats used in SNPRelate

To support efficient memory management for genome-wide numerical data, the gdsfmt package provides the genomic data structure (GDS) file format for array-oriented bioinformatic data, which is a container for storing annotation data and SNP genotypes. In this format each byte encodes up to four SNP genotypes thereby reducing file size and access time. The GDS format supports data blocking so that only the subset of data that is being processed needs to reside in memory. GDS formatted data is also designed for efficient random access to large data sets.

```
> # load the R packages: gdsfmt and SNPRelate
> library(gdsfmt)
> library(SNPRelate)
```

Here is a typical GDS file:

```
> snpgdsSummary(snpgdsExampleFileName())

The total number of samples: 279
The total number of SNPs: 9088
SNP genotypes are stored in individual-major mode.
```

snpgdsExampleFileName() returns the file name of a GDS file used as an example in SNPRelate, and it is a subset of data from the HapMap project and the samples were genotyped by the Center for Inherited Disease Research (CIDR) at Johns Hopkins University and the Broad Institute of MIT and Harvard University (Broad). **snpgdsSummary()** summarizes the genotypes stored in the GDS file. “Individual-major mode” indicates listing all SNPs for an individual before listing the SNPs for the next individual, etc. Conversely, “SNP-major mode” indicates listing all individuals for the first SNP before listing all individuals for the second SNP, etc. Sometimes “SNP-major mode” is more computationally efficient than “individual-major model”. For example, the calculation of genetic covariance matrix deals with genotypic data SNP by SNP, and then “SNP-major mode” should be more efficient.

```
> # open a GDS file
> (genofile <- openfn.gds(snpgdsExampleFileName()))

file name: /Library/Frameworks/R.framework/Versions/2.15/Resources/library/SNPRelate/ext

+         [   ]
|--+ sample.id        [ FStr8 279 ZIP(23.10%) ]
|--+ snp.id       [ Int32 9088 ZIP(34.76%) ]
|--+ snp.rs.id        [ FStr8 9088 ZIP(42.66%) ]
```

```
|--+ snp.position        [ Float64 9088 ZIP(51.77%) ]
|--+ snp.chromosome        [ Int32 9088 ZIP(0.33%) ]
|--+ snp.allele        [ FStr8 9088 ZIP(14.45%) ]
|--+ genotype        [ Bit2 9088x279 ]
|--+ sample.annot        [   ] *
|  |--+ sample.id        [ FStr8 279 ZIP(23.10%) ]
|  |--+ family.id        [ FStr8 279 ZIP(28.37%) ]
|  |--+ geneva.id        [ Int32 279 ZIP(80.29%) ]
|  |--+ father.id        [ FStr8 279 ZIP(12.98%) ]
|  |--+ mother.id        [ FStr8 279 ZIP(12.86%) ]
|  |--+ plate.id        [ FStr8 279 ZIP(1.29%) ]
|  |--+ sex        [ FStr8 279 ZIP(28.32%) ]
|  |--+ pop.group        [ FStr8 279 ZIP(7.89%) ]
```

The output lists all variables stored in the GDS file. At the first level, it saves variables **sample.id**, **snp.id**, etc. The second-level variables **sex** and **pop.group** are both stored in the directory of **sample.annot**. All of the functions in SNPRelate require a minimum set of variables in the SNP annotation data. The minimum required variables are

- **snp.id**, a unique identifier for each SNP;
- **snp.chromosome**, an integer mapping for each chromosome, with values 1-26, mapped in order from 1-22, 23=X,24=XY (the pseudoautosomal region), 25=Y, 26=M (the mitochondrial probes), and 0 for probes with unknown positions; it does not allow NA.
- **snp.position**, the base position of each SNP on the chromosome, and 0 for unknown position; it does not allow NA.

```
> # Take out snp.id
> head(read.gdsn(index.gdsn(genofile, "snp.id")))

[1] 1 2 3 4 5 6

> # Take out snp.rs.id
> head(read.gdsn(index.gdsn(genofile, "snp.rs.id")))

[1] "rs1695824"  "rs13328662" "rs4654497"  "rs10915489" "rs12132314"
[6] "rs12042555"
```

There are two additional variables:

- **snp.rs.id**, a character string identifier for the SNP that may not be unique.
- **snp.allele**, it is not necessary for the analysis, but it is necessary when merging genotypes from different platforms. The format of **snp.allele** is "A allele/B allele", like "T/G" where T is A allele and G is B allele.

There are four possible values stored in the variable **genotype**: 0, 1, 2 and 3. “0” indicates two B alleles, “1” indicates one A allele and one B allele, “2” indicates two A alleles, and “3” is a missing genotype. “Bit2” indicates that each byte encodes up to four SNP genotypes since one byte consists of eight bits.

```
> # Take out genotype data for the first 3 samples and the first 5 SNPs
> (g <- read.gdsn(index.gdsn(genofile, "genotype"), start=c(1,1), count=c(5,3)))

     [,1] [,2] [,3]
[1,]    2    1    2
[2,]    1    1    1
[3,]    0    0    1
[4,]    1    1    2
[5,]    2    2    2

> # read population information
> pop <- read.gdsn(index.gdsn(genofile, c("sample.annot", "pop.group")))
> table(pop)

pop
CEU HCB JPT YRI
 92  47  47  93

> # close the GDS file
> closefn.gds(genofile)
```

2.2 Create a GDS File of Your Own

The function **snpgdsCreateGeno** helps to create a GDS file of you own. There are possible values stored in the input genotype matrix: 0, 1, 2 and other values. “0” indicates two B alleles, “1” indicates one A allele and one B allele, “2” indicates two A alleles, and other values indicate a missing genotype. For example,

```
> # load data
> data(hapmap.geno)
> # create a gds file
> with(hapmap.geno, snpgdsCreateGeno("test.gds", genmat=genotype,
+     sample.id=sample.id, snp.id=snp.id, snp.chromosome=snp.chromosome,
+     snp.position=snp.position, snp.allele=snp.allele, snpfirstorder=TRUE))
> # open the gds file
> (genofile <- openfn.gds("test.gds"))

file name: test.gds
```

```
+         [   ]
|--+ sample.id        [ FStr8 279 ZIP(23.10%) ]
|--+ snp.id        [ FStr8 1000 ZIP(45.02%) ]
|--+ snp.position        [ Float64 1000 ZIP(55.97%) ]
|--+ snp.chromosome        [ Int32 1000 ZIP(2.00%) ]
|--+ snp.allele        [ FStr8 1000 ZIP(17.37%) ]
|--+ genotype        [ Bit2 1000x279 ] *

> # close the genotype file
> closefn.gds(genofile)
```

2.3 Format conversion from PLINK binary files

The SNPRelate package provides a function **snpgdsBED2GDS** for converting a PLINK binary file to a GDS file:

```
> # the PLINK BED file
> bed.fn <- system.file("extdata", "plinkhapmap.bed", package="SNPRelate")
> bim.fn <- system.file("extdata", "plinkhapmap.bim", package="SNPRelate")
> fam.fn <- system.file("extdata", "plinkhapmap.fam", package="SNPRelate")
> # convert
> snpgdsBED2GDS(bed.fn, fam.fn, bim.fn, "test.gds")

Start snpgdsBED2GDS ...
        open /Library/Frameworks/R.framework/Versions/2.15/Resources/library/SNPRelate/e
        open /Library/Frameworks/R.framework/Versions/2.15/Resources/library/SNPRelate/e
        open /Library/Frameworks/R.framework/Versions/2.15/Resources/library/SNPRelate/e
Fri Jul 20 04:48:08 2012         store sample id, snp id, position, and chromosome.
        start writing ...
         Fri Jul 20 04:48:08 2012        0%
         Fri Jul 20 04:48:08 2012        100%
Fri Jul 20 04:48:08 2012         Done.

> # summary
> snpgdsSummary("test.gds")

The total number of samples: 279
The total number of SNPs: 5000
SNP genotypes are stored in individual-major mode.
```

2.4 Format conversion from Sequence VCF files

The SNPRelate package provides a function **snpgdsVCF2GDS** for converting a VCF file to a GDS file:

```
> # the VCF file
> vcf.fn <- system.file("extdata", "sequence.vcf", package="SNPRelate")
> # convert
> snpgdsVCF2GDS(vcf.fn, "test.gds")

Start snpgdsVCF2GDS ...
        Open /Library/Frameworks/R.framework/Versions/2.15/Resources/library/SNPRelate/e
        Scanning ...
Fri Jul 20 04:48:09 2012        store sample id, snp id, position, and chromosome.
        start writing ...
Fri Jul 20 04:48:09 2012        Done.

> # summary
> snpgdsSummary("test.gds")

The total number of samples: 3
The total number of SNPs: 2
SNP genotypes are stored in SNP-major mode.
```

3 Data Analysis

We developed gdsfmt and SNPRelate (high-performance computing R packages for multi-core symmetric multiprocessing computer architectures) to accelerate two key computations in GWAS: principal component analysis (PCA) and relatedness analysis using identity-by-descent (IBD) measures.

```
> # open the GDS file
> genofile <- openfn.gds(snpgdsExampleFileName())
```

3.1 LD-based SNP pruning

It is important to use a pruned set of SNPs which are in approximate linkage equilibrium with each other to avoid the strong influence of SNP clusters in principal component analysis and relatedness analysis.

```
> set.seed(1000)
> # try different LD thresholds for sensitivity analysis
> snpset <- snpgdsLDpruning(genofile, ld.threshold=0.2)

SNP pruning based on LD:
        Sliding window: 500000 basepairs, Inf SNPs
        |LD| threshold: 0.2
Removing 365 non-autosomal SNPs
Removing 1 SNPs (monomorphic, < MAF, or > missing rate)
Working space: 279 samples, 8722 SNPs
Chromosome 1: 75.42%, 540/716
Chromosome 2: 72.24%, 536/742
Chromosome 3: 74.71%, 455/609
Chromosome 4: 73.31%, 412/562
Chromosome 5: 77.03%, 436/566
Chromosome 6: 75.58%, 427/565
Chromosome 7: 75.42%, 356/472
Chromosome 8: 71.31%, 348/488
Chromosome 9: 77.88%, 324/416
Chromosome 10: 74.33%, 359/483
Chromosome 11: 77.40%, 346/447
Chromosome 12: 76.81%, 328/427
Chromosome 13: 75.58%, 260/344
Chromosome 14: 76.95%, 217/282
Chromosome 15: 76.34%, 200/262
Chromosome 16: 72.66%, 202/278
Chromosome 17: 74.40%, 154/207
Chromosome 18: 73.68%, 196/266
Chromosome 19: 85.00%, 102/120
Chromosome 20: 71.62%, 164/229
Chromosome 21: 76.98%, 97/126
Chromosome 22: 75.86%, 88/116
6547 SNPs are selected in total.
```

```
> names(snpset)
```

```
 [1] "chr1"  "chr2"  "chr3"  "chr4"  "chr5"  "chr6"  "chr7"  "chr8"  "chr9"
[10] "chr10" "chr11" "chr12" "chr13" "chr14" "chr15" "chr16" "chr17" "chr18"
[19] "chr19" "chr20" "chr21" "chr22"
```

```
> head(snpset$chr1)  # snp.id
```

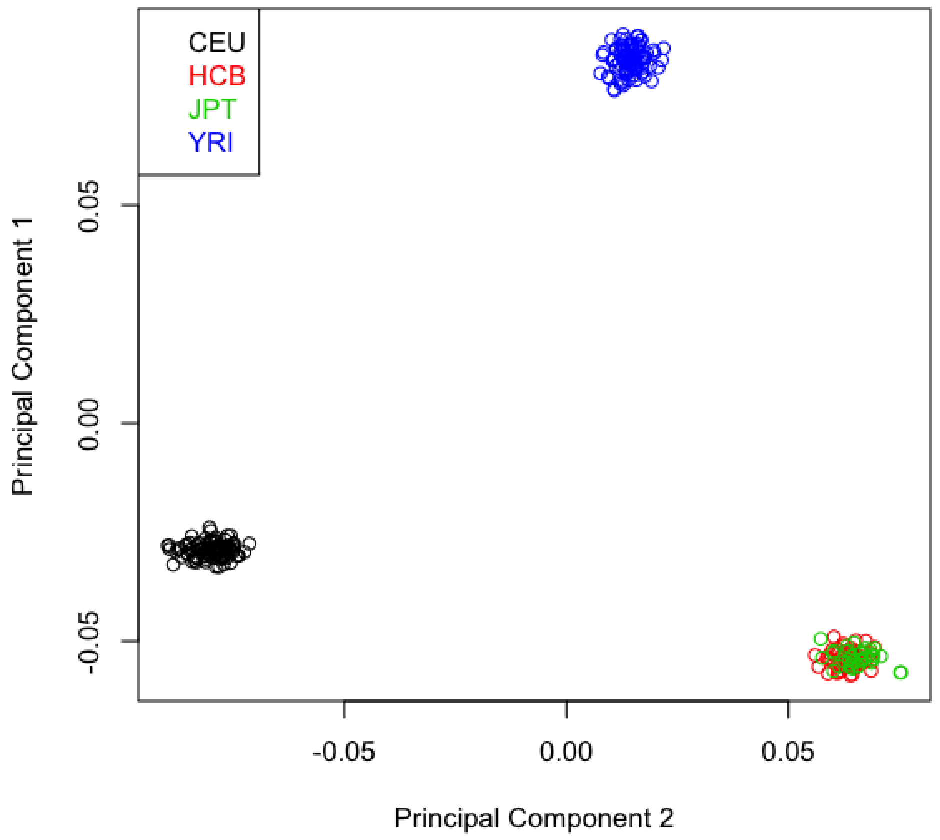
```
[1]  1  2  4  5  7 10
```

```
> # get all selected snp id
> snpset.id <- unlist(snpset)
```

3.2 Principal Component Analysis

The functions in SNPRelate for PCA include calculating the genetic covariance matrix from genotypes, computing the correlation coefficients between sample loadings and genotypes for each SNP, calculating SNP eigenvectors (loadings), and estimating the sample loadings of a new dataset from specified SNP eigenvectors.

```
> pca <- snpgdsPCA(genofile, maf=0.05, missing.rate=0.05,
+         snp.id=snpset.id, num.thread=2)

Principal Component Analysis (PCA) on SNP genotypes:
Removing 1112 SNPs (monomorphic, < MAF, or > missing rate)
Working space: 279 samples, 5435 SNPs
        Using 2 CPU cores.
PCA:        the sum of all working genotypes = 1520750
PCA:        Fri Jul 20 04:48:10 2012        0%
PCA:        Fri Jul 20 04:48:10 2012        100%
PCA:        Fri Jul 20 04:48:10 2012        Begin (eigenvalues and eigenvectors)
PCA:        Fri Jul 20 04:48:10 2012        End (eigenvalues and eigenvectors)

> plot(pca$eigenvect[,2], pca$eigenvect[,1], xlab="Principal Component 2",
+         ylab="Principal Component 1", type="n")
> # uses different colors with respect to ethnicities
> race <- as.factor(read.gdsn(index.gdsn(genofile,
+         c("sample.annot", "pop.group"))))
> points(pca$eigenvect[,2], pca$eigenvect[,1], col=race)
> legend("topleft", legend=levels(race), text.col=1:nlevels(race))
```

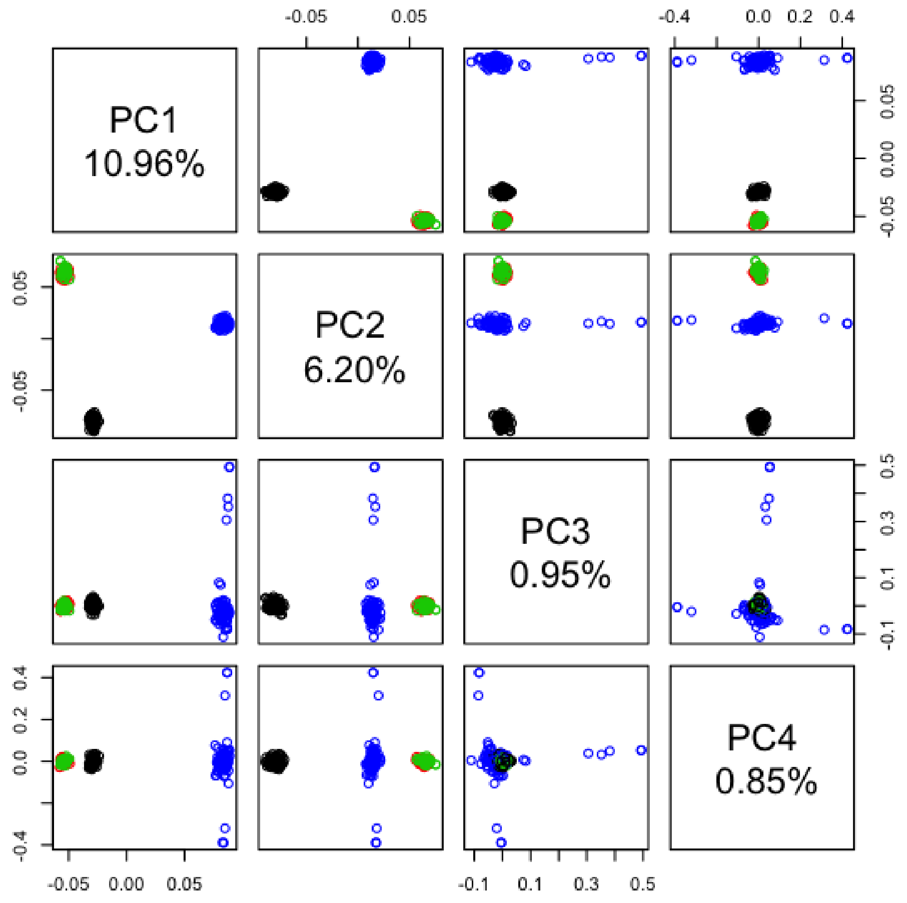


The code below shows how to calculate the percent of variation is accounted for by the principal component for the first 16 PCs. It is clear to see the first two eigenvectors hold the largest percentage of variance among the population, although the total variance accounted for is still less the one-quarter of the total.

```
> pc.percent <- 100 * pca$eigenval[1:16]/sum(pca$eigenval)
> pc.percent

 [1] 10.9608123  6.1973872  0.9480086  0.8526434  0.7927158  0.7804631
 [7]  0.7496657  0.7139979  0.6759434  0.6680874  0.6605878  0.6591129
[13]  0.6501947  0.6441370  0.6372978  0.6281596
```

Plot the principal component pairs for the first four PCs:

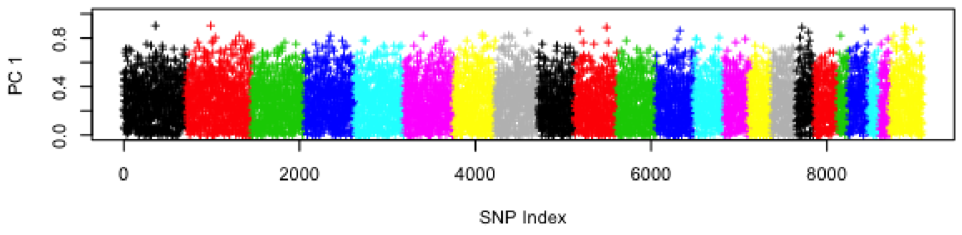
```
> lbls <- paste("PC", 1:4, "\n", format(pc.percent[1:4], digits=2), "%", sep="")
> pairs(pca$eigenvect[,1:4], col=race, labels=lbls)
```

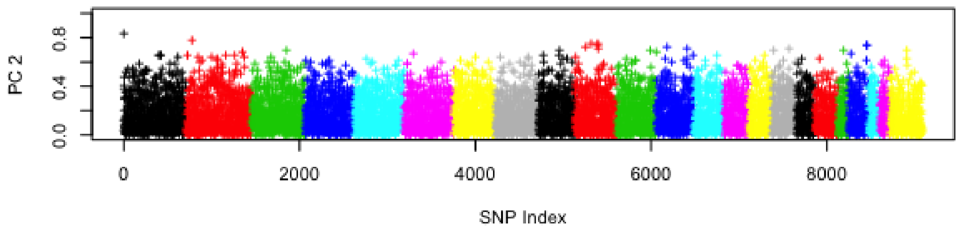


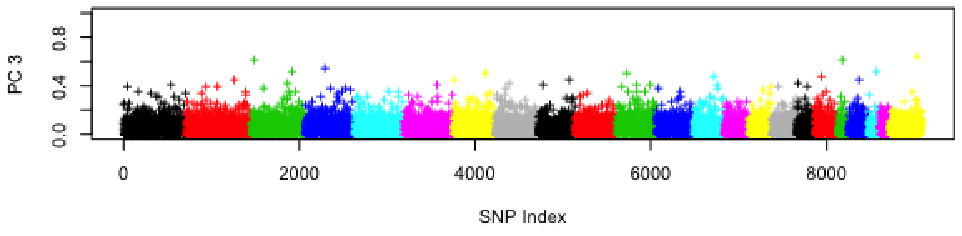
To calculate the SNP correlations between eigenvactors and SNP genotypes:

```
> # get chromosome index
> chr <- read.gdsn(index.gdsn(genofile, "snp.chromosome"))
> CORR <- snpgdsPCACorr(pca, genofile, eig.which=1:4)

SNP correlations:
Working space: 279 samples, 9088 SNPs
        Using 1 CPU core.
        Using the top 32 eigenvectors.
SNP Correlations:        the sum of all working genotypes = 2553065
SNP Correlations:        Fri Jul 20 04:48:11 2012        0%
SNP Correlations:        Fri Jul 20 04:48:11 2012        100%

> par( mfrow=c(3,1))
> for (i in 1:3)
+ {
```

```
+           plot(abs(CORR$snpcorr[i,]), ylim=c(0,1), xlab="SNP Index",
+                   ylab=paste("PC", i), col=chr, pch="+")
+ }
```







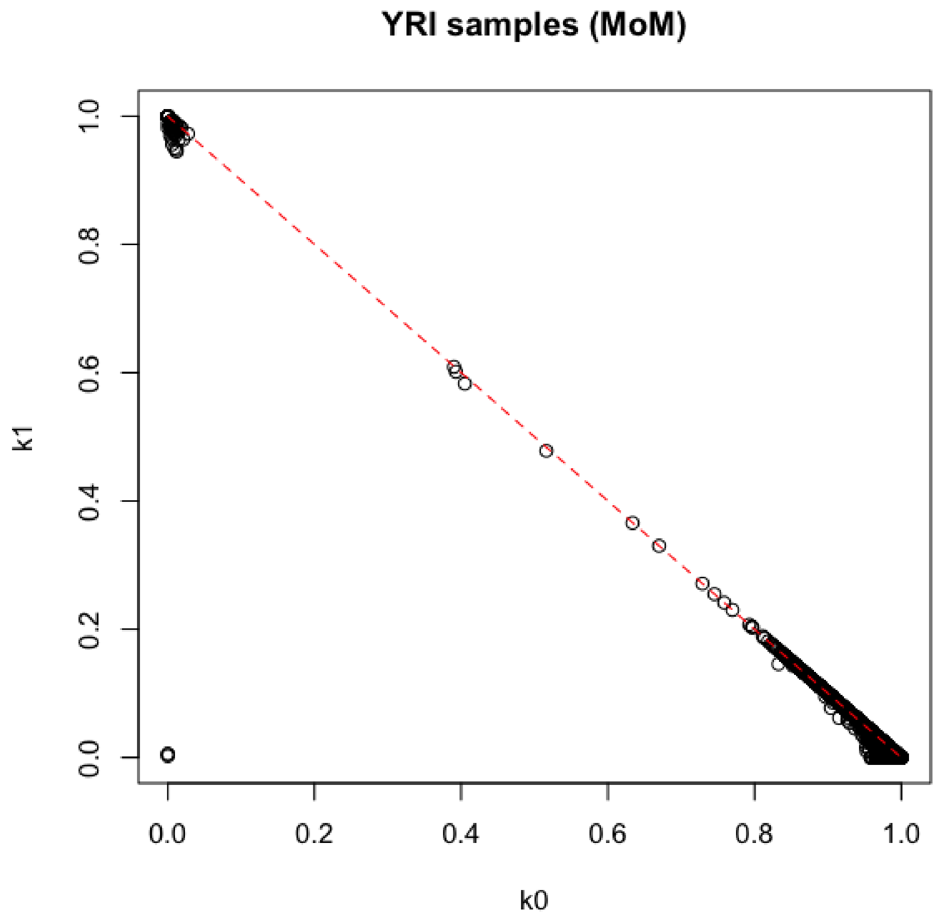
3.3 Relatedness Analysis

For relatedness analysis, identity-by-descent (IBD) estimation in SNPRelate can be done by either the method of moments (MoM) (Purcell et al., 2007) or maximum likelihood estimation (MLE) (Milligan, 2003; Choi et al., 2009). Although MLE estimates are more reliable than MoM, MLE is significantly more computationally intensive. For both of these methods it is preffered to use a LD pruned SNP set.

```
> # YRI samples
> sample.id <- read.gdsn(index.gdsn(genofile, "sample.id"))
> YRI.id <- sample.id[read.gdsn(index.gdsn(genofile,
+           c("sample.annot", "pop.group"))) == "YRI"]
```

3.3.1 PLINK method of moments (MoM)

```
> # estimate IBD coefficients
> ibd <- snpgdsIBDMoM(genofile, sample.id=YRI.id, snp.id=snpset.id,
+          maf=0.05, missing.rate=0.05)

Identity-By-Descent analysis (PLINK method of moment) on SNP genotypes:
Removing 1285 SNPs (monomorphic, < MAF, or > missing rate)
Working space: 93 samples, 5262 SNPs
        Using 1 CPU core.
PLINK IBD:       the sum of all working genotypes = 484520
PLINK IBD:       Fri Jul 20 04:49:20 2012        0%
PLINK IBD:       Fri Jul 20 04:49:20 2012        100%

> ibd.coeff <- snpgdsIBDSelection(ibd)
> head(ibd.coeff)

  sample1 sample2        k0         k1 kinshipcoeff
1 NA19152 NA19139 0.9548539 0.04514610  0.011286524
2 NA19152 NA18912 1.0000000 0.00000000  0.000000000
3 NA19152 NA19160 1.0000000 0.00000000  0.000000000
4 NA19152 NA18515 0.9234541 0.07654590  0.019136475
5 NA19152 NA19222 1.0000000 0.00000000  0.000000000
6 NA19152 NA18508 0.9833803 0.01661969  0.004154922

> plot(ibd.coeff$k0, ibd.coeff$k1, xlim=c(0,1), ylim=c(0,1),
+          xlab="k0", ylab="k1", main="YRI samples (MoM)")
> lines(c(0,1), c(1,0), col="red", lty=2)
```

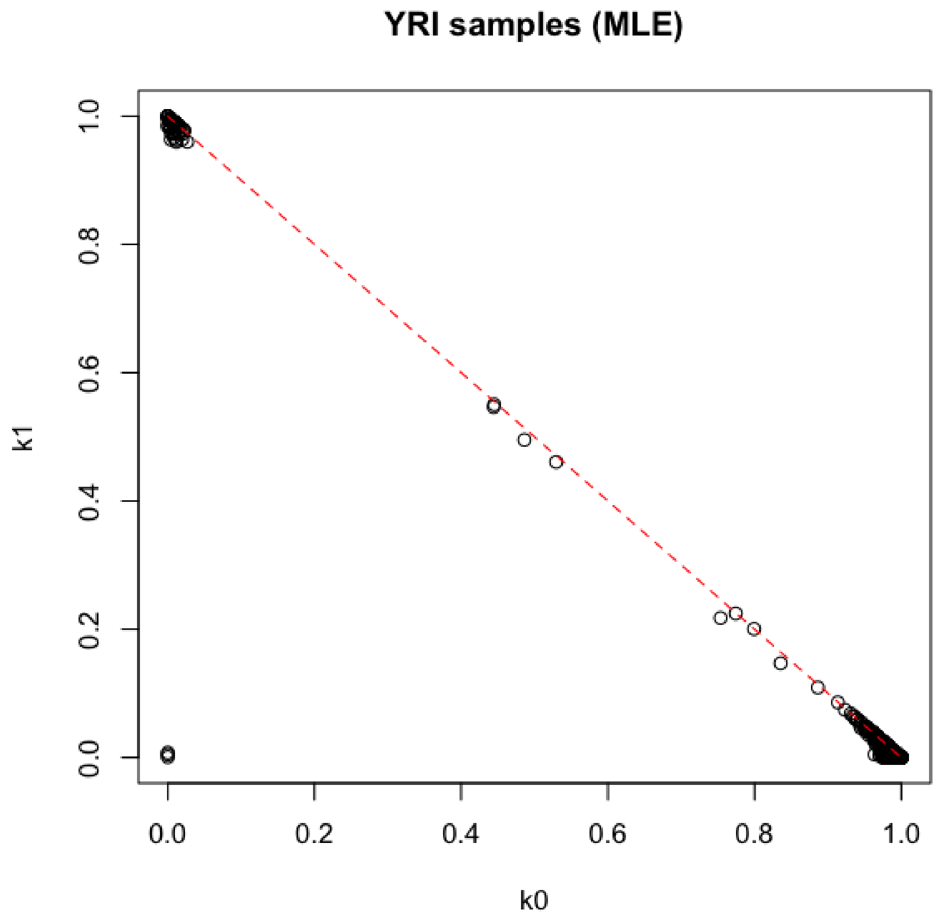


3.3.2 IBD Using Maximum Likelihood Estimation (MLE)

```
> # estimate IBD coefficients
> set.seed(1000)
> snp.id <- sample(snpset.id, 5000)  # random 5000 SNPs
> ibd <- snpgdsIBDMLE(genofile, sample.id=YRI.id, snp.id=snp.id,
+          maf=0.05, missing.rate=0.05)

Identity-By-Descent analysis (MLE) on SNP genotypes:
Removing 981 SNPs (monomorphic, < MAF, or > missing rate)
Working space: 93 samples, 4019 SNPs
         Using 1 CPU core.
MLE IBD:         the sum of all working genotypes = 369973
MLE IBD:         Fri Jul 20 04:49:21 2012        0%
MLE IBD:         Fri Jul 20 04:49:45 2012        6%
MLE IBD:         Fri Jul 20 04:50:08 2012        12%
MLE IBD:        Fri Jul 20 04:50:29 2012        19%
MLE IBD:        Fri Jul 20 04:50:57 2012        26%
MLE IBD:        Fri Jul 20 04:51:19 2012        32%
MLE IBD:        Fri Jul 20 04:51:42 2012        38%
MLE IBD:        Fri Jul 20 04:52:11 2012        45%
MLE IBD:        Fri Jul 20 04:52:41 2012        51%
MLE IBD:        Fri Jul 20 04:53:09 2012        57%
MLE IBD:        Fri Jul 20 04:53:35 2012        64%
MLE IBD:        Fri Jul 20 04:54:02 2012        70%
MLE IBD:        Fri Jul 20 04:54:26 2012        76%
MLE IBD:        Fri Jul 20 04:54:53 2012        83%
MLE IBD:        Fri Jul 20 04:55:14 2012        90%
MLE IBD:        Fri Jul 20 04:55:39 2012        97%
MLE IBD:        Fri Jul 20 04:55:45 2012        100%
```

```
> ibd.coeff <- snpgdsIBDSelection(ibd)
```

```
> plot(ibd.coeff$k0, ibd.coeff$k1, xlim=c(0,1), ylim=c(0,1),
+         xlab="k0", ylab="k1", main="YRI samples (MLE)")
> lines(c(0,1), c(1,0), col="red", lty=2)
```



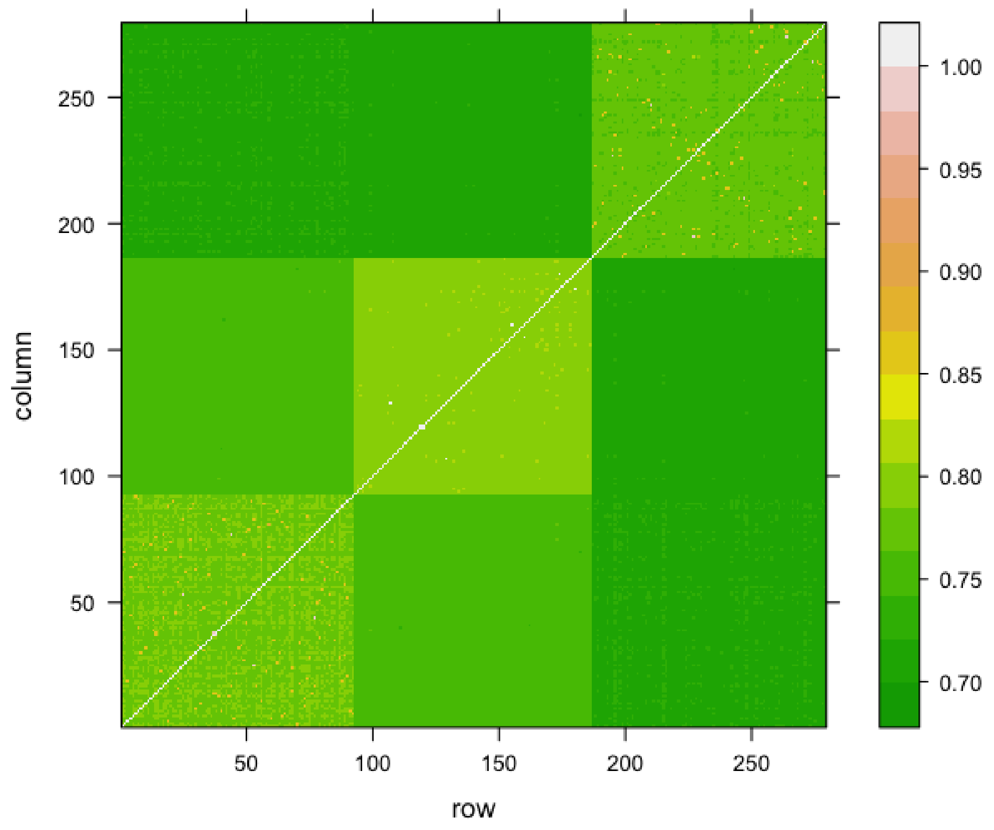
3.4 Identity-By-State Analysis

For the n individuals in a sample, **snpgdsIBS** can be used to create a $n \times n$ matrix of genome-wide average IBS pairwise identities:

```
> ibs <- snpgdsIBS(genofile, num.thread=2)

Identity-By-State (IBS) analysis on SNP genotypes:
Removing 365 non-autosomal SNPs
Removing 1 SNPs (monomorphic, < MAF, or > missing rate)
Working space: 279 samples, 8722 SNPs
        Using 2 CPU cores.
IBS:        the sum of all working genotypes = 2446510
IBS:        Fri Jul 20 04:55:47 2012        0%
IBS:        Fri Jul 20 04:55:48 2012        100%

> pop <- read.gdsn(index.gdsn(genofile, c("sample.annot", "pop.group")))
```

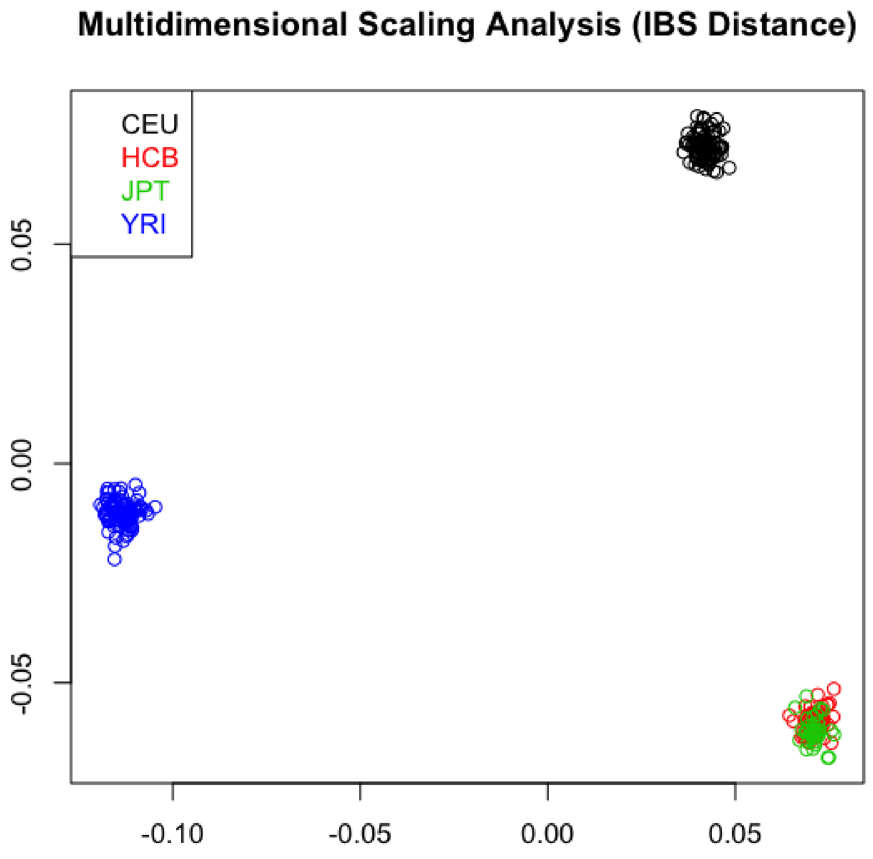
The heat map is shown:

```
> library(lattice)
> L <- order(pop)
> levelplot(ibs$ibs[L, L], col.regions = terrain.colors)
```



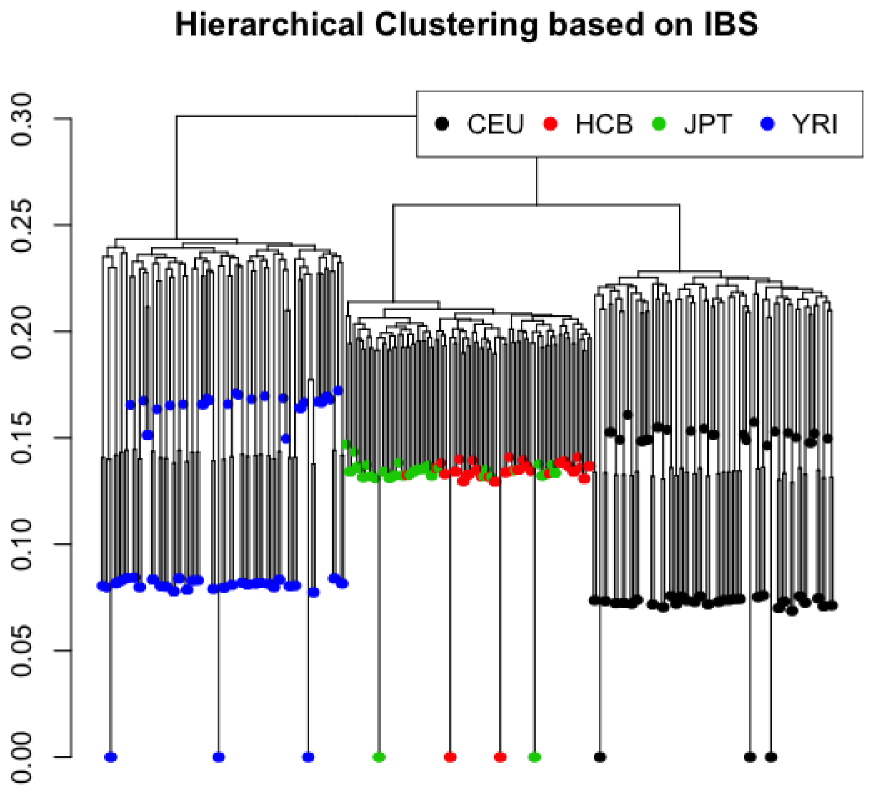
To perform multidimensional scaling analysis on the $n \times n$ matrix of genome-wide IBS pairwise distances:

```
> loc <- cmdscale(1 - ibs$ibs, k = 2)
> x <- loc[, 1]; y <- loc[, 2]
> race <- as.factor(pop)
> plot(x, y, col=race, xlab = "", ylab = "",
+         main = "Multidimensional Scaling Analysis (IBS Distance)")
> legend("topleft", legend=levels(race), text.col=1:nlevels(race))
```



To perform cluster analysis on the $n \times n$ matrix of genome-wide IBS pairwise distances:

```
> d <- 1 - ibs$ibs
> colnames(d) <- rownames(d) <- pop
> hc <- hclust(as.dist(d))
> obj <- as.dendrogram(hc, hang=0.2)
> add.point <- function(n, group) {
+   if (is.leaf(n))
+   {
+     attr(n, "nodePar") <- list(pch=20, col=match(attr(n, "label"), group))
+   }
+   n
+ }
> obj <- dendrapply(obj, add.point, group=levels(race))
> plot(obj, leaflab="none", main="Hierarchical Clustering based on IBS")
> legend("topright", legend=levels(race), col=1:nlevels(race), pch=19, ncol=4)
```



```
> # close the GDS file
> closefn.gds(genofile)
```

4 Resources

1. CoreArray project: `http://corearray.sourceforge.net/`
2. gdsfmt R package: `http://cran.r-project.org/web/packages/gdsfmt/index.html`
3. SNPRelate R package: `http://cran.r-project.org/web/packages/SNPRelate/index.html`

5 Acknowledgements

The author would like to thank members of the GENEVA consortium (`http://www.genevastudy.org`) for access to the data used for testing the gdsfmt and SNPRelate packages.

Funding: National Institutes of Health (HG 004446).