Data formats in GWASTools

Stephanie M. Gogarten

October 29, 2024

The central classes of the *GWASTools* package are `GenotypeData` and `IntensityData`. They are designed to link all parts of a GWAS analysis (genotype data, SNP information, and sample information) in a single S4 object, even when the genotype data is too large to be stored in R's memory at one time. In designing *GWASTools*, we took care to separate the application programming interface (API) of the `GenotypeData` and `IntensityData` classes from the format in which the data are stored.

Each class contains a data slot (for a `GenotypeReader` or `IntensityReader` object, respectively) and annotation slots (a `SnpAnnotationReader` and a `ScanAnnotationReader`). These Reader classes are actually class unions, allowing multiple options for storing data and enabling new storage methods to be added without changing any code that uses `GenotypeData` and `IntensityData` objects. The class unions are currently defined as follows:

- `GenotypeReader`: `NcdfGenotypeReader`, `GdsGenotypeReader`, or `MatrixGenotypeReader`
- `IntensityReader`: `NcdfIntensityReader` or `GdsIntensityReader`
- `SnpAnnotationReader`: `SnpAnnotationDataFrame` or `SnpAnnotationSQLite`
- `ScanAnnotationReader`: `ScanAnnotationDataFrame` or `ScanAnnotationSQLite`

We use the term "scan" to indicate a unique genotyping instance, as the same DNA sample may be genotyped more than once. Each SNP and scan must have a unique integer ID ("snpID" and "scanID") that serves as the primary key between the genotype data and the annotation. Validity methods ensure that these IDs, as well as chromosome and base position of SNPs, are consistent between the data and annotation slots. Chromosome and position values must be integers, so all classes which include SNP data have slots to record integer codes for non-autosomal chromosome types (X, Y, pseudoautosomal, and mitochondrial).

1 Genotype data formats

1.1 NetCDF

The Network Common Data Form (NetCDF, http://www.unidata.ucar.edu/software/netcdf/) allows array-oriented data to be stored on disk with fast access to subsets of the data in R using the *ncdf4* package. The `NcdfReader` class provides an S4 wrapper for `ncdf` objects. `NcdfGenotypeReader` and `NcdfIntensityReader` extend `NcdfReader` with methods specific to genotype and intensity data.

All NetCDF files created for `GWASTools` have two dimensions, one called `snp` and one titled `sample`. Further, all NetCDF files have three variables in common: `sampleID, chromosome` and `position`. The `sampleID` is used for indexing the columns of the two dimensional values stored in the NetCDF files (genotype calls, for example). The index to the SNP probes in the NetCDF file is the `snpID`, which is stored as values of the SNP dimension.

1.2 GDS

Genomic Data Structure (GDS, http://corearray.sourceforge.net/) is a storage format for bioinformatics data similar to NetCDF. An R interface is provided with the `gdsfmt` package. The *GWASTools* functions `convertNcdfGds` and `convertGdsNcdf` allow conversion between NetCDF and GDS format. GDS format is required for the *SNPRelate* package, which computes relatedness and PCA as demonstrated in the "GWAS Data Cleaning" vignette, so it may be convenient to store data in this format from the start. The `GdsReader` class provides a wrapper for *gdsfmt* objects with the same API as the `NcdfReader` class. `GdsGenotypeReader` and `GdsIntensityReader` extend `GdsReader` with methods specific to genotype and intensity data.

All GDS files created for `GWASTools` have variables `sample.id, snp.id, snp.chromosome`, and `snp.position`. For genotype files, which store the count of the "A" allele in two bits, the A and B alleles are stored in the `snp.allele` variable in the form "A/B". Character SNP identifiers are often stored in the variable `snp.rs.id`.

1.3 Matrix

The `MatrixGenotypeReader` class is convenient for analyses on smaller data sets which can easily fit into R's memory. It combines a matrix of genotypes with scanID, snpID, chromosome, and position.

2 Annotation

SNP and scan annotation can be stored in either of two formats: an annotated data frame, or a SQLite database. Either format may be supplied to the `snpAnnot` and `scanAnnot` slots of a `GenotypeData` or `IntensityData` object. Each annotation object consists of two component data frames (or tables). The main annotation data frame has one row for each SNP (or scan) and columns containing variables such as (for SNPs) snpID, chromosome, position, rsID, A and B alleles and (for scans) scanID, subject ID (to link duplicate scans of the same subject), sex, and phenotype. The metadata data frame has one row for each column in the annotation data frame, and (at minimum) a column containing a description of the variable. Both formats share methods to return annotation columns and metadata.

2.1 Annotated data frames

The `SnpAnnotationDataFrame` and `ScanAnnotationDataFrame` classes extend the `AnnotatedDataFrame` class in the `Biobase` package. In addition to GWASTools methods, all methods defined for `AnnotatedDataFrame` are available to these classes, including operators which allow these objects to be used like standard data frames in many ways. This format provides some built-in functionality from `AnnotatedDataFrame` to ensure that the annotation and metadata data frames are consistent.

2.2 SQLite databases

The `ScanAnnotationSQLite` and `ScanAnnotationSQLite` classes provide an alternate means of storing annotation that is portable across multiple platforms. In addition to the methods shared with the annotation data frame classes, these classes have `getQuery` methods to pass any SQL query to the database.

3 Input

3.1 Plain text

Data in plain text format (for example, FinalReport files produced by Illumina's GenomeStudio) can be converted to NetCDF or GDS files using the function `createDataFile`. See the "GWAS Data Cleaning" and "Preparing Affymetrix Data" vignettes for examples.

3.2 PLINK

PLINK ped/map files can be converted to NetCDF with accompanying SNP and scan annotation using the function `plinkToNcdf`. `plinkToNcdf` will automatically convert between the sex chromosome codes used by PLINK and the default codes used by *GWASTools*.

`snpgdsBED2GDS` in the *SNPRelate* package converts binary PLINK to GDS. `snpgdsBED2GDS` is significantly faster than `plinkToNcdf`, and the resulting GDS file may be used with *SNPRelate* as well. The option `cvt.snpid="int"` is required to generate integer snpIDs. Chromosome codes are not converted.

```
> library(GWASTools)
> library(SNPRelate)
> bed.fn <- system.file("extdata", "plinkhapmap.bed.gz", package="SNPRelate")
> fam.fn <- system.file("extdata", "plinkhapmap.fam.gz", package="SNPRelate")
> bim.fn <- system.file("extdata", "plinkhapmap.bim.gz", package="SNPRelate")
> gdsfile <- "snps.gds"
> snpgdsBED2GDS(bed.fn, fam.fn, bim.fn, gdsfile, family=TRUE,
+               cvt.chr="int", cvt.snpid="int", verbose=FALSE)
```

Now that the file has been created, we can access it in *GWASTools* using the `GdsGenotypeReader` class. We create sample and SNP annotation from the variables stored in the GDS file. Note that PLINK sex chromosome coding is different from the *GWASTools* default, so specify codes if your file contains Y or pseudoautosomal SNPs.

```
> (gds <- GdsGenotypeReader(gdsfile, YchromCode=24L, XYchromCode=25L))

File: /private/var/folders/db/4tvgx8jx4z3fm1gzlnlzw9rc0000gq/T/RtmpSReCjD/Rbuild19a4423c7469/GWA
+    [  ] *
|--+ sample.id   { Str8 60 LZMA_ra(53.8%), 265B }
|--+ snp.id   { Int32 5000 LZMA_ra(11.7%), 2.3K }
|--+ snp.rs.id   { Int32 5000 LZMA_ra(20.8%), 4.1K }
|--+ snp.position   { Int32 5000 LZMA_ra(78.5%), 15.3K }
```

```
|--+ snp.chromosome   { UInt8 5000 LZMA_ra(3.00%), 157B } *
|--+ snp.allele   { Str8 5000 LZMA_ra(13.8%), 2.7K }
|--+ genotype   { Bit2 60x5000, 73.2K } *
\--+ sample.annot   [ data.frame ] *
   |--+ family   { Int32 60 LZMA_ra(35.8%), 93B }
   |--+ father   { Int32 60 LZMA_ra(35.8%), 93B }
   |--+ mother   { Int32 60 LZMA_ra(35.8%), 93B }
   |--+ sex   { Str8 60 LZMA_ra(136.7%), 89B }
   \--+ phenotype   { Int32 60 LZMA_ra(37.5%), 97B }
```

```
> scanID <- getScanID(gds)
> family <- getVariable(gds, "sample.annot/family")
> father <- getVariable(gds, "sample.annot/father")
> mother <- getVariable(gds, "sample.annot/mother")
> sex <- getVariable(gds, "sample.annot/sex")
> sex[sex == ""] <- NA # sex must be coded as M/F/NA
> phenotype <- getVariable(gds, "sample.annot/phenotype")
> scanAnnot <- ScanAnnotationDataFrame(data.frame(scanID, father, mother,
+                                                 sex, phenotype,
+                                                 stringsAsFactors=FALSE))
> snpID <- getSnpID(gds)
> chromosome <- getChromosome(gds)
> position <- getPosition(gds)
> alleleA <- getAlleleA(gds)
> alleleB <- getAlleleB(gds)
> rsID <- getVariable(gds, "snp.rs.id")
> snpAnnot <- SnpAnnotationDataFrame(data.frame(snpID, chromosome, position,
+                                               rsID, alleleA, alleleB,
+                                               stringsAsFactors=FALSE),
+                                    YchromCode=24L, XYchromCode=25L)
> genoData <- GenotypeData(gds, scanAnnot=scanAnnot, snpAnnot=snpAnnot)
> getGenotype(genoData, snp=c(1,5), scan=c(1,5))
```

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

```
> close(genoData)
```

3.3 VCF

Bi-allelic SNP data from Variant Call Format (VCF) can be converted to GDS using the function `snpgdsVCF2GDS` in the `SNPRelate` package.

```
> library(GWASTools)
> library(SNPRelate)
> vcffile <- system.file("extdata", "sequence.vcf", package="SNPRelate")
> gdsfile <- "snps.gds"
> snpgdsVCF2GDS(vcffile, gdsfile, verbose=FALSE)
```

Now that the file has been created, we can access it in *GWASTools* using the `GdsGenotypeReader` class. We create a `SnpAnnotationDataFrame` from the variables stored in the GDS file.

```
> (gds <- GdsGenotypeReader(gdsfile))

File: /private/var/folders/db/4tvgx8jx4z3fm1gzlnlzw9rc0000gq/T/RtmpSReCjD/Rbuild19a4423c7469/GWA
+    [  ] *
|--+ sample.id   { Str8 3 LZMA_ra(375.0%), 97B }
|--+ snp.id   { Int32 2 LZMA_ra(975.0%), 85B }
|--+ snp.rs.id   { Str8 2 LZMA_ra(745.5%), 89B }
|--+ snp.position   { Int32 2 LZMA_ra(975.0%), 85B }
|--+ snp.chromosome   { Str8 2 LZMA_ra(1300.0%), 85B }
|--+ snp.allele   { Str8 2 LZMA_ra(975.0%), 85B }
|--+ genotype   { Bit2 3x2, 2B } *
\--+ snp.annot   [  ]
   |--+ qual   { Float32 2 LZMA_ra(975.0%), 85B }
   \--+ filter   { Str8 2 LZMA_ra(911.1%), 89B }

> getScanID(gds)

[1] "NA00001" "NA00002" "NA00003"

> snpID <- getSnpID(gds)
> chromosome <- as.integer(getChromosome(gds))
> position <- getPosition(gds)
> alleleA <- getAlleleA(gds)
> alleleB <- getAlleleB(gds)
> rsID <- getVariable(gds, "snp.rs.id")
> qual <- getVariable(gds, "snp.annot/qual")
> filter <- getVariable(gds, "snp.annot/filter")
> snpAnnot <- SnpAnnotationDataFrame(data.frame(snpID, chromosome, position,
+                                               rsID, alleleA, alleleB,
+                                               qual, filter,
+                                               stringsAsFactors=FALSE))
> genoData <- GenotypeData(gds, snpAnnot=snpAnnot)
> getGenotype(genoData)

     [,1] [,2] [,3]
[1,]    2    1    0
[2,]    2    1    2

> close(genoData)
```

3.4 Imputed genotypes

Genotype probabilities or dosages from IMPUTE2, BEAGLE, or MaCH can be converted into A allele dosage and stored in NetCDF or GDS with the function `imputedDosageFile`.

4 Output

4.1 PLINK

A `GenotypeData` object can be written to PLINK ped/map files with the function `plinkWrite`.

4.2 VCF

A `GenotypeData` object can be written to VCF with the function `vcfWrite`. `genoDataAsVCF` converts a `GenotypeData` object to a `VCF` object for use with the *VariantAnnotation* package.

4.3 snpStats

`asSnpMatrix` converts a `GenotypeData` object to a `SnpMatrix` object for use with the *snpStats* package.