Converting VCF data for use in GWASTools

Stephanie M. Gogarten

April 22, 2014

This vignette demonstrates how to convert a Variant Call Format (VCF) file into a format compatible with *GWASTools*. As *GWASTools* expects genotype data represented as the number of A alleles, with values of 0, 1, 2, or missing, only bi-allelic SNP data can be preserved in the conversion. Other types of variants will be ignored. The VCF format is described here: http://www.1000genomes.org/wiki/Analysis/Variant%20Call%20Format/vcf-variant-call-format-version-41.

1 Converting to GDS

The simplest way to convert a VCF file is to use the function `convertVcfGds`. This function will extract sample name, SNP ID, chromosome, position, reference and alternate alleles, and genotypes from a VCF file and store them in a GDS file (described in the *gdsfmt* package). Unique integer IDs are generated for all samples and SNPs.

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

Now that the file has been created, we can access it in *GWASTools* using the `GdsGenotypeReader` class.

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

```
File: /private/tmp/RtmpYFLIfN/Rbuild2791586f1dbe/GWASTools/vignettes/snps.gds
+         [  ]
|--+ sample.id        { Int32 3 ZIP(141.67%) }
|--+ sample.name        { VStr8 3 ZIP(87.50%) }
|--+ snp.id         { Int32 2 ZIP(175.00%) }
|--+ snp.rs.id        { VStr8 2 ZIP(166.67%) }
|--+ snp.position         { Int32 2 ZIP(200.00%) }
|--+ snp.chromosome         { UInt8 2 ZIP(500.00%) }
|--+ snp.allele        { VStr8 2 ZIP(175.00%) }
|--+ genotype        { Bit2 2x3 } *
```

```
> getScanID(gds)
```

```
[1] 1 2 3

> getSnpID(gds)

[1] 1 2

> getChromosome(gds)

[1] 20 20

> getPosition(gds)

[1] 14370 17330

> getVariable(gds, "sample.name")

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

> getVariable(gds, "snp.rs.id")

[1] "rs6054257" "."

> getVariable(gds, "snp.allele")

[1] "G/A" "T/A"

> getGenotype(gds)

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

> close(gds)
```

2 Converting to NetCDF

In some cases it may be desirable to extract additional information from a VCF file, e.g., monomorphic SNPs, quality scores, etc. In that case one can use the *VariantAnnotation* package to read the VCF file. We then create a NetCDF file with SNP and scan annotation for use in *GWASTools*.

```
> library(VariantAnnotation)
> vcffile <- system.file("extdata", "ex2.vcf", package="VariantAnnotation")
> (vcf <- readVcf(vcffile, "hg18"))
```

```
class: CollapsedVCF
dim: 5 3
rowData(vcf):
  GRanges with 5 metadata columns: paramRangeID, REF, ALT, QUAL, FILTER
info(vcf):
  DataFrame with 6 columns: NS, DP, AF, AA, DB, H2
info(header(vcf)):
      Number Type    Description
   NS 1      Integer Number of Samples With Data
   DP 1      Integer Total Depth
   AF A      Float   Allele Frequency
   AA 1      String  Ancestral Allele
   DB 0      Flag    dbSNP membership, build 129
   H2 0      Flag    HapMap2 membership
geno(vcf):
  SimpleList of length 4: GT, GQ, DP, HQ
geno(header(vcf)):
      Number Type    Description
   GT 1      String  Genotype
   GQ 1      Integer Genotype Quality
   DP 1      Integer Read Depth
   HQ 2      Integer Haplotype Quality
```

Retrieve the reference and alternate alleles. Since we are keeping only bi-allelic SNPs, we discard any variants where the reference or alternate sequences have length > 1, and variants with more than one alternate allele.

```
> # width of ref seq
> rw <- width(ref(vcf))
> # width of first alt seq
> aw <- unlist(lapply(alt(vcf), function(x) {width(x[1])}))
> # number of alternate genotypes
> nalt <- elementLengths(alt(vcf))
> # select only bi-allelic SNPs (monomorphic OK, so aw can be 0 or 1)
> snp <- rw == 1 & aw <= 1 & nalt == 1
> # subset vcf
> vcf <- vcf[snp,]
```

Retrieve the row data from the VCF, which describes the SNPs.

```
> chrom <- as.vector(seqnames(rowData(vcf)))
> position <- start(ranges(rowData(vcf)))
> alleleA <- as.character(ref(vcf))
> alleleB <- as.character(unlist(alt(vcf)))
> QUAL <- qual(vcf)
> FILTER <- filt(vcf)
```

Chromosome must be stored as an integer in the NetCDF file.

```
> chromosome <- chrom
> chromosome[chrom == "X"] <- 23
> chromosome[chrom == "XY"] <- 24 # pseudoautosomal
> chromosome[chrom == "Y"] <- 25
> chromosome[chrom == "M"] <- 26 # mitochrondrial
> chromosome <- as.integer(chromosome)
```

Create a data.frame from the row data, as well as metadata describing each column. Since FILTER is a single column, its metadata is combined into a single character string. The SNPs are ordered by the integer value of chromosome and position. The "snpID" column contains a unique integer ID that will be used to index SNPs in the NetCDF file.

```
> # make annotation
> snp.df <- data.frame(chromosome, position, alleleA, alleleB, QUAL, FILTER,
+                      row.names=rownames(vcf), stringsAsFactors=FALSE)
> snp.ord <- order(snp.df$chromosome, snp.df$position)
> snp.df <- snp.df[snp.ord,]
> snp.df$snpID <- 1:nrow(snp.df)
> cols <- c("snpID", "chromosome", "position", "alleleA", "alleleB", "QUAL", "FILTER")
> hdr <- exptData(vcf)$header
> desc <- c("unique integer ID = snp dimension values in NetCDF file",
+   "chromosome as integer where X=23, XY=24, Y=25, M=26",
+   "base position",
+   "allele A (the reference allele)",
+   "allele B (the alternate allele)",
+   "phred-scaled quality score for the assertion made in ALT. i.e. -10log_10 prob(call in ALT
+   paste(paste(row.names(fixed(hdr)$FILTER), fixed(hdr)$FILTER$Description, sep="="), collapse
> meta <- data.frame("labelDescription"=desc, row.names=cols)
> snp.df <- snp.df[,cols]
```

The INFO field of the VCF may contain many columns, some of which may contain many values. Here we keep only columns which have a single value for each SNP (Number=0, 1, or A). 'A' means one value per alternate allele, but we have kept only variants with one alternate allele.

```
> info <- info(vcf)
> info.meta <- info(hdr)
> vals <- list()
> for (i in 1:nrow(info.meta)) {
+   field <- row.names(info.meta)[i]
+   number <- info.meta$Number[i]
+   if (number == "A") {
+     # one element per alternate allele
+     # should only have one value per variant
+     dat <- unlist(info[[field]])
+     if (length(dat) > nrow(info)) next
+     vals[[field]] <- dat
+   } else if (number %in% 0:1) {
```

```
+     vals[[field]] <- info[[field]]
+   }
+ }
> snp.df <- cbind(snp.df, vals)
> meta.vals <- as.data.frame(info.meta[names(vals), "Description", drop=FALSE])
> names(meta.vals) <- "labelDescription"
> meta <- rbind(meta, meta.vals)
```

Combine the SNP annotation and metadata into a `SnpAnnotationDataFrame`.

```
> library(GWASTools)
> snpAnnot <- SnpAnnotationDataFrame(snp.df, meta)
```

Create a unique integer scanID that will be used to index samples in the NetCDF file, and store it in a `ScanAnnotationDataFrame`.

```
> scan.df <- data.frame("scanID"=1:ncol(vcf), row.names=colnames(vcf))
> meta.scan <- data.frame("labelDescription"="unique integer ID = sampleID in NetCDF file",
+                         row.names=names(scan.df))
> scanAnnot <- ScanAnnotationDataFrame(scan.df, meta.scan)
```

Extract the genotypes from the VCF, and convert to the number of REF alleles so each genotype may be stored as a single byte. Phase information (denoted by "|" or "/") is lost.

```
> geno <- geno(vcf)$GT
> geno.ab <- geno
> geno.ab[,] <- NA
> geno.ab[geno %in% c("0/0", "0|0")] <- 2 # AA = REF/REF
> geno.ab[geno %in% c("0/1", "0|1", "1/0", "1|0")] <- 1 # AB = REF/ALT
> geno.ab[geno %in% c("1/1", "1|1")] <- 0 # BB = ALT/ALT
> # new snp order (by chrom and position)
> geno.ab <- geno.ab[snp.ord,]
> # geno.ab should be integer, not character
> mode(geno.ab) <- "integer"
```

Extract the genotype quality scores from the VCF.

```
> gq <- geno(vcf)$GQ
```

Create a genotype NetCDF file and add the integer sample IDs, genotypes, and quality scores.

```
> ncfile <- "ex2.nc"
> ncdfCreate(snp.annotation=snp.df, ncdf.filename=ncfile,
+            n.samples=ncol(vcf), variables=c("genotype", "quality"))
> nc <- open.ncdf(ncfile, write=TRUE)
> put.var.ncdf(nc, "sampleID", scanAnnot$scanID, start=1, count=-1)
> put.var.ncdf(nc, "genotype", geno.ab)
> put.var.ncdf(nc, "quality", gq)
> close.ncdf(nc)
```

```
[[1]]
[1] 196608
```

Create a `GenotypeData` object to check that everything worked correctly. This object can be used in the *GWASTools* functions.

```
> nc <- NcdfGenotypeReader(ncfile)
> (genoData <- GenotypeData(nc, snpAnnot=snpAnnot, scanAnnot=scanAnnot))

An object of class GenotypeData
 | data:
[1] "file ex2.nc has 2 dimensions:"
[1] "sample   Size: 3"
[1] "snp   Size: 3"
[1] "------------------------"
[1] "file ex2.nc has 5 variables:"
[1] "int sampleID[sample]  Longname:sampleID Missval:0"
[1] "int position[snp]  Longname:position Missval:-1"
[1] "int chromosome[snp]  Longname:chromosome Missval:-1"
[1] "byte genotype[snp,sample]  Longname:genotype Missval:-1"
[1] "double quality[snp,sample]  Longname:quality Missval:-9999"
 | SNP Annotation:
An object of class 'SnpAnnotationDataFrame'
  snps: rs6054257 20:17330_T/A 20:1230237_T/.
  varLabels: snpID chromosome ... H2 (13 total)
  varMetadata: labelDescription
 | Scan Annotation:
An object of class 'ScanAnnotationDataFrame'
  scans: NA00001 NA00002 NA00003
  varLabels: scanID
  varMetadata: labelDescription

> pData(snpAnnot)

               snpID chromosome position alleleA alleleB QUAL FILTER NS DP
rs6054257          1         20    14370       G       A   29   PASS  3 14
20:17330_T/A       2         20    17330       T       A    3    q10  3 11
20:1230237_T/.     3         20  1230237       T           47   PASS  3 13
                  AF   AA    DB    H2
rs6054257      0.500 <NA>  TRUE  TRUE
20:17330_T/A   0.017 <NA> FALSE FALSE
20:1230237_T/.    NA    T FALSE FALSE

> pData(scanAnnot)

        scanID
NA00001      1
NA00002      2
NA00003      3
```

```
> getGenotype(genoData)
```

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

```
> getVariable(genoData, "quality")
```

```
     [,1] [,2] [,3]
[1,]   48   48   43
[2,]   49    3   41
[3,]   54   48   61
```

```
> close(genoData)
```

3 References

The variant call format and VCFtools. Danecek P, Auton A, Abecasis G, Albers CA, Banks E, DePristo MA, Handsaker RE, Lunter G, Marth GT, Sherry ST, McVean G, Durbin R; 1000 Genomes Project Analysis Group. Bioinformatics. 2011 Aug 1;27(15):2156-8. Epub 2011 Jun 7.

GDS home page: http://corearray.sourceforge.net

NetCDF home page: http://www.unidata.ucar.edu/software/netcdf