MCB536 Lecture 16 (Part 3): Read Variant Call Format (VCF) Files in R

Gavin Ha

11/22/2020

We will learn to read VCF files within R using a publicly available dataset of genomic variant calls for the infamous individual, NA12878. The Genome-in-a-Bottle Consortium has compiled consensus variant calls on this individual's genome and released this data for researchers to use. One of the main purposes of this data is to provide a benchmark for those to develop computational tools and analysis of human genomes. See https://github.com/genome-in-a-bottle/giab_latest_release

Variant Call Format (VCF) is a very common format for representing genomic variation data. See Lecture 16: Slides 12.

0. Install and load the `VariantAnnotation` Bioconductor package

Load the `VariantAnnotation` package

```
#BiocManager::install("VariantAnnotation")
library(VariantAnnotation)
```

1. Prepare parameters for reading VCF file.

There are a lot of variants in this file `GIAB_highconf_v.3.3.2.vcf.gz`, so we want to restrict to a smaller region for this example.

a. Setup parameters for scanning the VCF file.

First, we need to set up a `ScanVcfParam` object to read within `17:35500000-36000000`.

```
vcfFile <- "GIAB_highconf_v.3.3.2.vcf.gz"
vcfHead <- scanVcfHeader(vcfFile)
myGRange4 <- GRanges(seqnames = "17", ranges = IRanges(start = 35500000, end = 36000000))
vcf.param <- ScanVcfParam(which = myGRange4)
```

2. Read the VCF file.

```
vcf <- readVcf(vcfFile, genome = "hg19", param = vcf.param)
```

The `vcf` variable is of class `CollapsedVCF` and will contain header information and data. Let's see what information has been parsed by `readVcf`.

3. Extract the contents of the VCF entries.

a. Return the variants in this region as a `GRanges` object.

The `rowRanges` function will return a `GRanges` object containing the coordinates, REF/ALT bases, quality, and filtering status of the variants.

```
rowRanges(vcf)
```

```
## GRanges object with 332 ranges and 5 metadata columns:
##                  seqnames            ranges strand | paramRangeID            REF
##                     <Rle>         <IRanges>  <Rle> |     <factor> <DNAStringSet>
##     rs2411161        17          35501799      * |           NA              C
##     rs8073074        17          35502949      * |           NA              A
##     rs4523972        17          35507230      * |           NA              C
##   rs111498996        17 35507465-35507466      * |           NA             CA
##     rs8077266        17          35509302      * |           NA              A
##           ...       ...               ...    ... .          ...            ...
##     rs8080225        17          35996195      * |           NA              T
##     rs8075378        17          35996582      * |           NA              G
##     rs6607281        17          35997126      * |           NA              T
##     rs4332783        17          35997674      * |           NA              A
##    rs71984199        17          35998800      * |           NA              C
##                               ALT      QUAL      FILTER
##                  <DNAStringSetList> <numeric> <character>
##     rs2411161                   T        50        PASS
##     rs8073074                   G        50        PASS
##     rs4523972                   T        50        PASS
##   rs111498996                   C        50        PASS
##     rs8077266                   G        50        PASS
##           ...                 ...       ...         ...
##     rs8080225                   C        50        PASS
##     rs8075378                   A        50        PASS
##     rs6607281                   C        50        PASS
##     rs4332783                   G        50        PASS
##    rs71984199                  CT        50        PASS
##   -------
##   seqinfo: 25 sequences from hg19 genome
```

b. Inspect the header information

The `INFO` column in the original VCF text file contains a semi-colon delimited set of custom fields with flexible format that algorithms will output. Here, it is parsed into usable format. First, let's look at what fields are available from the header.

```
info(vcf) # returns a DataFrame object
```

The `FORMAT` column in the original VCF text file contains the format and description of the genotype fields. Let's see what these are.

```
geno(header(vcf))
```

```
## DataFrame with 8 rows and 3 columns
##             Number        Type            Description
##        <character> <character>            <character>
## GT                1      String Consensus Genotype a..
## DP                1     Integer Total read depth sum..
## GQ                1     Integer Net Genotype quality..
## ADALL             R     Integer Net allele depths ac..
## AD                R     Integer Net allele depths ac..
## IGT               1      String Original input genot..
## IPS               1      String      Phase set for IGT
## PS                1      String       Phase set for GT
```

c. Inspect the genotype, read depth, and allele depth inforation.

To see the genotype `GT`, read depth `DP`, and allele depth `AD`, we access the the list.

```
geno(vcf)$GT[1:5]
```

```
## [1] "1|1" "1|1" "1|1" "1|1" "1|1"
```

```
geno(vcf)$DP[1:5]
```

```
## [1] 675 607 528 470 718
```

```
geno(vcf)$AD[1:5]
```

```
## [[1]]
## [1]  95 372
##
## [[2]]
## [1]  77 334
##
## [[3]]
## [1]  66 292
##
## [[4]]
## [1]   0 223
##
## [[5]]
## [1]  97 393
```

d. Combine all `geno` fields into a single table.

You can also combine all fields into a `data.frame` object. But this code only works if the VCF contains a single sample.

```
genoData <- data.frame(do.call(cbind, geno(vcf)))
colnames(genoData) <- rownames(geno(header(vcf)))
```

Exercise 3: Reading variants from a VCF file.

a. Create a range for `8:128747680-128753680`.

```
# GRanges()
```

b. Setup parameters to read VCF.

```r
# ScanVcfParam
```

c. Read the VCF file at `8:128747680-128753680`

```r
# readVcf
```

d. What is the RS id, genotype (`GT`) and depth (`DP`) at the SNP in this locus?

```r
# geno()
```