2. Using `filterVcf()` to Select Variants from VCF Files

Paul Shannon

Created: 20 February, 2013
Last modified: 22 October, 2015

Contents

1 Introduction

Whole genome Variant Call Format (VCF) files are very large, typically containing millions of called variants, one call per line of text. The actual number of variants relevant to a particular study (of a disease such as breast cancer, for instance) will often be far fewer. Thus the first task one faces when analyzing a whole genome VCF file is to identify and extract the relatively few variants which may be of interest, excluding all others. This vignette illustrates several techniques for doing this.

We demonstrate three methods: filtering by genomic region, filtering on attributes of each specific variant call, and intersecting with known regions of interest (exons, splice sites, regulatory regions, etc.). We are primarily concerned with the latter two. However, in order to create the small VCF data file we use here for demonstration purposes, we employed genomic region filtering, reducing a very large whole genome two-sample breast cancer VCF file of fourteen million calls, to a file containing fewer than ten thousand calls in a one million base pair region of chromosome 7. For the sake of reproducibility, and for completeness of exposition, we will illustrate this first step also.

2 The Data: Paired Tumor/Normal Breast Cancer Variants

Complete Genomics Inc.[1] states:

> To provide the scientific community with public access to data generated from two paired tumor/normal cancer samples, Complete Genomics sequenced and analyzed cell-line samples of patients with breast cancer (invasive ductal carcinomas). The cell line-derived DNA are housed at ATCC. Samples have been sequenced to an average genome-wide coverage of 123X for three of the samples, and 92X for for the fourth sample.[2]

A small (1M base) subset of this data is included in the current package, and used in the code presented below.

[1] http://www.completegenomics.com/public-data/cancer-data

[2] Data generated with version 2.0.0.32 of the Complete Genomics assembly software, on high molecular weight genomic DNA isolated from the HCC1187 breast carcinoma cell line ATCC CRL 2322. Sequencing methods documented in [Drmanac et al, 2010].

3 Filter by Genomic Region

We identified this subset in a prior exploration of the full data set (work not shown), learning that variants of biological interest suitable for our purpose are found in a 1M base pair region of chromosome seven. The appendix to this document (see below) shows the few lines of code required to extract variant calls in that small region, from the very large file obtained from Complete Genomics (*"somaticVcfBeta-HCC1187-H-200-37-ASM-T1-N1.vcf.gz"*) and write them to a new, small VCF file.

4 Introducing the `filterVcf` Method

The `filterVcf` method reads (by chunks, about which more below) through a possibly very large VCF file to write a new, smaller VCF file which includes only those variant calling rows which meet the criteria specified in `prefilters` and `filters`.

Reading "by chunks" is accomplished using a tabix [[**?**]] file. The `yieldSize` argument specifies how many variant lines are read into memory at each iteration.

```
tabix.file <- TabixFile(file.gz, yieldSize=10000)
filterVcf(tabix.file, genome, destination.file,
          prefilters=prefilters, filters=filters)
```

in which

1. *file.gz*: a gzipped vcf file with an accompanying Tabix index file.
2. *yieldSize*: the number of text (call variant) lines to read at a time.
3. *genome*: a string indicating the genome assembly, e.g., "hg19".
4. *prefilters*: one or more simple string-based filtering functions, each of which returns a logical vector corresponding to the vcf rows it will be passed (as simple character strings).
5. *filters*: one or more filtering function, each of which returns a logical vector, corresponding to the list of parsed vcf structures it will be passed.

4.1 Prefilters

Prefilters are conceptually very simple. They are functions which will be called with a single argument, a vector of character strings, each of which is an *unparsed* variant call line, as read from the input VCF file. We use `grepl` to return a logical vector equal in length to the incoming vector of unparsed VCF lines. Each prefilter and filter is called repeatedly, with `yieldSize` lines supplied on each invocation. `filterVcf` calls these functions repeatedly until the input file is exhausted.

Notice how the logic of these prefilters is very simple, using `grepl` to do fast, simple, fixed pattern matching:

```
> isGermlinePrefilter <- function(x) {
+     grepl("Germline", x, fixed=TRUE)
+ }
> notInDbsnpPrefilter <- function(x) {
+     !(grepl("dbsnp", x, fixed=TRUE))
+ }
```

4.2 Filters

Filters are more sophisticated than prefilters in that they assess *parsed* variant call lines for possible inclusion. Such parsing is intrinsically expensive but will be performed only on those lines which passed the prefilters. Therefore it pays to eliminate as many lines as possible using prefilters. Filters are useful when there exists detailed criteria for inclusion and exclusion. This can be seen below, especially in the `allelicDepth` function. Each filter must be written to return a logical vector as long as the number of rows in the input VCF argument; be sure your filter works with 0-row VCF instances.

```
> ## We will use isSNV() to filter only SNVs
>
> allelicDepth <- function(x) {
+     ##  ratio of AD of the "alternate allele" for the tumor sample
+     ##  OR "reference allele" for normal samples to total reads for
+     ##  the sample should be greater than some threshold (say 0.1,
+     ##  that is: at least 10% of the sample should have the allele
+     ##  of interest)
+     ad <- geno(x)$AD
+     tumorPct <- ad[,1,2,drop=FALSE] / rowSums(ad[,1,,drop=FALSE])
```

```
+     normPct <- ad[,2,1, drop=FALSE] / rowSums(ad[,2,,drop=FALSE])
+     test <- (tumorPct > 0.1) | (normPct > 0.1)
+     as.vector(!is.na(test) & test)
+ }
```

4.3 FilterRules

`FilterRules` allow you to combine a list of filters, or of prefilters so that they may be passed as parameters to *filterVcf*. We use them here to combine the *isGermlinePrefilter* with the *notInDbsnpPrefilter*, and the *isSNV* with the *AD* filter.

```
> library(VariantAnnotation)
> prefilters <- FilterRules(list(germline=isGermlinePrefilter,
+                                dbsnp=notInDbsnpPrefilter))
> filters <- FilterRules(list(isSNV=isSNV, AD=allelicDepth))
```

4.4 Create the Filtered file

```
> file.gz     <- system.file("extdata", "chr7-sub.vcf.gz",
+                            package="VariantAnnotation")
> file.gz.tbi <- system.file("extdata", "chr7-sub.vcf.gz.tbi",
+                            package="VariantAnnotation")
> destination.file <- tempfile()
> tabix.file <- TabixFile(file.gz, yieldSize=10000)
> filterVcf(tabix.file,  "hg19", destination.file,
+           prefilters=prefilters, filters=filters, verbose=TRUE)
```

5 Look for SNPs in Regulatory Regions

We have now created a file containing 29 novel, Germline, SNP variant calls, each with a reasonable allelic depth, extracted from the 3808 calls in *chr7-sub.vcf*. We examine those variants for overlap with regulatory regions, turning to the ENCODE project and using *Bioconductor*'s *AnnotationHub*.

The ENCODE project is a large, long-term effort to build a "parts list" of all the functional elements in the human genome. They have recently focused on regulatory elements. We use the *Bioconductor AnnotationHub* to download regulatory regions reported for a breast cancer cell line, with which to identify possibly functional, and possibly clinically relevant SNVs in the breast cancer tumor/normal genome we have been examining.

The *AnnotationHub* is a recent addition to *Bioconductor* that facilitates access to genome-scale resources like ENCODE.

5.1 Load CTCF Transcription Factor Binding Regions Identified in MCF-7 Breast Cancer Cell Line

The *MCF-7* Breast Cancer Cell line (http://en.wikipedia.org/wiki/MCF-7) was established forty years ago, and has since played a dominant role in breast cancer cell line studies. The University of Washington reports transcription factor binding sites (TFBS) for the CTCF protein, which often acts as a negative regulator of transcription via chromatin structure modifications Though the MCF-7 cell line is an imperfect match to the HCC1187 cell line sequenced by Complete Genomics, we combine these two breast-cancer related data sets here, didactically, in this exercise, to highlight the importance of cell-type-specific regulatory regions, and of the availability of such data from ENCODE. We shall see a SNP in the intronic binding region of the cancer-related gene, EGFR.

```
> library(AnnotationHub)
> hub <- AnnotationHub()
> id <- names(query(hub, "wgEncodeUwTfbsMcf7CtcfStdPkRep1.narrowPeak"))
> mcf7.gr <- hub[[tail(id, 1)]]
```

5.2 Find SNPs in CTCF Binding Regions

```
> vcf <- readVcf(destination.file, "hg19")
> seqlevels(vcf) <- paste("chr", seqlevels(vcf), sep="")
> ov.mcf7 <- findOverlaps(vcf, mcf7.gr)
```

There is just one SNV which overlaps with the MCF-7 regulatory regions. Find out where, if anywhere, it fits within a gene model.

```
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
> txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
> locateVariants(vcf[6,], txdb, AllVariants())
```

```
GRanges object with 0 ranges and 9 metadata columns:
   seqnames    ranges strand | LOCATION  LOCSTART    LOCEND   QUERYID      TXID
      <Rle> <IRanges>  <Rle> | <factor> <integer> <integer> <integer> <integer>
           CDSID      GENEID       PRECEDEID        FOLLOWID
   <IntegerList> <character> <CharacterList> <CharacterList>
  -------
  seqinfo: no sequences
```

6 Conclusion

This case study begins, somewhat artificially, with a very short region of chromosome seven, a section which we knew, from previous exploration, held an intronic regulatory SNV for EGFR, a receptor tryosine kinase implicated in some cancers. Though artificial, the case study illustrates all of the steps needed for broader, realistic surveys of whole genome variation data:

1. Filter by genomic coordinates.

2. Filter to extract only those variant calls which meet these criteria: Germline, novel (not in dbSnp), consisting of a single nucleotide, of sufficient allelic depth.
3. Intersect these variants with recognized DNA elements. In our case, we used short TFBS regulatory regions, but the same method can be used with exons, splice sites, DNaseI footprints, methylation sites, etc.

7 Appendix: Filter by Genomic Region

The most basic form of VCF file filtering is by genomic region. We demonstrate that here, extracting variant calls in a 1M base region of chromosome 7, writing them to a new file, compressing and then indexing that file. These steps created the small VCF file which accompanies this vignette, and is used in the code shown above.

Note that this code is *NOT* executed during the creation of this vignette: we do not supply the very large VCF file that it operates on. This code is here for tutorial purposes only, showing how you can filter by genomic region with a possibly large VCF file of your own.

```
library(VariantAnnotation)
file.gz <- "somaticVcfBeta-HCC1187-H-200-37-ASM-T1-N1.vcf.gz"
stopifnot(file.exists(file.gz))
file.gz.tbi <- paste(file.gz, ".tbi", sep="")
if(!(file.exists(file.gz.tbi)))
    indexTabix(file.gz, format="vcf")
start.loc <- 55000000
end.loc   <- 56000000
chr7.gr <- GRanges("7", IRanges(start.loc, end.loc))
params <- ScanVcfParam(which=chr7.gr)
vcf <- readVcf(TabixFile(file.gz), "hg19", params)
writeVcf(vcf, "chr7-sub.vcf")
bgzip("chr7-sub.vcf", overwrite=TRUE)
indexTabix("chr7-sub.vcf.gz", format="vcf")
```

This code creates the small gzipped vcf and index files used in the examples above.