VariantAnnotation and ensemblVEP

Valerie Obenchain\textsuperscript{1}
Fred Hutchinson Cancer Research Institute, Seattle, WA

December 13-14, 2012

\textsuperscript{1}vobencha@fhcrc.org
Topics

**VariantAnnotation**
- Reading and writing vcf files
- Identify gene-centric location
- New SnpMatrix functions

**ensemblVEP**
- Accessing Ensembl VEP annotation data

**VariantTools**
- Brief mention
Sample data from 1000 Genomes

```r
> library(VariantAnnotation)
> fl <- "~/Downloads/ALL.wgs.phase1_release_v3.20101123.snps_indels.sites.vcf.gz"
> hd <- scanVcfHeader(fl)
> hd

class: VCFHeader
samples(0):
meta(2): fileformat reference
fixed(1): ALT
info(22): LDAF AVGPOST ... VT SNPSOURCE
geno(3): GT DS GL
```

Read with `readVcf`

Identify desired variables by exploring the header:

```r
> head(info(hd), 3)
Dataframe with 3 rows and 3 columns

   Number       Type Description
   <character> <character> <character>
      LDAF     1  Float            MLE Allele Frequency
     AVGPOST   1  Float Average posterior probability
         RSQ   1  Float Genotype imputation quality
```

Read in a region of chromosome 6 and 3 variables from info:

```r
> chr6 <- GRanges("6", IRanges(1, 1e7))
> param <- ScanVcfParam(which=chr6, info=c("LDAF", "RSQ", "VT"))
> vcf <- readVcf(fl, "hg19", param)
```
Exploring the data

```r
> rowData(vcf)[1:3,-1]

GRanges with 3 ranges and 4 metadata columns:

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>REF</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Rle&gt;</td>
<td>&lt;IRanges&gt;</td>
<td>&lt;Rle&gt;</td>
<td>&lt;DNAStringSet&gt;</td>
</tr>
<tr>
<td>rs201634483</td>
<td>[73924, 73928]</td>
<td>*</td>
<td>AAGAG</td>
</tr>
<tr>
<td>rs189139747</td>
<td>[89919, 89919]</td>
<td>*</td>
<td>T</td>
</tr>
<tr>
<td>rs181574336</td>
<td>[89921, 89921]</td>
<td>*</td>
<td>C</td>
</tr>
</tbody>
</table>

ALT    QUAL   FILTER
<CompressedCharacterList>  <numeric>  <character>
rs201634483      A     113     PASS
rs189139747      G     100     PASS
rs181574336      T     100     PASS

---

seqlengths:
  6
  NA
```
Exploring the data

> \texttt{info(vcf)[1:4,]}

DataFrame with 4 rows and 3 columns

<table>
<thead>
<tr>
<th></th>
<th>LDAF</th>
<th>RSQ</th>
<th>VT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;numeric&gt;</td>
<td>&lt;numeric&gt;</td>
<td>&lt;character&gt;</td>
</tr>
<tr>
<td>rs201634483</td>
<td>0.1313</td>
<td>0.2643</td>
<td>INDEL</td>
</tr>
<tr>
<td>rs189139747</td>
<td>0.1311</td>
<td>0.2672</td>
<td>SNP</td>
</tr>
<tr>
<td>rs181574336</td>
<td>0.0087</td>
<td>0.1135</td>
<td>SNP</td>
</tr>
<tr>
<td>rs185142701</td>
<td>0.2027</td>
<td>0.3002</td>
<td>SNP</td>
</tr>
</tbody>
</table>

> \texttt{geno(vcf)}

SimpleList of length 3
names(3): GT DS GL
### Quality Measures: dbSNP membership

Compare quality measures between novel (i.e., not in dbSNP) and known (i.e., in dbSNP) variants. Membership in dbSNP is determined by matching against a `SNPlocs` package.

```r
> library(SNPlocs.Hsapiens.dbSNP.20101109)
> dbsnprd <- renameSeqlevels(rowData(vcf), c("6"="ch6"))
> ch6snps <- getSNPlocs("ch6")
> dbsnpchr6 <- sub("rs", ",", names(dbsnprd)) %in% ch6snps$RefSNP_id
> table(dbsnpchr6)

dbsnpchr6
FALSE TRUE
87581 64520

> ## Data frame of quality measures of interest
> metrics <- data.frame(QUAL=qual(vcf), inDbSNP=dbsnpchr6,
+ VT=info(vcf)$VT, LDAF=info(vcf)$LDAF, RSQ=info(vcf)$RSQ)
```
Quality Measures: imputation quality
Subset and `writeVcf`

- Keep variants in dbSNP:
  ```r
  > vcfQC <- vcf[dbsnpchr6]
  ```
- Write to a file for later:
  ```r
  > writeVcf(vcfQC, '~/Downloads/chr6.vcf')
  ```
Gene-centric location with `locateVariants`

```r
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
> txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
> rd <- renameSeqlevels(rowData(vcfQC), c("6"="chr6"))
> loc <- locateVariants(rd, txdb, AllVariants())
```
Gene-centric location with `locateVariants`

```r
> loc[c(3,817,819),]
GRanges with 3 ranges and 7 metadata columns:

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>LOCATION</th>
<th>QUERYID</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Rle&gt;</td>
<td>&lt;IRanges&gt;</td>
<td>&lt;Rle&gt;</td>
<td>&lt;factor&gt;</td>
<td>&lt;integer&gt;</td>
</tr>
<tr>
<td>chr6</td>
<td>[147691, 147691]</td>
<td>*</td>
<td>intergenic</td>
<td>2</td>
</tr>
<tr>
<td>chr6</td>
<td>[311645, 311645]</td>
<td>*</td>
<td>intron</td>
<td>708</td>
</tr>
<tr>
<td>chr6</td>
<td>[311938, 311938]</td>
<td>*</td>
<td>coding</td>
<td>709</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TXID</th>
<th>CDSID</th>
<th>GENEID</th>
<th>PRECEDEID</th>
<th>FOLLOWID</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;integer&gt;</td>
<td>&lt;integer&gt;</td>
<td>&lt;character&gt;</td>
<td>&lt;character&gt;</td>
<td>&lt;character&gt;</td>
</tr>
<tr>
<td>&lt;NA&gt;</td>
<td>&lt;NA&gt;</td>
<td>&lt;NA&gt;</td>
<td>56940</td>
<td>55770</td>
</tr>
<tr>
<td>23052</td>
<td>&lt;NA&gt;</td>
<td>56940</td>
<td>&lt;NA&gt;</td>
<td>&lt;NA&gt;</td>
</tr>
<tr>
<td>23051</td>
<td>72059</td>
<td>56940</td>
<td>&lt;NA&gt;</td>
<td>&lt;NA&gt;</td>
</tr>
</tbody>
</table>
```
Gene-centric location with `locateVariants`

> ## Which genes have splice site variants?
> unique(loc$GENEID[loc$LOCATION %in% "spliceSite" &
+       !is.na(loc$GENEID)])

[1] "340156" "670"    "63027"    "2162"

> ## Summarize the number of promoter variants by gene.
> promoters <- loc[loc$LOCATION %in% "promoter"]
> splt <- split(mcols(promoters)$QUERYID,
+       mcols(promoters)$GENEID)
> head(sapply(splt, function(x) length(unique(x))))

100124533 100500900 100506207 100507194 100526836 100526837
  14       17       17       12       23       14


**SnpMatrix-centric functions**

Contributed by Stephanie Gogarten:

- genotypeToSnpMatrix: convert genotype (GT) data in *VCF* class to *SnpMatrix* using probability information (GL, GP)
- probabilityToSnpMatrix: convert posterior genotype probabilities to *SnpMatrix*

Coming soon from Chris Wallace:

- SnpSummary: compute allele frequencies, HWE, MAF.
EnsemblVEP wraps the Ensembl Variant Effect Predictor (VEP) perl API. To use the package, the Ensembl VEP software must be installed in the user's path.

Default behavior returns GRanges

```r
> library(EnsemblVEP)
> fl <- '~/Downloads/chr6.vcf'
> gr <- EnsemblVEP(fl)
```
```r
> head(gr, 3)

GRanges with 3 ranges and 13 metadata columns:

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>Allele</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Rle&gt;</td>
<td>&lt;IRanges&gt;</td>
<td>&lt;Rle&gt;</td>
<td>&lt;factor&gt;</td>
</tr>
<tr>
<td>rs28546785</td>
<td>[ 89949, 89949]</td>
<td>*</td>
<td>A</td>
</tr>
<tr>
<td>rs12525498</td>
<td>[147691, 147691]</td>
<td>*</td>
<td>C</td>
</tr>
<tr>
<td>rs12525498</td>
<td>[147691, 147691]</td>
<td>*</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gene</th>
<th>Feature</th>
<th>Feature_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;factor&gt;</td>
<td>&lt;factor&gt;</td>
<td>&lt;factor&gt;</td>
</tr>
<tr>
<td>rs28546785</td>
<td>&lt;NA&gt;</td>
<td>&lt;NA&gt;</td>
</tr>
<tr>
<td>rs12525498</td>
<td>ENSG00000170590</td>
<td>ENST00000436899</td>
</tr>
<tr>
<td>rs12525498</td>
<td>ENSG00000217929</td>
<td>ENST00000406017</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Consequence</th>
<th>cDNA_position</th>
<th>CDS_position</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;factor&gt;</td>
<td>&lt;factor&gt;</td>
<td>&lt;factor&gt;</td>
</tr>
<tr>
<td>rs28546785</td>
<td>intergenic_variant</td>
<td>&lt;NA&gt;</td>
</tr>
<tr>
<td>rs12525498</td>
<td>upstream_gene_variant</td>
<td>&lt;NA&gt;</td>
</tr>
<tr>
<td>rs12525498</td>
<td>downstream_gene_variant</td>
<td>&lt;NA&gt;</td>
</tr>
</tbody>
</table>
```
Runtime options are set with a `VEPParam`:

```r
> param <- VEPParam(fl)
> param

class: VEPParam
basic(0):
  input(1): species
  database(1): host
output(0):
filterqc(0):
> t(basic(param))

  verbose quiet no_progress config    everything fork
[1,] FALSE   FALSE FALSE FALSE      Character,0 FALSE   FALSE   FALSE
```
Set the `output` option `'vcf' to TRUE to return a VCF object. Request sift and polyphen scores and predictions.

```r
> output(param) <- list(sift='b', polyphen='b', vcf=TRUE)
> vcf <- ensemblVEP(fl, param)
```

Consequences are returned as an unparsed info column.

```r
> head(info(vcf)$CSQ, 3)
CompressedCharacterList of length 3
[[1]] A|||intergenic_variant||| |
[[2]] C|ENSG00000170590|ENST00000436899|Transcript|upstream_...
[[3]] C|ENSG00000218577|ENST00000407941|Transcript|upstream_...
```
Use `parseCSQToGRanges` to parse into a `GRanges`.

> csq <- parseCSQToGRanges(vcf)

### A few SIFT levels
> levels(csq$SIFT)[5:10]
[1] "deleterious(0.04)" "deleterious(0.05)" "tolerated(0.05)"
[4] "tolerated(0.06)" "tolerated(0.07)" "tolerated(0.08)"

### A few PolyPhen levels
> levels(csq$PolyPhen)[105:110]
[1] "benign(0.399)" "benign(0.407)"
[3] "benign(0.431)" "possibly_damaging(0.441)"
[5] "possibly_damaging(0.454)" "possibly_damaging(0.469)"
Variant calling:

- Tally observed differences from the reference, exclude N calls, count reads above 13 MAPQ score
- Apply QC filters of minimum read count, unique cycle count etc.
- Call the variants using a binomial likelihood ratio test
- Perform a power test to determine if the region is a variant wildtype or no-call due to lack of coverage