VariantAnnotation and *ensemblVEP*

Valerie Obenchain[1]
Fred Hutchinson Cancer Research Institute, Seattle, WA

December 13-14, 2012

[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

Explore Header with `scanVcfHeader`

Sample data from 1000 Genomes [2]

```
> library(VariantAnnotation)
> fl <- "~/Downloads/ALL.wgs.phase1_release_v3.20101123.snps_ind
> 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
```

[2] ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/release/20110521/

Read with `readVcf`

Identify desired variables by exploring the header:

```
> 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:

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

Exploring the data

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

GRanges with 3 ranges and 4 metadata columns:
                seqnames          ranges strand |                 REF
                   <Rle>       <IRanges>  <Rle> | <DNAStringSet>
  rs201634483          6 [73924, 73928]      * |               AAGAG
  rs189139747          6 [89919, 89919]      * |                  T
  rs181574336          6 [89921, 89921]      * |                  C
                                      ALT      QUAL      FILTER
                <CompressedCharacterList> <numeric> <character>
  rs201634483                           A       113        PASS
  rs189139747                           G       100        PASS
  rs181574336                           T       100        PASS
  ---
  seqlengths:
    6
   NA
```

Exploring the data

```
> info(vcf)[1:4,]

DataFrame with 4 rows and 3 columns
                  LDAF       RSQ          VT
             <numeric> <numeric> <character>
rs201634483     0.1313    0.2643       INDEL
rs189139747     0.1311    0.2672         SNP
rs181574336     0.0087    0.1135         SNP
rs185142701     0.2027    0.3002         SNP

> geno(vcf)

SimpleList of length 3
names(3): GT DS GL
```

Quality Measures: dbSNP membership

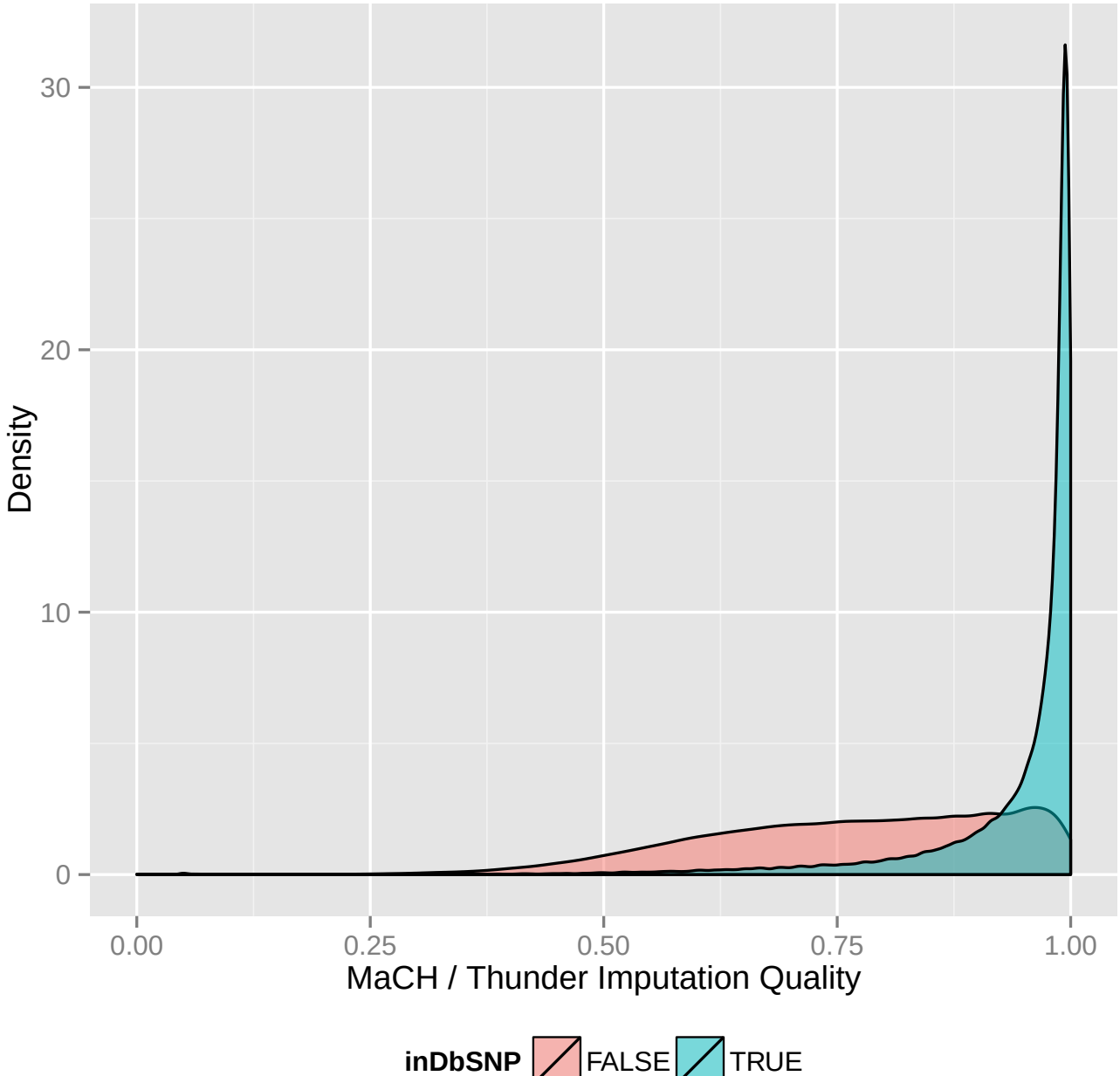
Compare quality measures bewteen novel (i.e., not in dbSNP) and known (i.e., in dbSNP) variants. Membership in dbSNP is determined by matching against a *SNPlocs* package.

```
> 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:

  ```
  > vcfQC <- vcf[dbsnpchr6]
  ```

- Write to a file for later:

  ```
  > writeVcf(vcfQC, '~/Downloads/chr6.vcf')
  ```

Gene-centric location with `locateVariants`

```
> 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`

```
> loc[c(3,817,819),]
GRanges with 3 ranges and 7 metadata columns:
      seqnames            ranges strand |   LOCATION    QUERYID
         <Rle>         <IRanges>  <Rle> |   <factor>  <integer>
          chr6 [147691, 147691]      * | intergenic          2
          chr6 [311645, 311645]      * |     intron        708
          chr6 [311938, 311938]      * |     coding        709

          TXID     CDSID      GENEID   PRECEDEID    FOLLOWID
     <integer> <integer> <character> <character> <character>
          <NA>      <NA>        <NA>       56940       55770
         23052      <NA>       56940        <NA>        <NA>
         23051     72059       56940        <NA>        <NA>
```

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

ensemblVEP wraps the Ensembl Variant Effect Predictor (VEP) perl API [3]. To use the package, the Ensembl VEP software must be installed in the user's path.

Default behavior returns GRanges

```
> library(ensemblVEP)
> fl <- '~/Downloads/chr6.vcf'
> gr <- ensemblVEP(fl)
```

[3] EnsemblVEP

ensemblVEP

```
> head(gr,3)
GRanges with 3 ranges and 13 metadata columns:
             seqnames            ranges strand |   Allele
                <Rle>         <IRanges>  <Rle> | <factor>
  rs28546785        6 [ 89949,  89949]      * |        A
  rs12525498        6 [147691, 147691]      * |        C
  rs12525498        6 [147691, 147691]      * |        C

                    Gene         Feature  Feature_type
                <factor>        <factor>      <factor>
  rs28546785        <NA>            <NA>          <NA>
  rs12525498 ENSG00000170590 ENST00000436899    Transcript
  rs12525498 ENSG00000217929 ENST00000406017    Transcript

                      Consequence cDNA_position  CDS_position
                         <factor>      <factor>      <factor>
  rs28546785      intergenic_variant          <NA>          <NA>
  rs12525498   upstream_gene_variant          <NA>          <NA>
  rs12525498 downstream_gene_variant          <NA>          <NA>
```

Runtime options are set with a *VEPParam*:

```
> 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       Character,0 FALSE      FALSE
```

ensemblVEP

Set the `output` option 'vcf' to TRUE to return a *VCF* object. Request sift and polyphen scores and predictions.

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

Consequences are returned as an unparsed info column.

```
> 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_...
```

ensemblVEP

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)"
```

VariantTools (Michael Lawrence)

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