Exploring the Complete Genomics Diversity panel

VJ Carey

October 15, 2015

Contents

1 Introduction

Complete Genomics Inc. distributes a collection of data on deeply sequenced genomes (from Coriell cell lines) from 11 different human populations.

```
> library(cgdv17)
> data(popvec)
> popvec[1:5]

NA19700 NA19020 NA19701 NA19025 NA19703
  "ASW"   "LWK"   "ASW"   "LWK"   "ASW"

> table(popvec)

popvec
ASW CEU CHB GIH JPT LWK MKK MXL TSI YRI
  5   5   4   4   4   4   4   5   4   7
```

The data are distributed with many details; VJC obtained the masterVar TSV files from the Complete Genomics ftp2 site, converted these to VCF 4.0 in Oct. 2011, using a tool noted at

```
http://community.completegenomics.com/tools/m/cgtools/219.aspx
```

The conversion tool used was released with various caveats. Perhaps the whole conversion should be redone with official tools.

The purpose of this note is to explore some basic structural features of the data, so that relevant genetic structures can be identified for analytic programming and interpretation. We focus on variant calls on chromosome 17.

Formal restrictions on publications related to these data are as follows.

```
1. The Coriell and ATCC Repository number(s) of the
cell line(s) or the DNA sample(s) must be cited in
publication or presentations that are based on the
use of these materials.
2. You must reference our Science paper (R. Drmanac,
et. al. Science 327(5961), 78. [DOI: 10.1126/
science.1181498])
3. You must provide the version number of the
Complete Genomics assembly software with which
the data was generated. This can be found in the
header of the summary.tsv file (\# Software_Version).
```

2 Contents of a VCF header

There is a lot of redundancy among the headers for the 46 files, so one was isolated for distribution.

```
> data(h1)
> h1

$NA21767_17.vcf.gz
List of length 3
names(3): Reference Sample Header

> h1[[1]]$Sample

[1] "GS21767-1100-37-ASM"

> h1[[1]]$Header$META

DataFrame with 5 rows and 1 column
                     Value
                 <character>
fileformat           VCFv4.1
fileDate            20111102
source     masterVar2VCFv40
reference    build37.fa.bz2
phasing              partial

> h1[[1]]$Header$INFO

DataFrame with 3 rows and 3 columns
        Number        Type                 Description
   <character> <character>                 <character>
NS           1     Integer Number of Samples With Data
DP           1     Integer                 Total Depth
DB           0        Flag dbSNP membership, build 131

> h1[[1]]$Header$FORMAT

DataFrame with 12 rows and 3 columns
            Number        Type                        Description
       <character> <character>                        <character>
GT               1      String                           Genotype
GQ               1     Integer                  Genotype Quality
DP               1     Integer                        Read Depth
```

HDP	2	Integer	Haplotype Read Depth
HQ	2	Integer	Haplotype Quality
...	...	...	...
mRNA	.	String	Overlaping mRNA
rmsk	.	String	Overlaping Repeats
segDup	.	String	Overlaping segmentation duplication
rCov	1	Float	relative Coverage
cPd	1	String	called Ploidy(level)

3 Variant calls for chromosome 17

3.1 Recording structural variation for an individual

We created a provisional container for the call data on chromosome 17. Tabix facilities were used to filter and index the data from the full VCF to all of chromosome 17.

At present it is not clear how to model a collection of deeply sequenced chromosomes. I have used VariantAnnotation:::readVcf, which must be applied separately for each individual, given the Complete Genomics distribution. The focus is on structural information in the rowRanges component of the VCF object returned by `readVcf`, which is a GRanges instance. From the elementMetadata I removed FILTER and added `geno()$GT` information. This gives us information to specific variants and phase for some variants, depending on the string content of the GT information.

The `getRVS` function will collect file references for the serialized GRanges.

```
> rv = getRVS("cgdv17")
> rv

raggedVariantSet instance with 46 elements.
some sampleNames:  NA06985 NA06994 ... NA21737 NA21767
```

Data on one individual can be extracted using `getrd()`. We will confine attention to variants with quality score in the top quartile of its distribution for this individual.

```
> R85 = getrd(rv, "NA06985")
> length(R85)

[1] 174744

> summary(elementMetadata(R85)$QUAL)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       0      98     117     166    1714

> kp = which(elementMetadata(R85)$QUAL >= 166)
> R85hiq = R85[kp]
```

A small excerpt gives us a sense of the sorts of variation to be encountered:

```
> elementMetadata(R85hiq)[11:20,]

DataFrame with 10 rows and 5 columns
              REF             ALT      QUAL        geno     depth
   <DNAStringSet> <CharacterList> <numeric> <character> <integer>
1               T               C       309         1|0        66
```

```
2               T             C       417        1/0        63
3               T             C       187        1/1        61
4               A             G       187        1/0        48
5             CTG       CCA,CCG       203        1/2        46
6               A             G       380        1/1        26
7               C             T       272        1/0        45
8               G             A       324        1/0        45
9            TAGT          TGGG       237        1/0        62
10              T             G       396        1|0        50
```

```
> refs = elementMetadata(R85hiq)$REF
> alts = elementMetadata(R85hiq)$ALT
> genos = elementMetadata(R85hiq)$geno
> table(nchar(refs))
```

```
    1     2     3     4     5     6     7     8     9    10    11    12    13
41848   646   645   347   193    63    33    21    25    17    16    18    15
   14    15    16    17    18    19    20    21    22    23    24    25    26
    9    10     7     5     3     1     2     2     1     2     2     2     1
   27    29    30    31    32    34   135
    2     1     2     1     2     1     1
```

```
> alts[grep(",",unlist(alts))]
```

```
CharacterList of length 120
[[1]] CCA,CCG
[[2]] CCC,CCG
[[3]] A,C
[[4]] ACA,ATG
[[5]] CTCG,CNCN
[[6]] GCA,GTG
[[7]] CGCA,CGCG
[[8]] C,G
[[9]] TGG,TCA
[[10]] CAAG,CGAG
...
<110 more elements>
```

Summary: references are recorded as DNAStrings, alternatives are compressed character strings with commas, and the phasing of the individual-level calls can be derived by parsing the `geno` component.

3.2 Isolating variants in the vicinity of a gene, for an individual

We are interested in gene ORMDL3. We will use the hg19 transcriptDb to obtain the locations and tabulate higher quality variants observed 100kb up and downstream of the transcript.

```
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
> tx19 = TxDb.Hsapiens.UCSC.hg19.knownGene
> library(org.Hs.eg.db)
> get("ORMDL3", revmap(org.Hs.egSYMBOL))

[1] "94103"

> ortx = transcriptsBy(tx19, "gene")$"94103"
> seqlevels(R85hiq) = "chr17"
> aro = subsetByOverlaps(R85hiq, ortx+100000)
> table(elementMetadata(aro)$geno)

0/. 0|. 1/0 1/1 1|0 1|2
  2   1 121  10  65   1

> alts = unlist(elementMetadata(aro)$ALT)
> alts[nchar(alts)>1]

 [1] "AGTG"    "CC"      "GG"      "CTGC"    "AT"      "AGG"     "AA"
 [8] "ACAC"    "GG"      "ACA,ACG" "GAA"
```

There are deletion and insertion events, but I don't see any simple way of isolating and counting them at the moment. Some code will be added to address this.

We can use VariantAnnotation to obtain structural contexts. It takes over a minute to use locateVariants, so I just show the code and results here for now.

```
mycache = new.env(hash=TRUE)
lvaro = locateVariants(aro, tx19, cache=mycache)
lvaro[1:4,]
table(lvaro$loca,sapply((lvaro$geneID), function(x)strsplit(x, ",")[[1]][1]))
DataFrame with 4 rows and 5 columns
    queryID location      txID                    geneID     cdsID
  <integer> <factor> <integer> <CompressedCharacterList> <integer>
1         1   intron     60959                     22806    189661
2         1   intron     60960                     22806    189661
3         1   intron     60961                     22806    189661
4         1   intron     60962                     22806    189661
```

```
           124626 1440 22806 284110 2886 55876 5709 94103 9862
  3'UTR         0    0     0      0    0     0    0     0   20
  5'UTR         0    0     0      1    0     8    0     0   10
  coding        0    0     0      6    0     0    0     0   10
  intergenic   34    1    10      4    1     2   25    10    0
  intron       28    0   105     42    0   138   99     6    0
```

We see that this search for variants near ORMDL3 identifies variants affecting other nearby genes.

4 Filtering and analyzing variants on multiple individuals

The analysis of a ragged variant set requires infrastructure. We will illustrate with a focused analysis of variants in the vicinity of ORMDL3. We have used the GGdata and hmyriB36 packages to collect expression data on 12 individuals in the diversity cohort, in the CY17 smlSet instance. This includes expression on all genes on chr17, and the HapMap phase 2 genotypes as well.

```
> suppressPackageStartupMessages(library(GGtools))
> data(CY17)
> CY17
```

```
SnpMatrix-based genotype set:
number of samples:  12
number of chromosomes present:  1
annotation: illuminaHumanv1.db
Expression data dims: 1291 x 12
Total number of SNP: 83889
Phenodata: An object of class 'AnnotatedDataFrame'
  rowNames: NA06985 NA06994 ... NA19129 (12 total)
  varLabels: mothid fathid isFounder male
  varMetadata: labelDescription
```

```
> sn = sampleNames(CY17)
```

4.1 Sample filtering

The ragged variant set can be filtered to these individuals.

```
> rv17 = rv[, sn]
> rv17
```

```
raggedVariantSet instance with 12 elements.
some sampleNames:  NA06985 NA06994 ... NA18517 NA19129
```

4.2 Counting variants in a specified region, with quality filtering

The variant counting function takes two key parameters in addition to the variant set: a region within which to count, and a lower bound on call quality for retained variants. A third additional parameter tells how to iterate over samples with an lapply-like function.

Since ORMDL3 is on the minus strand, the upstream region is to the right. We will create a region from start site to 50k upstream.

```
> if (length(ortx)>1) ortx = ortx[2]
> ortss = end(ortx)
> ortup50 = GRanges("chr17", IRanges(ortss, width=50000))
> cv50k = countVariants(rv17, ortup50, 160, lapply )

> cv50k

NA06985 NA06994 NA07357 NA10851 NA12004 NA18501 NA18502 NA18504 NA18505 NA18508
     65       0      51      10       9      26      72      68      56      56
NA18517 NA19129
     81      58
```

We see that the second sample seems to have a quality problem. We will now drop it from both the expression and variant structures.

```
> if (length(sampleNames(rv17))==12) rv17 = rv17[,-2]
> if (length(sampleNames(CY17))==12) CY17 = CY17[,-2]
> #redo

> cv50k = countVariants(rv17, ortup50, 160, lapply )
```

We can acquire the full data on variants in the region under the quality constraint using variantGRanges.

```
> vv50k = variantGRanges( rv17, ortup50, 160, lapply )

> vv50k[[1]][1:5]

GRanges object with 5 ranges and 5 metadata columns:
             seqnames               ranges strand |             REF
                <Rle>            <IRanges>  <Rle> | <DNAStringSet>
  rs12946393       17 [38087429, 38087429]      * |               T
  rs35557848       17 [38087439, 38087439]      * |               C
  rs56199421       17 [38090808, 38090808]      * |               C
   rs7207600       17 [38091660, 38091660]      * |               G
   rs6503525       17 [38095174, 38095174]      * |               G
```

```
                      ALT      QUAL        geno     depth
            <CharacterList> <numeric> <character> <integer>
  rs12946393               G       339         1/1        70
  rs35557848               T       339         1/0        56
  rs56199421               T       270         1/0        48
   rs7207600               A       470         1/0        66
   rs6503525               C       440         1/0        69
  -------
  seqinfo: 1 sequence from hg19 genome; no seqlengths
```

```
> sapply(vv50k,length)

NA06985 NA07357 NA10851 NA12004 NA18501 NA18502 NA18504 NA18505 NA18508 NA18517
     65      51      10       9      26      72      68      56      56      81
NA19129
     58
```

As a naive hint of a connection of "variant burden" with ORMDL3 expression, consider the following display.

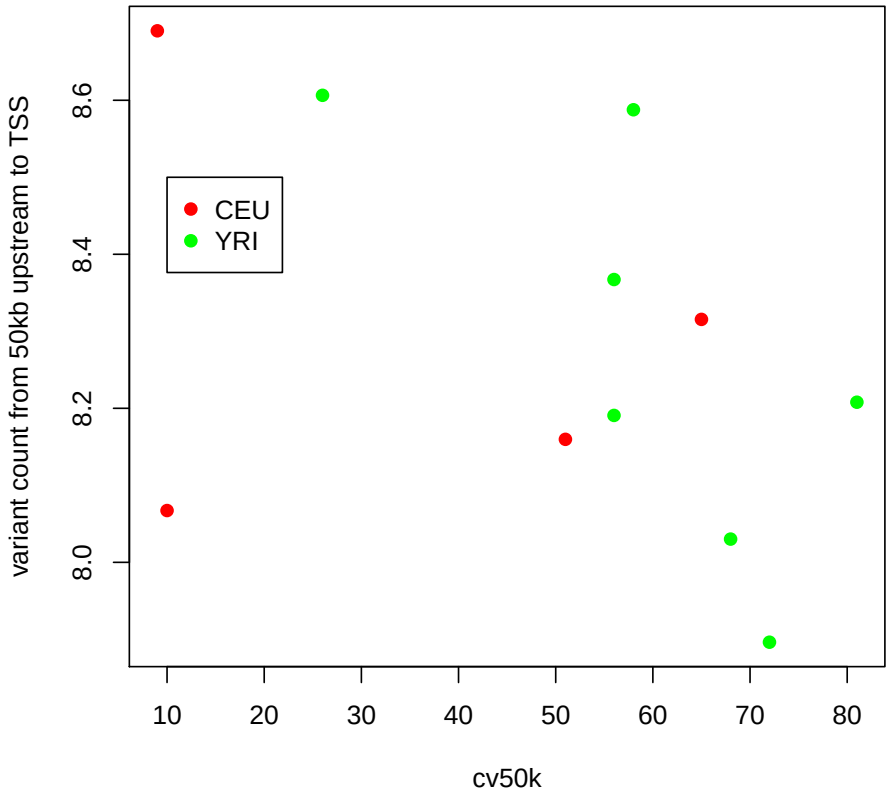
```
> ORMDL3ex = as.numeric(exprs(CY17[genesym("ORMDL3"),]))
> ygr = ifelse((1:11)<=4, "red", "green")
> plot(ORMDL3ex~cv50k, col=ygr, pch=19,
+       ylab="variant count from 50kb upstream to TSS")
> legend(10, 8.5, pch=19, col=c("red", "green"), legend=c("CEU", "YRI"))
> summary(lm(ORMDL3ex~cv50k*factor(ygr)))

Call:
lm(formula = ORMDL3ex ~ cv50k * factor(ygr))

Residuals:
     Min       1Q   Median       3Q      Max
-0.29969 -0.13281 -0.02538  0.12717  0.32077

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)          8.912418   0.355770  25.051 4.12e-08 ***
cv50k               -0.010792   0.005763  -1.873    0.103
factor(ygr)red      -0.520704   0.412601  -1.262    0.247
cv50k:factor(ygr)red  0.008317   0.007625   1.091    0.311
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1
```

```
Residual standard error: 0.2472 on 7 degrees of freedom
Multiple R-squared:  0.3528,        Adjusted R-squared:  0.07538
F-statistic: 1.272 on 3 and 7 DF,  p-value: 0.3557
```



4.3 Enumerating variants by structural context

Now we focus on variants in the ORMDL3 coding region.

```
> library(parallel)
> options(mc.cores=max(c(2, parallel::detectCores()-2)))
> vv = variantGRanges( rv17, ortx, 160, mclapply )
> vvv = lapply(vv, function(x) renameSeqlevels(x, c("17"="chr17")))
> mycache = new.env(hash=TRUE)
> locs = lapply(vvv, function(x) {
+   locateVariants(x, tx19, CodingVariants(),cache=mycache)
+ })
```

Further work: illustrate the predictCoding behavior, streamline the catalog of variants relevant to a given gene over the 46 individuals. Relate to population membership, and, where available, to expression variation.