

# Exploring data from the 1000 Genomes project in Bioconductor's *ind1KG* package

VJ Carey

April 4, 2013

## Contents

<b>1</b>	<b>Overview</b>	<b>1</b>
<b>2</b>	<b>External data acquisition</b>	<b>2</b>
2.1	Manual extraction of a multi-Mb chunk . . . . .	2
2.2	Programmatic extraction of annotated regions . . . . .	5
<b>3</b>	<b>Exploring a samtools pileup</b>	<b>6</b>
<b>4</b>	<b>Checking samtools-based calls against other calls</b>	<b>8</b>
4.1	HapMap Phase II calls . . . . .	8
4.2	Affy SNP 6.0 chip calls . . . . .	9
<b>5</b>	<b>Relating possibly novel variants to existing annotation</b>	<b>10</b>
5.1	Browser-based visualization . . . . .	10
5.2	Browser-based annotation extraction and comparison . . . . .	12
5.3	Exercises . . . . .	17

## 1 Overview

In this document we will look at high-coverage NGS data obtained on NA19240, because we have the HapMap phase II genotypes (4 mm SNP) for this individual in GGtools/hmyriB36, and we have an affy 6.0 SNP CEL file for this individual (and her cohort) as well.

There are three main objectives discussed in this document.

- We describe how data published in the 1000 genomes (1KG) project can be imported for investigations using R. This involves the use of the *Rsamtools* package. We provide serialized instances of various relevant data elements so that large objects distributed from the project need not be redistributed for these illustrations.

- We describe how information on variants can be related to existing annotation using *rtracklayer* to check for events in regulatory regions, for example.
- We discuss how information in the samtools 'pileup' format can be checked from a statistical perspective to explore how 'known' variants in the sample compare to the putatively newly discovered variants.

The reads examined in the document are all from the Illumina sequencing platform; additional work is sketched facilitating comparison with (released) read libraries based on 454 or ABI platforms.

## 2 External data acquisition

### 2.1 Manual extraction of a multi-Mb chunk

We will focus on this individual's chromosome 6. We acquired

```
NA19240.chrom6.SLX.maq.SRP000032.2009_07.bam
```

and the associated bai and bas files from

```
ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/pilot_data/data/
  NA19240/alignment/NA19240.chrom6.SLX.maq.SRP000032.2009_07.bam.bas
```

Note that it is possible to work with these files remotely in R, without moving them to the local machine, thanks to the remote access facilities built in to samtools and exposed in R.

We use

```
samtools view \
  NA19240.chrom6.SLX.maq.SRP000032.2009_07.bam |head -3000000 > na19240_3M.sam
```

to obtain a parsable text file, presumably of 3 million reads that aligned, using MAQ, nearest the 5' end of the p arm of chr6. This is because we expect the bam file to be sorted. We picked the number 3 million out of thin air.

This sam format file can be converted to bam format using the samtools import facility. We took chromosome 6 reference sequence from

```
ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/technical/
  reference/human_b36_female.fa.gz
```

and indexed it and used

```
samtools import femchr6.fa.fai na19240_3M.sam na19240_3M.bam
```

to generate the bam file.

We imported this into R using Bioconductor's *Rsamtools* with a straight application of `scanBam`. The result is saved in the package as `n240`.

```
> library(ind1KG)
> if (!exists("n240"))
+   data(n240)
```

This is a list of lists, and we check on the contents of the elements as follows:

```
> names(n240[[1]])

[1] "qname" "flag" "rname" "strand" "pos" "qwidth" "mapq" "cigar"
[9] "mrnm" "mpos" "isize" "seq" "qual"
```

We check the classes:

```
> sapply(n240[[1]], class)

      qname      flag      rname      strand      pos
"character" "integer" "factor" "factor" "integer"
      qwidth      mapq      cigar      mrnm      mpos
"integer" "integer" "character" "factor" "integer"
      isize      seq      qual
"integer" "DNAStringSet" "PhredQuality"
```

We get a small number of exemplars from each element:

```
> lapply(n240[[1]], head, 5)

$qname
[1] "EAS254_13:7:88:1639:15041" "EAS139_43:2:31:1128:9551"
[3] "EAS254_13:8:68:520:6861" "BGI-FC20AHFAAXX_6_26_477:352"
[5] "EAS139_43:6:71:1575:10961"

$flag
[1] 35 35 35 16 35

$rname
[1] 6 6 6 6 6
Levels: 6

$strand
[1] + + + - +
Levels: + - *
```

```

$pos
[1] 5001 5002 5004 5004 5005

$qwidth
[1] 51 51 51 36 51

$mapq
[1] 0 0 0 0 0

$ cigar
[1] "51M" "51M" "51M" "36M" "51M"

$mrnm
[1] 6 6 6 <NA> 6
Levels: 6

```

```

$mpos
[1] 5163 5203 5170 NA 5156

```

```

$ isize
[1] 214 253 218 NA 203

```

```

$seq
A DNASTringSet instance of length 5
width seq
[1] 51 GATCTTATATAACTGTGAGATTAATCTCAGATAATGACACAAAATATAGTG
[2] 51 ATCTTATATAACTGTGAGATTAATCTCAGATAATGACACAAAATATAGTGA
[3] 51 CTTATATAACTGTGAGATTAATCTCAGATAATGACACAAAATATAGTGAAG
[4] 36 TGTCATTATCTGAGATTAATCTCACAGTTATATAAG
[5] 51 GTATATAACTGTGAGATTAATCTCAGATAATGACACAAAATATAGTGAAGT

```

```

$qual
A PhredQuality instance of length 5
width seq
[1] 51 C<@?AB?A@B@A?C@B@?@?BC@BB@C@A??B@AB<??@?AABAA?@>=?
[2] 51 B???A@A@A@A?;??@?@>ACABAAC@;@=BABB@=@A@?A?>A?A?<@9>
[3] 51 B@@=A@A@@@B@A@=@?AB<AB@B@@@>A@AB@>@=?@A:AB@A??@>;?>
[4] 36 C=:>A=>>=A=8?7>@?@=: @?>8;>9?8>9><60
[5] 51 +?>@?A?A?B@A????BC@ABACA???B@BB@=?A@ABAAB@B@?A?=?=A

```

We can use R at this point to do matching to reference and filtering and so forth, but we will only do this in a *post mortem* fashion, as it seems to make more sense to use

samtools directly to do, for example, SNP calling.

## 2.2 Programmatic extraction of annotated regions

(This code segment suggested by Martin Morgan.)

We can use the *GenomicFeatures* package to obtain intervals defining various genomic elements.

```
> library(GenomicFeatures)
> library(TxDb.Hsapiens.UCSC.hg18.knownGene)
> txdb = TxDb.Hsapiens.UCSC.hg18.knownGene
> txdb
```

TranscriptDb object:

```
| Db type: TranscriptDb
| Supporting package: GenomicFeatures
| Data source: UCSC
| Genome: hg18
| Organism: Homo sapiens
| UCSC Table: knownGene
| Resource URL: http://genome.ucsc.edu/
| Type of Gene ID: Entrez Gene ID
| Full dataset: yes
| miRBase build ID: NA
| transcript_nrow: 66803
| exon_nrow: 266688
| cds_nrow: 221991
| Db created by: GenomicFeatures package from Bioconductor
| Creation time: 2013-03-08 09:41:32 -0800 (Fri, 08 Mar 2013)
| GenomicFeatures version at creation time: 1.11.14
| RSQLite version at creation time: 0.11.2
| DBSCHEMAVERSION: 1.0
```

The `transcripts` method will obtain ranges of transcripts with constraints.

```
> tx6 <- transcripts(txdb, vals = list(tx_chrom = "chr6"))
> chr6a <- head(unique(sort(ranges(tx6))), 50)
> chr6a
```

IRanges of length 50

```
      start      end width
[1]  237101  296355 59255
[2]  249628  296353 46726
```

```
[3] 336752 347637 10886
[4] 336752 356443 19692
[5] 342056 347637 5582
... ..
[46] 3064041 3098281 34241
[47] 3098901 3101481 2581
[48] 3098901 3102782 3882
[49] 3128053 3141000 12948
[50] 3169514 3172870 3357
```

With a local BAM file, the following counting procedure is quick. Note that `f1` could be a URL beginning

```
ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/pilot_data/data/NA19240/alignment...
```

and the computations would work, but completion speed would depend upon server load and network throughput.

```
> library(Rsamtools)
> p1 <- ScanBamParam(which=RangesList(`6`=chr6a))
> f1 <- "/mnt/data/stvjc/1000GENOMES/NA19240.chrom6.SLX.maq.SRP000032.2009_07.bam"
> unix.time(cnt <- countBam(f1, param=p1))
> sum(cnt$records) # 2103439
```

The following scan will yield a list with read and quality information on the 50 transcript regions requested in `chr6a` allocated to 50 list elements.

```
> res <- scanBam(f1, param=p1)
> length(res)
> names(res[[1]])
```

### 3 Exploring a samtools pileup

The pileup output derived from the 3 million reads is a 17GB (sic) text file derived as follows:

```
samtools pileup -cf femchr6.fa \
  NA19240.chrom6.SLX.maq.SRP000032.2009_07.bam > na19240F.pup
```

First 10 lines:

```
6      5001      G      G      4      0      0      1      ^!.      C
6      5002      A      A      7      0      0      2      .^!.      <B
6      5003      T      T      7      0      0      2      ..      @?
6      5004      C      C      14     0      0      4      ..^!.^!,      ??B0
```



## 4 Checking samtools-based calls against other calls

### 4.1 HapMap Phase II calls

We include information from the phase II HapMap calls for NA19240. We have a *snp.matrix* instance with the full genotyping for chromosome 6 and location information as supplied by HapMap.

```
> library(chopsticks)
> data(yri240_6)
> names(yri240_6)
```

```
[1] "hm2" "supp"
```

The following code gets all relevant HapMap calls in a generic format and isolates the SNP at which NA19240 is heterozygous.

```
> snps <- as(yri240_6[[1]], "character")
> hets <- which(snps == "A/B")
> rshet <- colnames(snps)[hets]
> smet <- yri240_6[[2]]
> smethet <- smet[hets,]
> head(smethet, 5)
```

	dbSNP	Palleles	Assignment	Chromosome	Position	Strand
rs12192290	A/T	A/T	chr6	95272	+	
rs1929630	A/C	A/C	chr6	99536	+	
rs719065	A/G	A/G	chr6	110632	+	
rs12209455	A/G	A/G	chr6	112510	+	
rs6909153	A/G	A/G	chr6	119769	+	

We also have the full pileup information for the first 500K loci computed by samtools pileup.

```
> data(pup240_500k)
> head(pup240_500k, 2)
```

```
      V2 V3 V4 V5  V9
1 5001  G  G  4  ^!.
2 5002  A  A  7  .^!.
```

This include some duplicated locations, which we remove.

```
> pup240_500ku <- pup240_500k[ !duplicated(pup240_500k[,1]),]
```

The pileups at which HapMap says our subject is heterozygous are



```
> hpup <- pup240_500ku[ pup240_500ku[,1] %in% smethet["Position"], ]
```

Are there any loci (in this very small region of chromosome 6) that HapMap says are heterozygous, but that are found to be homozygous by sequencing?

```
> hom <- hpup[ hpup[,2] == hpup[,3], ]
> hom
```

```

      V2 V3 V4 V5
158570 163386 C C 15
169883 174667 T T 89
219091 223773 C C 4
```

```

158570
169883
219091
V9
,,,,,,,t,,,,t..T.TtTa...^~^~T
,,,,,$.$,,,,,,C,,,,CAcc,,C,,,C,.C,,,,Cc,..^<^?,
,.t,,,,,$.Gt,,tt,,t,,,,,t,,,,,g.A,,,tt,,,,,T,t,,,,,Tt,,,,.gt.^g
```

```
> smethet[ smethet["Position"] %in% hom[,1], ]
```

	dbSNPalleles	Assignment	Chromosome	Position	Strand
rs1418703	C/T	C/T	chr6	163386	+
rs6915606	C/T	C/T	chr6	174667	+
rs815571	C/T	C/T	chr6	223773	+

## 4.2 Affy SNP 6.0 chip calls

We ran `cr1mm` to genotype all 90 YRI samples from 6.0 chips distributed by Affymetrix. The data for NA19240 chromosome 6 are available in the *ind1KG* package:

```
> data(gw6c6.snp240)
> head(gw6c6.snp240, 4)
```

	man_fsetid	dbsnp_rs_id	physical_pos	strand	allele_a	allele_b	call240
1	SNP_A-1984753	rs719065	110632	1	C	T	2
2	SNP_A-1984758	rs6927090	197145	1	A	C	3
3	SNP_A-1984759	rs815583	230695	1	C	T	2
4	SNP_A-1984760	rs1514346	334630	0	A	G	2

The heterozygous loci are

```
> hloc6 <- gw6c6.snp240[ gw6c6.snp240$call240 == 2, "physical_pos" ]
```

Let's see if the sequencing leads to the same decisions (at least with regard to heterozygous vs. homozygous):

```
> inds <- which(pup240_500k[,1] %in% hloc6)
> table(pup240_500k[inds, 3])
```

```
K M R S Y
6 2 17 2 14
```

For the loci called homozygous by crlmm, we have:

```
> oloc6 <- gw6c6.snp240[ gw6c6.snp240$call240 %in% c(1,3), "physical_pos" ]
> oinds <- which(pup240_500k[,1] %in% oloc6)
> table(pup240_500k[oinds, 3])
```

```
A C G T Y
21 49 38 22 1
```

## 5 Relating possibly novel variants to existing annotation

### 5.1 Browser-based visualization

There are many ways to use Bioconductor annotation resources to learn about contexts of variants. However, the UCSC genome browser is probably the most efficient place to start. We can convert our vector of locations of apparently new variants to a browser track as follows; this code is not executed in vignette construction, but you may run it manually if suitably networked.

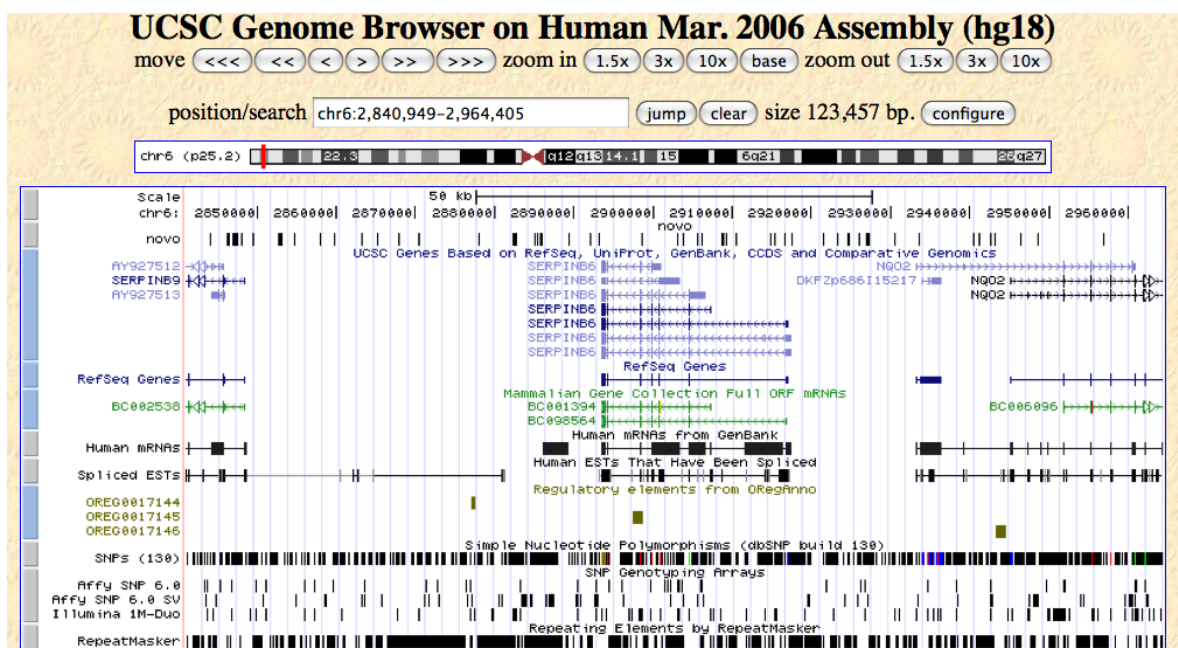
```
> library(IRanges)
> nloc <- nov$loc
> nrd <- RangedData( IRanges(nloc, nloc) )
> names(nrd) <- "chr6"
> library(rtracklayer)
> br <- browserSession("UCSC")
> br[["novo"]] <- nrd
> v1 <- browserView(br, GenomicRanges(1, 1e7, "chr6"))
```

This arranges the browser so that the custom track at the top of the display, 'novo', gives the locations of the possible novel variants.

Overall, we see that these novel variants occur regularly across the 10MB.



We can zoom in to the region around a given gene, here SERPINEB6.



## 5.2 Browser-based annotation extraction and comparison

Because the *rtracklayer* package gives a bidirectional interface, it is possible to programmatically check for coincidence of variant locations, gene regions, or regulatory elements, for example.

We can learn the names of all available tracks for the current session via code like the following.

```
> tn <- trackNames(br)
> grep("Genes", names(tn), value=TRUE) # many different gene sets
> tn["UCSC Genes"] # resolve indirection
```

For example, to get the symbols for genes in the 10 million bp excerpt that we are working with, we can use

```
> rsg <- track(br, "refGene")
> rsgdf <- as.data.frame(rsg)
```

This data frame has been serialized with the *ind1KG* package.

```
> data(rsgdf)
> names(rsgdf)

[1] "space"      "start"      "end"        "width"      "name"
[6] "score"     "strand"     "thickStart" "thickEnd"   "color"
[11] "blockCount" "blockSizes" "blockStarts"
```

```
> rsgdf[1:3,1:7]
```

```
  space  start    end  width    name  score  strand
1  chr6 237101 296355 59255 NM_020185    0      +
2  chr6 336752 356443 19692 NM_002460    0      +
3  chr6 430138 638109 207972 NM_018303    0      -
```

We see that the 'names' here are RefSeq identifiers. We may be able to resolve them to Entrez Gene Ids, and thence to symbols, as follows:

```
> library(org.Hs.eg.db)
> rsgn <- as.character(rsgdf$name)
> eid <- mget(rsgn, revmap(org.Hs.egREFSEQ), ifnotfound=NA)
> eid <- na.omit(unlist(eid))
> sym <- mget(eid, org.Hs.egSYMBOL, ifnotfound=NA)
> head(unlist(sym), 10)
```

```
      56940      3662      55770      135458      285768      285768
"DUSP22"  "IRF4"    "EXOC2"  "HUS1B" "LOC285768" "LOC285768"
      94234      2295      2296      2762
"FOXQ1"  "FOXF2"  "FOXC1"  "GMDS"
```

These names are consistent with what we see on the browser displays shown above.

We can use the *IRanges* infrastructure to check for intersection between novel variant locations and gene occupancy regions.

```
> nloc <- nov$loc # this one is evaluated
> nranges <- IRanges(nloc, nloc)
> granges <- IRanges(rsgdf$start, rsgdf$end) # no guarantee of annotation
> length(nranges)
```

```
[1] 4075
```

```
> length(granges)
```

```
[1] 73
```

```
> sum(nranges %in% granges)
```

```
[1] 0
```

```
> head(match(nranges,granges), 200)
```

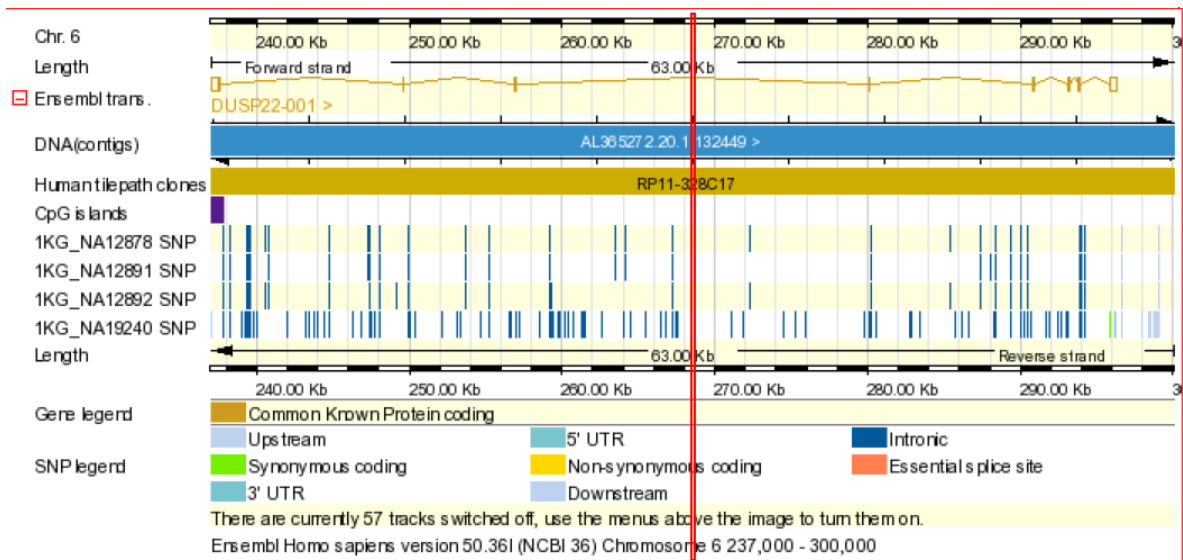
[1] NA  
[26] NA  
[51] NA  
[76] NA  
[101] NA  
[126] NA  
[151] NA  
[176] NA

We can see that there is a batch of variants present in the first gene, and this is confirmed

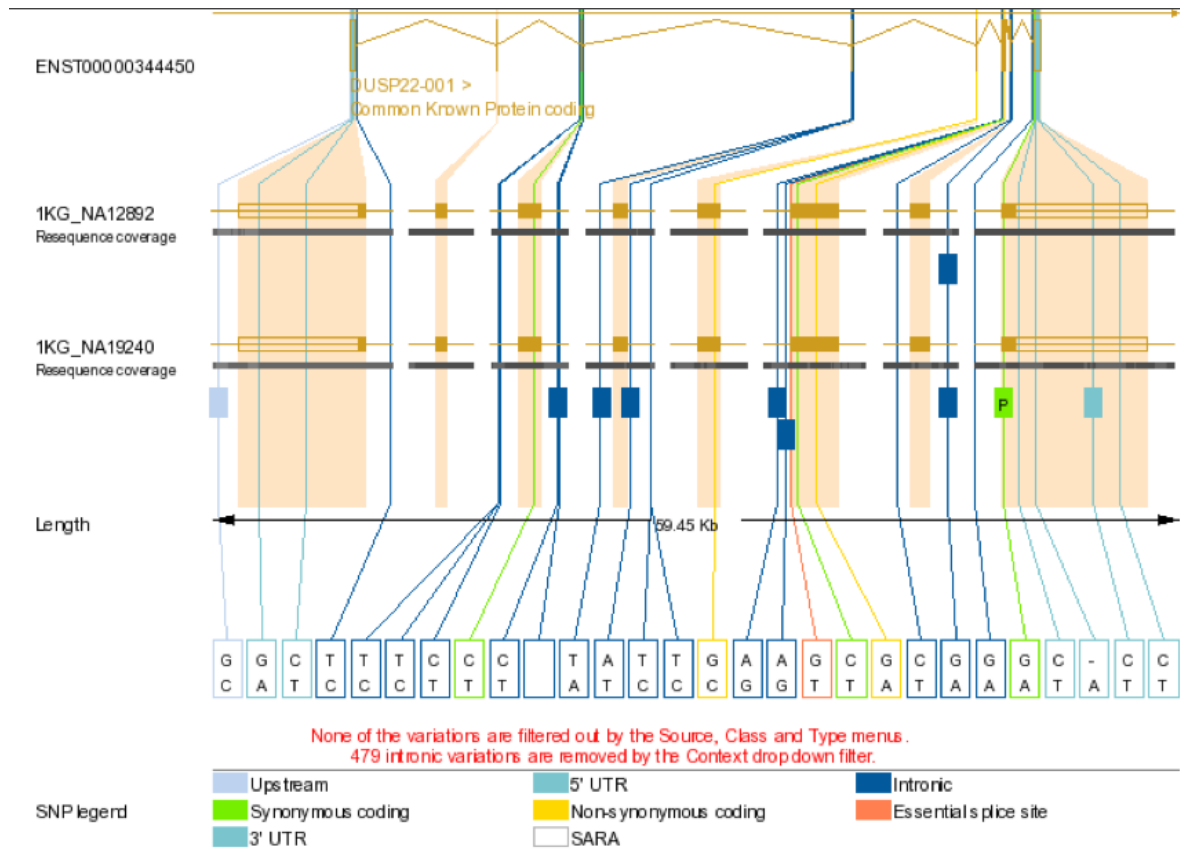
by checking the 1KG browser.

NCBI36	2201	TGGGTAATAAAATTTCTTGACTTTTATTATCTTGTGAATAGAGACTCACAG	2250
1KG_NA19240	2201	TGGGTAATAAAATTTCTTGACTTTTATTATCTTGTGAATAGAGACTCACAG	2250
NCBI36	2251	TGTATCAATGAAGAAATACATGTATAGAGAGATTTAGAGACTTCCCTGCT	2300
1KG_NA19240	2251	TGTATCAATGAAGAAATATGTATAGAGAGATTTAGAGACTTCCCTGCT	2300
NCBI36	2301	TCAACACTTATTGATTATGTACATACTATGAGCCAGGTGTTGTGCTAGTG	2350
1KG_NA19240	2301	TCAACACTTATTGATTATGTACATACTATGAGCCAGGTGTTGTGCTAGTG	2350
NCBI36	2351	GGTGGGGAGGGGGCAAAGGGAGTACAGTGGTAAATGACCCAAAGATTCTT	2400
1KG_NA19240	2351	GGTGGGGAGGGGGCAAAGGGAGTACAGTGGTAAATGACCCAAAGATTCTT	2400
NCBI36	2401	CACAGAACCTTCAGTACAGGTCAGCATTGTCCAGCACTAATATGATGCGA	2450
1KG_NA19240	2401	CACAGAACCTTCAGTACAGGTCAGCATTGTCCAGCACTAATATGATGCGA	2450
NCBI36	2451	GCTATATATGTAACCTTAAAAATGTTCCGGTAGCCACATTAATAAAATAAACC	2500
1KG_NA19240	2451	GCTATATATGTAACCTTAAAAATGTTCCGGTAGCCACATTAATAAAATAAACC	2500
NCBI36	2501	GGTGAATTAATTTTAATAATATATTTTATTTAACAATATATTAATAACTA	2550
1KG_NA19240	2501	GGTGAATTAATTTTAATAATATATTTTATTTAACAATATATTAATAACTA	2550
NCBI36	2551	TTGTTTCAACATGTAACCAATATAAAGTTATTCATGGAAGATTCCATG	2600
1KG_NA19240	2551	TTGTTTCAACATGTAACCAATATAAAGTTATTCATGGAAGATTCCATG	2600
NCBI36	2601	TTTTTACTATGTCTTTGTGAAAGGAAAAATCTCGGGACCCCCAGATCAC	2650
1KG_NA19240	2601	TTTTTACTATGTCTTTGTGAAAGGAAAAATCTCGGGACCCCCAGATCAC	2650

Looking in more detail, we have



and this can be exploded into the Ensembl variant browser view



with textual metadata view

Variations in 1KG_NA19240												
ID	Type	Chr: bp	Ref. allele	Individual genotype	Ambiguity	Transcript codon	CDS coord.	AA change	AA coord.	Class	Source	Validation
<a href="#">rs9405165</a>	UPSTREAM	6:237024	G	C/C	S	-	-	-	-	SNP	1KG_NA19240, dbSNP, ENSEMBL:Venter	-
<a href="#">rs1342789</a>	INTRONIC	6:257023	C	C/T	Y	-	-	-	-	SNP	ENSEMBL:Watson, HGvbase, 1KG_NA19240, dbSNP, ENSEMBL:Venter	cluster, frequency, submitter, doublehit, hapmap
<a href="#">rs7753848</a>	INTRONIC	6:280069	T	A/T	W	-	-	-	-	SNP	1KG_NA19240, dbSNP	frequency, hapmap
<a href="#">rs2671431</a>	INTRONIC	6:280175	A	T/T	W	-	-	-	-	SNP	ENSEMBL:Watson, HGvbase, 1KG_NA19240, dbSNP, ENSEMBL:Venter	cluster, doublehit, hapmap
<a href="#">rs1877172</a>	INTRONIC	6:293051	A	A/G	R	-	-	-	-	SNP	ENSEMBL:Watson, HGvbase, 1KG_NA19240, dbSNP, TSC	cluster, frequency, submitter, doublehit
<a href="#">rs2797333</a>	INTRONIC	6:293080	A	G/G	R	-	-	-	-	SNP	ENSEMBL:Watson, HGvbase, 1KG_NA19240, dbSNP, ENSEMBL:Venter	cluster, doublehit, hapmap
<a href="#">rs11242812</a>	INTRONIC	6:293906	G	A/G	R	-	-	-	-	SNP	ENSEMBL:Watson, 1KG_NA12892, 1KG_NA12891, 1KG_NA12878, 1KG_NA19240, dbSNP	cluster, doublehit
<a href="#">rs1129085</a>	SYNONYMOUS_CODING	6:295829	G	A/G	R	CCA	516	P	172	SNP	ENSEMBL:Watson, 1KG_NA19240, dbSNP, ENSEMBL:Venter	cluster, frequency
<a href="#">rs1046656</a>	3PRIME_UTR	6:296156	C	C/T	Y	-	-	-	-	SNP	ENSEMBL:Watson, HGvbase, 1KG_NA19240, dbSNP, ENSEMBL:Venter	cluster, frequency, doublehit



So it seems DUSP22 resides over plenty of known SNP; our computations are supposed to reveal hitherto unknown variants in this region for this individual.

### 5.3 Exercises

1. The `oregdf` *data.frame* is supplied in *ind1KG*, containing information on regulatory elements annotated in *oreganno*. How many novel variants for NA19240 lie in *oreganno* regulatory regions? What types of regions are occupied?
2. Derive a *data.frame* for regions of nucleosome occupancy in our 10 Mb segment and check how many of the novel variants lie in such regions.