Lab solutions. Rare variant concepts and tools with Bioconductor.

VJ Carey, ©2009

November 19, 2009

1 Exercises restated

1. Estimate the frequencies of dbSNP-catalogued SNP per base pair in intronic vs. exonic DNA on chromosome 17. Estimate frequencies stratified by GC content (i.e., tabulate by 0, 1, 2 bases G or C in SNP).

2. Write a function with parameters identifying a RangedData instance generated from a pileup, a gene symbol, a variant type, and a specification of feature scope, that reports on the variants present in the gene. Discuss infelicities of basic Bioconductor genomic features data structures that should be ameliorated to simplify solution of this exercise.

3. Whether or not you solve the previous exercise, characterize the variants in gene MYH3 for NA19240 in some concise way. It is advisable to focus on SNP; show that there are coding SNP present for this individual that are not identified in dbSNP.

4. Introduce and justify a mechanism for filtering variant reporting using quality information.

5. Assess how many of the MAQ-based SNP calls using the chromosome 6 pileup data are found at dbSNP locations. Is the distribution of quality scores for variants identified at dbSNP locations similar to that of putatively de novo variants?

6. Acquire the probe sequences for the illumina Human v1 expression array, perhaps by inverting the nuids found in the lumiHumanIDMapping metadata package. Determine the genomic positions of all probes interrogating genes on chromosome 17 using Biostrings matchPDict against the consensus genomic sequence for chromosome 17. Find all probes (on chr17) corresponding to sequence for which NA19240 is found by MAQ to harbor a variant (use the pileup noted in exercise presentation). We will call these probes "associated with sequence variants". Compute

expression Z-scores for expression levels obtained for NA19240 using mean and standard deviation based on log expression for the 89 individuals in hmyriB36 excluding NA19240. Can the distribution of expression Z-scores for probes associated with sequence variants be distinguished from the distribution of expression Z-scores for probes not associated with sequence variants.

Extra credit extension: Some probes define sequence associated with splice junctions. These 50mers will not align to consensus genomic sequence, but will align once introns are removed. Can you identify probes associated with splice junctions that are also associated with sequence variants? Does the expression Z-score for splice-junction-associated probes differ in distribution from the general distribution of expression Z-scores?

7. (Additional exercises.) Retrieve the SOLiD or 454-based short read archives for NA19240 and check the consistency of conclusions obtained in prior Solexa-based exercises with results based on these platforms.

8. (Special design exercise (attempt only after all other exercises have been solved correctly.)) Consider alternative representations of the SNP location/value data. The allele data could be represented as a single DNAString, and the location information as an IRanges instance. Assess the resource consumption and query resolution performance of these representations in comparison to the existing data.frame. Consider also a representation rooted in an RDBMS such as SQLite.

2 Solutions

1. (Stratified SNP frequency estimates.)

   The SNP location/code metadata are:

```
> library(SNPlocs.Hsapiens.dbSNP.20080617)
> c17s = getSNPlocs("chr17")
> c17s[1:4, ]

  RefSNP_id alleles_as_ambig  loc
1   1106176                R 6934
2   6420494                S 7214
3   6420495                K 7242
4  62054996                R 8611
```

   The locations of exons are given simply:

```
> library(GenomicFeatures)
> library(GenomicFeatures.Hsapiens.UCSC.hg18)
> library(IRanges)
```

```
> genes = geneHuman()
> g17 = genes[genes$chrom == "chr17", ]
> g17.e = exons(g17)
```

What is the nature of this structure? It is easy to look at:

```
> g17.e

RangedData with 34108 rows and 1 value column across 1 space
         space           ranges |        gene
   <character>        <IRanges> | <character>
1         chr17 [ 62294,  63714] |  uc002frd.1
2         chr17 [ 65445,  65593] |  uc002frd.1
3         chr17 [ 69414,  69527] |  uc002frd.1
4         chr17 [ 96902,  97076] |  uc002frd.1
5         chr17 [131559, 131645] |  uc002frd.1
6         chr17 [169211, 169340] |  uc002frd.1
7         chr17 [171063, 171206] |  uc002frd.1
8         chr17 [177258, 177378] |  uc002frd.1
9         chr17 [ 62294,  63714] |  uc002fre.1
10        chr17 [ 65445,  65593] |  uc002fre.1
...
<34098 more rows>
```

but note that record 9 has the same start and end as record 1. The UCSC known genes annotation has a certain redundancy – certainly there is a many-one relationship between UCSC known genes and Entrez genes (exercise: could you check if it is many-many?) For the question of interest here, we would like to partition the genome into exons and introns, and the UCSC "known gene" redundancies are not desirable.

To get a view of the scope of this problem, we can tabulate the start values in our exon table. A table would be quite long, but a table of appearance frequencies is useful:

```
> table(table(start(g17.e)))

   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
5738 3618 2070 1324  596  432  242   62   37    2    8    4    1    6   10    5
  17   18   19   20   21   26   27   28
   1    4    1    3    1   10   15    7
```

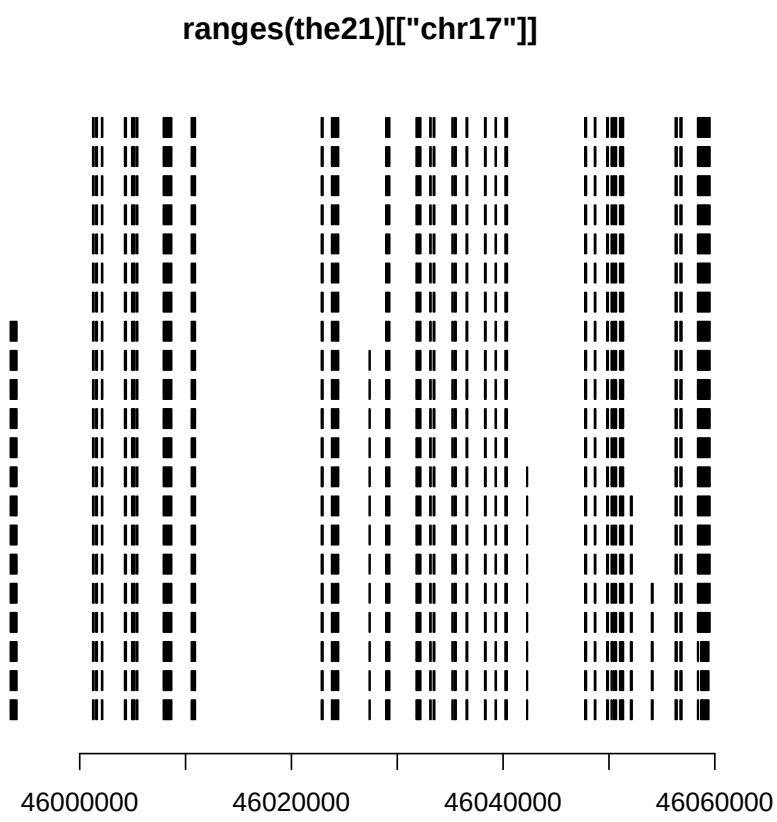
Note that there is one start address that appears 21 times. Let's find it:

```
> haslots = which(table(start(g17.e)) == 21)
> haslots

46040187
    9441

> st21 = g17.e[start(g17.e) == as.numeric(names(haslots)), ]
> st21

RangedData with 21 rows and 1 value column across 1 space
         space               ranges |        gene
   <character>            <IRanges> | <character>
1        chr17 [46040187, 46040400] |  uc002irj.1
2        chr17 [46040187, 46040379] |  uc002irk.1
3        chr17 [46040187, 46040379] |  uc002irl.1
4        chr17 [46040187, 46040400] |  uc002irm.1
5        chr17 [46040187, 46040379] |  uc002irn.1
6        chr17 [46040187, 46040400] |  uc002iro.1
7        chr17 [46040187, 46040379] |  uc002irp.1
8        chr17 [46040187, 46040379] |  uc002irq.1
9        chr17 [46040187, 46040379] |  uc002irr.1
10       chr17 [46040187, 46040400] |  uc002irs.1
...
<11 more rows>
```

The gene names here are distinct, but the genomic information is often similar. To visualize the situation, we can borrow the plotRanges function, but we need to get full exonic address information from the 21 genes:

```
> the21 = g17.e[g17.e$gene %in% st21$gene, ]
> plotRanges = function(x, xlim = x, main = deparse(substitute(x)),
+     col = "black", sep = 0.5, ...) {
+     height <- 1
+     if (is(xlim, "Ranges"))
+         xlim <- c(min(start(xlim)), max(end(xlim)))
+     bins <- disjointBins(IRanges(start(x), end(x) + 1))
+     plot.new()
+     plot.window(xlim, c(0, max(bins) * (height + sep)))
+     ybottom <- bins * (sep + height) - height
+     rect(start(x) - 0.5, ybottom, end(x) + 0.5, ybottom + height,
+         col = col, ...)
+     title(main)
+     axis(1)
+ }
```

```
> plotRanges(ranges(the21)[["chr17"]])
```



How can we eliminate the redundancy and retain the addresses of the most comprehensive exon model for the gene?

```
> newr = reduce(ranges(the21)[["chr17"]])
> newr

IRanges of length 38
        start      end width
[1]  45993448 45994061   614
[2]  46001230 46001341   112
[3]  46001525 46001658   134
[4]  46002066 46002163    98
[5]  46004238 46004397   160
[6]  46004914 46005214   301
[7]  46005381 46005473    93
[8]  46007903 46008686   784
```

```
[9]  46010548 46010924   377
...       ...      ...   ...
[30] 46050219 46050289    71
[31] 46050408 46050486    79
[32] 46050582 46050703   122
[33] 46051015 46051368   354
[34] 46052043 46052186   144
[35] 46054020 46054154   135
[36] 46056267 46056435   169
[37] 46056720 46056889   170
[38] 46058377 46059541  1165
```

Now if we really wanted to distinguish intronic and exonic sequence regions, we'd have to split the chromosome by gene, reduce within gene, compute the complement of the exonic region for each gene, and combine over non-redundant genes. A very simple approach that distinguishes exonic and non-exonic addresses is:

```
> library(org.Hs.eg.db)
> nb17 = org.Hs.egCHRLENGTHS["17"]
> redex17 = reduce(ranges(g17.e)[["chr17"]])
> all17 = IRanges(start = 1, end = nb17)
> rednonex17 = setdiff(all17, redex17)
```

Get the denominator for our rate estimates:

```
> library(org.Hs.eg.db)
```

Transform SNP metadata on location to IRanges structure, and compute requested intersections. We use an abuse of language, regarding non-exonic sequence as intronic.

```
> sranges = IRanges(start = c17s$loc, end = c17s$loc)
> s.in.exon = intersect(sranges, redex17)
> s.in.intron = intersect(sranges, rednonex17)
```

Basic frequency estimates:

```
> length(s.in.intron)/org.Hs.egCHRLENGTHS["17"]

        17
0.003306745

> length(s.in.exon)/org.Hs.egCHRLENGTHS["17"]
```

```
          17
0.0002191946
```

Stratification by CG content of SNP. First we analyze the ambiguity codes for GC content.

```
> # compute number C/G for each SNP allele assignment
> library(Biostrings)
> ncg = (1:15) %in% grep("C", IUPAC_CODE_MAP) +
+     (1:15) %in% grep("G", IUPAC_CODE_MAP)
```

Or you could write/use:

```
> GCcontent = function(cset) {
+     apply(alphabetFrequency(DNAStringSet(cset), baseOnly = TRUE)[,
+         2:3], 1, sum)
+ }
> Biostrings::IUPAC_CODE_MAP
```

```
     A      C      G      T      M      R      W      S      Y      K      V
   "A"    "C"    "G"    "T"   "AC"   "AG"   "AT"   "CG"   "CT"   "GT"  "ACG"
     H      D      B      N
 "ACT"  "AGT"  "CGT" "ACGT"
```

```
> GCcontent(IUPAC_CODE_MAP)
```

```
 [1] 0 1 1 0 1 1 0 2 1 1 2 1 1 2 2
```

But you need association:

```
> names(ncg) = names(IUPAC_CODE_MAP)
> al = c17s[, 2]
> ncgs = ncg[al]
> table(ncgs)
```

```
ncgs
     0      1      2
 19295 238373  25722
```

```
> SSS = split(sranges, ncgs)
> zer.i = intersect(SSS[["0"]], rednonex17)
> one.i = intersect(SSS[["1"]], rednonex17)
> two.i = intersect(SSS[["2"]], rednonex17)
> zer.e = intersect(SSS[["0"]], redex17)
```

```
> one.e = intersect(SSS[["1"]], redex17)
> two.e = intersect(SSS[["2"]], redex17)
> fmat = t(matrix(sapply(list(zer.i, one.i, two.i, zer.e, one.e,
+     two.e), length), nc = 2))
> dimnames(fmat) = list(c("intron", "exon"), c("#CG=0", "#CG=1",
+     "#CG=2"))
> fmat

       #CG=0  #CG=1 #CG=2
intron 18120 219997 23684
exon     820  14738  1825

> fmat[1, ]/sum(fmat[1, ])

     #CG=0      #CG=1      #CG=2
0.06921288 0.84032147 0.09046566

> fmat[2, ]/sum(fmat[2, ])

     #CG=0      #CG=1      #CG=2
0.04717252 0.84783984 0.10498763
```

Conclusion: On chr17, SNPs in exons are more likely to have G or C alleles than SNPs in introns.

2. (Function for reporting on variants by gene.)

```
> showVariants = function(genesym, pup,
+   scope=c("transcripts", "exons", "introns")) {
+  if (length(scope)>1) {
+    warning("multiple elements in scope parameter, using first")
+    scope = scope[1]
+  }
+  require(org.Hs.eg.db)
+  egid = get(genesym, revmap(org.Hs.egSYMBOL))
+  if (length(egid) > 1) {
+    warning("multiple Entrez IDs, using first")
+    egid = egid[1]
+    }
+  kgid = get(egid, org.Hs.egUCSCKG)
+  require(GenomicFeatures)
+  require(GenomicFeatures.Hsapiens.UCSC.hg18)
+  require(IRanges)
+  require(Rsamtools)
+  genes = geneHuman()
+  ginfo = genes[ genes$name == kgid, ]
+  chr = get(egid, org.Hs.egCHR )[1]
+  cchr = paste("chr", chr, sep="")
+  featloc = get(scope)(ginfo)  # watch out
+  featr = ranges(featloc)[[cchr]]
+  pup[ which(ranges(pup)[[chr]]%in% featr), ]
+ }
```

3. (Reporting on MYH3.)

```
> pup17 = gzfile(system.file("pileups/n240_17.pup.gz", package="ind1KG"))
> library(Rsamtools)
> c17p.s = readPileup(pup17, variant="SNP")
> c17p.s
```

```
RangedData with 133527 rows and 6 value columns across 1 space
         space       ranges | referenceBase consensusBase consensusQuality
   <character>    <IRanges> |      <factor>      <factor>        <integer>
1           17 [ 828,  828] |             T             C               14
2           17 [ 834,  834] |             G             C                4
3           17 [1869, 1869] |             A             T               88
4           17 [2041, 2041] |             G             A                3
5           17 [2220, 2220] |             G             A               48
```

```
6           17 [2564, 2564] |             A             G                68
7           17 [3587, 3587] |             G             A                45
8           17 [3936, 3936] |             A             G                 2
9           17 [4966, 4966] |             A             G                42
10          17 [5687, 5687] |             C             T                26
   snpQuality maxMappingQuality  coverage
    <integer>         <integer> <integer>
1          14                 0         4
2           4                 0         1
3         129                24        36
4          10                 0         4
5          48                13         8
6          90                24        25
7          45                12        26
8           2                 0        20
9          42                27         5
10         26                 0        12
...
<133517 more rows>
```

```
> pup17b = gzfile(system.file("pileups/n240_17.pup.gz", package="ind1KG"))
> c17p.i = readPileup(pup17b, variant="indel")
> c17p.i
```

```
RangedData with 22640 rows and 11 value columns across 1 space
         space         ranges | referenceBase consensusBase consensusQuality
   <character>      <IRanges> |      <factor>      <factor>        <integer>
1           17 [55518, 55518] |             A             A               98
2           17 [55994, 55994] |             T             T               70
3           17 [56014, 56014] |             A             R               37
4           17 [57801, 57801] |             G             G               62
5           17 [59631, 59631] |             C             C               72
6           17 [62489, 62489] |             G             G               61
7           17 [62491, 62491] |             C             C              123
8           17 [62495, 62495] |             C             C              107
9           17 [62498, 62498] |             A             A              106
10          17 [64559, 64559] |             T             T               68
   snpQuality maxMappingQuality  coverage   alleleOne alleleOneSupport
    <integer>         <integer> <integer> <character>        <integer>
1           0                47        31          -T                2
2           0                52        20          -A                6
3          37                46        18          +G                7
4           0                57        27          +A               17
```

```
5           0                60        21         -AT                11
6           0                56        36         +AC                 7
7           0                56        32         +AA                 1
8           0                56        33         +AG                 1
9           0                56        38         +CG                 1
10          0                51        33          +C                 1
     alleleTwo alleleTwoSupport additionalIndels
   <character>        <integer>        <integer>
1            *               29                0
2            *               14                0
3            *               11                0
4            *               10                0
5            *               10                0
6            *               29                0
7            *               31                0
8            *               32                0
9            *               37                0
10           *               31                1
...
<22630 more rows>
```

```
> showVariants("FAM83G", c17p.i, "transcripts")
```

```
RangedData with 7 rows and 11 value columns across 1 space
       space               ranges | referenceBase consensusBase
  <character>            <IRanges> |      <factor>      <factor>
1          17 [18815408, 18815408] |             T             T
2          17 [18815410, 18815410] |             C             C
3          17 [18834635, 18834635] |             T             T
4          17 [18834836, 18834836] |             G             G
5          17 [18834839, 18834839] |             C             C
6          17 [18834840, 18834840] |             T             T
7          17 [18846156, 18846156] |             C             C
  consensusQuality snpQuality maxMappingQuality  coverage   alleleOne
         <integer>  <integer>         <integer> <integer> <character>
1             107          0                55        38           *
2              81          0                55        39           *
3             100          0                56        49          +C
4             120          0                56        31          -T
5             108          0                56        31          -T
6              56          0                57        32          -C
7              65          0                57        56         +CCT
  alleleOneSupport   alleleTwo alleleTwoSupport additionalIndels
```

```
         <integer> <character>        <integer>        <integer>
1               37        +TCG                1                0
2               35        +GGT                4                0
3               13           *               36                0
4                1           *               30                0
5                1           *               30                0
6               19           *               13                0
7                3           *               53                0
```

```
> mv = showVariants("MYH3", c17p.s, "exons")
> mv
```

```
RangedData with 7 rows and 6 value columns across 1 space
        space               ranges | referenceBase consensusBase
  <character>            <IRanges> |      <factor>      <factor>
1          17 [10476743, 10476743] |             G             R
2          17 [10482240, 10482240] |             C             T
3          17 [10483196, 10483196] |             T             K
4          17 [10483490, 10483490] |             A             R
5          17 [10483611, 10483611] |             T             Y
6          17 [10484188, 10484188] |             T             Y
7          17 [10485141, 10485141] |             G             K
  consensusQuality snpQuality maxMappingQuality  coverage
         <integer>  <integer>         <integer> <integer>
1              135        135                56        30
2               98        141                55        40
3              141        141                59        42
4              110        110                59        40
5              160        160                55        53
6              151        151                55        35
7              108        108                57        45
```

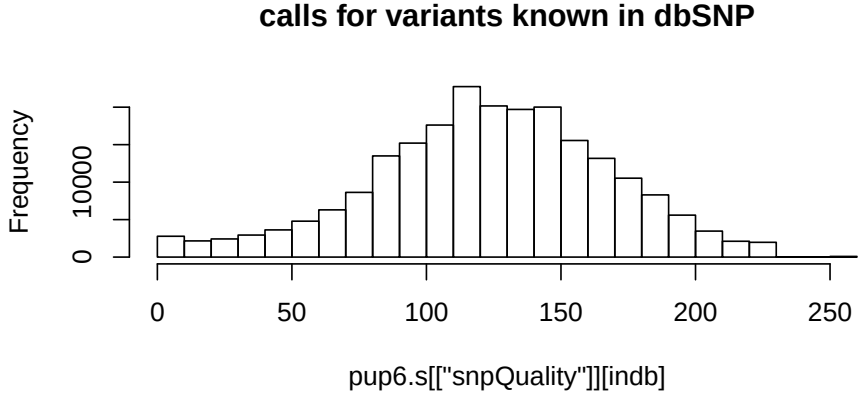
```
> all(ranges(mv)[["17"]] %in% sranges) # if true, nothing de novo in this pileup
```

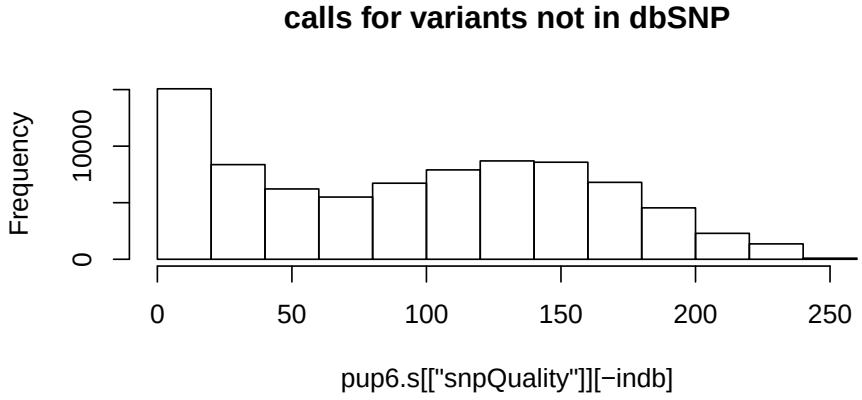
```
[1] TRUE
```

4. (Quality considerations 1.) Introduce and justify a mechanism for filtering variant reporting using quality information. On your own.

5. (Quality considerations 2.) Assess how many of the MAQ-based SNP calls using the chromosome 6 pileup data are found at dbSNP locations? Is the distribution of quality scores for variants identified at dbSNP locations similar to that of putatively de novo variants?

```
> pup6 = gzfile(system.file("pileups/n240_6.pup.gz", package = "ind1KG"))
> pup6.s = readPileup(pup6, variant = "SNP")
> s6 = getSNPlocs("chr6")
> sranges = IRanges(start = s6$loc, end = s6$loc)
> comm = intersect(ranges(pup6.s)[["6"]], sranges)
> nrow(s6)

[1] 695808

> length(comm)

[1] 221569

> indb = which(ranges(pup6.s)[["6"]] %in% sranges)

> par(mfrow = c(2, 1))
> hist(pup6.s[["snpQuality"]][indb], main = "calls for variants known in dbSNP")
> hist(pup6.s[["snpQuality"]][-indb], main = "calls for variants not in dbSNP")
```





6. (Concordance: MAQ vs HapMap phase II for NA19240.) Get the pileup and MAQ calls:

```
> library(Rsamtools)
> if (!exists("pup6.s")) {
+     pup6 = gzfile(system.file("pileups/n240_6.pup.gz", package = "ind1KG"))
+     pup6.s = readPileup(pup6, variant = "SNP")
+ }
```

The HapMap Phase II calls are:

```
> library(ind1KG)
> library(snpMatrix)
> data(yri240_6)
> yri240_6$hm2

A snp.matrix with  1 rows and  265955 columns
Row name:  NA19240
Col names:  rs4097465 ... rs4599694

> yri240_6$supp[1:4, ]

           dbSNPalleles Assignment Chromosome Position Strand
rs4097465          G/T        G/T       chr6    37012      -
rs7754266          A/G        A/G       chr6    94609      +
rs9393087          C/T        C/T       chr6    94901      +
rs12192290         A/T        A/T       chr6    95272      +
```

The Iranges representation of SNP locations is:

```
> dbranges6 = IRanges(start = yri240_6$supp$Position, end = yri240_6$supp$Positi
```

The indices of SNP variants in NA19240 that coincide with the HM2 SNP positions are

```
> inhm2 = which(ranges(pup6.s)[["6"]] %in% dbranges6)
```

Let's look at a few of the variants:

```
> few = pup6.s[inhm2[1:3], ]
> few

RangedData with 3 rows and 6 value columns across 1 space
        space         ranges | referenceBase consensusBase consensusQuality
  <character>      <IRanges> |      <factor>      <factor>        <integer>
```

```
1           6 [94609, 94609] |            G            A             108
2           6 [95272, 95272] |            A            W             138
3           6 [99536, 99536] |            A            M             116
  snpQuality maxMappingQuality  coverage
   <integer>         <integer> <integer>
1        135                43        37
2        202                58        33
3        116                49        42
```

To get the associated rs numbers:

```
> supp = yri240_6$supp
> matched = supp[supp$Position %in% start(ranges(few)[["6"]]),
+     ]
> matched

           dbSNPalleles Assignment Chromosome Position Strand
rs7754266           A/G        A/G       chr6    94609      +
rs12192290          A/T        A/T       chr6    95272      +
rs1929630           A/C        A/C       chr6    99536      +

> rsmat = rownames(matched)
```

The actual HapMap phase II calls for NA19240 are

```
> yri240_6[[1]][, rsmat]

Autosomal snp(s):
 rs7754266 rs12192290  rs1929630
     "A/A"      "A/B"      "A/B"
```

Now we write a function that characterizes discrepancies between calls in a HapMap import and the MAQ calls for that individual. We'll solve the problem for the case in which the MAQ call is heterozygous but the HM2 call is homozygous.

```
> chkhh = function(hm2list, pup, space = "6") {
+     hm2rng = IRanges(start = hm2list$supp$Position, end = hm2list$supp$Positio
+     puprng = ranges(pup)[[space]]
+     inhm2 = which(puprng %in% hm2rng)
+     pupsub = pup[inhm2, ]
+     supp = hm2list$supp
+     matched = supp[supp$Position %in% start(ranges(pupsub)[[space]]),
+         ]
+     rsid = rownames(matched)
```

```
+     hmcalls = as(hm2list[[1]][, rsid], "character")
+     dbsn = supp[rsid, "dbSNPalleles"]
+     ccbc = as.character(pupsub$consensusBase)
+     disc = which(hmcalls != "A/B" & !(ccbc %in% c("A", "C", "T",
+         "G")))
+     ans = pupsub[disc, ]
+     ans$rsid = rsid[disc]
+     ans$dbSNPallele = dbsn[disc]
+     ans
+ }
> cc = chkhh(yri240_6, pup6.s)
> cc

RangedData with 4281 rows and 8 value columns across 1 space
         space           ranges | referenceBase consensusBase consensusQuality
   <character>        <IRanges> |      <factor>      <factor>        <integer>
1            6 [202186, 202186] |             C             Y              228
2            6 [202246, 202246] |             C             Y              185
3            6 [202461, 202461] |             G             R              142
4            6 [203015, 203015] |             C             S              228
5            6 [203032, 203032] |             A             R              189
6            6 [206391, 206391] |             A             R              140
7            6 [219713, 219713] |             G             R              141
8            6 [226417, 226417] |             T             W              156
9            6 [239416, 239416] |             C             Y              198
10           6 [239423, 239423] |             G             R              215
   snpQuality maxMappingQuality  coverage        rsid dbSNPallele
    <integer>         <integer> <integer> <character>    <factor>
1         228                58        70   rs1011327         C/T
2         185                58        61   rs4959628         C/T
3         228                59        69   rs1011329         A/G
4         228                58        81   rs4320417         C/G
5         189                58        98   rs4596530         A/G
6         140                56        80   rs7740992         A/G
7         228                59        63   rs4959645         A/G
8         156                59        61   rs6922183         A/T
9         198                58        67  rs11242765         C/T
10        217                58        64  rs11752881         A/G
...
<4271 more rows>
```

Extra credit: Make this function more comprehensive.

7. (Effects of novel variants on transcript profiling.)

   We'll obtain all 48k sequences for 50mers on the illumina v1 chip.

```
> library(lumiHumanIDMapping)

This is mgcv  1.5-6 . For overview type `help("mgcv-package")'.

> con = lumiHumanIDMapping_dbconn()
> dbListTables(con)

 [1] "HumanHT12_V3_0_R1_11283641_A" "HumanHT12_V3_0_R2_11283641_A"
 [3] "HumanRef8_V2_0_R1_11223162_A" "HumanRef8_V2_0_R2_11223162_A"
 [5] "HumanRef8_V3_0_R0_11282963_A" "HumanRef8_V3_0_R1_11282963_A"
 [7] "HumanRef8_V3_0_R2_11282963_A" "HumanWG6_V1"
 [9] "HumanWG6_V2_0_R1_11223189_A"  "HumanWG6_V2_0_R2_11223189_A"
[11] "HumanWG6_V2_11223189_B"       "HumanWG6_V3_0_R0_11282955_A"
[13] "metadata"                     "nuID_MappingInfo"

> dbGetQuery(con, "select * from HumanWG6_V1 limit 5")

    Search_key        Target    ProbeId   Accession Symbol               nuID
1        PLAC3 GI_23097300-A 0002360044 NM_021936.1  PLAC3 cn0dn1Sqdb0UHE4nEY
2         COG4 GI_21070955-A 0003940446 NM_015386.1   COG4 iklSlJ.eTo60t35XQE
3 GI_4505876-A  GI_4505876-A 0006420736 NM_000445.1  PLEC1 NBHBeFupql_azWVUMA
4        PTPRD GI_18860893-A 0002630279 NM_130393.1  PTPRD KcSlfQzU6Ld94lMSpE
5       HS6ST2 GI_27597081-A 0003120162 NM_147174.2 HS6ST2 ZeMrPvoCSjgl4lLoAk

> allnu = dbGetQuery(con, "select nuID from HumanWG6_V1")
> allnuc = as.character(allnu[, 1])
> library(lumi)
> seqs = id2seq(allnuc)
> length(seqs)

[1] 47296
```

   Now let's get the ranges of their locations in consensus genome.

```
> library(BSgenome.Hsapiens.UCSC.hg18)
> s17 = Hsapiens$chr17
> library(Biostrings)
> d48k = PDict(forwSeqs <- DNAStringSet(unique(seqs)))
> rd48k = PDict(revSeqs <- reverseComplement(DNAStringSet(unique(seqs))))
> fok = which(countPDict(d48k, s17) == 1)
> rok = which(countPDict(rd48k, s17) == 1)
> length(intersect(rok, fok))
```

```
[1] 23
```

We'll have to be a bit careful. Some probes (evidently not palindromes) match in both forward and reverse forms. We will ignore these.

```
> f2keep = setdiff(fok, intersect(rok, fok))
> d48kok = PDict(forwSeqs <- DNAStringSet(unique(seqs)[f2keep]))
> fmat = matchPDict(d48kok, s17)
> r2keep = setdiff(rok, intersect(rok, fok))
> rd48kok = PDict(revSeqs <- reverseComplement(DNAStringSet(unique(seqs)[r2keep],
> rmat = matchPDict(rd48kok, s17)
> fstarts = unlist(startIndex(fmat))
> fends = unlist(endIndex(fmat))
> rstarts = unlist(startIndex(rmat))
> rends = unlist(endIndex(rmat))
> fints = IRanges(start = fstarts, end = fends)
> rints = IRanges(start = rstarts, end = rends)
```

We want to deal with 'novel' variants if there are enough. Thus we exclude variants for NA19240 that are known in dbSNP.

```
> sranges17 = getSNPlocs("chr17")
> sranges17 = IRanges(start = sranges17$loc, end = sranges17$loc)
> all17v = ranges(c17p.s)[["17"]]
> known = which(all17v %in% sranges17)
> new17v = all17v[-known, ]
```

We obtain the nuids of the probes whose forward or reverse sequences harbor novel variants in NA19240. This is easier than keeping track of the source sequences through the filtering process conducted here, although perhaps some container that would manage this should perhaps be devised.

```
> vapf = fints[fints %in% new17v]
> fext = substring(s17, start(vapf), end(vapf))
> fnu = seq2id(fext)
> fnu

TCCCCCACCTGGTTTTGACTAATCCTGCTTCCCTCTCTGGGCCTGGCTGC
                                "E1VF6.4cNefV3epenk"
ATCAGCGTCATCGGCGGGAACCTCACGGGCATCTTCATCCACCGGGTCAC
                                "9NJtNpqBdGpN9NRatE"
CCCGGCTTTGGTGCGGGGTACACAAGAGGGGATGAGTTGTGTGAATACCC
                                "lVp.rmqxEIqji_7gxU"
GCTGGGCCTCTGCTCTTTCCTGGGCTGACGTAAAGCGTTCTGCTCATTTA
```

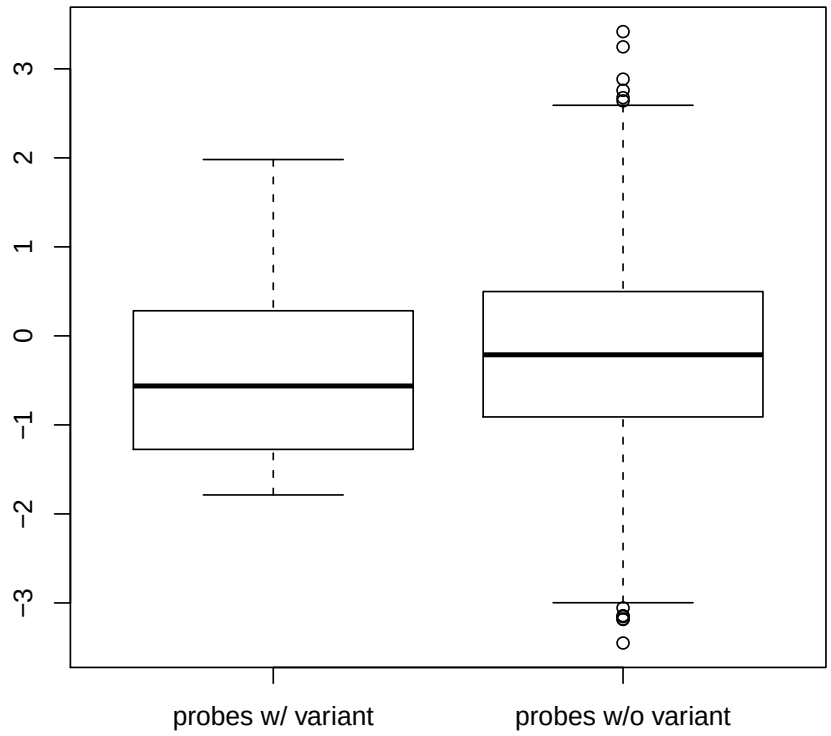
```
                                  "WnqXed.XqeGwJvedPw"
TCCCTGCAGATGAACAGCAACTACCCCGCCTTGCTCAAGTGCCCCGACTA
                                  "H1eSOBJBxVl_dC5Vhw"
CAGGAGATAGAACAGCCCGGCCTAGCCAGGAGAGACTGCAGGGACTCACT
                                  "fSiMgSVpclKIh5Kh0c"
AGACTCCCGCCTGCCCCCAGTGAAGGAATTGGTTACTCTCCCAGAGCCCT
                                  "0IdWXlVLgoPrx3VIlc"
TTCCAGACCCCTCCCCAAACATGCATATGTACCTGTCCGTCAGTGTGTGG
                                  "o9SFXVQE5M7F7W0u7o"
GCGGAAGCTATCAGGACCGATTACAATGGCAGGTGTGACGAGATCCACAG
                                  "6mgnNKFjxDpK7hiNRI"
TACCTGCTGGGCCGCAACCCCTTCTCGTGGTGCCTGGAGTGGGACGACAA
                                  "TxeepZBV9265ei6hhA"
AATGGTGGCAACTTCAGTAGGCTGACGGAGGAGAACCTGAGGGTACGGGG
                                  "QDrpB9LKeGiiBeKsao"
ATGAATCTTTACATGTCTGCACAGGCTCTAGGTGGGAAGGATGCCCGTGG
                                  "cODfxO3kSncrqCjlbo"
GTTGTGTTTAAAAATTATGGACAAGGCCGGGCACGGTGGCTCACGCCTGG
                                  "Qvu.ADzoQpaka6dGXo"
CGTCAGAGACGGTTGGCAGTGGGCAAGGACAAGTGTGAAGCTCTGTCTCC
                                  "ibSIa_kupChC7gne3U"
AACAACCAGCACCAAGCACCAGCCACGCAAGAAAGGCCATCCCAGGCCAT
                                  "rBBSRQkUlGQgKU1SlM"
CAGCGTCCCTCGTCTGCGGCCACTGTCTGCCCTTCATCTATGTTCTCAAG
                                  "uSbV23mlHt5XO3O90I"
GCTGTCTGAGATGCAGCACTTCTCAAATACATGTCTGACTCACTGGCAAG
                                  "0nt4jkkfdAxO3h0ekI"
GAGCAATGGACACTCTCCTTCCTGTATCTGTTAGTGAGGTTCCTGCTGAC
                                  "6iQ6Ed19eze8uK9eeE"
AGGGCTGTGACGGCATTTCTGGTGTTAGTGTCTCTGGCTGTGAGTTCTGC
                                  "KKnuGk.eu8u3enuL3k"
```

```
> vapr = rints[rints %in% new17v]
> rext = substring(s17, start(vapr), end(vapr))
> rnu = seq2id(as.character(reverseComplement(DNAStringSet(rext))))
> vallnu = c(fnu, rnu)
> all(vallnu %in% allnuc)
```

```
[1] TRUE
```

We now have all probes interrogating genes on chr17 that harbor novel variants in NA19240. Let's construct some Z-scores for expression on chr17 for NA19240. We have to play some annotation games.

```
> # connect nuid and target id used on the hmyriB36 smlSet
> allnu = dbGetQuery(con, "select nuID from HumanWG6_V1")
> allnuc = as.character(allnu[,1])
> alltarg = dbGetQuery(con, "select Target from HumanWG6_V1")
> alltarg = as.character(alltarg[,1])
> names(allnuc) = alltarg
> # get the expression data for all YRI
> library(hmyriB36)
> if (!exists("hmyriB36")) data(hmyriB36)
> library(illuminaHumanv1.db)
> # limit to chr17
> all17 = get("17", revmap(illuminaHumanv1CHR))
> fyri = featureNames(hmyriB36)
> fon17 = intersect(fyri, all17)
> ex17 = exprs(hmyriB36)[fon17,]
> lfon17 = allnuc[fon17]
> # translate rownames to nuid
> rownames(ex17) = lfon17
> # compute expression z-scores
> drop = which(colnames(ex17)=="NA19240")
> m17 = apply(ex17[,-drop],1,mean,na.rm=TRUE)
> sd17 = apply(ex17[,-drop],1,sd,na.rm=TRUE)
> z17.240 = (ex17[, "NA19240"] - m17)/sd17
> wvar = which(lfon17 %in% vallnu)

> par(mfrow = c(1, 1))
> boxplot(z17.240[wvar], z17.240[-wvar], names = c("probes w/ variant",
+     "probes w/o variant"))
```



If this code, which is too complex to be trustworthy on short acquaintance, is correct, then it seems clear that possessing variants in sequence used for measuring mRNA abundance does not have drastic effects on probe performance.

Additional extensions: Robustify the Z-scores. Generalize to a function that can operate on any chromosome or gene set. Make allowance for the nature of the variant: polyPhen predictions might be relevant.

```
> sessionInfo()

R version 2.10.0 Patched (2009-10-31 r50269)
i386-apple-darwin9.8.0

locale:
[1] C

attached base packages:
[1] splines   stats     graphics  grDevices utils     datasets  methods
```

```
[8] base

other attached packages:
 [1] illuminaHumanv1.db_1.4.0
 [2] hmyriB36_0.99.2
 [3] GGBase_3.5.8
 [4] GSEABase_1.7.4
 [5] graph_1.23.7
 [6] BSgenome.Hsapiens.UCSC.hg18_1.3.11
 [7] lumiHumanIDMapping_1.4.0
 [8] lumi_1.11.6
 [9] MASS_7.3-3
[10] preprocessCore_1.7.9
[11] mgcv_1.5-6
[12] affy_1.23.12
[13] annotate_1.23.4
[14] snpMatrix_1.9.5
[15] survival_2.35-7
[16] ind1KG_0.1.2
[17] Rsamtools_0.1.19
[18] BSgenome_1.13.16
[19] Biostrings_2.15.5
[20] org.Hs.eg.db_2.3.6
[21] RSQLite_0.7-3
[22] DBI_0.2-4
[23] AnnotationDbi_1.7.20
[24] Biobase_2.5.8
[25] IRanges_1.5.5
[26] GenomicFeatures.Hsapiens.UCSC.hg18_0.1.0
[27] GenomicFeatures_0.1.4
[28] rtracklayer_1.5.23
[29] RCurl_1.2-1
[30] bitops_1.0-4.1
[31] SNPlocs.Hsapiens.dbSNP.20080617_0.99.1

loaded via a namespace (and not attached):
[1] XML_2.6-0       affyio_1.13.5   grid_2.10.0     lattice_0.17-26
[5] nlme_3.1-96     tools_2.10.0    xtable_1.5-5
```