Genetics of gene expression: 2010 lab

VJ Carey

October 15, 2010

Contents

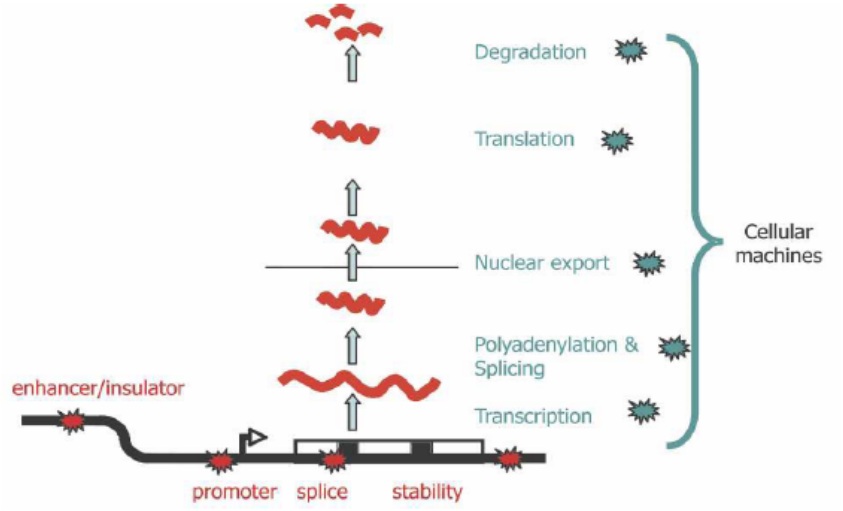


Figure 1. Plausible sites of action for genetic determinants of mRNA levels. Genetic variations influencing gene expression may reside within the regulatory sequences, promoters, enhancers, splice sites, and secondary structure motifs of the target gene and so be genetically in *cis* (red stars), or there may be variations in the molecular machinery that interact with *cis*-regulatory sequences and so act genetically in *trans* (blue stars).

Figure 1: A schematic illustrating various nonexclusive mechanisms by which DNA variants can affect transcript abundance (Williams et al., 2007).

1 eQTL concepts and discovery tools

The basic concern in the lab is the relationship between structural variation in DNA and variation in mRNA abundance. DNA variants of interest are primarily SNP as identified through

- direct genotyping in the Sanger sequencing paradigm (yielding HapMap phase II genotypes, for example)
- array-based genotyping (yielding HapMap phase III)
- NGS-based variant calling (as provided for 1000 genomes (1KG))
- hybrids of array-based and imputed genotypes (imputation to the 1KG panel)

mRNA variation is typically characterized using gene expression microarrays, but RNA-seq can also be considered.

A helpful schematic indicating possible impacts of structural variation on gene transcription is given in Williams et al. (2007); see Figure 1.

To elucidate impacts of DNA variations on regulation of gene transcription we need to be able to

- represent mRNA abundance and SNP genotypes (both observed and imputed) in a coordinated way for many samples
- perform suitably focused and calibrated tests of association between mRNA levels and genotype (e.g., allele count for a SNP)
- provide significance assessments and functional contexts for observed associations.

On the basis of excellent algorithms and software in the *snpMatrix* package of Clayton and Leung (Clayton and Leung, 2007)), the *GGtools* package provides tools to address all of these concerns.

1.1 Checking for informative variants for a specified gene

The following is a simple illustration of a focused workflow.

```
> library(GGtools)
> if (!exists("hmceuB36.2021")) data(hmceuB36.2021)
> f1 = gwSnpTests(genesym("CPNE1") ~ male, hmceuB36.2021, chrnum(20))
> topSnps(f1)
```

```
                  p.val
rs17093026 6.485612e-14
rs1118233  1.897898e-13
rs2425078  2.426168e-13
rs1970357  2.426168e-13
rs12480408 2.426168e-13
rs6060535  2.426168e-13
rs11696527 2.426168e-13
rs6058303  2.426168e-13
rs6060578  2.426168e-13
rs7273815  2.544058e-13
```

These findings can be visualized in the context of the chromosome in a so-called Manhattan plot; see Figure 2.

1.2 Components of the workflow; refinements

The primary workhorse thus far is `gwSnpTests`. We need to understand its interface and its return values.

```
> showMethods("gwSnpTests")
```

```
> plot(f1)
```

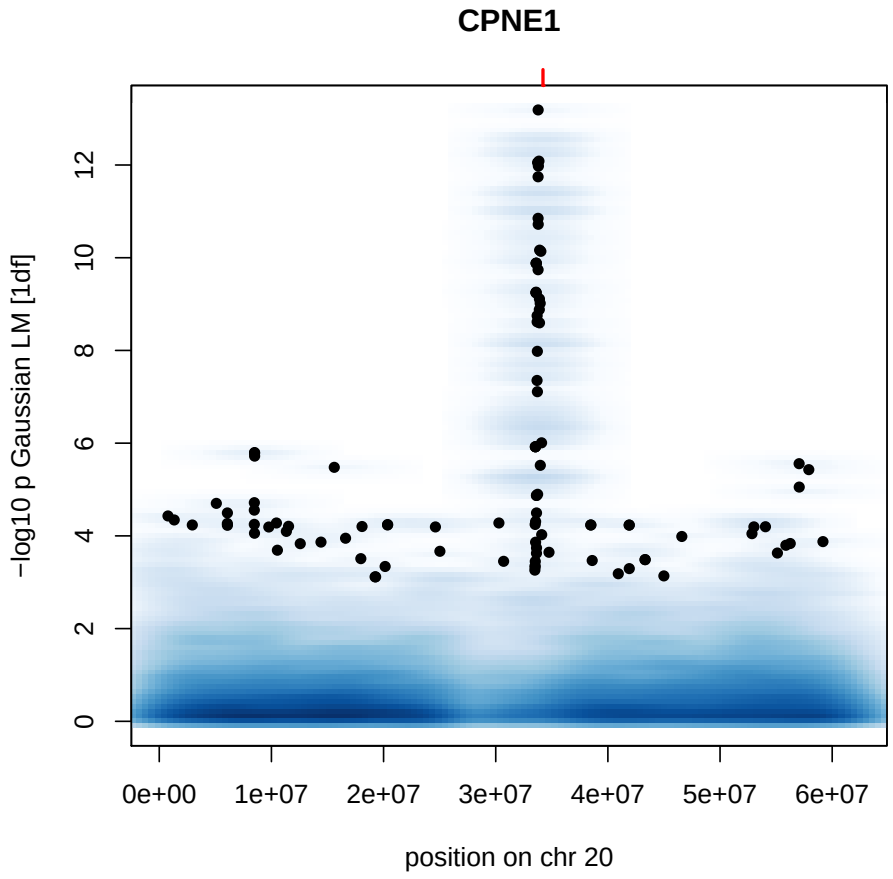


Figure 2: Manhattan plot for chromosome 20 eQTL for CPNE1. The y axis measures the significance of association of SNP allele copy number with expression for this gene. The association is measured by a $\chi^2(1)$ statistic transformed to a p-value, which is logarithmically transformed and then multiplied by -10.

```
Function: gwSnpTests (package GGtools)
sym="formula", sms="smlSet", cnum="chrnum", cs="missing"
    (inherited from: sym="formula", sms="smlSet", cnum="cnumOrMissing", cs="missing")
sym="formula", sms="smlSet", cnum="cnumOrMissing", cs="missing"
sym="formula", sms="smlSet", cnum="snpdepth", cs="chunksize"
sym="formula", sms="smlSet", cnum="snpdepth", cs="missing"
```

```
> f1
```

```
cwSnpScreenResult [chr 20 ] for gene  CPNE1  [probe  GI_23397697-A ]
```

```
> class(f1)
```

```
[1] "cwSnpScreenResult"
attr(,"package")
[1] "GGBase"
```

```
> getClass(class(f1))
```

```
Class "cwSnpScreenResult" [package "GGBase"]

Slots:

Name:        .Data      chrnum        gene        psid  annotation    testType
Class:        list      chrnum   character   character   character   character

Name:         call sessionInfo
Class:        call SessionInfo

Extends:
Class "gwSnpScreenResult", directly
Class "list", by class "gwSnpScreenResult", distance 2
Class "vector", by class "gwSnpScreenResult", distance 3
Class "AssayData", by class "gwSnpScreenResult", distance 3
```

1.2.1 Data representation: `smlSet`

The `smlSet` container provides coordinated management of expression, genotype, phenotype, and other metadata in a fashion broadly consistent with the `ExpressionSet` class. If `X` is an `smlSet` instance then

- `exprs(X)` and `pData(X)` have familiar roles of extracting a matrix of expression values, and a `data.frame` of sample-level data
- `X[,S]` subsets the samples according to predicate `S`

- `X[probeId(P),]` subsets the expression data to probes enumerated in predicate `P`
- `X[chrnum(C),]` subsets the genotype data to chromosome `C`
- `annotation(X)` returns the annotation package used for mapping expression probe identifiers

Clearly there are two kinds of feature in use here: expression probes and SNPs. The ambiguous handling of the `X[G,]` idiom has not posed a problem thus far. As the structure gains wider use, more features may be added to simplify filtering through bracket-based operations.

Operations targeted on the genotype data component include:

- `smList(X)` returns a list of `snp.matrix` objects, using a byte to encode each genotype
- `MAFfilter(X,...)` will remove SNP that fail to meet a minor allele frequency criterion
- `GTFfilter(X,...)` will remove SNP that fail to meet a minimum genotype frequency criterion

The *GGtools* package provides `hmceuB36.2021` as a modest-sized exemplar of the `smlSet` structure, with HapMap phase II genotype data from chromosomes 20 and 21, and GENEVAR expression data for 90 CEU individuals.

```
> hmceuB36.2021

snp.matrix-based genotype set:
number of samples:  90
number of chromosomes present:  2
annotation: illuminaHumanv1.db
Expression data dims: 47293 x 90
Phenodata: An object of class "AnnotatedDataFrame"
  sampleNames: NA06985 NA06991 ... NA12892 (90 total)
  varLabels: famid persid ... male (7 total)
  varMetadata: labelDescription

> names(pData(hmceuB36.2021))

[1] "famid"     "persid"    "mothid"    "fathid"    "sampid"    "isFounder"
[7] "male"

> table(hmceuB36.2021$isFounder)
```

```
FALSE  TRUE
   30    60
```

The expression data is managed exactly as in an `ExpressionSet` from *Biobase*. The genotype data has a concise representation. It and its coercions are defined in the *snpMatrix* package.

```
> s20 = smList(hmceuB36.2021)[["20"]]
> s20

A snp.matrix with  90 rows and  119921 columns
Row names:  NA06985 ... NA12892
Col names:  rs4814683 ... rs6090120

> as(s20, "matrix")[1:4,1:4]  # show raw

        rs4814683 rs6076506 rs6139074 rs1418258
NA06985        03        03        03        03
NA06991        02        03        02        02
NA06993        01        03        01        01
NA06994        01        03        01        01

> as(s20, "character")[1:4,1:4]

     [,1]  [,2]  [,3]  [,4]
[1,] "B/B" "B/B" "B/B" "B/B"
[2,] "A/B" "B/B" "A/B" "A/B"
[3,] "A/A" "B/B" "A/A" "A/A"
[4,] "A/A" "B/B" "A/A" "A/A"

> as(s20, "numeric")[1:4,1:4]

        rs4814683 rs6076506 rs6139074 rs1418258
NA06985         2         2         2         2
NA06991         1         2         1         1
NA06993         0         2         0         0
NA06994         0         2         0         0
```

Question 1: Use `plot_EvG` to visualize the specific relationship between expression and allele copy number for the best result for CPNE1 shown above.

Question 2: Create a new `smlSet` `hmff` limited to CEU founders and with genotypes having minimum genotype frequency at least 5%.

Question 3: Test for eQTL for CPNE1 in the filtered structure, saving the result in `f1f` for later conversions. Create relevant visualizations.

1.2.2 Analysis: `snp.rhs.tests`; permutation testing

The *snpMatrix* package supplied various utilities for general analysis of genome-wide association studies. `gwSnpTests` is just a wrapper for the `snp.rhs.tests` function. To see it in action,

```
> SS = (SS <- snp.rhs.tests(exprs(hmceuB36.2021)["GI_23397697-A",
+     ] ~ male, data = pData(hmceuB36.2021), snp.data = s20, family = "gaussian"))
> class(SS)

[1] "snp.tests.glm"
attr(,"package")
[1] "snpMatrix"

> length(names(SS))

[1] 119921

> sum(is.na(p.value(SS)))

[1] 53695

> SS[1:4, ]

          Chi.squared Df   p.value
rs4814683 0.071165981  1 0.7896466
rs6076506 0.002905309  1 0.9570141
rs6139074 0.680486311  1 0.4094193
rs1418258 0.077548726  1 0.7806472

> SS["rs6060535", ]

          Chi.squared Df      p.value
rs6060535    53.62528  1 2.426168e-13
```

This shows that for a given phenotypic response, large numbers of GWAS tests can be performed rapidly.

Question 4: Why do so many tests have p-value NA?

Which of our findings are significant in the context of so many tests? We can permute the expression data, keeping the genotype data fixed. Here is one way to obtain a sample from the permutation distribution of the smallest p-value for a specific gene.

```
> Mtests = sapply(1:100, function(x) topSnps(gwSnpTests(genesym("CPNE1") ~
+     male, sms = permEx(hmceuB36.2021), cnum = chrnum(20)))[[1]][1])
```

1.2.3 Context: *GenomicRanges* and *rtracklayer*

A major response to the transition to high-throughput sequencing assays is the *IRanges* infrastructure. The *GenomicRanges* package defines the `GRanges` class, to which we can convert results of an eQTL search:

```
> gr1 = as(f1, "GRanges")
> gr1[1:4, ]

GRanges with 4 ranges and 4 elementMetadata values
          seqnames          ranges strand |        type       group      score
             <Rle>       <IRanges>  <Rle> | <character> <character>  <numeric>
rs4814683    chr20 [ 9795,  9795]      * |      snpeff         gws 0.10256722
rs6076506    chr20 [11231, 11231]      * |      snpeff         gws 0.01908166
rs6139074    chr20 [11244, 11244]      * |      snpeff         gws 0.38783167
rs1418258    chr20 [11799, 11799]      * |      snpeff         gws 0.10754519
             universe
          <character>
rs4814683        hg18
rs6076506        hg18
rs6139074        hg18
rs1418258        hg18

seqlengths
 chr20
    NA

> gr2 = as(f1f, "GRanges")
```

Question 5: Export these GRanges instances as wig files and add to a UCSC browser session as custom tracks. You could do this with *rtracklayer* if your searchlist had a particular form. You should see something like Figure 3; R Track 1 is the search on all SNP; R Track 2 follows the restriction to minimum genotype frequency 5%.

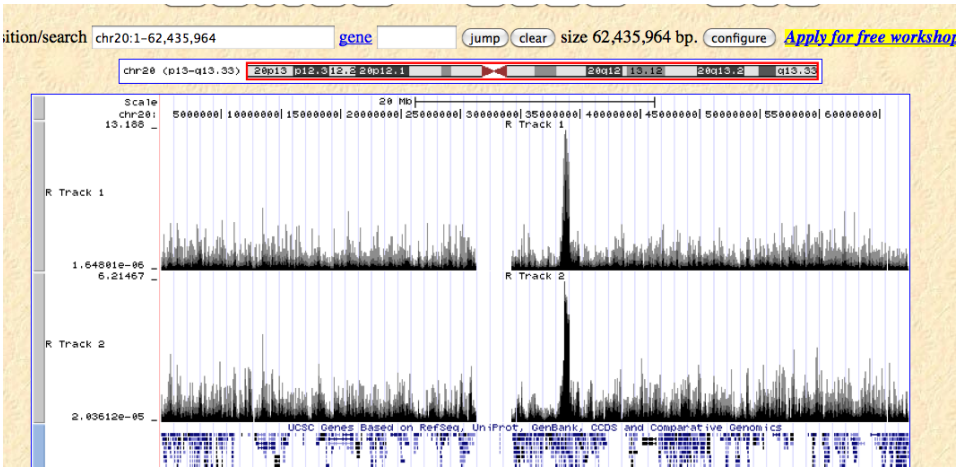


Figure 3: Browser tracks associated with eQTL screens `f1` and `f1f`.

2 Comprehensive eQTL surveys: performance issues

A comprehensive survey of eQTL would consider possible relationships between 20000 or so genes and 8 million or more SNP (the latter is the approximate size of the 1000 genomes SNP panel). Creating, storing, and interrogating the family of 160 billion results is a significant undertaking. We now illustrate a strategy for conducting comprehensive surveys.

2.1 A study of genes coresident on chromosome 20

We will select 50 genes at random from chromosome 20 and compute all tests for association with SNP on chromosome 20.

First we select the genes and filter the `smlSet`:

```
> set.seed(1234)
> library(illuminaHumanv1.db)
> g20 = get("20", revmap(illuminaHumanv1CHR))
> samp = sample(g20, size = 50, replace = FALSE)
> hlit = hmceuB36.2021[probeId(samp), ]
```

If you are using a POSIX-compliant system, you can speed up this process by distributing over cores. We will be writing data to a folder called `dem50`.

```
> try(system("rm -rf dem50"))
> if (try(require(multicore))) {
+     unix.time(e1 <- eqtlTests(hlit, ~male, geneApply = mclapply,
+         targdir = "dem50"))
+ }
```

If you do not have *multicore*, alter the code to:

```
> try(system("rm -rf dem50"))
> unix.time(e1 <- eqtlTests(hlit, ~male, geneApply = lapply, targdir = "dem50"))
```

```
   user  system elapsed
 40.464   2.146  45.885
```

The resulting object is:

```
> e1
```

```
eqtlTools results manager, computed Fri Oct 15 00:19:53 2010
There are 2 chromosomes analyzed.
some genes:  GI_26051259-I GI_42734342-S ... hmm26595-S GI_24041039-A
some snps:  rs4814683 rs6076506 ... rs6062370 rs6090120
```

```
> class(e1)

[1] "eqtlTestsManager"
attr(,"package")
[1] "GGtools"

> getClass(class(e1))

Class "eqtlTestsManager" [package "GGtools"]

Slots:

Name:       fflist        call        sess      exdate    shortfac    geneanno
Class:        list        call         ANY         ANY     numeric   character

Name:           df summaryList
Class:     numeric        list
```

We are using the *ff* package to use disk to maintain extremely voluminous outputs.

```
> e1@fflist[[1]][1:4, 1:4]

          GI_26051259-I GI_42734342-S GI_29570787-I GI_42662261-S
rs4814683            86             7            15             1
rs6076506           183            71             0            41
rs6139074            68            39             1            23
rs1418258           101             6            23             1

> e1@shortfac

[1] 100
```

The information shown above is not intended for public consumption. Instead we perform focused interrogation using brackets and casting:

```
> e1[rsid("rs4814683"), probeId("GI_26051259-I")]

$`20`
          GI_26051259-I
rs4814683          0.86
```

Typical use case: finding a collection of strongest associations:

```
> topFeats(probeId("GI_26051259-I"), mgr = e1, ffind = 1, anno = "illuminaHumanv1.db"
```

```
  rs708954   rs237422 rs11570311  rs6098015  rs6119624  rs6117923  rs6108773
     18.56      18.25      17.36      16.39      16.00      15.88      15.58
rs16987350  rs6049299  rs6018244
     15.26      14.66      14.45
```

```
> topFeats(rsid("rs708954"), mgr = e1, ffind = 1, anno = "illuminaHumanv1.db")
```

```
   KCNQ2     THBD     ATRN      SRC     NFS1 TP53INP2    PFDN4  EMILIN3
   18.56     6.97     5.13     4.13     3.20     2.99     2.86     2.74
   TCF15     CST2
    2.39     2.19
```

```
> topFeats(genesym("SRC"), mgr = e1, ffind = 1, anno = "illuminaHumanv1.db")
```

```
rs6117301 rs2983458 rs6116200  rs293554  rs293558  rs293559  rs293544  rs293543
    19.08     18.84     18.77     18.09     18.09     18.09     17.42     17.31
 rs293553 rs6141319
    17.28     16.84
```

2.2 Expanding to genes on multiple chromosomes

The analysis conducted just above uses genes on chromosome 20. We repeat with some genes on chromosome 21. First sample the genes:

```
> g21 = get("21", revmap(illuminaHumanv1CHR))
> samp21 = sample(g21, size = 50, replace = FALSE)
> hlit21 = hmceuB36.2021[probeId(samp21), ]
```

Now perform the tests, using a different target folder:

```
> try(system("rm -rf dem50.21"))
> unix.time(e2 <- eqtlTests(hlit21, ~male, geneApply = lapply,
+     targdir = "dem50.21"))
```

```
   user  system elapsed
 40.419   2.129  44.551
```

```
> e2
```

```
eqtlTools results manager, computed Fri Oct 15 00:20:38 2010
There are 2 chromosomes analyzed.
some genes:  GI_42822877-S GI_42662331-S ... GI_25952073-A GI_6912315-S
some snps:  rs4814683 rs6076506 ... rs6062370 rs6090120
```

To facilitate interrogation of the two sets of results simultaneously, the `eqtlTestsManager` instances need to be coordinated by a `cisTransDirector` instance.

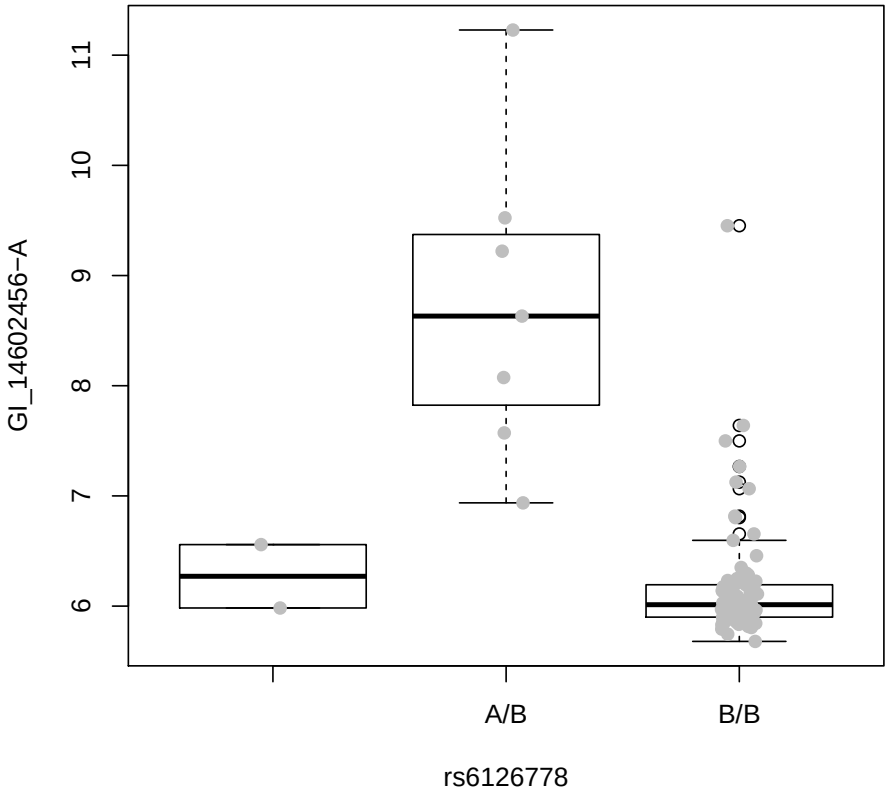
```
> try(system("rm -rf d2021_probetab.ff"))
> try(system("rm -rf d2021_snptab.ff"))
> d1 = mkCisTransDirector(list(e1, e2), "d2021", "snptab", "probetab",
+     "illuminaHumanv1.db")
> d1

eqtlTools cisTransDirector instance.
there are 2 managers.
Total number of SNP:  170086 ; total number of genes:  100
First:
eqtlTools results manager, computed Fri Oct 15 00:19:53 2010
There are 2 chromosomes analyzed.
some genes:  GI_26051259-I GI_42734342-S ... hmm26595-S GI_24041039-A
some snps:  rs4814683 rs6076506 ... rs6062370 rs6090120
---
use [ (rsnumvec), (geneidvec) ] to obtain chisq stats; topFeats(), etc.

> d1["rs6090120", "GI_42734342-S"]

          GI_42734342-S
rs6090120          1.22
```

You are now managing and interrogating 17 million test results. By increasing the numbers of genes and SNP held in the `smlSet` instances passed to `eqtlTests` you can increase this. Interfaces need further refinement.

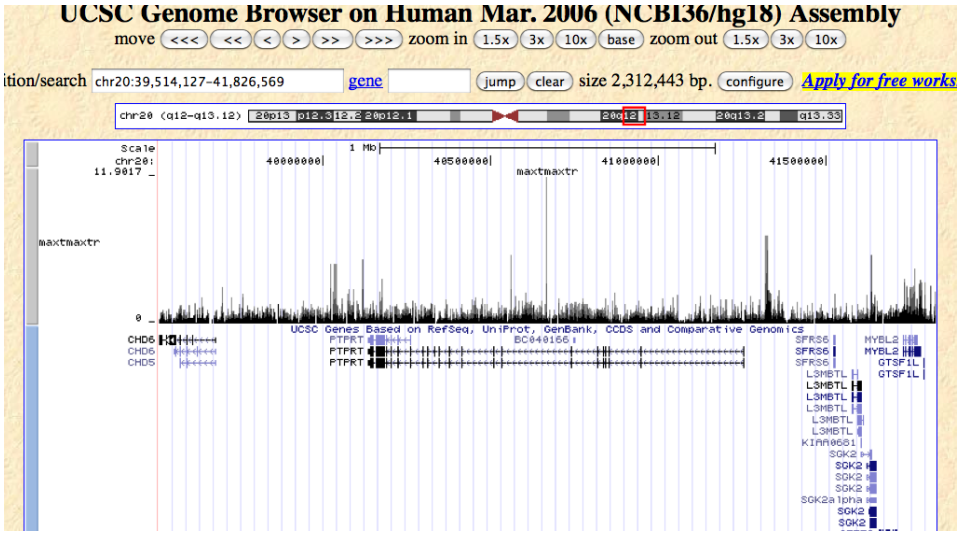
2.3 Exercises

Question 6. You can use `as.ram` from the ff package to work directly with data in `ff` arrays. You have to be careful or you will create large objects and defeat the purpose of `ff`. Show that the gene/SNP combination shown here has a special status in this series of tests.

```
 rs6126778  rs6034309  rs6123249   rs175786  rs1997924  rs6114695  rs1558384
     47.50      39.10      37.59      36.03      35.65      35.41      34.87
 rs6139509 rs13045352 rs11905049
     34.84      34.84      34.84
```



Question 7. Use `getGRanges` to obtain a Manhattan plot for the gene identified in question 6, displayed in the browser. You should see something like


UCSC Genome Browser on Human Mar. 2006 (NCBI36/hg18) Assembly
move <<< << < > >> >>> zoom in 1.5x 3x 10x base zoom out 1.5x 3x 10x
ition/search chr20:39,514,127-41,826,569 gene jump clear size 2,312,443 bp. configure Apply for free works
chr20 (q12-q13.12) 20p13 p12.3 p12.2 20p12.1 20q12 13.12 20q13.2 q13.33
Scale
chr20:
11,9017
1 Mb
40000000
40500000
41000000
41500000
maxtmaxtr
maxtmaxtr
0
UCSC Genes Based on RefSeq, UniProt, GenBank, CCDS and Comparative Genomics
CHD6
CHD6
CHD5
PTPRT
PTPRT
PTPRT
BC040166
SFRS6
SFRS6
SFRS6
L3MBTL
L3MBTL
L3MBTL
L3MBTL
L3MBTL
KIAA0661
SGK2
SGK2
SGK2
SGK2
SGK2a lpha
SGK2
SGK2
MYBL2
MYBL2
GTSF1L
GTSF1L

3 Investigating allele-specific expression with RNA-seq data

We consider a study of read mapping bias in RNA-seq (Degner et al., 2009). We will consider how to identify SNPs located in exons, how to analyze allele frequencies for such SNP, and how to check findings against existing sequencing data and eQTL statistics for a single individual and a HapMap population.

3.1 The data

We will use a samtools pileup representation of RNA-seq data that were aligned using MAQ and distributed by Degner and colleagues through GEO. We confine attention to chromosome 6 for individual NA19238, and use the "unmasked" alignment files. Data from two sequencing runs was combined using samtools merge, and then the pileup was generated against hg18.

```
> library(GGtools)
> library(Rsamtools)
> pup238_6 = readPileup(system.file("pup/combn238_chr6.pup", package = "GGtools"),
+     variant = "SNP")
> pup238_6[1:4, ]
```

```
GRanges with 4 ranges and 6 elementMetadata values
    seqnames         ranges strand | referenceBase consensusBase
       <Rle>      <IRanges>  <Rle> |      <factor>      <factor>
[1]     chr6 [40376, 40376]      * |             C             A
[2]     chr6 [43893, 43893]      * |             A             C
[3]     chr6 [84124, 84124]      * |             T             C
[4]     chr6 [84125, 84125]      * |             G             A
    consensusQuality snpQuality maxMappingQuality  coverage
           <integer>  <integer>         <integer> <integer>
[1]                4          4                 0         1
[2]                4          4                 0         1
[3]                4          4                 0         1
[4]                4          4                 0         1

seqlengths
 chr6
   NA
```

It will be useful to have the specific pileup of calls as well.

```
> getPupCalls = function(pupfn) {
+     tmp = strsplit(readLines(pupfn), "\t")
```

```
+     sapply(tmp, "[", 9)
+ }
> callp = getPupCalls(system.file("pup/combn238_chr6.pup", package = "GGtools"))
> elementMetadata(pup238_6)$callp = callp
> pup238_6[144:148, ]

GRanges with 5 ranges and 7 elementMetadata values
    seqnames           ranges strand | referenceBase consensusBase
       <Rle>        <IRanges>  <Rle> |      <factor>      <factor>
[1]     chr6 [249623, 249623]      * |             C             A
[2]     chr6 [249625, 249625]      * |             T             A
[3]     chr6 [249662, 249662]      * |             G             A
[4]     chr6 [256482, 256482]      * |             C             T
[5]     chr6 [256873, 256873]      * |             C             T
    consensusQuality snpQuality maxMappingQuality  coverage         callp
           <integer>  <integer>         <integer> <integer>   <character>
[1]              25         25                27         3     ^6A^?A^?A
[2]              60         60                38        11 AAAAAAAAAA^UA
[3]              33         33                56         2           A$A
[4]               4          4                 0         1             T
[5]               4          4                40         1          ^It

seqlengths
 chr6
   NA
```

3.2 Filtering to "coding SNP"; checking for de novo variants

We have derived a `GRanges` instance with addresses of exons on chromosome 6, using the *GenomicFeatures* facilities for extracting feature data from UCSC tables.

```
> library(GenomicFeatures)
> data(ex6)
> ex6[1:4, ]

GRanges with 4 ranges and 1 elementMetadata value
    seqnames           ranges strand |   exon_id
       <Rle>        <IRanges>  <Rle> | <integer>
[1]     chr6 [292101, 292560]      + |     83906
[2]     chr6 [304628, 304661]      + |     83907
[3]     chr6 [311880, 311962]      + |     83908
[4]     chr6 [335114, 335163]      + |     83909
```

```
seqlengths
                  chr1                   chr2 ... chr18_gl000207_random
             249250621              243199373 ...                   4262
```

```
> exp_6 = pup238_6[which(ranges(pup238_6) %in% ranges(ex6)), ]
```

How many of the SNP variants in this individual reside in exons? Restrict the pileup data to these.

```
> sum(ranges(pup238_6) %in% ranges(ex6))
```

```
[1] 2581
```

```
> isExonic = which(ranges(pup238_6) %in% ranges(ex6))
> pup238_6x = pup238_6[isExonic, ]
```

How many of these exonic variants are already catalogued in the May 2009 dnSNP database?

```
> library(SNPlocs.Hsapiens.dbSNP.20100427)
> s6 = getSNPlocs("ch6")
> s6[1:5, ]
```

```
  RefSNP_id alleles_as_ambig   loc
1    845718                Y 88813
2    845717                R 88857
3  28836829                R 88920
4  28770826                S 88972
5  56715894                R 89010
```

```
> knownLocs = IRanges(s6$loc, s6$loc)
> indbsnp = ranges(pup238_6x) %in% knownLocs
> sum(indbsnp)
```

```
[1] 17
```

What are the heterozygosity frequencies at known and potentially novel SNP locations?

```
> p6xknown = pup238_6x[which(indbsnp), ]
> p6xnovel = pup238_6x[-which(indbsnp), ]
> nKnown = length(p6xknown)
> nhetKnown = sum(!(elementMetadata(p6xknown)$consensusBase %in%
+     c("A", "C", "G", "T")))
> nhetKnown/nKnown
```

```
[1] 0
```

```
> nNovel = length(p6xnovel)
> nhetNovel = sum(!(elementMetadata(p6xnovel)$consensusBase %in%
+     c("A", "C", "G", "T")))
> nhetNovel/nNovel
```

```
[1] 0.03510140
```

3.3 Assessing allelic imbalance in transcripts harboring SNPs

What exactly are we looking for? A guide is available through the table published by Degner et al.

```
> data(degnerASE01)
> antab = with(degnerASE01, degnerASE01[chr == "chr6" & indiv ==
+     "GM19238", 1:8])
> antab
```

```
       rsnum refreads nonrefreads miscall  chr      loc     gene   indiv
2  rs1042448       72           4       0 chr6 33162320 HLA-DPB1 GM19238
29 rs3025040       56         133       7 chr6 43861029    VEGFA GM19238
46    rs7192      212         317       0 chr6 32519624  HLA-DRA GM19238
48 rs7739387        4          32       0 chr6 34730399 C6orf106 GM19238
50    rs8084      388         518       2 chr6 32519013  HLA-DRA GM19238
```

This shows 5 loci related to 4 genes for which there is evidence of allele-specific expression, in that the individual is heterozygous at the locus, but the transcript abundance measures are skewed towards one of the two alleles present.

To check for this on the basis of our pileup data, we have a little tabulation function:

```
> tabCalls = function(pup, ind) {
+     ac = as.character
+     empup = elementMetadata(pup)
+     ref = ac(empup$referenceBase[ind])
+     maqcall = ac(empup$consensusBase[ind])
+     puptag = gsub("\\^.", "", empup$callp[ind])
+     list(ref = ref, maqcall = maqcall, calls = table(strsplit(puptag,
+         "")))
+ }
```

In the following, we tind overlaps between our exonic loci and the tabulated locations, and tabulate the available calls. First we build a little utility that returns nucleotides given an ambiguous code.

```
> library(Biostrings)
> decodeIU = function(x) {
+     if (length(x) > 1)
+         warning("only handling first entry")
+     strsplit(IUPAC_CODE_MAP[toupper(x)], "")[[1]]
+ }
```

Now we identify those locations from the Degner table that coincide with our pileup-based ranges. There is an idiosyncratic but intuitive matrix result wrapped in S4.

```
> ol = findOverlaps(IRanges(antab$loc, width = 1), ranges(pup238_6x))
> ol

An object of class "RangesMatching"
Slot "matchMatrix":
     query subject
[1,]     4    1489

Slot "DIM":
[1]    5 2581
```

In the following, we generate a little report to summarize the comparisons.

```
> for (i in 1:nrow(ol@matchMatrix)) {  # ONEOFF
+  cat("query", i, "\n")
+  print(t1 <- antab[ ol@matchMatrix[i, "query"], ])
+  t2 <- tabCalls( pup238_6x, ol@matchMatrix[i, "subject"])
+  print(t2[["calls"]])
+  dpref = t1[1,2]/(t1[1,2]+t1[1,3])
+  cat("Degner refFreq =", round(dpref,4),"\n")
+  dpuprefinds = which(toupper(names(t2[["calls"]])) %in% c(",", "."))
+  targcodes = decodeIU( t2[["maqcall"]] )
+  altcode = setdiff(targcodes, t2[["ref"]])
+  dpupaltinds = which(toupper(names(t2[["calls"]])) == altcode)
+  dpuppref = sum(t2[["calls"]][dpuprefinds])/(sum(t2[["calls"]][dpuprefinds])+
+     sum(t2[["calls"]][dpupaltinds]))
+  cat("Pileup refFreq =", round(dpuppref,4),"\n")
+  cat("---\n")
+ }

query 1
       rsnum refreads nonrefreads miscall  chr      loc     gene   indiv
48 rs7739387        4          32       0 chr6 34730399 C6orf106 GM19238
```

```
 ,  .  C  c
 2  1 11  5
Degner refFreq = 0.1111
Pileup refFreq = 0.1579
---
```

The correspondence is very close. Unfortunately there is relatively low coverage for the most extreme imbalances.

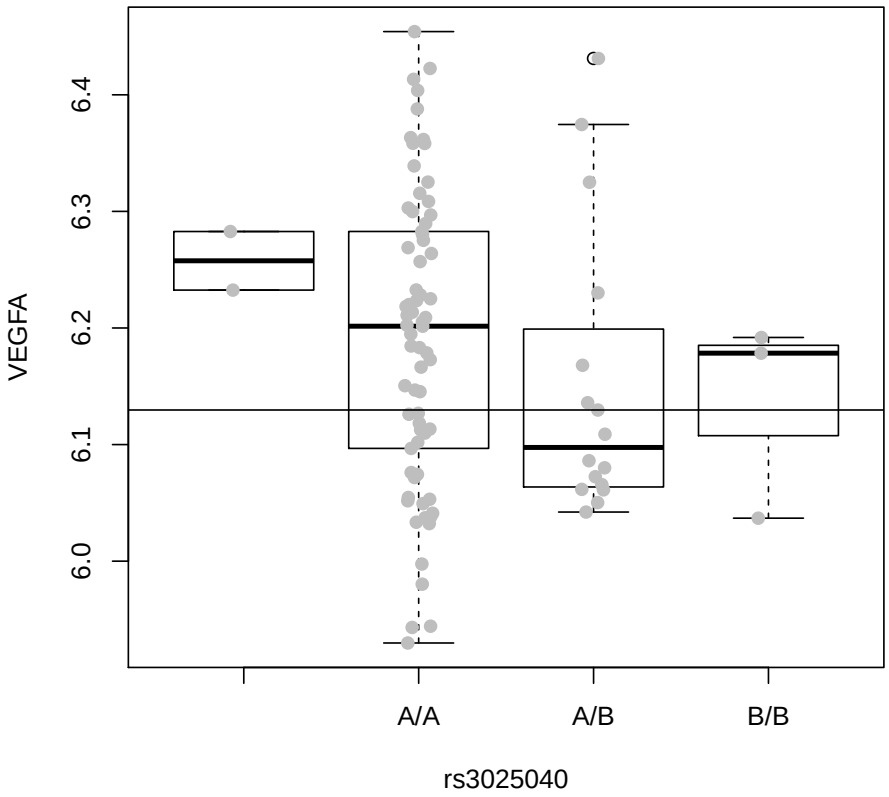
3.4 Checking consistency of findings with GENEVAR expression arrays

The hmyriB36 package of Bioconductor's experimental data archive includes expression and genotyping data on 90 YRI individuals including NA19238. The expression data are distributed by the GENEVAR project (Stranger et al., 2007).

```
> library(hmyriB36)
> if (!exists("hmyriB36")) data(hmyriB36)
> library(illuminaHumanv1.db)
> pidVEGFA = get("VEGFA", revmap(illuminaHumanv1SYMBOL))
> ve = exprs(hmyriB36)[pidVEGFA, "NA19238"]
```

The following code can be used to check for relationship between allele copy number for a SNP and array-based expression values:

```
> plot_EvG(genesym("VEGFA"), rsid("rs3025040"), hmyriB36)
> abline(h = ve)
```



3.5 Exercises

Question 8. Check the array-based expression configurations for the other four genes with potential allele-specific expression.

Question 9. Write code that scans all the exonic SNP on chr6 and identifies loci with potential imbalance. You might modify the code snippet marked `ONEOFF` above. What statistical tests and multiple testing corrections should you use?

Question 10. Acquire the data on NA19239 and reproduce some of the key entries of the Degner table.

4 Appendix: samtools pileup format, from samtools distribution site

Pileup Format

Pileup format is first used by Tony Cox and Zemin Ning at the Sanger Institute. It describes the base-pair information at each chromosomal position. This format facilitates SNP/indel calling and brief alignment viewing by eyes.

The pileup format has several variants. The default output by SAMtools looks like this:

```
seq1 272 T 24  ,.$.....,.,.,...,,,.,..^+. <<<+;<<<<<<<<<<=<;<;7<&
seq1 273 T 23  ,.....,.,.,...,,,.,..A <<<;<<<<<<<<<3<=<<<;<<+
seq1 274 T 23  ,.$...,.,.,...,,,.,....    7<7;<;<<<<<<<<<=<;<;<<6
seq1 275 A 23  ,$...,.,.,...,,,.,...^l.  <+;9*<<<<<<<<<=<<:;<<<<
seq1 276 G 22  ...T,.,.,...,,,.,....  33;+<<7=7<<7<&<<1;<<6<
seq1 277 T 22  ....,.,.,.C.,.,.,..G.  +7<;<<<<<<<&<=<<:;<<&<
seq1 278 G 23  ....,.,.,...,,,.,....^k.   %38*<<;<7<<7<=<<<;<<<<<
seq1 279 C 23  A..T,.,.,...,,,.,..... ;75&<<<<<<<<<=<<<9<<:<<
```

where each line consists of chromosome, 1-based coordinate, reference base, the number of reads covering the site, read bases and base qualities. At the read base column, a dot stands for a match to the reference base on the forward strand, a comma for a match on the reverse strand, `ACGTN' for a mismatch on the forward strand and `acgtn' for a mismatch on the reverse strand. A pattern `\+[0-9]+[ACGTNacgtn]+' indicates there is an insertion between this reference position and the next reference position. The length of the insertion is given by the integer in the pattern, followed by the inserted sequence. Here is an example of 2bp insertions on three reads:

```
seq2 156 A 11  .$......+2AG.+2AG.+2AGGG    <975;:<<<<<
```

Similarly, a pattern `-[0-9]+[ACGTNacgtn]+' represents a deletion from the reference. Here is an exmaple of a 4bp deletions from the reference, supported by two reads:

```
seq3 200 A 20 ,,,,,,.,.-4CACC.-4CACC....,.,,.^~. ==<<<<<<<<<<<::<;2<<
```

Also at the read base column, a symbol `^' marks the start of a read segment which is a contiguous subsequence
on the read separated by `N/S/H' CIGAR operations. The ASCII of the character following `^' minus 33 gives the mapping quality. A symbol `$' marks the end of a read segment. Start and end markers of a read are largely inspired by Phil Green's CALF format. These markers make it possible to reconstruct the read sequences from pileup.

5 Session information

```
> toLatex(sessionInfo())
```

- R version 2.13.0 Under development (unstable) (2010-10-12 r53293), `x86_64-apple-darwin10.4.0`
- Locale: `C`
- Base packages: base, datasets, grDevices, graphics, methods, splines, stats, tools, utils
- Other packages: AnnotationDbi 1.11.8, Biobase 2.9.2, Biostrings 2.17.47, DBI 0.2-5, GGBase 3.9.4, GGtools 3.7.66, GenomicFeatures 1.1.12, GenomicRanges 1.1.29, IRanges 1.7.35, RCurl 1.4-3, RSQLite 0.9-2, Rsamtools 1.1.17, SNPlocs.Hsapiens.dbSNP.20100427 0.99.2, annotate 1.27.1, bit 1.1-6, bitops 1.0-4.1, codetools 0.2-2, digest 0.4.2, ff 2.1-2, hmyriB36 0.99.4, illuminaHumanv1.db 1.6.0, org.Hs.eg.db 2.4.5, rtracklayer 1.9.9, snpMatrix 1.13.3, survival 2.35-8, weaver 1.15.0
- Loaded via a namespace (and not attached): BSgenome 1.17.7, GSEABase 1.11.2, KernSmooth 2.23-3, XML 3.1-1, annaffy 1.21.0, biomaRt 2.5.1, graph 1.27.25, xtable 1.5-6

References

David Clayton and Hin-Tak Leung. An r package for analysis of whole-genome association studies. *Hum Hered*, 64(1):45–51, Jan 2007. doi: 10.1159/000101422.

J Degner, J Marioni, A Pai, J Pickrell, E Nkadori, Y Gilad, and J Pritchard. Effect of read-mapping biases on detecting allele-specific expression from rna-sequencing data. *Bioinformatics*, Oct 2009. doi: 10.1093/bioinformatics/btp579.

Barbara E Stranger, Alexandra C Nica, Matthew S Forrest, Antigone Dimas, Christine P Bird, Claude Beazley, Catherine E Ingle, Mark Dunning, Paul Flicek, Daphne Koller, Stephen Montgomery, Simon Tavaré, Panos Deloukas, and Emmanouil T Dermitzakis. Population genomics of human gene expression. *Nat Genet*, 39(10):1217–24, Oct 2007. doi: 10.1038/ng2142.

R. B.H Williams, E. K.F Chan, M. J Cowley, and P. F.R Little. The influence of genetic variation on gene expression. *Genome Research*, 17(12):1707–1716, Dec 2007. doi: 10.1101/gr.6981507.