Bioc 2009 lab session: genetics of gene expression

July 25, 2009

Contents

1 Introduction

Numerous publications have investigated the relationships between SNP allele frequencies and variation in expression, (Cheung et al., 2005; Stranger et al., 2007; Dixon et al., 2007; Göring et al., 2007). Interpretation of findings in such studies is complicated (Williams et al., 2007; Kliebenstein, 2009). This lab reviews how Bioconductor resources may be used to conduct further research in this domain.

A figure from Williams (2007) illustrates some of the ways polymorphisms in DNA might affect expression variation:

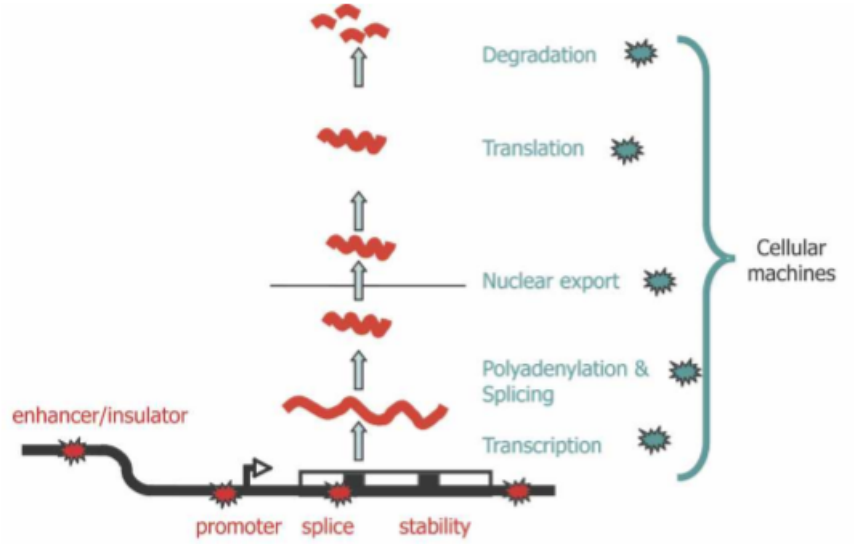


Figure 1. Plausible sites of action for genetic determinants of mRNA levels. Genetic variations influencing gene expression may reside within the regulatory sequences, promoters, enhancers, splice sites, and secondary structure motifs of the target gene and so be genetically in *cis* (red stars), or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

2 Key resources for discovering and interpreting eQTL

2.1 General metadata

2.1.1 Consensus sequence

It is often useful to have access to consensus genomic sequence. We will focus on data in humans.

```
> library(BSgenome.Hsapiens.UCSC.hg18)
> Hsapiens$chr20
```

```
  62435964-letter "MaskedDNAString" instance (# for masking)
seq: ####################################...ATTCTTCTTTGAATGTCTGATAGAATTCCGCGGATC
masks:
  maskedwidth maskedratio active names                                desc
1     2930711 0.046939469   TRUE AGAPS                         assembly gaps
2           0 0.000000000   TRUE   AMB   intra-contig ambiguities (empty)
3    29017078 0.464749419  FALSE    RM                          RepeatMasker
4      496674 0.007954934  FALSE   TRF Tandem Repeats Finder [period<=12]
all masks together:
  maskedwidth maskedratio
     31975395    0.512131
all active masks together:
  maskedwidth maskedratio
      2930711  0.04693947
```

2.1.2 Gene enumeration, location, and other mappings

Human genes are enumerated and described using a SQLite-based package that provides information from NCBI Entrez Gene.

```
> library(org.Hs.eg.db)
> cpeg = get("CPNE1", revmap(org.Hs.egSYMBOL))
> cpeg

[1] "8904"

> get(cpeg, org.Hs.egGENENAME)

[1] "copine I"

> get(cpeg, org.Hs.egCHRLOC)

       20        20
-33677381 -33677379
```

You may notice two start positions for this gene, depending on the annotation package version in use (see the `sessionInfo()` in the appendix for details.) To explain this, the documentation available through `help(org.Hs.egCHRLOC)` has:

> Since some genes have multiple start sites, this field can map to multiple locations.
>
> Mappings were based on data provided by: UCSC Genome Bioinformatics (Homo sapiens) (ftp://hgdownload.cse.ucsc.edu/goldenPath/currentGenomes/Homo_sapiens) on 2009-Sep3

The full set of mappings for Entrez Gene entities can be listed:

```
> org.Hs.eg()

Quality control information for org.Hs.eg:


This package has the following mappings:

org.Hs.egACCNUM has 29534 mapped keys (of 40596 keys)
org.Hs.egACCNUM2EG has 592232 mapped keys (of 592232 keys)
org.Hs.egALIAS2EG has 103280 mapped keys (of 103280 keys)
org.Hs.egCHR has 40359 mapped keys (of 40596 keys)
org.Hs.egCHRLENGTHS has 25 mapped keys (of 25 keys)
org.Hs.egCHRLOC has 20048 mapped keys (of 40596 keys)
org.Hs.egCHRLOCEND has 20048 mapped keys (of 40596 keys)
org.Hs.egENSEMBL has 20198 mapped keys (of 40596 keys)
org.Hs.egENSEMBL2EG has 19886 mapped keys (of 19886 keys)
org.Hs.egENSEMBLPROT has 19876 mapped keys (of 40596 keys)
org.Hs.egENSEMBLPROT2EG has 44845 mapped keys (of 44845 keys)
org.Hs.egENSEMBLTRANS has 19912 mapped keys (of 40596 keys)
org.Hs.egENSEMBLTRANS2EG has 44905 mapped keys (of 44905 keys)
org.Hs.egENZYME has 2021 mapped keys (of 40596 keys)
org.Hs.egENZYME2EG has 870 mapped keys (of 870 keys)
org.Hs.egGENENAME has 40596 mapped keys (of 40596 keys)
org.Hs.egGO has 17593 mapped keys (of 40596 keys)
org.Hs.egGO2ALLEGS has 10614 mapped keys (of 10614 keys)
org.Hs.egGO2EG has 7831 mapped keys (of 7831 keys)
org.Hs.egMAP has 36457 mapped keys (of 40596 keys)
org.Hs.egMAP2EG has 2944 mapped keys (of 2944 keys)
org.Hs.egOMIM has 14184 mapped keys (of 40596 keys)
org.Hs.egOMIM2EG has 16569 mapped keys (of 16569 keys)
org.Hs.egPATH has 4758 mapped keys (of 40596 keys)
org.Hs.egPATH2EG has 201 mapped keys (of 201 keys)
org.Hs.egPFAM has 23980 mapped keys (of 40596 keys)
org.Hs.egPMID has 28269 mapped keys (of 40596 keys)
org.Hs.egPMID2EG has 238987 mapped keys (of 238987 keys)
org.Hs.egPROSITE has 23980 mapped keys (of 40596 keys)
org.Hs.egREFSEQ has 28038 mapped keys (of 40596 keys)
org.Hs.egREFSEQ2EG has 91461 mapped keys (of 91461 keys)
org.Hs.egSYMBOL has 40596 mapped keys (of 40596 keys)
org.Hs.egSYMBOL2EG has 40579 mapped keys (of 40579 keys)
org.Hs.egUNIGENE has 24563 mapped keys (of 40596 keys)
```

```
org.Hs.egUNIGENE2EG has 25172 mapped keys (of 25172 keys)
org.Hs.egUNIPROT has 20666 mapped keys (of 40596 keys)


Additional Information about this package:

DB schema: HUMAN_DB
DB schema version: 2.0
Organism: Homo sapiens
Date for NCBI data: 2009-May7
Date for GO data: 200904
Date for KEGG data: 2009-May6
Date for Golden Path data: 2009-Sep3
Date for IPI data: 2009-May07
Date for Ensembl data: 2009-Mar6
```

More details on other annotation resources can be obtained using the *biomaRt* package.

2.1.3 SNP enumeration and location

```
> library(SNPlocs.Hsapiens.dbSNP.20080617)
> data(package = "SNPlocs.Hsapiens.dbSNP.20080617")$results[, 3]

 [1] "SNPcount"      "chr10_snplocs" "chr11_snplocs" "chr12_snplocs"
 [5] "chr13_snplocs" "chr14_snplocs" "chr15_snplocs" "chr16_snplocs"
 [9] "chr17_snplocs" "chr18_snplocs" "chr19_snplocs" "chr1_snplocs"
[13] "chr20_snplocs" "chr21_snplocs" "chr22_snplocs" "chr2_snplocs"
[17] "chr3_snplocs"  "chr4_snplocs"  "chr5_snplocs"  "chr6_snplocs"
[21] "chr7_snplocs"  "chr8_snplocs"  "chr9_snplocs"  "chrX_snplocs"
[25] "chrY_snplocs"

> c1chk = getSNPlocs("chr20")[1:10, ]
> c1chk

   RefSNP_id alleles_as_ambig   loc
1   28753379                Y  8572
2   28579812                Y  8646
3    6078030                Y  9098
4    4814683                K  9795
5    6047235                Y 10100
6   34147676                M 10731
7    6076506                K 11231
```

```
8    6139074                M 11244
9    1418258                Y 11799
10   7274499                M 12150
```

```
> apply(c1chk, 2, class)
```

```
       RefSNP_id alleles_as_ambig              loc
     "character"      "character"      "character"
```

The dbSNP identifiers are encoded without the 'rs' to save space.

We can learn the detailed allele assignments using

```
> 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"
```

Additional metadata on SNP can be obtained using biomaRt, if you have a network connection.

For example, we can get some metadata on a given dbSNP id as follows:

```
> m = useMart("snp")
> m = useDataset("hsapiens_snp", m)
> att = listAttributes(m)
> getBM(c("refsnp_id", "chr_name", "allele_1", "associated_gene",
+     "chrom_start", "chrom_strand"), filt = "refsnp", values = "rs6060535",
+     m)
```

The response is:

```
  refsnp_id chr_name allele_1 associated_gene chrom_start chrom_strand
1 rs6060535       20        C              NA    33698936            1
```

You will need to examine `att` and the results of `listFilters(m)` to determine how to make more detailed queries.

2.1.4 SNP chip annotation

We focus on the genomewide SNP 6.0 chip from Affymetrix.

```
> library(pd.genomewidesnp.6)
> objects("package:pd.genomewidesnp.6")
```

```
[1] "pd.genomewidesnp.6"
```

```
> get(objects("package:pd.genomewidesnp.6"))

Class........: AffySNPCNVPDInfo
Manufacturer.: Affymetrix
Genome Build.: NCBI Build 36
Chip Geometry: 2572 rows x  2680 columns
Annotation...:

> con6 = get(objects("package:pd.genomewidesnp.6"))@getdb()
> dbListTables(con6)

[1] "featureSet"    "featureSetCNV" "pmfeature"     "pmfeatureCNV"
[5] "sequence"      "sequenceCNV"   "sqlite_stat1"  "table_info"
```

We use SQL queries to discover table contents and attribute formats. Here we look at some probe sequences, determine the feature set in which a given probe resides, and check genomic metadata regarding this feature set.

```
> dbGetQuery(con6, "select * from sequence limit 8")

  fid offset tstrand tallele                       seq
1   7      1       f       T GTGCTTACAATACAGTGGTTTCCTT
2   8      1       f       C GTGCTTACAATACGGTGGTTTCCTT
3   9     -1       r       G CAGCTACAAATGAGAGTTTTCTAGT
4  10     -1       r       A CAGCTACAAATGAAAGTTTTCTAGT
5  11     -1       r       C CCATCATAACAGACAGTTGTATTAG
6  12     -1       r       A CCATCATAACAGAAAGTTGTATTAG
7  13     -2       r       T CGCATCAGAAAAAATGTTTTATGGC
8  14     -2       r       G CGCATCAGAAAAAAGGTTTTATGGC

> dbGetQuery(con6, "select * from pmfeature where fid = 10")

  fid strand allele fsetid pos x y
1  10      1      0 391678   1 9 0

> dbGetQuery(con6, "select * from featureSet where fsetid = 391678")

  fsetid    man_fsetid affy_snp_id dbsnp_rs_id chrom physical_pos strand
1 391678 SNP_A-1856539          NA  rs11135923     8     26116296      1
  cytoband allele_a allele_b
1    p21.2        A        G
                                                gene_assoc fragment_length
1 ENST00000380737 // intron // 0 // --- // --- // --- // ---              NA
  fragment_length2 dbsnp  cnv
1              979     0 <NA>
```

We see that this feature set is used to genotype a SNP located on chromosome 8. We can verify that the probe sequence corresponding to `fid` 10 is present, modulo the polymorphism, using Bioconductor's efficient genomic sequence representation.

```
> library(BSgenome.Hsapiens.UCSC.hg18)
> c8 = Hsapiens$chr8
> c8

  146274826-letter "MaskedDNAString" instance (# for masking)
seq: GCAATTATGACACAAAAAATTAAACAGTGCAGACTG...ATGAATCTGGGTGCTCCTGTATTGGGTGCATATATA
masks:
  maskedwidth maskedratio active names                                desc
1     3662000 0.025035067   TRUE AGAPS                        assembly gaps
2           0 0.000000000   TRUE   AMB   intra-contig ambiguities (empty)
3    68913019 0.471120157  FALSE    RM                          RepeatMasker
4      949199 0.006489148  FALSE   TRF Tandem Repeats Finder [period<=12]
all masks together:
  maskedwidth maskedratio
     72624214   0.4964915
all active masks together:
  maskedwidth maskedratio
      3662000  0.02503507

> f10s = dbGetQuery(con6, "select seq from sequence where fid = 10")[1, 1]
> f10s

[1] "CAGCTACAAATGAAAGTTTTCTAGT"

> matchPattern(reverseComplement(DNAString(f10s)), c8)

  Views on a 146274826-letter DNAString subject
subject: GCAATTATGACACAAAAAATTAAACAGTGCAGAC...GAATCTGGGTGCTCCTGTATTGGGTGCATATATA
views: NONE

> matchPattern(reverseComplement(DNAString(f10s)), c8, max.mismatch = 1)

  Views on a 146274826-letter DNAString subject
subject: GCAATTATGACACAAAAAATTAAACAGTGCAGAC...GAATCTGGGTGCTCCTGTATTGGGTGCATATATA
views:
       start      end width
[1] 26116285 26116309    25 [ACTAGAAAACTCTCATTTGTAGCTG]
```

The attempt to find a perfect match failed, but the search allowing a single mismatch succeeds.

2.1.5 Expression array metadata

We are working with the illumina expression platform (human v1).

```
> library(illuminaHumanv1.db)
> get("CPNE1", revmap(illuminaHumanv1SYMBOL))
```

```
[1] "GI_23397697-A"
```

2.2 A HapMap-based integrative genomics package for the YRI cohort

The `hmyriB36` object in the *hmyriB36* package is an instance of class `smlSet`, which coordinates transcriptwide expression measures with genomewide SNP data.

```
> library(hmyriB36)
> if (!exists("hmyriB36")) data(hmyriB36)
> hmyriB36
```

```
snp.matrix-based genotype set:
number of samples:  90
number of chromosomes present:  24
annotation: illuminaHumanv1.db
Expression data dims: 47293 x 90
Phenodata: An object of class "AnnotatedDataFrame"
  rowNames: NA18500, NA18501, ..., NA19240  (90 total)
  varLabels and varMetadata description:
    fam: NA
    samp: NA
    ...: ...
    isFounder: NA
    (8 total)
```

This object answers to methods familiar in the use of ExpressionSets:

```
> dim(exprs(hmyriB36))
```

```
[1] 47293    90
```

```
> featureNames(hmyriB36)[1:4]
```

```
[1] "GI_10047089-S" "GI_10047091-S" "GI_10047093-S" "GI_10047099-S"
```

```
> dim(pData(hmyriB36))
```

```
[1] 90  8
```

```
> annotation(hmyriB36)
```

```
[1] "illuminaHumanv1.db"
```

Genotype data is managed using structures defined in the *snpMatrix* package by David Clayton.

```
> library(snpMatrix)
> objects("package:snpMatrix")
```

```
 [1] "chi.squared"           "col.summary"           "deg.freedom"
 [4] "effect.sign"           "effective.sample.size" "epsout.ld.snp"
 [7] "filter.rules"          "glm.test.control"      "ibs.stats"
[10] "ibsCount"              "ibsDist"               "imputation.maf"
[13] "imputation.r2"         "impute.snps"           "ld.snp"
[16] "ld.with"               "misinherits"           "niceprint"
[19] "p.value"               "pair.result.ld.snp"    "plot"
[22] "plot.snp.dprime"       "pool"                  "pool2"
[25] "print.snp.dprime"      "qq.chisq"              "read.HapMap.data"
[28] "read.pedfile.info"     "read.pedfile.map"      "read.plink"
[31] "read.snps.chiamo"      "read.snps.long"        "read.snps.pedfile"
[34] "read.wtccc.signals"    "row.summary"           "sample.size"
[37] "single.snp.tests"      "snp.cbind"             "snp.cor"
[40] "snp.imputation"        "snp.lhs.tests"         "snp.post"
[43] "snp.pre"               "snp.rbind"             "snp.rhs.tests"
[46] "summary"               "switch.alleles"        "tdt.snp"
[49] "test.allele.switch"    "write.snp.matrix"      "xxt"
```

We use the `snp.matrix` class to manage high-density genotypes:

```
> snps(hmyriB36, chrnum("1"))
```

```
A snp.matrix with  90 rows and  305929 columns
Row names:  NA18500 ... NA19240
Col names:  rs10399749 ... rs7534839
```

```
> class(snps(hmyriB36, chrnum("1")))
```

```
[1] "snp.matrix"
```

```
> names(smList(hmyriB36))
```

```
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15"
[16] "16" "17" "18" "19" "20" "21" "22" "X"  "Y"
```

Raw bytes are used to represent genotypes:

```
> snps(hmyriB36, chrnum("1"))@.Data[1:4, 1:4]

        rs10399749 rs2949421 rs2691310 rs4030303
NA18500         01        03        03        01
NA18501         01        03        03        01
NA18502         01        03        03        01
NA18503         01        03        03        01
```

Various coercions can be performed:

```
> as(snps(hmyriB36, chrnum("1"))[1:5, 1:5], "character")

        rs10399749 rs2949421 rs2691310 rs4030303 rs4030300
NA18500 "A/A"      "B/B"     "B/B"     "A/A"     "A/A"
NA18501 "A/A"      "B/B"     "B/B"     "A/A"     "A/A"
NA18502 "A/A"      "B/B"     "B/B"     "A/A"     "A/A"
NA18503 "A/A"      "B/B"     "B/B"     "A/A"     "A/A"
NA18504 "A/A"      "B/B"     "B/B"     "A/A"     "A/A"

> as(snps(hmyriB36, chrnum("1"))[1:5, 1:5], "numeric")

        rs10399749 rs2949421 rs2691310 rs4030303 rs4030300
NA18500          0         2         2         0         0
NA18501          0         2         2         0         0
NA18502          0         2         2         0         0
NA18503          0         2         2         0         0
NA18504          0         2         2         0         0
```

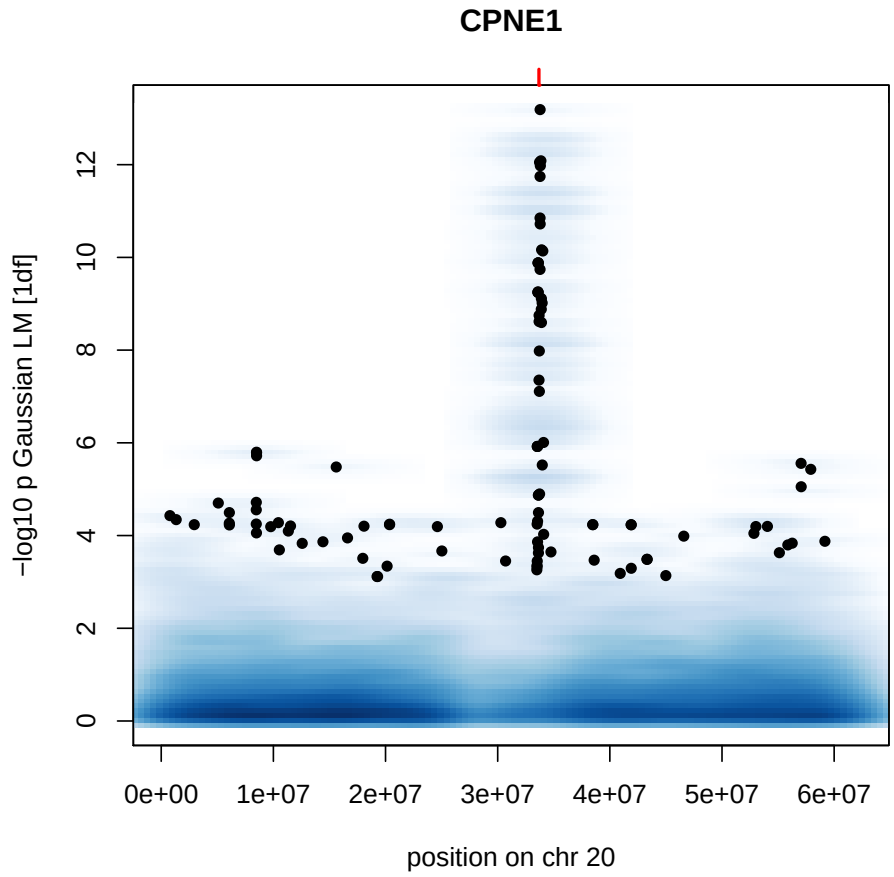
The latter computation gives us the number of copies of the B allele. We also have

```
> table(getAlleles(hmyriB36, rsid("rs10399749")))

  A/A
  7  83
```

2.3 GGtools and rtracklayer: eQTL discovery and context

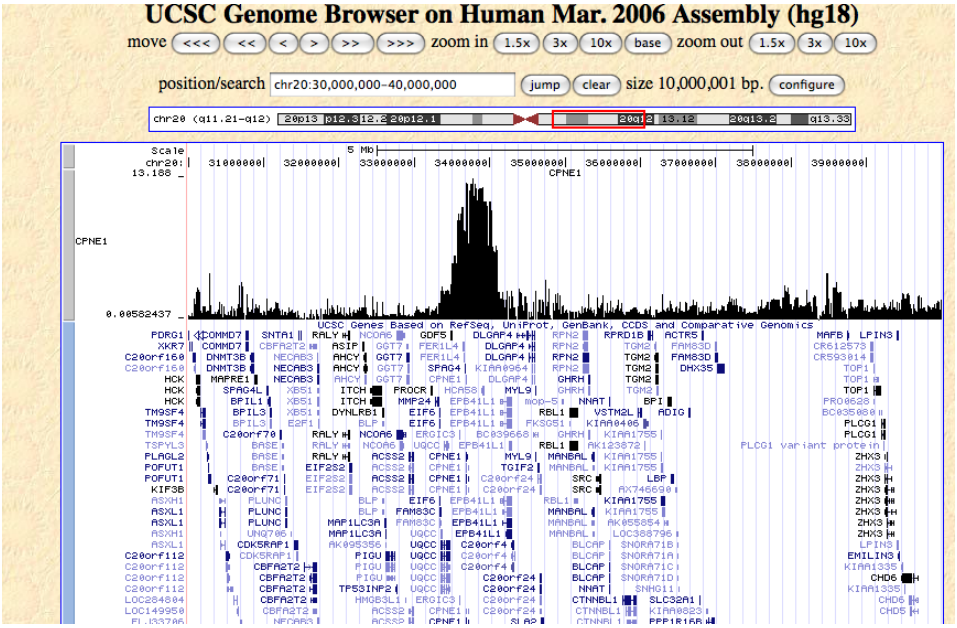
We now illustrate how to search for expression QTL. We focus on gene copine I, resident on chromosome 20, using the CEU cohort.

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

                  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

> tsn = rownames(tt)[1]

> plot(f1)
```



The following code will create a view of the UCSC genome browser:

```
> library(rtracklayer)
> f1d = as(f1, "RangedData")
> s1 = browserSession("UCSC")
> s1[["CPNE1"]] = f1d
> v1 = browserView(s1, GenomicRanges(3e+07, 4e+07, "chr20"))
```



2.4 Problems

2.4.1 Sample filtering and validity

The use of simple linear regression tests to assess eQTLs with the 90 CEPH CEU participants is invalid because of the familial structure present.

a. Use the phenoData of hmceuB36.2021 to reduce the data to the parental units of the various families (use the isFounder variable). Recompute the chromosome-wide tests for eQTL for CPNE1 and compare to the results shown above.

b. A test that is valid (*a priori*) under the assumption that trios are independent could be obtained if a random effect of family were incorporated. Use `lme` to carry this out for the top scoring SNP. Compare the new p-value to the one naively obtained above.

Warning: the analyses performed in part b might make certain assumptions about the samples with missing genotypes. Be careful.

c. The mixed effects model allows estimation of the intrafamilial correlation of expression after adjustment for covariates. What is the effect of adjusting for genotype on estimated intrafamilial correlation of CPNE1?

2.4.2 SNP filtering and robustness

a. Use code resembling

```
> as(snps(hmceuB36.2021, chrnum(20))[, {
+     top_rsnum
+ }], "character")
```

to determine the distribution of genotypes for the putative eQTL determined using all 90 samples above.

b. Use `plot_EvG` to see the distribution of CPNE1 stratified by genotype for the top SNP.

c. Use `MAFfilter` to limit SNPs tested to those with minor allele frequency greater than 20%. Recompute the top SNP list. Perform again with a 10% minor allele frequency limit and compare to the original list.

d. (Requires programming.) Emulate the code for MAFfilter in GGBase to define a GTFfilter that filters on the frequencies of genotypes rather than alleles. Apply to the CPNE1 screen and assess the effects of choice of the inclusion threshold.

2.4.3 Somatic deletion

In a multipopulation survey of copy number variation, a condition of "somatic deletion" present in cultured cell lines was described (**?**):

> [W]e sought signals of somatic deletions within the SNP genotypes of HapMap trios. A somatic deletion in a parental genome manifests as a cluster of SNPs at which alleles present in the offspring are not found in either parent. We assessed all of our preliminary CNV calls in 120 trio parents and found that 17 (of 4,758) fell in genomic regions that harbour highly significant clusters of HapMap Phase II SNP genotypes compatible with a somatic deletion in a parental genome (Supplementary Table 5A, Supplementary Fig. 5 and Supplementary Note).

The following code can be used to check a trio structure for evidence of somatic deletion at a specific SNP.

```
> somDel = function (pan, fou, ind)
+ {
+ # pan is matrix with three rows of genotype data in form
+ # "[AB]/[AB]" (or ""); length(fou)==3, and fou == 1 for rows corresponding
+ # to parents; ind is the column to be checked
+     parg = pan[which(fou == 1), ind]  # parental genotypes
+     if (any(nchar(parg) == 0))
+         return(NA)   # bail out if parental genotypes incomplete
+     para = unique(unlist(strsplit(as.character(parg), "/")))
+     offg = pan[which(fou == 0), ind]  # offspring genotype
+     offa = unique(unlist(strsplit(as.character(offg), "/")))
+     isTRUE(!all(offa %in% para))   # return TRUE if an offspring allele is de novo
+ }
```

A quick application:

```
> f1.20 = snps(hmyriB36, chrnum("20"))[1:3, ]
> cf1.20 = as(f1.20, "character")
> psomd = which(sapply(1:5500, function(x) somDel(cf1.20, hmyriB36$isFounder[1:3],
+     x)))
> cf1.20[, psomd[1]]

NA18500 NA18501 NA18502
  "A/B"   "B/B"   "B/B"
```

What is the total number of possible somatic deletion events for the first family's cell lines? Create a more general checking tool that operates over several families. Do locations of putative somatic deletion events cluster in genomic coordinates?

2.4.4 gseQTL – gene set expression QTLs

The *gglabpack* package (distributed only for this lab) includes a serialization of the Broad's MSIGDB 2.5.

```
> library(gglabpack)
> data(msig2.5)
> irf2set = msig2.5[["V$IRF2_01"]]
> details(irf2set)

setName: V$IRF2_01
geneIds: MAPK6, ATF3, ..., DDX58 (total: 130)
geneIdType: Symbol
collectionType: Broad
  bcCategory: c3 (Motif)
  bcSubCategory:  NA
setIdentifier: c3:1047
description: Genes with promoter regions [-2kb,2kb] around transcription start site con
  (longDescription available)
organism: Human,Mouse,Rat,Dog
pubMedIds:
urls: msigdb_v2.5.xml
contributor: Xiaohui Xie
setVersion: 0.0.1
creationDate: Mon Jul  6 11:51:51 2009
```

Rohan Williams and colleagues (Williams et al., 2007) report a "regulon analysis" showing that genes with the IRF2 binding motif frequently exhibit eQTL on chromosome 8 when assayed in liver.

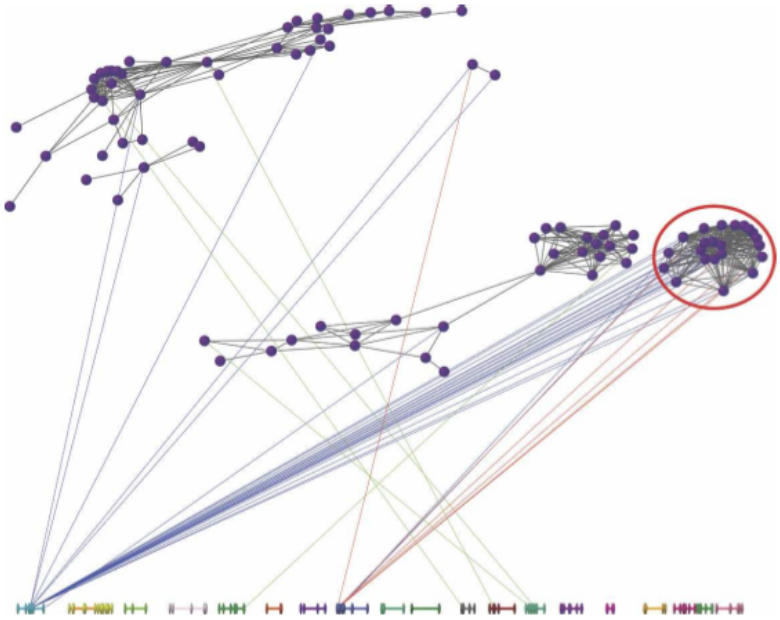


Figure 2. Regulon analysis of genes. Following sparse latent factor analysis, each gene is represented as a purple vertex connected to other genes by a gray line if the posterior probability of being correlated across the three tissues is ≥0.90. Linkage is drawn to the chromosomes below (1–19, X, *left* to *right*) if $P < 10^{-4}$: line color indicates the relevant tissue (blue, brain; green, kidney; red, liver). Note the cluster of genes on the *right* (circled), influenced by chromosome 1 in the brain, but chromosome 8 in the liver. In this cluster of genes, there is enrichment for the Irf2-binding motif; which is also located within the linkage region on chr 8; and is expressed in all three tissues.

Using the Yoruba data, check whether this might be true for genes assayed in EBV-transformed B-cells. You can use a transformation of `irf2set` to an annotation-identifier gene set object as dependent variable in `gwSnpTests`. Limit testing to chromosome 8, and eliminate 75% of genes with low variation to save time.

Here is the gene set and its modification:

```
> irf2sa = irf2set
> geneIdType(irf2sa) = AnnotationIdentifier("illuminaHumanv1.db")
```

We need the coercion method

```
> setAs("smlSet", "ExpressionSet", function(from) {
+     ex = exprs(from)
+     pd = phenoData(from)
+     ans = new("ExpressionSet", exprs = ex, phenoData = pd)
+     annotation(ans) = from@annotation
+     ans
+ })
> yex = as(hmyriB36, "ExpressionSet")
```

and then invoke filtering:

```
> library(genefilter)
> yex = yex[irf2sa, ]
> yex = nsFilter(yex, require.entrez = FALSE, remove.dupEntrez = FALSE,
+     var.cutoff = 0.75)[[1]]
> yrlit = hmyriB36[chrnum(8), ]
> yrlit@assayData = assayDataNew("lockedEnvironment", exprs = exprs(yex))
> irf2filt = GeneSet(geneIds = featureNames(yrlit))
> geneIdType(irf2filt) = AnnotationIdentifier("illuminaHumanv2.db")
```

Now we can check all genes in the filtered gene set for eQTL.

```
> irft8l = gwSnpTests(irf2filt ~ male, yrlit, chrnum(8))
> irft8l
```

3 Getting acquainted with affy's genomewide 6.0 chip

3.1 Raw import and visualization

The hapmapsnp6 experimental data package includes CEL files from three 6.0 chips, to which DNA from three HapMap CEU samples were hybridized. Here instead we use gglabpack (custom for the course) as a source of CEL files from YRI.

```
> library(affyio)
> dir(system.file("celfiles", package = "gglabpack"))

[1] "NA18500_GW6_Y.CEL" "NA18501_GW6_Y.CEL" "NA18502_GW6_Y.CEL"
```

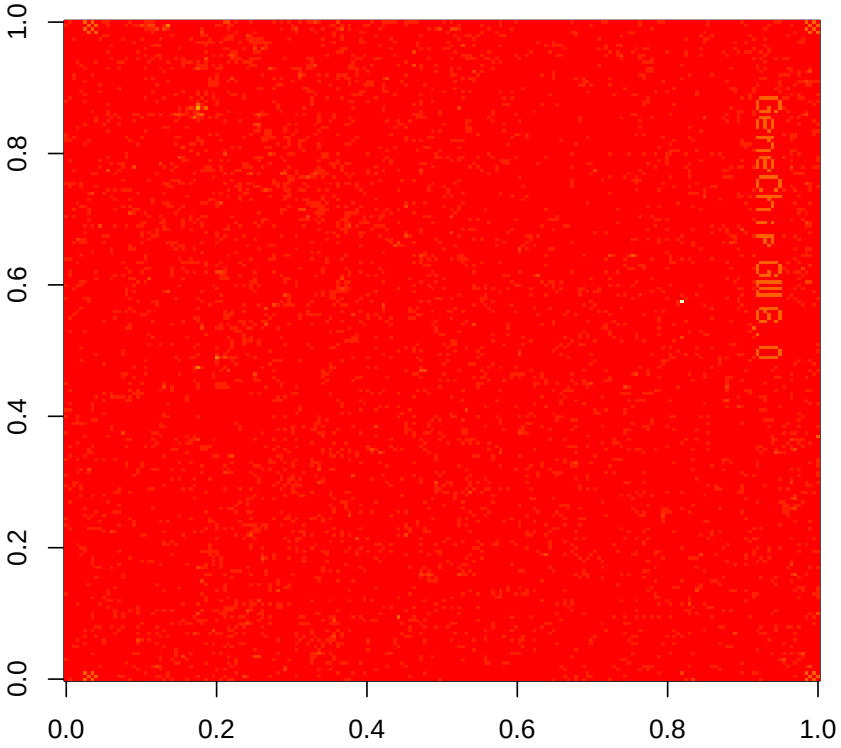
We can use standard CEL file import tools to import:

```
> library(affyio)
> f1 = dir(system.file("celfiles", package = "gglabpack"), full = TRUE)[1]

> cel1 = read.celfile(f1)

> names(cel1)

[1] "HEADER"    "INTENSITY" "MASKS"     "OUTLIERS"

> names(cel1[[2]])

[1] "MEAN"    "STDEV"   "NPIXELS"
```

Our job is to create meaningful estimates of genotypes and copy numbers using the intensity data.

As a sanity check, we can recover some etching on the chip:

```
> image(matrix(cel1[[2]]$MEAN, nr = 2680)[200:1, 2372:2572])
```



3.2 Grouping intensities; the M vs S plot

The mean intensities will be used directly. We have a small quantity of metadata derived from the pd.genomewidesnp.6 package.

```
> library(gglabpack)
> data(affmeta10k)
> affmeta10k[1:3, ]
```

```
  fsetid    man_fsetid affy_snp_id dbsnp_rs_id chrom physical_pos strand    x
1      1 SNP_A-2131660          NA   rs2887286     1      1145994      0 1688
```

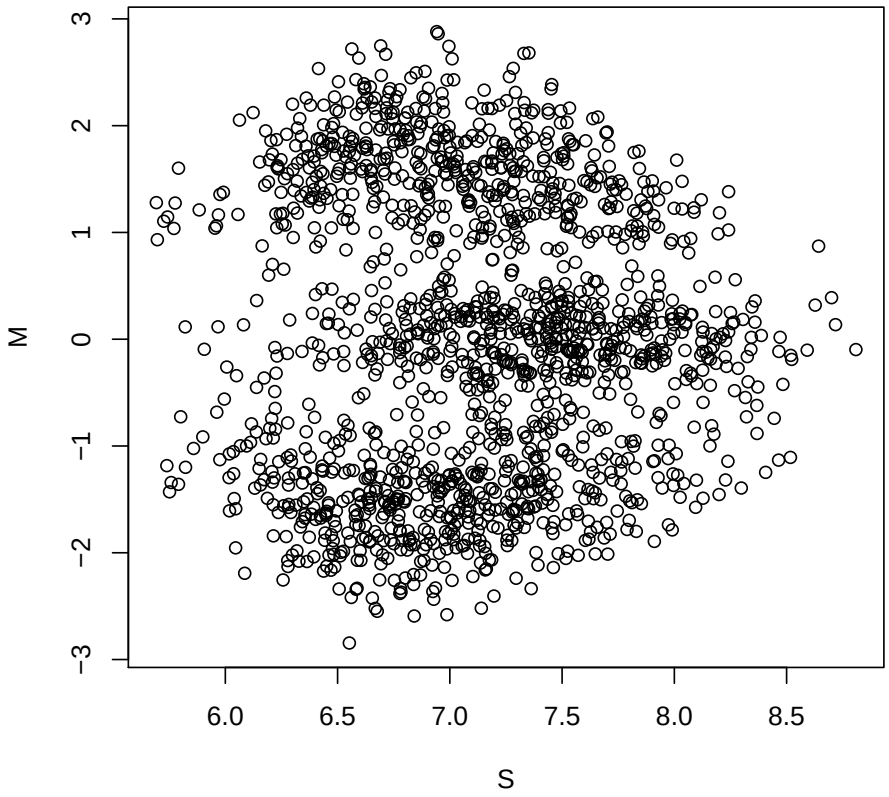
```
2      1 SNP_A-2131660          NA   rs2887286     1      1145994      0 1689
3      1 SNP_A-2131660          NA   rs2887286     1      1145994      0  262
    y pos     fid allele allele_a allele_b cytoband
1 218   6  585929      1        C        T   p36.33
2 218   5  585930      0        C        T   p36.33
3 938   4 2514103      1        C        T   p36.33
```

The `fid` column indexes the raw intensities, which can be grouped by allele within featureset.

```
> int = cel1[[2]]$MEAN
> oint = int[affmeta10k$fid]
> isA = which(affmeta10k$allele == 0)
> isB = which(affmeta10k$allele == 1)
> oint.A = oint[isA]
> oint.B = oint[isB]
> set.A = affmeta10k$man_fsetid[isA]
> set.B = affmeta10k$man_fsetid[isB]
> sintA = split(oint.A, set.A)
> sintB = split(oint.B, set.B)
> medA = sapply(sintA, median)
> medB = sapply(sintB, median)
> medA[1:4]

SNP_A-1782432 SNP_A-1786207 SNP_A-1813299 SNP_A-1815814
          645           358          1496          2181

> medB[1:4]

SNP_A-1782432 SNP_A-1786207 SNP_A-1813299 SNP_A-1815814
          724          2453           207          1975

> M = log(medA) - log(medB)
> S = 0.5 * (log(medA) + log(medB))
```

The M vs S plot is an analog of the M vs A plot that we saw with expression arrays. It shows data on many SNP on one chip.

```
> plot(S, M)
```



3.3 crlmm for genotyping

The latest genotyping algorithm due to Irizarry, Carvalho and colleagues at Johns Hopkins SPH, is called CRLMM. It can produce calls at a rate of 10000 chips/day. Here we apply it to three CEL files.

```
> library(crlmm)
> allf = dir(system.file("celfiles", package = "gglabpack"), patt = "CEL$",
+     full = TRUE)

> c1 = crlmm(allf)

  |
  |=======================                                               |  33%
  |
  |===============================================                       |  67%
  |
```

```
  |======================================================================| 100%
Calling 906600 SNPs for recalibration... Calling 906600 SNPs...
```

```
> getClass(class(c1))
```

```
Class "SnpSet" [package "Biobase"]

Slots:

Name:           assayData          phenoData        featureData
Class:          AssayData AnnotatedDataFrame AnnotatedDataFrame

Name:      experimentData         annotation       protocolData
Class:              MIAME          character AnnotatedDataFrame

Name:   .__classVersion__
Class:           Versions

Extends:
Class "eSet", directly
Class "VersionedBiobase", by class "eSet", distance 2
Class "Versioned", by class "eSet", distance 3
```

```
> calls(c1)[1:10, ]
```

	NA18500_GW6_Y.CEL	NA18501_GW6_Y.CEL	NA18502_GW6_Y.CEL
SNP_A-2131660	2	2	2
SNP_A-1967418	3	3	3
SNP_A-1969580	3	3	3
SNP_A-4263484	1	2	1
SNP_A-1978185	2	2	1
SNP_A-4264431	2	2	1
SNP_A-1980898	3	3	2
SNP_A-1983139	1	1	1
SNP_A-4265735	1	1	1
SNP_A-1995832	3	3	3

3.4 Problems

3.4.1 M vs S granularity

What is the difference between this approach to the M vs S plot and the one given above?

```
> data(affmeta10k)
> affmeta10k[1:3, ]
> c1 = dir(system.file("celfiles", package = "gglabpack"), full = TRUE)[1]
> library(affyio)
> rc1 = read.celfile(c1)
> data(affmeta10k)
> ss = split(affmeta10k, affmeta10k[, 2])
> fid1 = lapply(ss, function(x) x$fid[x$allele == 1])
> fid0 = lapply(ss, function(x) x$fid[x$allele == 0])
> int1 = sapply(1:1611, function(x) rc1[[2]]$MEAN[fid1[[x]]])
> int0 = sapply(1:1611, function(x) rc1[[2]]$MEAN[fid0[[x]]])
> uint1 = unlist(int1)
> uint0 = unlist(int0)
> plot(0.5 * log(uint1 * uint0), log(uint1/uint0))
```

3.4.2 Calling via clustering

Use k-means clustering on an M vs S plot for a HapMap sample to call genotypes for SNP identified in affmeta10k. Compare your calls to those of crlmm.

4 Imputation

Missing data are common in genotyping studies, and when data from different platforms or populations are to be combined, harmonization of SNP panels by using the intersection of sets of SNPs covered entails loss of information. The correlation structure among SNPs can be exploited to informatively impute missing genotypes.

The snp.imputation man page from snpMatrix package reads in part:

> The routine first carries out a series of step-wise regression analyses in which each Y SNP is regressed on the nearest 'try' regressor (X) SNPs. If 'phase' is 'TRUE', the regressions are calculated at the chromosome (haplotype) level, variances being simply p(1-p) and covariances estimated using the same algorithm used in 'ld.snp'. Otherwise, the analysis is carried out at the diplotype level based on conventional variance and covariance estimates using the '"all.obs"' missing value treatment (see 'cov'). New SNPs are added to the regression until either (a) the value of R^2 exceeds the first parameter of 'stopping', (b) the number of "tag" SNPs has reached the maximum set in the second parameter of 'stopping', or (c) the change in R^2 does not achieve the target set by the third parameter of 'stopping'. If the third parameter of 'stopping' is 'NA', this last test is replaced by a

```
     test for improvement in the Akaike information criterion (AIC).

     If the prediction  as measure by  R^2, has not achieved a
     threshold (the first parameter of 'use.hap') using more than one
     tag SNP, then a second imputation method is tried. Phased
     haplotype frequencies are estimated for the Y SNP plus the tag
     SNPs. The  R^2 for prediction of the Y SNP using these haplotype
     frequencies is then calculated. If the (1-R^2) is reduced by a
     proportion exceeding the second parameter of 'use.hap', then the
     haplotype imputation rule is saved in preference to the faster
     regression rule. The argument 'em.cntrl' controls convergence
     testing for the EM algorithm for fitting haplotype frequencies.
     The first parameter is the maximum number of iterations, and the
     second parameter is the threshold for the change in log likelihood
     below which the iteration is judged to have converged.
```

SNP locations are important for this process (distance may be specified by base-pair addressing or linkage disequilibrium measures). Here add a little bit of code to compute base-pair addresses.

```
> snpAddr = function(x, c) {
+     df = getSNPlocs(c)
+     intg = as.integer(gsub("rs", "", x))
+     ans = df[which(df[, 1] %in% intg), ]
+     nms = paste("rs", ans[, 1], sep = "")
+     locs = ans[, 3]
+     names(locs) = nms
+     locs
+ }
```

How does this work with the YRI data? Let's see how well this imputation procedure works to recover known genotypes.

First we use snpMatrix to develop the imputation rules:

```
> h20p = MAFfilter(hmyriB36[chrnum(20),], lower=.01)
> y20 = snps(h20p, chrnum("20"))
> set.seed(1234)
> y20dropinds = sample(1:ncol(y20), size=5)
> dropids = colnames(y20)[y20dropinds]
> dropmat = y20[,y20dropinds]
> droplocs = snpAddr(dropids, "chr20")
> kpids = colnames(y20)[-y20dropinds]
> kpmat = y20[,-y20dropinds]
> kplocs = snpAddr(kpids, "chr20")
```

```
> # not all snps have locations
> impeq = snp.imputation(kpmat[,names(kplocs)], dropmat, kplocs, droplocs,
+   stopping=c(.98, 4, NA))
> impeq

rs6085565  ~  rs6117275 (MAF = 0.02247191, R-squared = 1)
rs1033643  ~  rs2425464+rs3092525+rs6072574+rs2206419 (MAF = 0.07777778, R-squared = 0.
rs12625460  ~  rs6102897+rs1157347+rs2064677+rs6102891 (MAF = 0.1222222, R-squared = 0.
rs6093682  ~  rs6030354+rs13039393+rs6102919+rs6030350 (MAF = 0.2333333, R-squared = 0.
rs6099033  ~  rs6069635 (MAF = 0.01666667, R-squared = 1)
```

Now perform the imputations based on the kept SNP:

```
> imps = impute.snps(impeq, kpmat)
> dim(imps)

[1] 90  5

> table(getAlleles(h20p, rsid(dropids[4])), round(imps[, 4], 0))

      0  1  2
  A/A  1  3  0
  A/B  1 30  3
  B/B  0  7 41
```

Other predictive procedures can be constructed for this task. In the following we use physical distance to identify candidate predictors and then let CART prune to a parsimonious set.

```
> impByTree = function(targ, radius, sms, chrnum, ...) {
+     if (!is(targ, "rsid"))
+         stop("targ must be instance of rsid")
+     preds = snpsNear(rsid(targ), radius, chrnum)
+     snm = smList(sms)[[chrnum]]
+     actual = c(targ, intersect(preds, colnames(snm)))
+     df = data.frame(as(snm[, actual], "character"))
+     fmla = x ~ .
+     fmla[[2]] = as.name(as(targ, "character"))
+     rpart(fmla, data = df, ...)
+ }
```

Appendix

```
> sessionInfo()

R version 2.10.0 Under development (unstable) (2009-06-22 r48823)
x86_64-apple-darwin9.7.0

locale:
[1] C

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

other attached packages:
 [1] genomewidesnp6Crlmm_1.0.2
 [2] affyio_1.13.3
 [3] genefilter_1.25.7
 [4] gglabpack_0.0.3
 [5] crlmm_1.3.15
 [6] GGtools_3.3.4
 [7] annaffy_1.17.1
 [8] KEGG.db_2.3.0
 [9] GO.db_2.3.0
[10] rpart_3.1-44
[11] RColorBrewer_1.0-2
[12] hmyriB36_0.99.1
[13] GGBase_3.5.5
[14] snpMatrix_1.9.1
[15] survival_2.35-4
[16] GSEABase_1.7.1
[17] graph_1.23.3
[18] annotate_1.23.1
[19] illuminaHumanv1.db_1.3.2
[20] pd.genomewidesnp.6_0.4.2
[21] oligoClasses_1.7.7
[22] SNPlocs.Hsapiens.dbSNP.20080617_0.99.1
[23] org.Hs.eg.db_2.3.0
[24] RSQLite_0.7-1
[25] DBI_0.2-4
[26] AnnotationDbi_1.7.7
[27] Biobase_2.5.5
[28] BSgenome.Hsapiens.UCSC.hg18_1.3.11
```

```
[29] BSgenome_1.13.10
[30] Biostrings_2.13.28
[31] IRanges_1.3.43
[32] weaver_1.11.0
[33] codetools_0.2-2
[34] digest_0.3.1

loaded via a namespace (and not attached):
[1] KernSmooth_2.23-2    XML_2.5-3            ellipse_0.3-5
[4] mvtnorm_0.9-7        preprocessCore_1.7.4 xtable_1.5-5
```

References

Vivian G Cheung, Richard S Spielman, Kathryn G Ewens, Teresa M Weber, Michael Morley, and Joshua T Burdick. Mapping determinants of human gene expression by regional and genome-wide association. *Nature*, 437(7063):1365–9, Oct 2005. doi: 10.1038/nature04244.

Anna L Dixon, Liming Liang, Miriam F Moffatt, Wei Chen, Simon Heath, Kenny C C Wong, Jenny Taylor, Edward Burnett, Ivo Gut, Martin Farrall, G Mark Lathrop, Gonçalo R Abecasis, and William O C Cookson. A genome-wide association study of global gene expression. *Nat Genet*, 39(10):1202–1207, Oct 2007. doi: 10.1038/ng2109.

Harald H H Göring, Joanne E Curran, Matthew P Johnson, Thomas D Dyer, Jac Charlesworth, Shelley A Cole, Jeremy B M Jowett, Lawrence J Abraham, David L Rainwater, Anthony G Comuzzie, Michael C Mahaney, Laura Almasy, Jean W Maccluer, Ahmed H Kissebah, Gregory R Collier, Eric K Moses, and John Blangero. Discovery of expression qtls using large-scale transcriptional profiling in human lymphocytes. *Nat Genet*, 39(10):1208–1216, Oct 2007. doi: 10.1038/ng2119.

Dan Kliebenstein. Quantitative genomics: analyzing intraspecific variation using global gene expression polymorphisms or eqtls. *Annual review of plant biology*, 60:93–114, Jan 2009. doi: 10.1146/annurev.arplant.043008.092114.

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):1217–1224, 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.