ISMB 2011: Tutorial on genetics of gene expression in humans: data structures, algorithms, inference – version for R 2.15/GGtools 4.2.x

VJ Carey `stvjc at channing.harvard.edu`

January 27, 2012

Contents

1 Basic concepts with array-based data: cis-associated variants

1.1 Functional relations between DNA variants and mRNA abundance

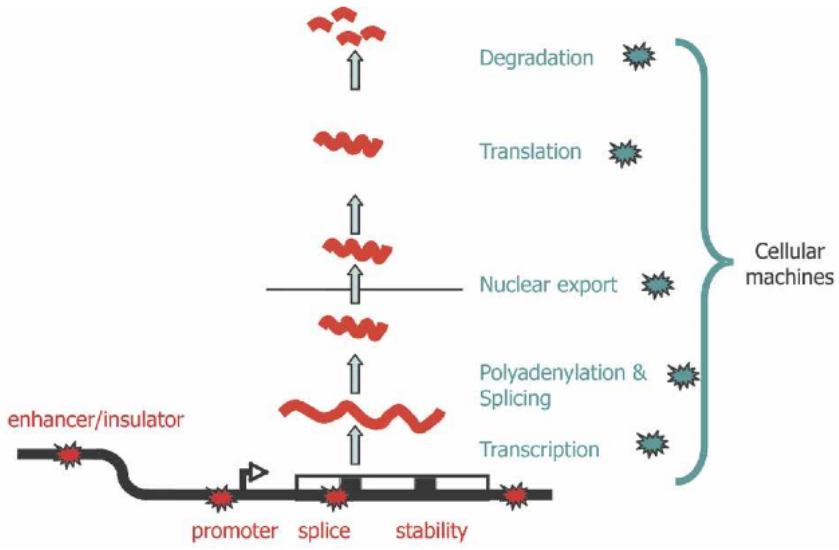


Figure 1. Plausible sites of action for genetic determinants of mRNA levels. Genetic variations influencing gene expression may reside within the regulatory sequences, promoters, enhancers, splice sites, and secondary structure motifs of the target gene and so be genetically in *cis* (red stars), or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

From Williams et al. (2007).

1.2 Direct computation to search for eQTL

The following table is excerpted from Supplement to Stranger et al. (2007). It shows three genes on chromosome 17 that possess eQTL according to certain criteria.

CHI3L2	GI_11993934-S	rs942697	1	111503545	5615	6.81	0.3696	6.99	5.98	4.82	3.94	CEU
CHRNE	GI_38327653-S	rs2302321	17	4728587	13594	10.4232	0.5243	9.2	7.05	5.42	4.3	CEU
COPS7A	GI_7705329-S	rs714774	12	6489545	221475	5.4052	0.2975	6.37	5.35	4.5	3.74	CEU
CP110	GI_34222317-S	rs1475963	16	19412783	59263	6.262	0.3424	6.13	5.37	4.32	3.58	CEU
CPEB4	GI_32698754-S	rs10516107	5	173280762	36825	10.8238	0.539	5.97	5.68	4.58	3.82	CEU
CPNE1	GI_23397697-A	rs6060535	20	33698936	21264	12.4017	0.5927	7.23	5.63	4.29	3.36	CEU
CRIPT	GI_41350204-S	rs1451156	2	46744253	19148	8.1268	0.4309	6.93	5.72	4.54	3.84	CEU
CRNKL1	GI_30795219-S	rs2073123	20	19978870	15505	5.7006	0.3133	7.28	5.53	4.5	3.73	CEU
CSTB	GI_20357564-S	rs1041456	21	44030783	12403	8.2125	0.4403	6.75	5.76	4.68	3.87	CEU
CSTF3	GI_4557494-S	rs3758741	11	33063192	29	5.3589	0.295	5.85	5.3	4.38	3.58	CEU
CTNS	GI_4826681-S	rs161370	17	3477002	34422	6.562	0.3574	6.7	5.44	4.54	3.81	CEU
CTSH	GI_23110956-A	rs1036938	15	77024302	22891	8.4809	0.4464	6.3	5.45	4.56	3.74	CEU
CTSK	GI_23110958-S	rs1136774	1	147551270	30875	7.2458	0.3906	6.89	5.43	4.24	3.45	CEU
DCLRE1B	GI_24431998-S	rs878129	1	114171237	2889	8.7891	0.4595	7.2	5.63	4.44	3.67	CEU
DCTD	GI_4503276-S	rs7277	4	184186662	176	8.596	0.4513	6.89	5.77	4.65	3.8	CEU
DERP6	GI_44662825-I	rs4562	17	7104463	8022	8.5043	0.4474	7.01	5.91	4.56	3.78	CEU

We can check aspects of these findings using publicly available data. The GENEVAR project distributed expression data for CEPH CEU cell lines (immortalized B cells), and genotype results from Phase II HapMap archives have be associated with these in the Bioconductor *GGdata* package. We obtain the full expression data and genotypes for chromosome 17 as follows:

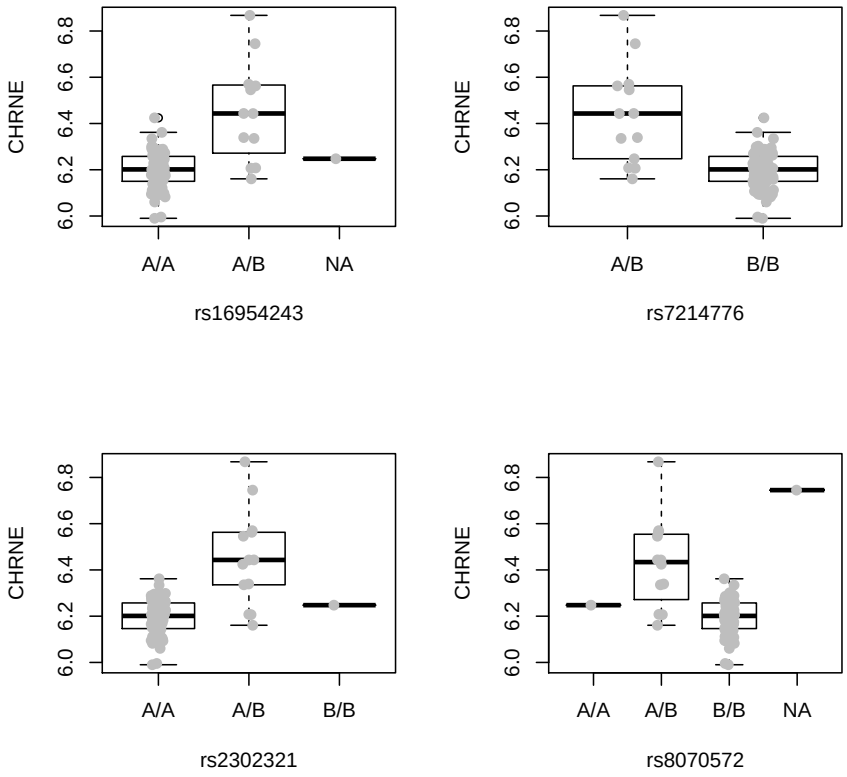
```
> intsave = function(...) NULL # could set as save to speed runs
> date()

[1] "Fri Jan 27 01:22:41 2012"

> library(GGtools)
> library(GGdata)
> c17 = getSS("GGdata", "17", renameChrs="chr17")
> class(c17)  # smlSet links SnpMatrix instances and expression data

[1] "smlSet"
attr(,"package")
[1] "GGBase"

> c17

SnpMatrix-based genotype set:
number of samples:  90
number of chromosomes present:  1
annotation: illuminaHumanv1.db
Expression data dims: 47293 x 90
Total number of SNP: 89701
Phenodata: An object of class "AnnotatedDataFrame"
  sampleNames: NA06985 NA06991 ... NA12892 (90 total)
  varLabels: famid persid ... male (7 total)
  varMetadata: labelDescription
```

We can use an additive genetic model to test for association between allelic dosage and mean expression for CHRNE, adjusting for gender, as follows:

```
> t1 = gwSnpTests(genesym("CHRNE") ~ male, c17, chrnum("chr17"))
> topSnps(t1)

$chr17
                  p.val
rs16954243 2.925728e-09
rs7214776  7.564315e-09
rs8081611  7.564315e-09
rs2302321  4.838786e-08
rs8070572  2.505861e-07
rs7225684  4.087961e-07
rs2243093  8.156751e-07
rs16954257 9.001695e-07
rs2243100  9.273871e-07
rs8077875  1.474871e-06
```

We can visualize some of the associations directly as follows:

```
> par(mfrow = c(2, 2))
> plot_EvG(genesym("CHRNE"), rsid("rs16954243"), c17)
> plot_EvG(genesym("CHRNE"), rsid("rs7214776"), c17)
> plot_EvG(genesym("CHRNE"), rsid("rs2302321"), c17)
> plot_EvG(genesym("CHRNE"), rsid("rs8070572"), c17)
> par(mfrow = c(1, 1))
```



1.2.1 Exercises 1

- What are the p-values returned by `topSnps`? To compute related tests, by hand, using very standard R code, we proceed as follows. First, we acquire the expression data to use as the 'response' in a linear regression model.

```
> chrneExpr = as.numeric(exprs(c17[genesym("CHRNE"), ]))
> summary(chrneExpr)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  5.990   6.159   6.207   6.232   6.272   6.868
```

Then we need a suitable representation of genotype for a SNP of interest. This can be somewhat cumbersome:

```
> num243 = as(smList(c17)[["chr17"]][, "rs16954243"], "numeric")
> table(num243)
```

```
num243
 0  1
77 12
```

Now we obtain the fit:

```
> summary(lm(chrneExpr ~ num243 + male, data = pData(c17)))

Call:
lm(formula = chrneExpr ~ num243 + male, data = pData(c17))

Residuals:
     Min       1Q   Median       3Q      Max
-0.28055 -0.04835 -0.00189  0.06057  0.40423

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.18678    0.01639 377.553  < 2e-16 ***
num243       0.25449    0.03326   7.651 2.68e-11 ***
maleTRUE     0.02210    0.02274   0.972    0.334
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1072 on 86 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared: 0.4097,        Adjusted R-squared: 0.396
F-statistic: 29.84 on 2 and 86 DF,  p-value: 1.434e-10
```

The code underlying `gwSnpTests` is

```
> snp.rhs.tests(chrneExpr~1, data=pData(c17),
+    snp.data=smList(c17)[["chr17"]][, "rs16954243"],
+    family="gaussian")

           Chi.squared Df      p.value
rs16954243    35.48146  1 2.574894e-09
```

The p-value here is based on a statistically and computationally efficient score test.

Use this 'direct access' to the expression and genotype data to visualize aspects of the data leading to:

```
> var.test(chrneExpr[num243 == 0], chrneExpr[num243 == 1])
```

```
        F test to compare two variances

data:  chrneExpr[num243 == 0] and chrneExpr[num243 == 1]
F = 0.1308, num df = 76, denom df = 11, p-value = 2.085e-08
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.04391078 0.28349184
sample estimates:
ratio of variances
         0.1307975
```

What is the consequence of genotype-dependent differential dispersion?

- Describe how to verify findings of Stranger et al. on genes CTNS and DERP6 with similar resources. Note that DERP6 symbol translation is difficult; Stranger and colleagues helpfully provide the probe identifier, which can be used with the `probeId()` cast. Use packages *GGtools* and *GGdata* to carry out the tests.
- In the 90 individuals assayed, many SNP are monomorphic. Use `col.summary` in the *snpStats* package to determine which.

1.3 Transcriptome-wide searches for eQTL

1.3.1 Managing millions of test results; resolving focused queries

A comprehensive search for *cis* and *trans* eQTL using expression microarrays would examine about $5 \cdot 10^4$ mRNA abundance measures against each of $8 \cdot 10^6$ SNP loci. Parallel computing can be used to do this in a reasonable amount of time, but managing the results for convenient interrogation is challenging. We provide one approach to managing large numbers of results in the `eqtlTestsManager` class and related structures in *GGtools*. A small chunk of results related to chromosome 17 is provided in the *ggtut* package.

```
> library(ggtut)

To get a tailored smlSet, use getSS("hmyriB36", [chrvec])
available chromosomes are named  10, 11, ..., X, Y

> f1 = observed17ceu()
> f1

eqtlTestsManager  computed Fri May  6 16:05:50 2011
gene annotation: illuminaHumanv1.db
some genes (out of 498): GI_10190685-S GI_10835020-S ... hmm23927-S hmm5188-S
some snps (out of 60967): rs6565733 rs1106175 ... rs7502145 rs4986109
```

```
> f1@call

eqtlTests(smlSet = c17, rhs = ~male, targdir = "c17c", geneApply = mclapply,
    genegran = 1)
```

The object `f1` holds results of 30361566 tests for expression-genotype association. Note that results for only 498 probes are present. This reflects adoption of a non-specific expression filtering policy (Bourgon et al., 2010) used only to reduce volume for this tutorial.

We can recover results on the probe called DERP6 by Stranger as follows:

```
> topFeats(probeId("GI_44662825-I"), mgr = f1, ffind = 1)

   rs4562  rs222851 rs2106842  rs222843 rs2074217 rs2074218  rs402514  rs222835
    45.04     44.77     40.37     39.86     39.86     39.86     39.86     39.24
rs8067500  rs222850
    34.07     33.65
```

1.3.2 Surveying transcriptome-wide test collections

This suggests an approach to surveying the entire (filtered) transcriptome for chromosome 17 for eQTL.

```
> options(digits=4)
> bestApply = lapply
> if ("multicore" %in% installed.packages()[,1]) {
+   ncores = multicore:::volatile$detectedCores
+   if (ncores >= 3) {
+    library(multicore)
+    options(cores=min(c(ncores-1,10)))  # check!
+    bestApply = mclapply
+    }
+ }
> allpro = probesManaged(f1,1)
> if (!exists("tops")) tops = bestApply( allpro,
+   function(x) topFeats(probeId(x), mgr=f1, ffind=1, n=5 ) )
> names(tops) = allpro
```

The values here are, under the null, individually χ^2 with 1 degree of freedom. The associated p-values would be, for example,

```
> lapply(tops[1:4], function(x) 1 - pchisq(x, 1))
```

```
$`GI_10190685-S`
rs4794214  rs163372 rs6504700 rs3865264  rs489698
2.175e-05 3.738e-05 3.758e-05 4.495e-05 8.164e-05

$`GI_10835020-S`
rs9916609 rs6502743  rs220471  rs220470 rs1178563
2.784e-05 2.858e-05 2.888e-05 3.192e-05 4.471e-05

$`GI_10835100-S`
rs2685524 rs9898312 rs9911505 rs7207897 rs9908211
0.0001790 0.0002424 0.0002530 0.0004070 0.0004340

$`GI_10864026-S`
rs11869731 rs17637018  rs2270517 rs12453418  rs2097970
 9.418e-05  1.075e-04  2.360e-04  3.199e-04  4.386e-04
```

1.3.3 Assessing false discovery rates using statistics computed after permutation

Are these small enough to be regarded as significant? To help reason about this, a manager of test statistics computed after permutation of expression against genotype is provided.

```
> permf1 = onePerm17ceu()
> if (!exists("permtops")) permtops = bestApply( allpro,
+    function(x) topFeats(probeId(x),
+    mgr=permf1, ffind=1, n=5))
> names(permtops) = allpro
> lapply(permtops[1:4], function(x)1-pchisq(x,1))

$`GI_10190685-S`
  rs183209   rs199146 rs11077688  rs7219399   rs180102
 1.328e-05  2.843e-05  1.241e-04  2.111e-04  2.286e-04

$`GI_10835020-S`
 rs7216823    rs11245  rs7221190  rs4792722 rs12951345
 3.660e-05  3.778e-05  9.983e-05  1.771e-04  1.838e-04

$`GI_10835100-S`
 rs7212938  rs2908948  rs2969243 rs11651692 rs12603358
 0.0001959  0.0002055  0.0002055  0.0002450  0.0003796

$`GI_10864026-S`
```

```
rs4426406 rs4447484 rs4327112 rs4246426 rs2255865
8.745e-05 1.134e-04 1.134e-04 1.417e-04 2.348e-04
```

As one might expect, given the large number of tests, the minimum p-values achieved for the first four probes investigated are similar in magnitude to those obtained after permutation. The collection of tests obtained under permutation can be used to assess the false discovery rate for various types of claims.

To illustrate this idea, consider the collection of the gene-specific maximum association statistics. It is:

```
> maxassoc = sapply(tops, "[", 1)
> maxassoc[1:5]

 GI_10190685-S.rs4794214  GI_10835020-S.rs9916609  GI_10835100-S.rs2685524
                   18.03                    17.56                    14.04
GI_10864026-S.rs11869731  GI_11038675-A.rs2584597
                   15.25                    21.40
```

R helpfully provides mangled names allowing us to determine both the gene and SNP associated with any score.

We can obtain the same set of quantities for the permuted tests:

```
> maxaperm = sapply(permtops, "[", 1)
> maxaperm[1:5]

  GI_10190685-S.rs183209  GI_10835020-S.rs7216823  GI_10835100-S.rs7212938
                   18.97                    17.04                    13.87
 GI_10864026-S.rs4426406 GI_11038675-A.rs17719981
                   15.39                    16.00
```

The 99th percentile of the distribution of maximal gene-specific scores computed under permutation will be used as a threshold for asserting the existence of at least one eQTL for a gene, corresponding to a false discovery rate of approximately one percent.

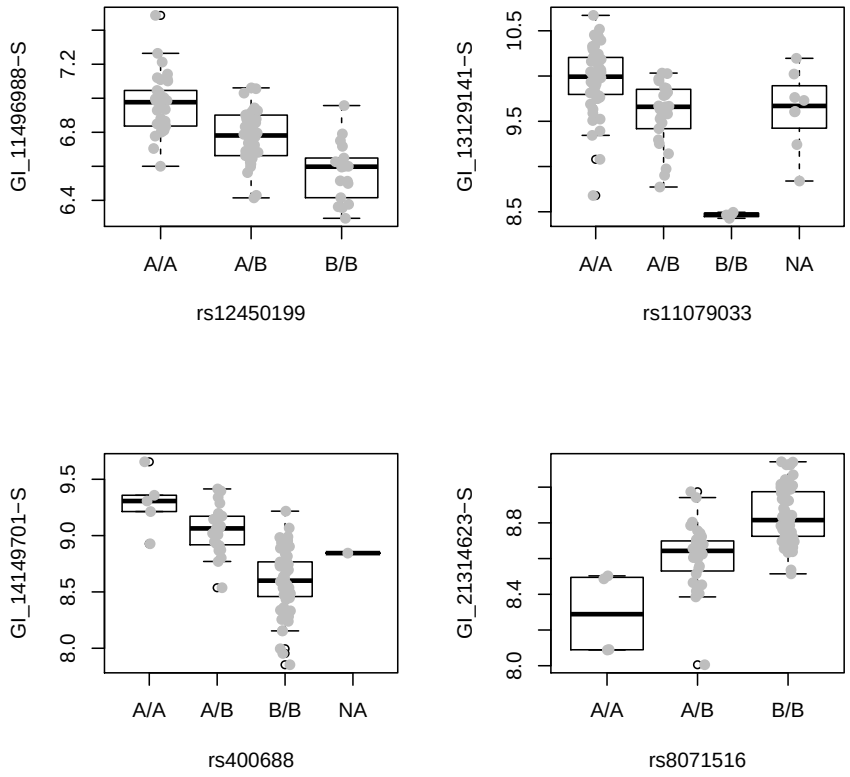
```
> p99 = quantile(maxaperm, 0.99)
> p99

  99%
30.75

> sum(maxassoc > p99)

[1] 22
```

We claim that there are 22 genes with eQTL under this rubric, with an approximate FDR of 0.01. They are

```
> haseqtl = which(maxassoc > p99)
> pweq = allpro[haseqtl]
> unlist(mget(pweq, illuminaHumanv1SYMBOL))

GI_11496988-S GI_13129141-S GI_14149701-S GI_21314623-S GI_31377797-S
        "GAA"       "DHX58"      "RNF167"        "PGS1"      "RABEP1"
GI_31542719-S GI_31542722-S GI_31543284-S GI_31543557-S GI_32528304-I
      "ACSF2"     "SPATA20"       "NDEL1"      "RPH3AL"     "PIP4K2B"
GI_34147471-S GI_38142463-S GI_38348363-S GI_42661209-S GI_42661225-S
     "NT5C3L"      "HEATR6"       "MXRA7"   "LOC645638"       "PRKCA"
GI_42661283-S GI_44662825-I  GI_4506832-S  GI_4826681-S  GI_4885062-S
   "C17orf97"    "C17orf81"        "CCL1"        "CTNS"       "ALDOC"
 GI_7705938-S   Hs.379903-S
   "RAPGEFL1"      "ZSWIM7"
```

To visualize some of the associations, we use

```
> tmp = names(maxassoc)[haseqtl]
> tmp = gsub(".rs", "%.rs", tmp)
> pids = sapply(strsplit(tmp, "%."), "[", 1)
> rsids = sapply(strsplit(tmp, "%."), "[", 2)
> par(mfrow = c(2, 2))
> for (i in 1:4) plot_EvG(probeId(pids[i]), rsid(rsids[i]), c17)
> par(mfrow = c(1, 1))
```



1.3.4 Locations and contexts: the eQTL landscape of a chromosome

We have access to dbSNP-archived SNP locations for chromosome 17 in hg19. The `snpgr17` structure was created using *SNPlocs.Hsapiens.dbSNP.20100427*, based on hg19.

```
> library(SNPlocs.Hsapiens.dbSNP.20100427)
> snpgr17 = getSNPlocs("chr17", as.GRanges = TRUE)
> names(snpgr17) = paste("rs", elementMetadata(snpgr17)$RefSNP_id,
+     sep = "")
> length(snpgr17)
> snpgr17[1:3]
> length(intersect(names(snpgr17), snpsManaged(f1, 1)))
```

We can use this location information to organize and interpret collections of eqtlTests.

We also have information on gene ranges, developed by the somewhat tedious code noted here. The most important component to understand is

```
> library(GenomicFeatures)
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
```

```
> txdb = TxDb.Hsapiens.UCSC.hg19.knownGene
> txg = transcriptsBy(txdb, "gene")
> txg[1:3]

GRangesList of length 3:
$1
GRanges with 2 ranges and 2 elementMetadata values:
      seqnames               ranges strand |     tx_id     tx_name
         <Rle>            <IRanges>  <Rle> | <integer> <character>
  [1]    chr19 [58858174, 58864865]      - |     68902  uc002qsd.3
  [2]    chr19 [58859832, 58874120]      - |     68904  uc002qsf.1

$10
GRanges with 1 range and 2 elementMetadata values:
      seqnames               ranges strand | tx_id    tx_name
  [1]     chr8 [18248755, 18258723]      + | 29976 uc003wyw.1

$100
GRanges with 2 ranges and 2 elementMetadata values:
      seqnames               ranges strand | tx_id    tx_name
  [1]    chr20 [43248164, 43280376]      - | 70158 uc002xmj.2
  [2]    chr20 [43248164, 43280376]      - | 70159 uc010ggt.2

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

which shows how UCSC tables can be used with GenomicFeatures package infrastructure to collect information on transcripts. The recoding from transcript sets to approximate gene regions is:

```
> pn = probesManaged(f1,1)
> library(illuminaHumanv1.db)
> pn.eg = unlist(mget(pn, illuminaHumanv1ENTREZID))
> pn.eg = na.omit(pn.eg)
> eg.pn = names(pn.eg)
> names(eg.pn) = pn.eg
> txg17 = txg[ intersect(names(txg), pn.eg) ]
> extents = function(x) {
+   y = x[seqnames(x)=="chr17"]; c(min(start(y)),max(end(y)))
+   }  # watch for random
> ssnr = lapply( txg17, function(z) try(extents(z)) )
```

```
> firsts = sapply(ssnr, function(x) {if(is.finite(x[1])) return(x[1]); NA})
> if (any(is.na(firsts))) ssnr = ssnr[-which(is.na(firsts))]
> firsts = sapply(ssnr, function(x) {if(is.numeric(x[1])) return(x[1]); NA})
> lasts = sapply(ssnr, function(x) {if(is.numeric(x[2])) return(x[2]); NA})
> g17rngsnr = GRanges(seqnames="chr17",
+     IRanges(firsts,lasts), probeid=eg.pn[names(ssnr)])
```

which you can avoid by using

```
> data(g17rngsnr)
> g17rngsnr
```

```
GRanges with 475 ranges and 1 elementMetadata value:
                seqnames              ranges strand   |       probeid
                   <Rle>           <IRanges>  <Rle>   |   <character>
  GI_21237796-A    chr17 [39509647, 39556540]      *   | GI_21237796-A
   GI_4885638-S    chr17 [50333051, 50394327]      *   |  GI_4885638-S
  GI_22035666-S    chr17 [46294586, 46300338]      *   | GI_22035666-S
  GI_17572809-S    chr17 [77439016, 77442758]      *   | GI_17572809-S
  GI_30410793-A    chr17 [38229969, 38249303]      *   | GI_30410793-A
  GI_20070210-S    chr17 [37098653, 37101424]      *   | GI_20070210-S
   GI_5032212-S    chr17 [45133689, 45140527]      *   |  GI_5032212-S
   GI_5031728-S    chr17 [58981554, 59025373]      *   |  GI_5031728-S
  GI_33519473-S    chr17 [44263371, 44297228]      *   | GI_33519473-S
            ...      ...                 ...    ... ...           ...
  GI_41281459-S    chr17 [ 7180597,  7195517]      *   | GI_41281459-S
  GI_41281472-S    chr17 [70964259, 71008128]      *   | GI_41281472-S
  GI_40538727-S    chr17 [40869049, 40923893]      *   | GI_40538727-S
   GI_7662287-S    chr17 [ 6422369,  6484971]      *   |  GI_7662287-S
  GI_37543271-S    chr17 [ 2187556,  2231098]      *   | GI_37543271-S
   GI_7662241-S    chr17 [12633554, 12835685]      *   |  GI_7662241-S
   GI_7661883-S    chr17 [62497016, 62671781]      *   |  GI_7661883-S
  GI_16554576-S    chr17 [42550310, 42621664]      *   | GI_16554576-S
   GI_4827043-S    chr17 [57374748, 57497425]      *   |  GI_4827043-S
  ---
  seqlengths:
   chr17
      NA
```

Our objective here is to give a chromosome-wide picture of SNP-mediated expression variation. We will employ two constraints on SNP-gene distance. The `best.cis.eQTLs` function accomplishes this with a specified cis radius. We also filter probes to the upper half of the non-specific variation distribution, implicitly, through use of `nsFilter`

requiring Entrez annotation for probes to be considered, and remove SNP with MAF less than 0.05. This procedure uses a plug-in method for FDR estimation, using by default 2 permutations of expression against genotype.

```
> library(multicore)
> options(mc.cores = 10)
> myfilt = function(x) nsFilter(MAFfilter(x, lower = 0.05), var.cutoff = 0.5)
> b1 = best.cis.eQTLs("GGdata", ~male, radius = 50000, chrnames = "17",
+     geneApply = mclapply, schrpref = "chr", smFilter = myfilt)
> b1
```

The number of probes with FDR < 0.05 is given by

```
> ok = which(elementMetadata(b1@scoregr)$fdr < 0.05)
> length(ok)
```

```
[1] 19
```

These SNP are obtained with

```
> rsb1hi = elementMetadata(b1@scoregr)$snpid[ok]
> rsb1hi[1:10]
```

```
 [1] "rs6565724" "rs4792709" "rs8066107" "rs3026133" "rs760273"  "rs8067500"
 [7] "rs452825"  "rs1476162" "rs7216490" "rs1054083"
```

1.3.5 High-level tools for locus annotation: ChIPpeakAnno

We can use a facility intended for ChIP-seq analysis to obtain structural metadata about SNPs that have interesting scores.

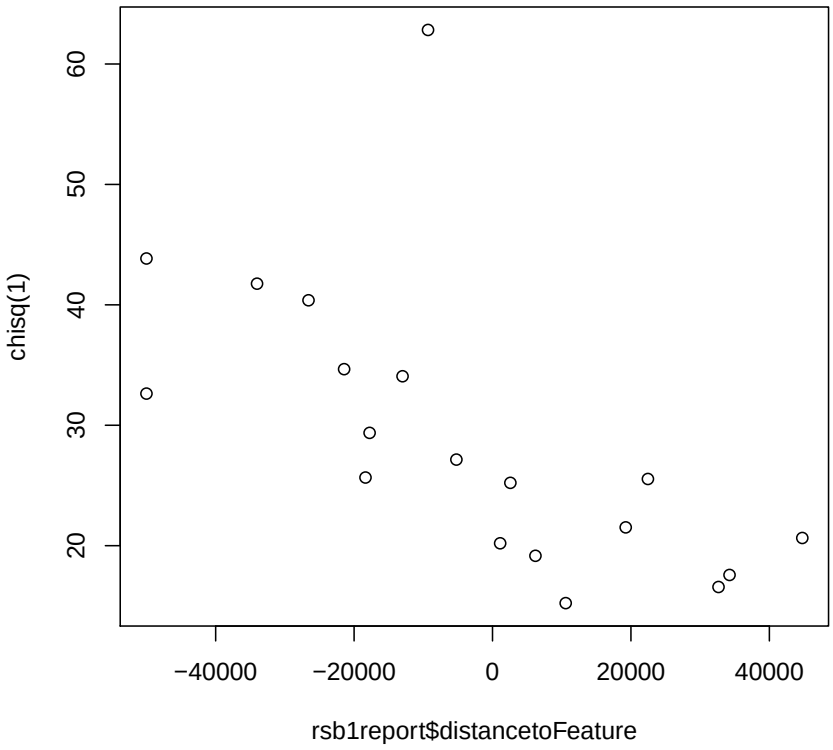
```
> library(ChIPpeakAnno)
> data(TSS.human.GRCh37)
> TSS.human.GRCh37[1:3,]
```

```
RangedData with 3 rows and 2 value columns across 72 spaces
                   space         ranges |    strand
                <factor>      <IRanges> | <integer>
ENSG00000223972        1 [11874, 14412] |         1
ENSG00000227232        1 [14363, 29570] |        -1
ENSG00000243485        1 [29554, 31109] |         1


ENSG00000223972 DEAD/H (Asp-Glu-Ala-Asp/His) box polypeptide 11 like 10 [Source:HGNC Sy
ENSG00000227232                 WAS protein family homolog 5 pseudogene [Source:HGNC Sy
ENSG00000243485
```

```
> rsb1rd = as(b1@scoregr[ok], "RangedData")  # convert for use
> rsb1report = annotatePeakInBatch(rsb1rd, AnnotationData=TSS.human.GRCh37)
> rsb1report[1:4,]

RangedData with 4 rows and 9 value columns across 1 space
                               space               ranges |          peak
                            <factor>            <IRanges> |   <character>
GI_14149701-S ENSG00000205710       17 [ 4793630,  4898515] | GI_14149701-S
GI_22547158-S ENSG00000011295       17 [15852782, 15982096] | GI_22547158-S
GI_22547188-A ENSG00000228000       17 [18181195, 18316856] | GI_22547188-A
GI_22748758-S ENSG00000187688       17 [16292301, 16394523] | GI_22748758-S
                              strand         feature start_position
                            <character>     <character>      <numeric>
GI_14149701-S ENSG00000205710           + ENSG00000205710        4802948
GI_22547158-S ENSG00000011295           + ENSG00000011295       15902782
GI_22547188-A ENSG00000228000           + ENSG00000228000       18215218
GI_22748758-S ENSG00000187688           + ENSG00000187688       16318888
                            end_position  insideFeature distancetoFeature
                               <numeric>    <character>         <numeric>
GI_14149701-S ENSG00000205710     4806227 includeFeature             -9318
GI_22547158-S ENSG00000011295    15948336 includeFeature            -50000
GI_22547188-A ENSG00000228000    18215965 includeFeature            -34023
GI_22748758-S ENSG00000187688    16340317 includeFeature            -26587
                            shortestDistance fromOverlappingOrNearest
                                   <numeric>              <character>
GI_14149701-S ENSG00000205710             9318             NearestStart
GI_22547158-S ENSG00000011295            33760             NearestStart
GI_22547188-A ENSG00000228000            34023             NearestStart
GI_22748758-S ENSG00000187688            26587             NearestStart

> table(rsb1report$insideFeature)

includeFeature   overlapStart
            12              7
```

To check relationship between 'strength' of association and proximity to gene:

```
> plot(rsb1rd$score ~ rsb1report$distancetoFeature, ylab = "chisq(1)")
```


chisq(1)
20
30
40
50
60
−40000
−20000
0
20000
40000
rsb1report$distancetoFeature

2 Identifying and reducing expression heterogeneity for enhanced eQTL discovery

2.1 Unsupervised approach: PCA for covariates

Stegle et al. (2010) have described a variety of approaches that attempt to isolate and remove non-genetic sources of expression variation prior to testing for eQTL. One very simple approach involves using principal components of expression variation as covariates in models for effects of allelic dose. The assumption of this approach is that the bulk of variation exhibited in a set of microarrays is non-genetic in origin. Searching for genetic signals residual to non-genetic variation should be more productive.

To illustrate this idea, we take a gene, CD79B, found by Stranger et al (2007) to have an eQTL on chromosome 17. First, we do a simple search for eQTL.

```
> library(GGtools)
> c17 = getSS("GGdata", "17")
> get("CD79B", revmap(illuminaHumanv1SYMBOL))

[1] "GI_11038673-I" "GI_11038675-A"
```

We see that this gene is represented by two probes; Stranger et al provided the probe identifier:

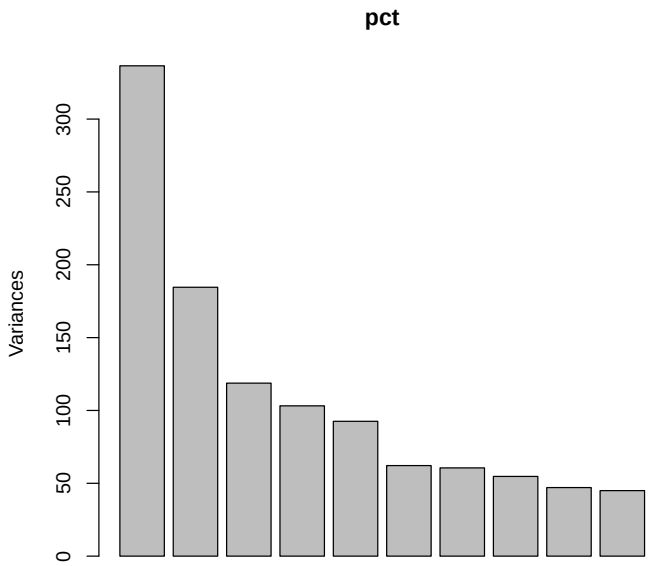
```
> lkcd1 = gwSnpTests(probeId("GI_11038675-A") ~ male, c17, chrnum("17"))
> topSnps(lkcd1)

$`17`
               p.val
rs2584597  3.714e-06
rs1376110  4.484e-06
rs2665850  5.086e-06
rs11654841 5.463e-06
rs3817182  5.495e-06
rs2257281  5.796e-06
rs12946669 6.378e-06
rs2854184  6.987e-06
rs7209608  8.117e-06
rs2236737  8.205e-06
```

We see reasonable evidence of signal. Now we will use all available expression data to compute principal components.

```
> pct = prcomp(t(exprs(c17)))
```

We can plot the relative magnitudes of variation partitioned into the PCs:

```
> plot(pct)
```



We use this figure to decide that the first 4 principal components are capturing a reasonable fraction of variation. We add this information to the `smlSet`:

```
> DF = data.frame(pct$x[, 1:4])
> pData(c17) = cbind(pData(c17), DF)
```

and now we fit an enhanced model:

```
> lkcd2 = gwSnpTests(probeId("GI_11038675-A") ~ PC1 + PC2 + PC3 +
+     PC4 + male, c17, chrnum("17"))
> topSnps(lkcd2)

$`17`
               p.val
rs12946669 9.023e-11
rs7209608  1.389e-10
rs2665850  2.986e-10
rs2286565  3.701e-10
rs11079515 5.341e-10
rs2727350  5.708e-10
rs4968674  7.127e-10
rs2854184  8.304e-10
rs2584597  1.229e-09
rs2257281  1.467e-09
```

We see a fairly striking reduction in the p-values of the strongest hits.

Exercise: Show how to recompute the counts of eQTL in a systematic survey, using this unsupervised adjustment for expression heterogeneity.

2.2 Supervised approach: surrogate variable analysis

Leek and Storey (2007) describe an approach to reduction of heterogeneity in expression that is supervised in the sense that the extraneous variance is estimated residual to a specified structural source of variation. An iterative algorithm for establishing significance criteria for "surrogate variables" that carry extraneous expression variation is developed. We will use it in conjunction with verification of assessment of eQTL for CD79B.

A basic model is lkcd1 computed above. The top SNP for that model was rs2584597. We will seek variation residual to that carried by this SNP. First exclude samples with missing genotypes.

```
> table(nsn <- as(smList(c17)[[1]][, "rs2584597"], "numeric"))

 0  1  2
11 42 32

> drop = which(is.na(nsn))
```

Now perform SVA.

```
> mod = model.matrix(~as(smList(c17)[[1]][, "rs2584597"], "numeric"))
> mod0 = model.matrix(~1, data = pData(c17[, -drop]))
> library(sva)
> if (!exists("SVA1") & file.exists("SVA1.rda")) load("SVA1.rda") else {
+     SVA1 = sva(exprs(c17[, -drop]), mod, mod0)
+     intsave(SVA1, file = "SVA1.rda")
+ }
```

This code can take a long time, and is not evaluated in this vignette. Instead:

```
> if (!exists("SVA1")) data(SVA1)
```

We can see the number of surrogate variables identified:

```
> SVA1$n.sv

[1] 14
```

Now we append them to the pData of c17 and retest.

```
> SVDF = data.frame(SVA1$sv)
> c17d = c17[, -drop]
> pData(c17d) = cbind(pData(c17d), SVDF)
> lkcd3 = gwSnpTests(probeId("GI_11038675-A") ~ X1 + X2 + X3 +
+     X4 + X5 + X6 + X7 + X8 + X9 + X10 + X11 + X12 + X13 + X14,
+     c17d, chrnum("17"))
> topSnps(lkcd3)

$`17`
               p.val
rs7209608  5.806e-11
rs12946669 1.101e-10
rs2286565  2.458e-10
rs2854184  3.619e-10
rs4968674  3.997e-10
rs11079515 5.350e-10
rs2727350  8.213e-10
rs2584597  1.175e-09
rs2665850  1.175e-09
rs1043127  3.152e-09
```

Exercise: if we make the bold assumption that the surrogate variables identified for rs2584597 are valid for all SNP on chromosome 17, what happens to our basic assessment of the number of genes for which eQTL exist given above, once we include this adjustment for extraneous variation?

3 Investigating trans associations

The combinatorics of assessing trans associations are daunting. We use a 'scratch pad' approach. For every locus $1, \ldots, L_c$ on chromosome c, all the genes on a given chromosome d are tested for association, and the scores and gene identifiers for the top K tests are retained in the scratch pad. All the genes on the next chromosome are tested against all loci, and the scratch pad is updated to retain the top K scores and identifiers seen so far. The updating process is done with small blocks, so the quantity of memory consumed is always fairly small. In the end two out-of-memory matrices of size $L_c \times K$ are created, holding the best scores achieved after surveying all chromosomes, and the gene (index) associated with each score.

In the following example, we continue to focus on loci on chromosome 17 as potential regulators of genes on chromosomes 1 or 9. These were selected to allow an assessment of findings of Cheung et al. (2010), where three loci on chr17 were associated with expression levels of genes on chr1 or chr9.

This set of tests can be done quickly with a multicore system. To avoid contending with stray expression values observed on a very rare genotype, we limit the loci investigated to those with MAF at least 10%. This is accomplished by setting `wrapperEndo`.

```
> library(GGtools)
> library(GGdata)
> t17f = transScores("GGdata", "17", rhs=~male,
+   chrnames=c("chr1", "chr9"), wrapperEndo=
+   function(x) MAFfilter(x, low=.1),
+   targdirpref="twfilt", geneApply=bestApply)
> intsave(t17f, file="t17f.rda")
```

The result is made available in the ggtut package.

```
> tr17 = tr17_1_9()
> tr17

transManager instance, created Tue Oct 11 10:24:22 2011
dimension of scores component:
 number of loci checked: 44638; genes retained: 20
the call was:
transScores(smpack = "GGdata", snpchr = "17", rhs = ~male, targdirpref = "twfilt",
    geneApply = mclapply, chrnames = c("chr1", "chr9"), wrapperEndo = function(x) MAFfi
        low = 0.1))
```

We have an analgous set of scores computed with a permutation of expression against genotype:

```
> tr17_perm = tr17_1_9_perm()
```

The distribution of highest scores per locus, with the permuted data, can be obtained via

```
> psco = topScores(tr17_perm)
> summary(psco)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    6.5    10.5    11.8    12.2    13.5    27.6

> tp99 = quantile(psco, 0.99)
```

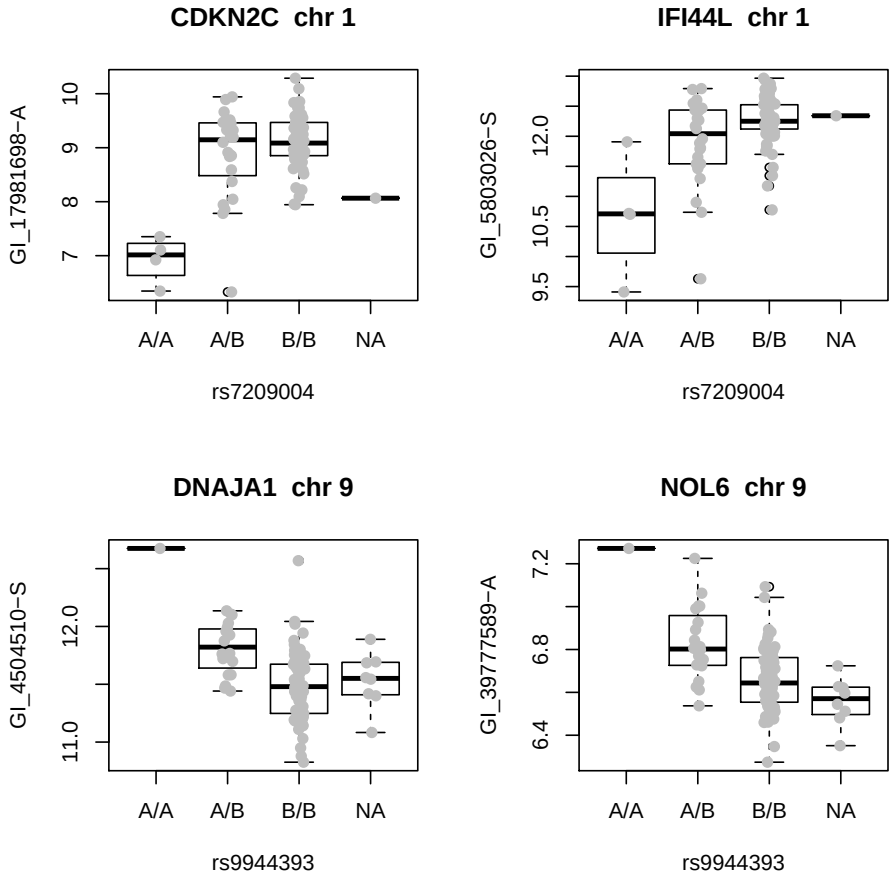
This gives an informal threshold against which to compare extreme trans scores over loci.

We would like to investigate SNP which appear to be significantly associated with more than one gene in trans. To find these, we will use the 19.1 threshold seen above.

```
> locw2 = which(nthScores(tr17, 2) > tp99)
> g1inds = geneIndcol(tr17, 1)[locw2]
> g2inds = geneIndcol(tr17, 2)[locw2]
> g1names = GGtools::geneNames(tr17)[g1inds]
> g2names = GGtools::geneNames(tr17)[g2inds]
> locnames = locusNames(tr17)[locw2]
```

Here we display the expression by genotype distributions for two pairs of genes that are associated with a single locus on chromosome 17.

```
> library(illuminaHumanv1.db)
> nicevg = function(pr, rs, sms) {
+     sym = get(pr, illuminaHumanv1SYMBOL)
+     chr = get(pr, illuminaHumanv1CHR)
+     plot_EvG(probeId(pr), rsid(rs), sms, main = paste(sym, " chr",
+         chr))
+ }
> par(mfrow = c(2, 2))
> nicevg(g1names[1], locnames[1], c17)
> nicevg(g2names[1], locnames[1], c17)
> nicevg(g1names[3], locnames[3], c17)
> nicevg(g2names[3], locnames[3], c17)
```



To return to the findings of Cheung et al, we note that they found that variation in expression of FDPS was associated with SNP rs7212322 (Supplementary table 2). To check whether this gene was noted among top trans associations tested here, we proceed as follows.

```
> pid = get("FDPS", revmap(illuminaHumanv1SYMBOL))
> ind = which(GGtools::geneNames(tr17) == pid)
> isatop = any(geneIndcol(tr17, 1) == ind)
> isasec = any(geneIndcol(tr17, 2) == ind)
> isatop

[1] FALSE

> isasec

[1] TRUE
```

We see that there is some locus (on chromosome 17, with MAF $> 10\%$) for which FDPS gives a second-strongest association.

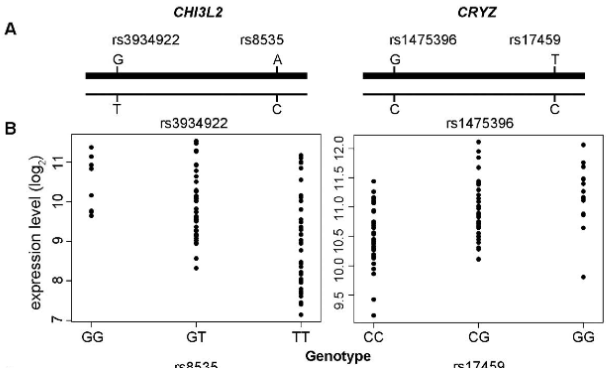
Exercises. Find a locus giving a χ^2_1 statistic exceeding 11.8 for association with FDPS; for this locus FDPS is the fourth-largest association score. Display the expression-vs-genotype plot. How would you perform a more comprehensive check for FDPS without restricting to MAF $> 10\%$?

4 Leveraging RNA-seq: details of transcriptomic diversity

Cheung et al. (2010) describe the use of RNA-seq in conjunction with dense genotyping to assess the frequency with which transcript variation is governed by *cis*-regulatory variants. We have excerpted the aligned data published by Cheung et al. to illustrate some of the data-analytic considerations.

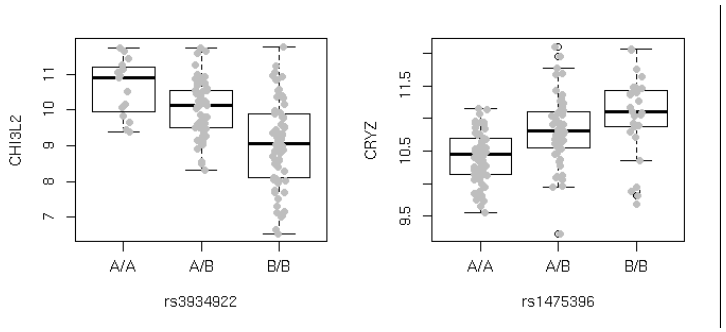
4.1 Some key observations and their approximate reproduction

Figure 2b of Cheung et al includes this display of array-based measures of expression of two genes.

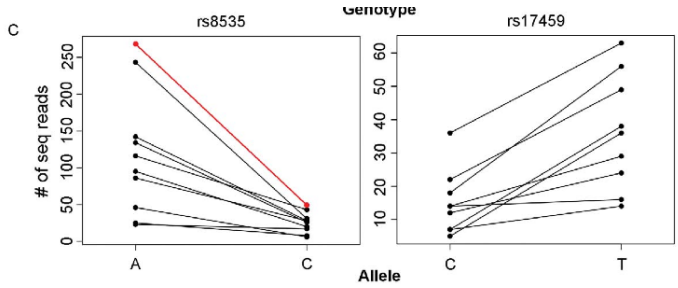


Using Cheung et al. expression quantitations, we can qualitatively recover this display.

```
> c1 = getSS("cheung2010", "chr1")
> annotation(c1) = "hgfocus.db"
> annotation(CHE1) = "hgfocus.db"
> par(mfrow = c(2, 2))
> plot_EvG(genesym("CHI3L2"), rsid("rs3934922"), CHE1)
> plot_EvG(genesym("CRYZ"), rsid("rs1475396"), CHE1)
```



The authors dig into RNA-seq data on a subset of samples to display the following:



To permit recovery aspects of this display from publicly available data, we have created a collection of BAM files derived from the MAQ alignments published in GEO GSE16921. Briefly, samtools maq2sam-short was used to translate MAQ alignments to SAM format, which were then re-indexed to hg18 and filtered to include all reads for a contiguous segment of chromosome 1 including genes DENND2D, CHI3L2, and CHIA. *Rsamtools* can be used to interrogate these reads.

```
> library(Rsamtools)
```

First we prepare to compute 'pileups' for the 41 BAM files.

```
> bff=PileupFiles(dir(system.file("bam", package="ggtut"),
+    patt="bam$", full=TRUE))
> # fix up names
> bp = sapply(plpFiles(bff), path)
> no1 = gsub(".*bam.lit..", "", bp)
> no2 = gsub("_1.bam", "", no1)
> no3 = gsub(".bam", "", no2)
> nanames = paste("NA", no3, sep="")
> names(bff) = nanames
> bff

class: PileupFiles
names: NA06985, ..., NA12891 (41 total)
plpFiles: litgm06985.bam, ..., litgm12891_1.bam (41 total)
plpParam: class PileupParam

> # now have sample names
```

Corresponding to hg18, SNP rs8535 is at position 111587452. We will acquire the pileups of base calls at this location using the *Rsamtools* `applyPileups` method.

```
> which = GRanges(seqnames="chr1", IRanges(111587452,width=1))
> param = PileupParam(which=which)
> pinfo = function(x) {
+    x[["seq"]][,,1]  # reduce to matrix
+    }
> ans = applyPileups(bff, pinfo, param=param)[[1]]
> colnames(ans) = nanames
> ans
```

	NA06985	NA06993	NA06994	NA07000	NA07022	NA07034	NA07055	NA07056	NA07345
A	66	0	86	51	0	1	0	119	0
C	0	193	30	1	39	251	53	5	267
G	0	0	1	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0

	NA11004	NA11829	NA11830	NA11831	NA11832	NA11839	NA11881	NA11882	NA11992
A	144	0	0	22	33	119	0	0	217
C	27	7	52	8	1	47	3	26	26
G	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0

	NA11993	NA12003	NA12004	NA12005	NA12006	NA12043	NA12044	NA12056	NA12057
A	203	0	111	0	47	0	0	2	1
C	43	2	0	266	0	51	118	263	271
G	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0

	NA12144	NA12145	NA12155	NA12264	NA12716	NA12717	NA12750	NA12762	NA12813
A	215	119	46	0	0	105	39	193	81
C	15	0	6	238	166	0	0	0	0
G	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0

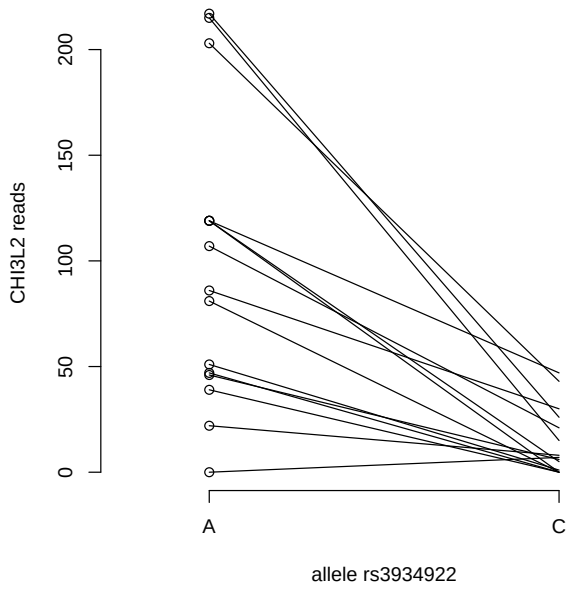
	NA12814	NA12815	NA12872	NA12874	NA12891
A	251	107	0	0	0
C	1	21	53	25	179
G	0	0	0	0	0
T	0	0	0	0	0
N	0	0	0	0	0

Thus we have the calls at rs8535. We do not have ready access to genotype data on our samples at this locus, but we can use the data on the linked locus rs3934922. We use the following code to couple the two data sources:

```
> c1 = getSS("cheung2010", "chr1")
> c1s = smList(c1)[[1]]
> rownames(c1s) = gsub("GM", "NA", c1$GMno)
> c1ss = c1s[intersect(rownames(c1s), nanames), ]
> c1sss = c1ss[, "rs3934922"]
> THESN = as(c1sss[, 1], "character")
> rownames(THESN) = rownames(c1sss)
> tt = as(THESN, "character")
> names(tt) = rownames(c1ss)
> ttt = tt[intersect(names(tt), colnames(ans))]
> anss = ans[, names(ttt)]
```

Now we obtain the names and associated genotype and count data for heterozygotes only, and make the plot.

```
> anssh = anss[, ttt == "A/B"]
> plot(rep(1, ncol(anssh)), anssh[1, ], xlim = c(0.75, 2.25), axes = FALSE,
+     xlab = "allele rs3934922", ylab = "CHI3L2 reads")
> axis(1, at = c(1, 2), labels = c("A", "C"))
> axis(2)
> segments(1, anssh[1, ], 2, anssh[2, ])
```



4.2 Surveying a read set for transcript variants

In this section we will examine reads overlapping exons for within-subject variation. Again the workhorse is `applyPileups`.

We begin by acquiring exon ranges.

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

The segment that we have archived is

```
> myr = GRanges(seqnames = "chr1", IRanges(111500000, 111700000))
```

and the exons within this interval are

```
> exinseg = allex[allex %in% myr]
```

which we assign as a pileup application parameter:

```
> Pexinseg = PileupParam(which = exinseg)
```

We now define a function that will tell us which exon intervals harbor locations at which some individuals exhibit multiple distinct bases. We require that there be at least five reads for each variant.

```
> hasTxDivers = function(x) {
+     nuccts = x[["seq"]]
+     apply(nuccts, 3, function(w) any(apply(w, 2, function(z) sum(z >
+         5) > 1)))
+ }
> ans = applyPileups(bff, hasTxDivers, param = Pexinseg)
```

The exonic regions with variants are:

```
> ediv = exinseg[which(sapply(ans, any))]
> ediv
```

```
GRanges with 6 ranges and 1 elementMetadata value:
      seqnames                ranges strand |   exon_id
         <Rle>             <IRanges>  <Rle> | <integer>
  [1]     chr1 [111528336, 111529247]      + |     13107
  [2]     chr1 [111579784, 111579908]      + |     13129
  [3]     chr1 [111582895, 111583077]      + |     13131
  [4]     chr1 [111585472, 111585588]      + |     13132
  [5]     chr1 [111587362, 111587585]      + |     13134
```

```
  [6]     chr1 [111532276, 111532515]      - |     13112
  ---
  seqlengths:
            chr1   chr1_random         chr10 ...   chrX_random          chrY
       247249719       1663265     135374737 ...       1719168      57772954
```

We then refine the pileup analysis to focus on these exons within which some individuals exhibit allelic heterogeneity, and identify the locations at which variants are found.

```
> whichLocTxDivers = function(x) {
+     nuccts = x[["seq"]]
+     which(apply(apply(nuccts, 3, function(w) apply(w, 2, function(z) sum(z >
+         5) > 1)), 2, any))
+ }
> Nparam = PileupParam(which = exinseg[which(sapply(ans, any))])
> ans2 = applyPileups(bff, whichLocTxDivers, param = Nparam)
```

The actual locations are:

```
> divlocs = lapply(1:length(ediv), function(x) (start(ediv[x]):end(ediv[x]))[ans2[[x]
> divlocs

[[1]]
[1] 111528392 111528710

[[2]]
[1] 111579846 111579848

[[3]]
[1] 111582997 111583000 111583002 111583004 111583009 111583020

[[4]]
[1] 111585505 111585554

[[5]]
[1] 111587452 111587453 111587535 111587578

[[6]]
[1] 111532424

> alldiv = unlist(divlocs)
> length(alldiv)
```

```
[1] 17
```

How many of these locations correspond to known SNP for hg18?

```
> library(SNPlocs.Hsapiens.dbSNP.20090506)
> c1locs = getSNPlocs("chr1")
> sum(alldiv %in% c1locs$loc)
```

```
[1] 8
```

```
> date()
```

```
[1] "Fri Jan 27 01:24:40 2012"
```

Cheung and colleagues remark that the literature suggests fractions of genes exhibiting differential allelic expression as ranging from 18% to 26%. These examples show that relatively compact code exploiting the R-samtools interface can be used to investigate this phenomenon.

5 Summary

- When dense genotypes and transcript profiles are acquired on the same subjects, integrative analysis of genetics of gene expression can provide insight into mechanisms of expression regulation.
- Computational versatility is required to filter and manage billions of statistical tests arising with comprehensive searches for variant-expression association. R/Bioconductor provide suitable facilities for divide and conquer approaches involving simple concurrent execution of embarrassingly parallel algorithms.
- Ranges and tracks can be used to record association statistics in genomic coordinates for assessment of genomic context of expression-associated DNA variants.
- Imputation using haplotype models and other aspects of dependencies among DNA features can improve resolution of regulatory signals.
- Techniques for modeling and reducing effects of expression heterogeneity can have important impacts on power of eQTL discovery processes.
- Simple comprehensive surveys of *trans* associations can be undertaken in light of the sparsity of real associations.
- Mechanics of working with RNA-seq data to unearth allelic imbalance and other variations in transcript structure are reasonably manageable with streaming/filtering approaches to computation in data analysis.

Appendix: Preparing the 1000 genome imputation rules

The basic idea is that some collection of individuals is available with a very rich genotype panel, and another collection of individuals is genotyped on only a subset of the rich set of loci. The data on the rich panel are used to build predictive models for loci not present in the subset.

We begin by processing the calls (ignoring quality and uncertainty) provided by the 1000 genomes project for 629 individuals. The following code creates 8 blocks spanning 10 million bases each, to cover the 79Mb chromosome 17. each

```
> library(GGtools)
> library(Rsamtools)
> exts  = seq(1, 80e6+10, by=10e6)
> st = exts[-length(exts)]
> en = exts[-1]-1
> # following file is 66 GB from 1000genomes.org
> tf = TabixFile("ALL.2of4intersection.20100804.genotypes.vcf.gz")
> gg = GRanges(seqnames="17", IRanges(st,en))
> for (i in 1:length(gg)) {
+   vv = vcf2sm(tf, gr=gg[i], nmetacol=9L)
+   intsave(vv, file=paste("vv", i, ".rda", sep=""))
+ }
```

Each block is stored as a SnpMatrix instance, with a byte encoding each genotype call, and the blocks are combined using `cbind`.

The following code then constructs the rules provided in the ggtut package.

```
> getRules = function(tkgsm, basesm, locvec, try = 200, em.cntrl = c(1000,
+     0.005, 1000, 0.005), use.hap = c(0.99, 0.01)) {
+     sntkg = colnames(tkgsm)
+     snbase = colnames(basesm)
+     baseok = intersect(names(locvec), snbase)
+     tkok = intersect(names(locvec), sntkg)
+     tkgsm = tkgsm[, tkok]
+     basesm = basesm[, baseok]
+     sntkg = colnames(tkgsm)
+     snbase = colnames(basesm)
+     toimp = setdiff(sntkg, snbase)
+     usepred = setdiff(snbase, toimp)
+     yloc = locvec[toimp]
+     xloc = locvec[usepred]
+     rules = snp.imputation(basesm[, usepred], tkg[, toimp], xloc,
```

```
+         yloc, try = 200, em.cntrl = c(1000, 0.005, 1000, 0.005),
+         use.hap = c(0.99, 0.01))
+ }
> c17 = getSS("GGdata", "17", wrapperEndo = dropMonomorphies)
> base = smList(c17)[[1]]
> tkg = get(load("ceu1kg_17.rda"))
> base = base[rownames(tkg), ]
> library(SNPlocs.Hsapiens.dbSNP.20090506)
> loc17 = getSNPlocs("chr17")
> snin = colnames(base)
> nams = paste("rs", loc17[, 1], sep = "")
> loc17 = loc17$loc
> names(loc17) = nams
> sninC = colnames(tkg)[grep("chr", colnames(tkg))]
> kgloc = as.numeric(gsub("chr17:", "", sninC))
> names(kgloc) = sninC
> allloc = c(loc17, kgloc)
> rules.n43 = getRules(tkg, base, allloc)
```

6 Session information

```
> sessionInfo()

R Under development (unstable) (2012-01-11 r58090)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.iso88591       LC_NUMERIC=C
 [3] LC_TIME=en_US.iso88591        LC_COLLATE=en_US.iso88591
 [5] LC_MONETARY=en_US.iso88591    LC_MESSAGES=en_US.iso88591
 [7] LC_PAPER=C                    LC_NAME=C
 [9] LC_ADDRESS=C                  LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.iso88591 LC_IDENTIFICATION=C

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

other attached packages:
 [1] TxDb.Hsapiens.UCSC.hg18.knownGene_2.6.2
 [2] TxDb.Hsapiens.UCSC.hg19.knownGene_2.6.2
 [3] multicore_0.1-7
 [4] ggtut_0.0.37
 [5] hmyriB36_0.99.12
 [6] SNPlocs.Hsapiens.dbSNP.20090506_0.99.6
 [7] cheung2010_0.0.21
 [8] hgfocus.db_2.6.3
 [9] Rsamtools_1.7.19
[10] ChIPpeakAnno_2.3.1
[11] limma_3.11.7
[12] GO.db_2.6.1
[13] BSgenome.Ecoli.NCBI.20080805_1.3.17
[14] BSgenome_1.23.2
[15] Biostrings_2.23.5
[16] multtest_2.11.1
[17] biomaRt_2.11.1
[18] gplots_2.10.1
[19] KernSmooth_2.23-7
[20] caTools_1.12
[21] bitops_1.0-4.1
[22] gdata_2.8.2
[23] gtools_2.6.2
```

```
[24] GenomicFeatures_1.7.10
[25] GenomicRanges_1.7.15
[26] IRanges_1.13.20
[27] ff_2.2-3
[28] bit_1.1-7
[29] GGdata_1.0.17
[30] illuminaHumanv1.db_1.12.1
[31] org.Hs.eg.db_2.6.4
[32] RSQLite_0.11.1
[33] DBI_0.2-5
[34] AnnotationDbi_1.17.14
[35] snpStats_1.5.2
[36] Matrix_1.0-2
[37] lattice_0.20-0
[38] survival_2.36-10
[39] Biobase_2.15.3
[40] BiocGenerics_0.1.4
[41] GGtools_4.2.6
[42] GGBase_3.16.1
[43] weaver_1.21.0
[44] codetools_0.2-8
[45] digest_0.5.1
[46] BiocInstaller_1.3.5

loaded via a namespace (and not attached):
[1] annotate_1.33.1    genefilter_1.37.0  MASS_7.3-16        RCurl_1.9-4
[5] rtracklayer_1.15.6 XML_3.7-3          xtable_1.6-0       zlibbioc_1.1.0
```

References

Richard Bourgon, Robert Gentleman, and Wolfgang Huber. Independent filtering increases detection power for high-throughput experiments. *Proc Natl Acad Sci USA*, 107(21):9546–51, May 2010. doi: 10.1073/pnas.0914005107.

Vivian G Cheung, Renuka R Nayak, Isabel Xiaorong Wang, Susannah Elwyn, Sarah M Cousins, Michael Morley, and Richard S Spielman. Polymorphic cis- and trans-regulation of human gene expression. *PLoS Biol*, 8(9):e1000480, Sep 2010. doi: 10.1371/journal.pbio.1000480.t005.

Jeffrey T Leek and John D Storey. Capturing heterogeneity in gene expression studies by surrogate variable analysis. *PLoS Genet*, 3(9):1724–35, Sep 2007. doi: 10.1371/journal.pgen.0030161. URL `http://www.plosgenetics.org/article/info%253Adoi%252F10.1371%252Fjournal.pgen.0030161`.

Oliver Stegle, Leopold Parts, Richard Durbin, and John Winn. A bayesian framework to account for complex non-genetic factors in gene expression levels greatly increases power in eqtl studies. *PLoS Comput Biol*, 6(5):e1000770, May 2010. doi: 10.1371/journal.pcbi.1000770. URL `http://www.ploscompbiol.org/article/info%253Adoi%252F10.1371%252Fjournal.pcbi.1000770`.

B. E Stranger, M. S Forrest, M Dunning, C. E Ingle, C Beazley, N Thorne, R Redon, C. P Bird, A De Grassi, C Lee, C Tyler-Smith, N Carter, S. W Scherer, S Tavare, P Deloukas, M. E Hurles, and E. T Dermitzakis. Relative impact of nucleotide and copy number variation on gene expression phenotypes. *Science*, 315(5813):848–853, Feb 2007. doi: 10.1126/science.1136678.

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.