Using *GGtools* for eQTL discovery and interpretation

VJ Carey stvjc at channing.harvard.edu

March 17, 2017

Contents

1 Overview and installation

This document addresses data structure and analytic workflow for investigations of genetic sources of expression variation. Key background references are **?** for general biologic overview, **?** and **?** for key applications, and **?**, **?**, and **?** for various methodological issues. **?** reviews potentials of eQTL investigations with expression measures based on RNA sequencing.

This document is constructed using R version 3.3.3. See the session information at the end of the document for full details. Using a comparable version of R, you can obtain the software needed for the production of this document using

```
source("http://www.bioconductor.org/biocLite.R")
biocLite("GGtools", dependencies=TRUE)
```

2 Data structures

2.1 Reference data supplied with Bioconductor

A collection of 30 trios of central European ancestry was genotyped for 4 million SNP loci in HapMap phase II. Immortalized B-cell lines were assayed for gene expression using Illumina's HumanWG6v1 bead array. Digital data on expression and genotype for the 90 CEU individuals is distributed in Bioconductor package *GGdata*; the expression data were retrieved from the GENEVAR website of Wellcome Trust, e.g.,

`ftp://ftp.sanger.ac.uk/pub/genevar/CEU_parents_norm_march2007.zip`

and the genotype data were obtained directly from hapmap.org at build 36:

`ftp://ftp.ncbi.nlm.nih.gov/hapmap/genotypes/2008-03/forward/non-redundant/`

The data in GGdata are likely derived from r23, while r23a is now distributed. Some effort at updating genotypes may be supplied in the future.

Acquire the genome-wide expression data and the genotype data for chromosome 20 as follows:

```
> suppressPackageStartupMessages(library(GGtools))
> library(parallel)

> g20 = getSS("GGdata", "20")

> g20

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: 119921
Phenodata: An object of class 'AnnotatedDataFrame'
  sampleNames: NA06985 NA06991 ... NA12892 (90 total)
  varLabels: famid persid ... male (7 total)
  varMetadata: labelDescription

> class(g20)

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

The `smlSet` class was designed in 2006 as an experiment in unifying high-throughput expression and genotype data. A key resource was the *snpMatrix* (now *snpStats*) package of David Clayton, which defined an 8-bit representation of genotype calls (including uncertain calls obtained by statistical imputation), import and coercion infrastructure allowing use of the 8-bit representation with popular genetic data formats (pedfiles, mach and beagle outputs, etc.), and statistical testing infrastructure employing this representation.

The expression and sample-level data are handled just as with familiar `ExpressionSet` instances:

```
> exprs(g20)[1:5,1:5]

               NA06985   NA06991   NA06993   NA06994   NA07000
GI_10047089-S  5.983962  5.939529  5.912270  5.891347  5.906675
GI_10047091-S  6.544493  6.286516  6.244446  6.277397  6.330893
GI_10047093-S  9.905235 10.353804 10.380972  9.889223 10.155686
GI_10047099-S  7.993935  7.593970  8.261215  6.598430  6.728085
GI_10047103-S 11.882265 12.204753 12.249708 11.798415 12.015252
```

```
> pData(g20)[1:4,]
```

```
        famid persid mothid fathid  sampid isFounder  male
NA06985  1341     14      0      0 NA06985      TRUE FALSE
NA06991  1341      2     14     13 NA06991     FALSE FALSE
NA06993  1341     13      0      0 NA06993      TRUE  TRUE
NA06994  1340      9      0      0 NA06994      TRUE  TRUE
```

The genotype data are held in a list with elements intended to represent chromosomes, and the list is stored in an environment to reduce copying efforts.

```
> smList(g20)
```

```
$`20`
A SnpMatrix with  90 rows and  119921 columns
Row names:  NA06985 ... NA12892
Col names:  rs4814683 ... rs6090120
```

```
> as(smList(g20)[[1]][1:5,1:5], "matrix")
```

```
        rs4814683 rs6076506 rs6139074 rs1418258 rs7274499
NA06985        03        03        03        03        03
NA06991        02        03        02        02        03
NA06993        01        03        01        01        03
NA06994        01        03        01        01        03
NA07000        03        03        03        03        03
```

The leading zeroes in the display above indicate that raw bytes are used to represent the genotypes per sample (rows) per SNP (columns). Coercions:

```
> as(smList(g20)[[1]][1:5,1:5], "numeric")
```

```
        rs4814683 rs6076506 rs6139074 rs1418258 rs7274499
NA06985         2         2         2         2         2
NA06991         1         2         1         1         2
NA06993         0         2         0         0         2
NA06994         0         2         0         0         2
NA07000         2         2         2         2         2
```

```
> as(smList(g20)[[1]][1:5,1:5], "character")
```

```
        rs4814683 rs6076506 rs6139074 rs1418258 rs7274499
NA06985 "B/B"     "B/B"     "B/B"     "B/B"     "B/B"
NA06991 "A/B"     "B/B"     "A/B"     "A/B"     "B/B"
NA06993 "A/A"     "B/B"     "A/A"     "A/A"     "B/B"
NA06994 "A/A"     "B/B"     "A/A"     "A/A"     "B/B"
NA07000 "B/B"     "B/B"     "B/B"     "B/B"     "B/B"
```

Any number of chromosomes can be held in the genotype list component, but the design allows working with only one chromosome at a time and examples emphasize this approach. Amalgamation of results across chromosomes is generally straightforward.

2.2 Working with your own data

The `make_smlSet` function can be used to bind a list of suitably named `SnpMatrix` instances with an `ExpressionSet` instance to create a single `smlSet` instance covering multiple chromosomes.

The `externalize` function can be applied to such an `smlSet` instance to create a new *package* which can be installed for use with `getSS`. This is the preferred way of managing work with large genotyping panels (such as the 10 million locus panel achievable with "thousand genomes imputation").

Briefly, `externalize` arranges a DESCRIPTION file and system of folders that can be installed as an R package. The expression data are stored as object `ex` in data/eset.rda, and the `SnpMatrix` instances are stored separately as .rda files in inst/parts. The `getSS` function will create `smlSet` instances on the fly from the externalize-generated package.

2.3 Filters and permutation support

Here "filter" is used to refer to any function that processes an `smlSet` instance and returns an `smlSet` instance after altering the contents, which may involve eliminating probes or SNPs, performing numerical transformations to expression or genotype measures, transforming sample data or eliminating samples.

2.3.1 Bracket operations

Coordinated manipulations of genotype, expression, and phenotype information on samples can be accomplished with the second subscript to the bracket operator. Thus `g20[,1:5]` is the restriction of `g20` to the first five samples. Sample names may also be used for such manipulations.

Reduction of the expression component can be accomplished with the first subscript to the bracket operator. Thus `g20[1:5,]` is the restriction of `g20` to five expression probes; all other data are unaltered. If it is desired to use feature names for such manipulations, the character vector of feature names must be cast to class `probeId` with the `probeId()` method. At present no such operations can be used to alter the genotype data contents.

2.3.2 Large scale filters

It is known that non-specific filtering (removal of probes with low variation across samples, without regard to sample phenotype or class information) can increase sensitivity

and specificity of certain differential expression test procedures (**?**). The `nsFilter` function of the *genefilter* package has been adapted to work with `smlSet` instances.

SNPs can be filtered out of the `smlSet` instance on the basis of observed minor allele or genotype frequencies using `MAFfilter` and `GTFfilter` respectively.

Various approaches to reduction of "expression heterogeneity" can be examined. `clipPCs(x, vec)` will form the singular value decomposition of the expression matrix and remove principal components enumerated in `vec` by reassembling the expression matrix after setting eigenvalues in `vec` to zero. It is also possible to employ any computed quantities such as principal components or surrogate variables identified in SVA (**?**) as covariates in the formula element of analysis functions in `GGtools`, but note that simple permutations do not lead to valid permutation tests of SNP effects in the presence of covariates (see (**?**) who focus on interaction, but describe the problem for main effects models, with references, early in the paper.)

The introduction of novel approaches to expression transformation can be accomplished using code similar to the following, illustrating of the use of PEER (**?**):

```
> library(peer)
> model = PEER()
> PEER_setPhenoMean(model, t(exprs(g20)))
> PEER_setNk(model, 10)
> PEER_setCovariates(model, matrix(1*g20$male,nc=1))
> PEER_update(model)
> resid=t(PEER_getResiduals(model))
> rownames(resid) = featureNames(g20)
> colnames(resid) = sampleNames(g20)
> g20peer10 = g20
> g20peer10@assayData = assayDataNew("lockedEnvironment", exprs=resid)
```

At this point, `g20peer10` holds expression data with 10 latent factors removed after adjustment for gender.

2.3.3 Permutation of expression against genotype

Because the *snpStats* testing procedures defensively match predictor to response variable orderings using sample labels, special steps must be taken to ensure that tests use properly permuted responses. The `permEx` function takes care of this, using the current state of the random number generator.

2.4 Post-analysis data structures

While it is possible to construe the results of an eQTL search as a static report, it is more productive to conceptualize the result as a data object for further analysis.

Unfortunately, the number of tests to be managed can be very large – at least hundreds of millions, and these must be joinable with location metadata to be maximally useful.

Several data structures for managing post-analysis results have emerged as this package has matured. Of particular concern are those that use *ff* out-of-memory archiving for test statistic values or effect estimates and those that use the `GRanges` infrastructure to facilitate efficient query resolution in genomic coordinates. These will be described along with the related analytic workflow steps.

3 Focused analyses

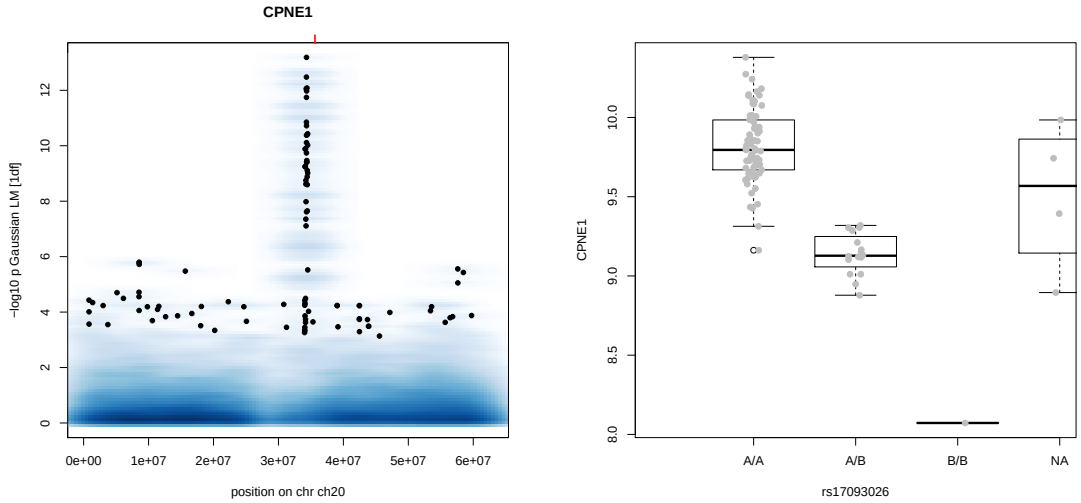
A specific gene can be checked for eQTL on a given chromosome or set of chromosomes with `gwSnpTests`. There are various convenience facilities. In the call to `gwSnpTests` below, a gene symbol is used to pick out an expression element, and adjustment for gender is commodated in an additive genetic model for effects of B allele copy number on expression of CPNE1. One degree of freedom chi-squared tests are computed.

```
> t1 = gwSnpTests(genesym("CPNE1")~male, g20, chrnum("20"))
> t1

gwSnpScreenResult for gene  CPNE1  [probe  GI_23397697-A ]

> topSnps(t1)

$`20`
                  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
```

It is possible to compute tests for this specific gene for association with SNP across several chromosomes if desired; change the value of the third argument to a vector.

There are a few approaches to visualization of the results that are relevant, but complications arise in relation to choice of genomic coordinates.

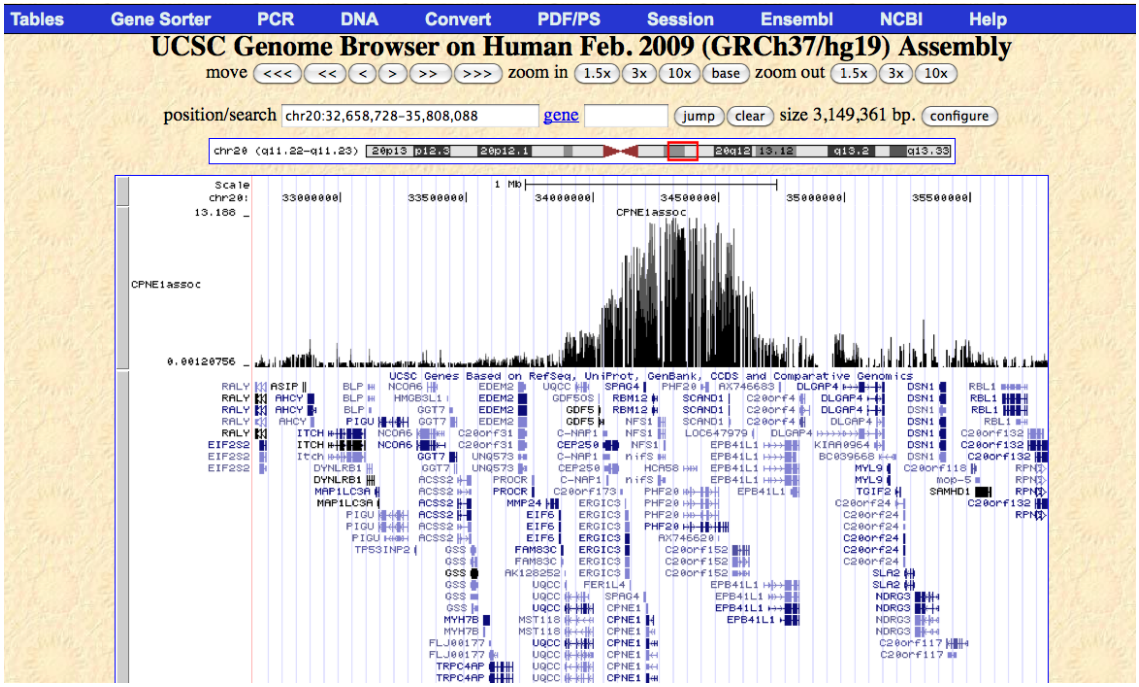
```
> plot(t1, snplocsDefault())
> plot_EvG(genesym("CPNE1"), rsid("rs17093026"), g20)
```



Code like the following can be used to display scores ($-\log_{10} p$) on the genome browser, here with hg19 locations.

```
> library(snplocsDefault(), character.only=TRUE)
> sl = get(snplocsDefault())
> S20 = snplocs(sl, "ch20", as.GRanges=TRUE)
> GR20 = makeGRanges(t1, S20)
> library(rtracklayer)
> export(GR20, "~/cpne1new.wig")
```

With this code, it will be necessary to manually alter the chr assignment in the wig file, and place an informative title to get the following display.


Tables
Gene Sorter
PCR
DNA
Convert
PDF/PS
Session
Ensembl
NCBI
Help
UCSC Genome Browser on Human Feb. 2009 (GRCh37/hg19) Assembly
move
zoom in
zoom out
position/search
chr20:32,658,728-35,808,088
gene
jump
clear
size 3,149,361 bp.
configure
CPNE1assoc
UCSC Genes Based on RefSeq, UniProt, GenBank, CCDS and Comparative Genomics

4 Comprehensive surveys

4.1 A set of genes vs. all SNP on a chromosome

The performance of *snpStats* `snp.rhs.tests` is very good and so our principle for large-scale searches is to compute all association statistics, save them in truncated form, and filter results later. This is carried out with the `eqtlTests` function. To illustrate, the expression data is sharply filtered to the 50 most variable genes on chromosome 20 as measured by cross-sample median absolute deviation, SNP with MAF < 0.05 are removed, and then all SNP-gene association tests are executed.

```
> g20 = GGtools:::restrictProbesToChrom(g20, "20")
> mads = apply(exprs(g20),1,mad)
> oo = order(mads, decreasing=TRUE)
> g20 = g20[oo[1:50],]
> tf = tempfile()
> dir.create(tf)
> e1 = eqtlTests(MAFfilter(g20, lower=0.05), ~male,
+     geneApply=mclapply, targdir=tf)
> e1

eqtlTestsManager  computed Fri Mar 17 18:58:13 2017
gene annotation: illuminaHumanv1.db
some genes (out of 50): hmm9615-S GI_31077201-S ... GI_34335276-S GI_32483384-A
some snps (out of 55039): rs4814683 rs6139074 ... rs6062363 rs4809418
```

On a two-core macbook pro, this computation completes in less than a minute. The details of the underlying data structure are involved. Briefly, a short integer is used to represent each chi-squared statistic obtained in the 1 tests computed, in an `ff` archive. Use `topFeats` to manually harvest this.

```
> pm1 = colnames(e1@fffile)
> tops = sapply(pm1, function(x) topFeats(probeId(x), mgr=e1, n=1))
> top6 = sort(tops, decreasing=TRUE)[1:6]

> print(top6)

 GI_17149835-I.rs6041750  GI_34147330-S.rs6037097  GI_41327717-S.rs2223883
                   40.30                    36.49                    24.87
 GI_31563517-A.rs6085541  GI_29826316-A.rs6135980 GI_13540544-S.rs11905405
                   21.38                    19.70                    19.20
```

R has propagated the names of probes and SNPs with the scores so that a table can be created as follows:

```
> nms = strsplit(names(top6), "\\.")
> gn = sapply(nms,"[",1)
> sn = sapply(nms,"[",2)
> tab = data.frame(snp=sn,score=as.numeric(top6))
> rownames(tab) = gn
> tab

                     snp score
GI_17149835-I  rs6041750 40.30
GI_34147330-S  rs6037097 36.49
GI_41327717-S  rs2223883 24.87
GI_31563517-A  rs6085541 21.38
GI_29826316-A  rs6135980 19.70
GI_13540544-S rs11905405 19.20
```

Statistical interpretation of the scores in this table is not clear as the data structure includes familial aggregation in trios and extended pedigrees, and may include population stratification, Nevertheless, consistency of these findings with other published results involving multiple populations can be checked. *GGtools* includes a table published by Stranger and colleagues in 2007 enumerating multipopulation eQTL (**?**).

```
> data(strMultPop)
> strMultPop[ strMultPop$rsid %in% tab$snp, ]

           rsid  genesym      illv1pid snpChr snpCoordB35 probeMidCoorB35
11306 rs6041750   FKBP1A GI_17149835-I     20     1298709         1300503
11398 rs6037097     PYGB GI_34577123-S     20    25295221        25226352
25370 rs6037097 C20orf22 GI_34147330-S     20    25295221        25223609
      snp2probe minuslog10p  adjR2 assocGrad permThresh          popSet
11306      1794     21.6949 0.3798    0.4708    12.4815 CEU-CHB-JPT-YRI
11398     68869      8.9349 0.1645    0.1324     6.5410 CEU-CHB-JPT-YRI
25370     71612      8.9684 0.2257    0.3396     6.5535     CEU-CHB-JPT
```

Thus the top two SNP in the table computed here are identified as multipopulation eQTL by Stranger. The other association scores are not very strong and likely do not correspond to genuine associations.

4.2 Tabulating best associated cis-rSNP with permutation-based FDR: small example

The workhorse for identifying genes to which can be associated putatively regulating SNP (rSNP) is `best.cis.eQTLs`. This can be used for genome-wide analysis, but here an alternative table is created for the sharply filtered chromosome 20 data given above.

This call says that gene location information will be acquired from the Bioconductor TxDb.Hsapiens annotation package for hg19 UCSC known genes, and that tests for association within 1 Mbase of the coding region for each gene will be considered. The expression data will be permuted against genotype data in two independent draws to assemble the null reference distribution for association scores; these are used to enumerate false significance claims at various magnitudes of the distribution of association scores. The plug-in procedure for estimating FDR XXX cite Hastie Tibs Friedman is used.

```
> fn = probeId(featureNames(g20))

> if (file.exists("db2")) unlink("db2", recursive=TRUE)
> fn = probeId(featureNames(g20))
> exTx = function(x) MAFfilter( x[fn, ], lower=0.05)
> b1 = best.cis.eQTLs("GGdata", ~male,  radius=1e6,
+    folderstem="db2", nperm=2, geneApply=mclapply,
+    smFilter= exTx, chrnames="20", snpannopk=snplocsDefault())

> b1

GGtools mcwBestCis instance.  The call was:
best.cis.eQTLs(smpack = "GGdata", rhs = ~male, folderstem = "db2",
    radius = 1e+06, chrnames = "20", geneApply = mclapply, snpannopk = snplocsDefault()
    smFilter = exTx, nperm = 2)
Best loci for 50 probes are recorded.
There were 83969 gene:snp tests.
Top  4 probe:SNP combinations:
GRanges object with 4 ranges and 6 metadata columns:
                seqnames              ranges strand |     score       snpid
                   <Rle>           <IRanges>  <Rle> | <numeric> <character>
  GI_34147330-S       20 [24280834, 26371618]      * |     36.49   rs6037097
     hmm26961-S       20 [61665697, 63671315]      * |     16.78   rs3810504
  GI_17149835-I       20 [  352356,  2373816]      * |     16.67  rs13043344
  GI_31077201-S       20 [61679079, 63680979]      * |     14.78  rs13044229
                   snploc radiusUsed      nsnp       fdr
                <integer>  <numeric> <integer> <numeric>
  GI_34147330-S  25347221      1e+06      1387      0.00
     hmm26961-S  62678549      1e+06       783      0.00
  GI_17149835-I    544417      1e+06      2382      0.00
  GI_31077201-S  61738288      1e+06       766      0.25
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
====
use chromsUsed(), fullreport(), etc. for additional information.
```

5 Comprehensive reporting on FDR for all SNP cis to a set of genes

We want to support improved control of cis-eQTL searches and better reflectance of cis-eQTL search outputs. The basic idea is that we can, in most modern computing facilities, retain fairly substantial archives resulting from comprehensive searches; previous versions of GGtools emphasized gene-centric analysis which could then be extended to higher levels of resolution in a targeted manner. We can do a single comprehensive search and then filter to understand sensitivity of FDR measures to aspects of upstream filtering.

5.1 Configuring a search

```
> suppressPackageStartupMessages(library(GGtools))
> ini = new("CisConfig")
> ini

CisConfig instance; genome  hg19 .  Key parameters:
smpack =  GGdata ; chrnames =  22
nperm =  3 ; radius =  50000
====
Configure using
 [1] "smpack<-"        "rhs<-"           "nperm<-"         "folderStem<-"
 [5] "radius<-"        "shortfac<-"      "chrnames<-"      "smchrpref<-"
 [9] "gchrpref<-"      "schrpref<-"      "geneApply<-"     "geneannopk<-"
[13] "snpannopk<-"     "smFilter<-"      "exFilter<-"      "keepMapCache<-"
[17] "SSgen<-"         "genome<-"        "excludeRadius<-" "estimates<-"
[21] "extraProps<-"    "useME<-"         "MEpvot<-"

> radius(ini) = 75000L
> smFilter(ini) = function(x) nsFilter(x, var.cutoff=.98)
> smpack(ini) = "GGtools"
> chrnames(ini) = "20"
> library(parallel)  # to define mclapply
> geneApply(ini) = mclapply
> ini

CisConfig instance; genome  hg19 .  Key parameters:
smpack =  GGtools ; chrnames =  20
nperm =  3 ; radius =  75000
====
Configure using
 [1] "smpack<-"        "rhs<-"           "nperm<-"         "folderStem<-"
 [5] "radius<-"        "shortfac<-"      "chrnames<-"      "smchrpref<-"
```

```
 [9] "gchrpref<-"      "schrpref<-"      "geneApply<-"     "geneannopk<-"
[13] "snpannopk<-"     "smFilter<-"      "exFilter<-"      "keepMapCache<-"
[17] "SSgen<-"         "genome<-"        "excludeRadius<-" "estimates<-"
[21] "extraProps<-"    "useME<-"         "MEpvot<-"
```

5.2 Executing a search

5.2.1 Computing and serializing scored GRanges

The FDR reported are for the specific search (here one chromosome). Various tools for combining multiple searches are used to get FDRs for more comprehensive searches.

```
> options(mc.cores=max(1, detectCores()-3))
> system.time(t20 <- All.cis(ini))

get data...build map...run smFilter...filter probes in map...tests...get data...build m
229.806  21.267 138.240

> t20

cisRun object with 5806 ranges and 15 metadata columns:
                seqnames              ranges strand |         snp   snplocs
                   <Rle>           <IRanges>  <Rle> | <character> <integer>
  GI_11496985-S    chr20 [23004355, 23161340]      - |   rs2746621  23004620
  GI_11496985-S    chr20 [23004355, 23161340]      - |   rs2567613  23004858
  GI_11496985-S    chr20 [23004355, 23161340]      - |  rs16984838  23005819
           ...      ...                 ...    ... .         ...       ...
      hmm9615-S    chr20 [15121857, 16128196]      + |   rs6043760  16128165
                  score      ests        se       fdr       probeid
                <numeric> <numeric> <numeric> <numeric>   <character>
  GI_11496985-S      0.78     -0.30      0.34 0.8635648 GI_11496985-S
  GI_11496985-S      4.12      0.26      0.12 0.9038031 GI_11496985-S
  GI_11496985-S      0.00      <NA>      <NA> 0.7098274 GI_11496985-S
           ...       ...       ...       ...       ...           ...
      hmm9615-S         0      <NA>      <NA> 0.7098274     hmm9615-S
                      MAF  dist.mid   mindist genestart   geneend permScore_1
                  <numeric> <numeric> <numeric> <integer> <integer>   <numeric>
  GI_11496985-S 0.04444444   78227.5     74735 -23079355 -23086340        0.16
  GI_11496985-S 0.41111111   77989.5     74497 -23079355 -23086340        0.29
  GI_11496985-S 0.00000000   77028.5     73536 -23079355 -23086340        0.00
           ...        ...       ...       ...       ...       ...         ...
      hmm9615-S          0  503138.5     74969  15196857  16053196           0
                permScore_2 permScore_3
                  <numeric>   <numeric>
```

```
  GI_11496985-S        0.97        1.50
  GI_11496985-S        0.12        0.13
  GI_11496985-S        0.00        0.00
            ...         ...         ...
      hmm9615-S           0           0
  -------
  seqinfo: 93 sequences (1 circular) from hg19 genome
```

Note that the observed score is retained along with all associated scores achieved under permutation of expression against genotype.

We can update to a different chromosome:

```
> chrnames(ini) = "21"
> system.time(t21 <- All.cis(ini))
```

For realistic and comprehensive filtering settings (radius 250000 or greater, MAF bound .005, the results of All.cis can be quite large, so most downstream utilities assume the results are serialized to disk.

```
> td = tempdir()
> save(t20, file=paste0(td, "/t20.rda"))
> #save(t21, file=paste0(td, "/t21.rda"))
```

5.2.2 Amalgamation of chromosome-specific results, recomputation of FDR

We will use a collection method that focuses on sensitivity analysis, generating FDRs for different tuning parameters within the scope of the search.

```
> fns = dir(td, full=TRUE, patt="^t2.*rda$")
> cf = collectFiltered(fns, mafs=c(.02,.03,.1), hidists=c(1000,10000,75000))

.................................

> class(cf)

[1] "list"

> names(cf)  # MAFs are primary organization, distances secondary

[1] "0.02" "0.03" "0.1"

> names(cf[[1]])

[1] "1000"  "10000" "75000"
```

```
> sapply(cf, sapply, function(x) sum(x$fdr <= 0.05))  # best per gene
```

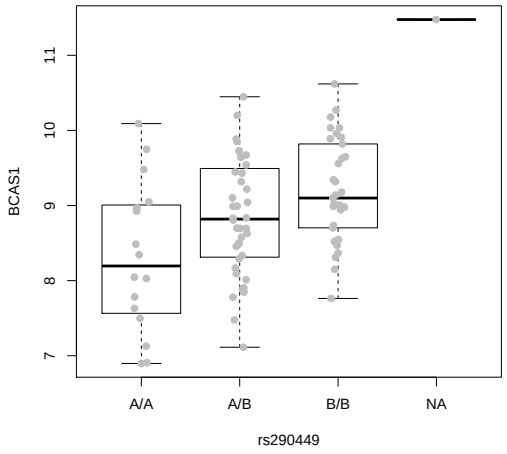
```
      0.02 0.03 0.1
1000     0    0   0
10000    0    0   0
75000    0    0   0
```

```
> of = order(cf[[3]][[1]]$fdr)
> cf[[3]][[1]][of,][1:4,]  # shows best hits
```

```
                     genes   bestsnp   chr       fdr scores MAF.observed
GI_11496985-S GI_11496985-S  rs844820 chr20 0.6666667   7.96    0.1348315
hmm9615-S         hmm9615-S rs7272442 chr20 0.6666667  10.81    0.4367816
GI_18379361-A GI_18379361-A rs3092750 chr20 0.7777778   2.63    0.1166667
GI_18426910-S GI_18426910-S rs2076555 chr20 0.8666667   3.04    0.1091954
              permScore_1 permScore_2 permScore_3
GI_11496985-S        3.04        1.76        4.03
hmm9615-S           12.32       10.87        9.82
GI_18379361-A        3.47        1.12        3.50
GI_18426910-S        3.41        4.58        1.85
```

```
> g20 = getSS("GGtools", "20")
```

```
> plot_EvG(probeId("GI_4502372-S"), rsid("rs290449"), g20)
```



`collectFiltered` has a default `filterFun` which isolates the best scoring SNP per gene. A SNP-centric FDR can also be computed.

```
> cf2 = collectFiltered(fns, mafs=c(.02,.05,.1), hidists=c(10000,75000),
+   filterFun=cis.FDR.filter.SNPcentric)
> sapply(cf2, sapply, function(x) sum(x$fdr<=0.01))
```

```
      0.02 0.05 0.1
10000    0    0   0
75000    0    0   0
```

6 Full search on an SGE-based cluster

The stages are:

- create the smlSet-based package via externalize
- set up the chromosome-specific run as a function

```
> onepopConfig = function(chrn="22", nperm=3L, MAF=.05,
+     npc=10, radius=50000, exclRadius=NULL) {
+    if (!is.null(npc))
+      bf = basicFilter = function(ww) MAFfilter(clipPCs(ww, 1:npc), lower=MAF)[
+    else bf = basicFilter = function(ww) MAFfilter(ww, lower=MAF)[, which(
+     ww$isFounder==1)]
+
+    ssm(library(GGtools))
+    iniconf = new("CisConfig")
+    smpack(iniconf) = "GGdata"
+    rhs(iniconf) = ~1
+    folderStem(iniconf) = paste0("cisScratch_", chrn)
+    chrnames(iniconf) = as.character(chrn)
+    geneannopk(iniconf) = "illuminaHumanv1.db"
+    snpannopk(iniconf) = snplocsDefault()
+    smchrpref(iniconf) = ""
+    geneApply(iniconf) = mclapply
+    exFilter(iniconf) = function(x)x
+    smFilter(iniconf) = bf
+    nperm(iniconf) = as.integer(nperm)
+    radius(iniconf) = radius
+    estimates(iniconf) = TRUE
+    MAFlb(iniconf) = MAF
+
+    library(parallel)
+    options(mc.cores=3)
+    options(error=recover)
+    set.seed(1234)
+    tmp = All.cis( iniconf )
+    metadata(tmp)$config = iniconf
+    obn = paste("pop2_", "np_", nperm, "_maf_", MAF, "_chr_", chrn,
```

```
+     "_npc_", npc, "_rad_", radius, "_exc_", exclRadius, sep="")
+    fn = paste(obn, file=".rda", sep="")
+    assign(obn, tmp)
+    save(list=obn, file=fn)
+ }
```

- Create a script that invokes the function

```
#!/bin/bash
RCMD=/udd/stvjc/bin/R3
CHR=$1
MAF=$2
NPERM=$3
NPC=$4
RAD=$5
EXCL=$6
LANG=C
echo "source('onepopConfig.R'); \
    onepopConfig(chrn=${CHR}, MAF=${MAF}, nperm=${NPERM}, \
        npc=${NPC}, radius=${RAD}, exclRadius=${EXCL})" | ${RCMD} \
       --no-save >& lk${CHR}_${MAF}_${NPERM}_${RAD}_${EXCL}.txt
```

- Use the scheduler implicitly (or somehow)

```
#!/bin/bash
for i in {1..22};
 do qsub -cwd -l lx6,large ./doOne.sh $i .005 3L 18 250000L NULL; echo $i;
done;
```

  At conclusion, 22 .rda files will be present for harvesting using the collectFiltered function.

6.1 Removal of empirically identified expression heterogeneity

? describe implications of batch effects in high-throughput experimental contexts. Various methods for adjustment of responses have been proposed; a very simple but evidently risky approach is nonspecific removal of principal components of variation. To maximize information on possible batch effects, the entire expression matrix is restored and decomposed for principal component removal.

```
> if (file.exists("db2")) unlink("db2", recursive=TRUE)
> g20 = getSS("GGdata", "20")
> exTx = function(x) MAFfilter( clipPCs(x,1:10)[fn, ], lower=0.05)
> g20f = exTx(g20)
```

```
> b2 = best.cis.eQTLs("GGdata", ~male,  radius=50000,
+    folderstem="db3", nperm=2, geneApply=mclapply,
+    smFilter= exTx, chrnames="20", snpannopk=snplocsDefault())

> b2

GGtools mcwBestCis instance.  The call was:
best.cis.eQTLs(smpack = "GGdata", rhs = ~male, folderstem = "db3",
    radius = 50000, chrnames = "20", geneApply = mclapply, snpannopk = snplocsDefault()
    smFilter = exTx, nperm = 2)
Best loci for 50 probes are recorded.
There were 7838 gene:snp tests.
Top  4 probe:SNP combinations:
GRanges object with 4 ranges and 6 metadata columns:
                seqnames              ranges strand |     score       snpid
                   <Rle>           <IRanges>  <Rle> | <numeric> <character>
  GI_34147330-S       20 [25230834, 25421618]      * |     36.49   rs6037097
     hmm26961-S       20 [62615697, 62721315]      * |     16.78   rs3810504
  GI_22538441-S       20 [57520242, 57632309]      * |     10.46    rs163782
      hmm9615-S       20 [13926146, 16083841]      * |     10.84   rs7272442
                   snploc radiusUsed      nsnp       fdr
                <integer>  <numeric> <integer> <numeric>
  GI_34147330-S  25347221      50000        94 0.0000000
     hmm26961-S  62678549      50000        47 0.0000000
  GI_22538441-S  57565943      50000        66 0.1111111
      hmm9615-S  15275538      50000      2480 0.1250000
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
====
use chromsUsed(), fullreport(), etc. for additional information.
```

An improvement in sensitivity can be seen after this adjustment. The probes with FDR at 0.13 (used for demonstration purposes with this drastically reduced data) or lower are identified by the helper function

```
> goodProbes = function(x) names(x@scoregr[elementMetadata(x@scoregr)$fdr<0.13])
```

All probes identified as significant (at FDR $\leq$ 0.13) before the PCA adjustment are identified as such after it:

```
> setdiff(goodProbes(b2), goodProbes(b1))

[1] "GI_22538441-S" "hmm9615-S"
```

The adjustment can lead to loss of significance for some probes.

```
> setdiff(goodProbes(b1), goodProbes(b2))

[1] "GI_17149835-I"
```

The effects of the adjustment for genes that were significant only in the adjusted analysis can be visualized:

```
> newp = setdiff(goodProbes(b2), goodProbes(b1))
> np = length(newp)
> bestSnp = function(pn, esm) elementMetadata(esm@scoregr[pn])$snpid
> par(mfrow=c(2,2))
> plot_EvG(probeId(newp[1]), rsid(bestSnp(newp[1], b2)), g20, main="raw")
> plot_EvG(probeId(newp[1]), rsid(bestSnp(newp[1], b2)), g20f, main="PC-adjusted")
> plot_EvG(probeId(newp[np]), rsid(bestSnp(newp[np], b2)), g20, main="raw")
> plot_EvG(probeId(newp[np]), rsid(bestSnp(newp[np], b2)), g20f, main="PC-adjusted")
```

7 Exercises

1. All computations performed above ignore familial structure in the data that can be determined using the `famid`, `mothid`, `fathid` variables in the `pData(g20)`. Reduce the `smlSet` instance used for eQTL testing to parents only, who have parent identifiers equal to zero, and recompute the main tables.

2. For selected eQTLs that are were significant with low FDR in the full data ignoring, but are not significant in the analysis of the reduced data, use a reasonably specified variance components model on the full data with familial structure and compute a third test statistic. Is the restriction to parents only a good policy for eQTL discovery? Is there evidence of substantial familial aggregation in expression after heterogeneity reduction?

3. How can we select in a principled way the number of principal components to be removed for heterogeneity reduction?

8 Appendix: building your own structures

The following code illustrates construction of a minimal smlSet instance.

```
> library(Biobase)
> suppressPackageStartupMessages(library(GGtools))
> ex = matrix(0, nr=5, nc=3)
> pd = data.frame(v1 = 1:3, v2=5:7)
> colnames(ex) = rownames(pd) = LETTERS[1:3]
> adf = AnnotatedDataFrame(pd)
> rownames(ex) = letters[1:5]
> es = ExpressionSet(ex, phenoData=adf)
> exprs(es)

  A B C
a 0 0 0
b 0 0 0
c 0 0 0
d 0 0 0
e 0 0 0

> pData(es)

  v1 v2
A  1  5
B  2  6
C  3  7

> library(snpStats)
> mysnps = matrix(rep(1:3, 10), nr=3)  # note 1=A/A, ... 0 = NA
> rownames(mysnps) = colnames(ex)
> mysm = new("SnpMatrix", mysnps)

coercing object of mode  numeric  to SnpMatrix

> as(mysm, "character")

  [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9]  [,10]
A "A/A" "A/A" "A/A" "A/A" "A/A" "A/A" "A/A" "A/A" "A/A" "A/A"
B "A/B" "A/B" "A/B" "A/B" "A/B" "A/B" "A/B" "A/B" "A/B" "A/B"
C "B/B" "B/B" "B/B" "B/B" "B/B" "B/B" "B/B" "B/B" "B/B" "B/B"

> as(mysm, "numeric")
```

```
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
A    0    0    0    0    0    0    0    0    0     0
B    1    1    1    1    1    1    1    1    1     1
C    2    2    2    2    2    2    2    2    2     2
```

```
> sml = make_smlSet(es, list(c1=mysm))
> annotation(sml)
```

```
character(0)
```

```
> colnames(smList(sml)[[1]])
```

```
NULL
```

9 Session information

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

- R version 3.3.3 (2017-03-06), `x86_64-apple-darwin13.4.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.36.2, BSgenome 1.42.0, Biobase 2.34.0, BiocGenerics 0.20.0, Biostrings 2.42.1, GGBase 3.36.0, GGtools 5.10.1, GO.db 3.4.0, GenomeInfoDb 1.10.3, GenomicFeatures 1.26.3, GenomicRanges 1.26.4, Homo.sapiens 1.3.1, IRanges 2.8.2, Matrix 1.2-8, OrganismDbi 1.16.0, S4Vectors 0.12.2, SNPlocs.Hsapiens.dbSNP144.GRCh37 0.99.11, TxDb.Hsapiens.UCSC.hg19.knownGene 3.2.2, XVector 0.14.1, data.table 1.10.4, illuminaHumanv1.db 1.26.0, org.Hs.eg.db 3.4.0, rtracklayer 1.34.2, snpStats 1.24.0, survival 2.41-2
- Loaded via a namespace (and not attached): AnnotationHub 2.6.5, BiocInstaller 1.24.0, BiocParallel 1.8.1, DBI 0.6, Formula 1.2-1, GenomicAlignments 1.10.1, Gviz 1.18.2, Hmisc 4.0-2, KernSmooth 2.23-15, R6 2.2.0, RBGL 1.50.0, RColorBrewer 1.1-2, RCurl 1.95-4.8, ROCR 1.0-7, RSQLite 1.1-2, Rcpp 0.12.9, Rsamtools 1.26.1, SummarizedExperiment 1.4.0, VariantAnnotation 1.20.3, XML 3.98-1.5, acepack 1.4.1, annotate 1.52.1, assertthat 0.1, backports 1.0.5, base64enc 0.1-3, biglm 0.9-1, biomaRt 2.30.0, biovizBase 1.22.0, bit 1.1-12, bitops 1.0-6, caTools 1.17.1, checkmate 1.8.2, cluster 2.0.6, colorspace 1.3-2, dichromat 2.0-0, digest 0.6.12, ensembldb 1.6.2, ff 2.2-13, foreign 0.8-67, gdata 2.17.0, genefilter 1.56.0, ggplot2 2.2.1, gplots 3.0.1, graph 1.52.0, grid 3.3.3, gridExtra 2.2.1, gtable 0.2.0, gtools 3.5.0, hexbin 1.27.1, htmlTable 1.9, htmltools 0.3.5, htmlwidgets 0.8, httpuv 1.3.3, httr 1.2.1, interactiveDisplayBase 1.12.0, iterators 1.0.8, knitr 1.15.1, lattice 0.20-34, latticeExtra 0.6-28, lazyeval 0.2.0, magrittr 1.5, matrixStats 0.51.0, memoise 1.0.0, mime 0.5, munsell 0.4.3, nnet 7.3-12, plyr 1.8.4, reshape2 1.4.2, rpart 4.1-10, scales 0.4.1, shiny 1.0.0, splines 3.3.3, stringi 1.1.2, stringr 1.2.0, tibble 1.2, tools 3.3.3, xtable 1.8-2, yaml 2.1.14, zlibbioc 1.20.0